Self-Managing Energy-Efficient Multicast Support in MANETs under End-to-End Reliability Constraints

Tridib Mukherjee, Georgios Varsamopoulos, Sandeep K. S. Gupta [*]

Impact Lab
School of Computing and Informatics
Arizona State University
Tempe, Arizona, USA

Abstract

Dynamic networks, e.g. Mobile Ad Hoc Networks (MANETs), call for self-healing routing protocols to tolerate topological changes imposed by node mobility. Moreover, emerging time-critical MANET applications such as disaster response and rescue, and battlefield operations, require support for real-time, reliable data streaming, while maintaining energy efficiency. However, most of the energy-efficient routing protocols rely on configuration parameters which need to be estimated and specified before the deployment phase. This paper proposes a **self-managing, energy-efficient multicast routing suite** based on the self stabilization paradigm. This suite uses (i) WECM, a *Waste Energy Cost Metric* designed for energy-efficient route selection, (ii) SS-SPST-E, a *Self-Stabilizing, Shortest-Path Spanning Tree protocol for Energy efficiency* based on WECM to maintain an energy-efficient, self-healing routing structure, (iii) SS-SPST-E*fc*, an enhanced SS-SPST-E with fault containment to decrease stabilization latency, (iv) AMO, an *Analytical Model for Optimization* framework to reduce the energy overhead of the route maintenance mechanism, and (v) self-configuration mechanisms that observe, estimate and disseminate the optimization parameters.

The WECM's innovation is that it considers the overhearing energy wasted. The AMO framework considers the link state change rate, application data traffic intensity, application packet delivery requirements, and the stabilization latency. Numerical evaluations show that SS-SPST-E slightly increases the energy consumption when compared with non-adaptive energy-efficient protocols such as EWMA because of its mechanism to handle mobility. Simulation results show that SS-SPST-E*fc* achieves the maximum balance between the energy-reliability trade-off while conforming to the end-to-end packet delivery requirement with an accuracy between 80% and 100%. The energy-reliability balance, measured in terms of the Packet Delivery Ratio (PDR) per millijoules of energy expended, is at least 24% and 27% higher in SS-SPST-E and SS-SPST-E*fc*, respectively, when compared to the MAODV and ODMRP protocols.

Key words: Self Management, Energy Efficiency, Mobile Ad hoc Network, Self-stabilization, Multicasting

1. Introduction

Autonomy and self-management are envisioned to reduce the operational cost of large distributed and complex computing systems [24]. In recent years, with the growth and complexity of the Internet and other computer networking

[*] Corresponding author.
Email address: sandeep.gupta@asu.edu (Sandeep K. S. Gupta).
URL: impact.asu.edu (Sandeep K. S. Gupta).

paradigms such as *Mobile Ad hoc NETworks* (MANETs), research has been focused on the self-manageability of the networks [1]. This is especially challenging in MANETs, where existing resources become unreachable and new resources become available in a dynamic manner. Autonomic computing paradigm envisions self-healing behavior in the network to detect, diagnose, and repair localized faults caused by such dynamic changes. A principal challenge is to—a) automatically organize the routing structure in the presence of: i) the dynamics in the wireless medium, ii) mobility of the wireless nodes and iii) the limitations of energy consumption in the battery-powered mobile nodes, and b) adapt and optimize the route maintenance mechanism to these parameters. This paper intends to *incorporate a self-managing behavior in the routing structure for MANETs* in the context of multicast—an important group communication primitive for which the broadcast nature of the wireless medium can be used to save energy and reduce end-to-end delay [39]. The defining advantage of the self-management behavior for multicasting in MANETs is the automatic adaptation of the multicast routing structure to the aforementioned network dynamics. Applications of such autonomic multicast communication include critical applications such as disaster rescue and military applications, and traditional applications such as software updating, teleconferencing and on-line gaming.

1.1. *Motivation*

The lack of any fixed infrastructure in MANETs increases the difficulty of the multicasting problem described in the previous section, since routing information among the distributed mobile nodes need to be exchanged regularly to update any changes in the network. In current routing protocols, this exchange is performed either locally among the neighbors (in distance-vector protocols) or through broadcast among all the nodes (link-state protocols) [32]. In any case, flooding of route-updates takes place to diffuse the updates across the network [7,30]. This however consumes a lot of battery power, leading to depletion of node energy, subsequently leading to low scalability, especially when the number of receivers is high [28]. Instead, a more autonomous approach is required, where the nodes are designed to take local actions based on the neighborhood information without requiring communication across the network. An efficient way to ensure adaptivity and self-healing to topological changes based on local autonomous actions is the use of *self-stabilization* [28], originally proposed by Dijkstra for distributed systems [12].

Self-stabilization has been applied to multicast routing for MANETs in [19]. In that scheme, each node periodically exchanges beacon messages (consisting of the local knowledge of the network) with the one-hop neighbors. Local actions are performed at each node based on the i) local knowledge of the network, and ii) information in the beacon messages from the one-hop neighbors. The actions are designed in such a way that they converge to a valid multicast tree, which is one of the efficient distribution structures for multicasting [18, 31], from any faulty state, without any further exchange of routing information. Topological changes in MANETs can be thought of as faults and thus self-stabilization has the ability to enable adaptation in an autonomic manner to topological changes [18, 28]. Self stabilization alone is not adequate to address the routing problem in MANETs; energy efficiency—a prime concern in MANETs—is not considered in the self-stabilizing multicast protocols [18]; moreover, due to their repetitive nature, when self-stabilizing algorithms are implemented as message-passing protocols, the frequency of iterations determines the energy overhead of the protocol [28]; additionally, self-stabilizing protocols suffer from the problem of *fault propagation* [15, 16]. This paper enhances our previous work [28] to cover all these gaps.

Developing a *self-managing*, *reliable*, *energy-efficient* routing protocol should address the following functional and performance requirements:

– *Find energy-efficient routes*. Construction of an optimal energy-efficient broadcast/multicast tree is an NP-complete problem in Wireless Ad hoc NETworks (WANETs). WANETs are a special type of MANETs without node mobility. Thus, the minimum energy tree construction problem becomes even harder in MANETs due to the node mobility. Several heuristic protocols have been proposed [10, 25], which develop energy-efficient trees based on transmission and reception energy consumptions at the nodes in a multicast tree. These heuristics are applied as a refinement process over an initial tree configuration. However, topological changes can invalidate the initial tree configuration, making these protocols non-adaptive to these changes. Further, the wastage of energy in receiving unwanted packets (at the non-tree nodes) due to the broadcast nature of the wireless medium is not considered by these protocols. Self-stabilizing multicast has to therefore develop proper heuristics for the tree construction, along with adaptivity to topological changes. These changes are detected through exchanging periodic beacon messages among the neighbors which lead to energy-overhead.

– *Minimize the protocol's energy overhead*. There is a trade-off between adaptivity and the overhead energy consumplete [28]. Self-stabilizing protocols rely on the perpetual and periodic exchange of messages to reach, i.e. stabilize to, a proper and correct global state from any arbitrary state. The *frequency* of this periodic exchange, also known as *beaconing*, along with the per-packet energy, is the dominant factor that determines the energy overhead over time.
– *Respect the reliability requirements of the application*. The application reliability requirements, normally measured by the percentage of successfully delivered packets or *Packet Delivery Ratio (PDR)* [34], can affect the beacon overhead [27].
– *Minimize the route unavailability time*. In addition to the energy overhead, a major contributing factor to the beaconing is the stabilization latency which is defined as the beacon-interval (i.e. the time between two consecutive beacon messages) multiplied by the complexity (in terms of the number of beacon intervals) of the protocols to recover (stabilize) from topological changes [15]. Self-stabilizing networking protocols, in general, can lead to high stabilization latency [16]. This is due to their diffusing nature which enables faults to get propagated across the network before getting corrected. This is a direct consequence of asynchrony in the local actions of the distributed nodes allowing undesired sequence of executions [15, 16]. An efficient way to reduce stabilization latency is the fault-containment scheme [16], which enforces correct order of operation among the asynchronous nodes so that any change can be *locally contained* without any propagation.
– *Be free of requiring tuning from external (human) agents*. Most routing protocols require tuning of parameters to achieve optimal performance. A well-known example is the problem of tuning the *route update frequency* of proactive routing protocols [27]: such tuning requires knowledge of the data traffic pattern and the link breakage pattern. Therefore, tuning a self-stabilizing routing protocol, it being a proactive protocol with respect to data traffic, requires the knowledge of those parameters. However, a self-managing protocol should not rely on pre-configured parameters and it must adapt to the changing environment. This calls for the protocol's own-ability to (i) monitor the conditions, (ii) estimate the required optimization parameters, (iii) perform the optimizing calculations, and (iv) configure all the distributed nodes with the new values.

A preliminary version of this paper has focused on developing a self-stabilizing energy-efficient multicast tree in an autonomic manner [28] for MANETs. This paper enhances the preliminary work by applying fault-containment in the self-stabilizing energy-efficient multicast tree to minimize the route unavailability time. Additionally, the overhead in constructing the energy-efficient multicast tree is minimized while respecting the end-to-end reliability requirements. By integrating all these components, a self-managing, reliable, and energy-efficient multicast suite is proposed (Figure 1). The suite involves parallel local operations at each mobile node to emerge to an energy-efficient multicast tree with minimum energy-overhead while respecting an end-to-end reliability requirement at the network layer.

1.2. *Proposed scheme and contributions*

We address the aforementioned routing requirements in the following ways:

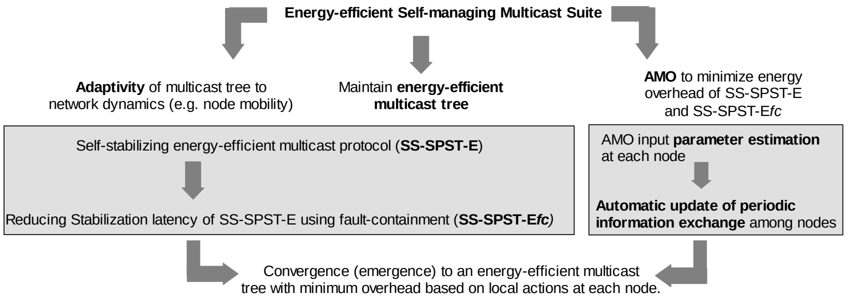
Energy efficiency: We use the Wastage Energy based Cost Metric (WECM) [28], which considers the energy wastage for the unnecessary reception of packets at non-intended nodes. The WECM model is considered to be an improvement over the *farthest neighbor* and *maximum transmission* costs [28]. Our simulations show that WECM allows for the construction of routes with a lesser impact on the nodes energy.

Route adaptivity to mobility: We use the self-stabilization paradigm and we improve previous work on Self-Stabilizing Shortest-Path Spanning Tree construction [17, 19, 28] to develop the SS-SPST-E protocol, which is the use of WECM in the SS-SPST protocol. Delivery to member nodes is achieved by pruning the spanning tree of the branches with no member nodes.

Minimization of stabilization time: We use the related work on fault-containment [15, 16] to define the *novel* SS-SPST-E*fc*, the fault-containing SS-SPST-E algorithm.

Minimization of protocol overhead *under reliability requirements*: We use our previous work on optimizing the periodic maintenance of routes [27] and propose AMO, the Analytical Model of Optimization framework to address the trade-off between stabilization time and overhead, and calculate the minimum overhead given a constraint on the stabilization time and reliability requirements, expressed as the required PDR.

Self management: We use the related work on Bayesian sequential estimation [29] to yield the traffic distribution



Energy-efficient Self-managing Multicast Suite
Adaptivity of multicast tree to network dynamics (e.g. node mobility)
Maintain energy-efficient multicast tree
AMO to minimize energy overhead of SS-SPST-E and SS-SPST-Efc
Self-stabilizing energy-efficient multicast protocol (SS-SPST-E)
Reducing Stabilization latency of SS-SPST-E using fault-containment (SS-SPST-Efc)
AMO input parameter estimation at each node
Automatic update of periodic information exchange among nodes
Convergence (emergence) to an energy-efficient multicast tree with minimum overhead based on local actions at each node.

Fig. 1. **Paper contributions**: the bold-faced texts signify the paper contributions. There are two aspects for developing an energy-efficient multicast suite – i) maintaining an adaptive and energy-efficient multicast tree, and ii) energy-overhead minimization. SS-SPST-E and SS-SPST-E*fc* address the first aspect and leads to the same energy-efficient tree with different stabilization time. AMO minimizes the overhead by optimizing the period of information exchange at each node. SS–SPST-E*fc* allows lower energy overhead due to low stabilization time. Together, SS-SPST-E*fc* and AMO emerge (converge) to an energy-efficient multicast tree with minimum energy overhead; thus the overall energy consumption is reduced.

parameters, and exponential moving average to estimate the link breakage rate as observed from the beaconing. Combining the above approaches, we build a self-managing, energy-efficient self-healing multicast routing protocol. Figure 1 summarizes the contributions.

We compare the performance of SS-SPST-E and SS-SPST-E*fc* with other MANET multicast protocols such as Multicast Operation of the Ad Hoc On-Demand Distance Vector routing (MAODV) [32] and On-Demand Multicast Routing Protocol (ODMRP) [14]. Our simulation results show that, in general, self-stabilization leads to higher scalability (with respect to the number of nodes in the multicast group). Furthermore, in case of Constant Bit Rate (CBR) multicast traffic and when mobile nodes move between the walking and in-city driving speeds, SS-SPST-E and SS-SPST-E*fc* achieve at least 18% and 23% reduction in energy consumption, respectively, than the ODMRP protocol. Although the energy consumption in MAODV is lower, the reliability in packet delivery is however increased by at least 5% and 25%, respectively in the SS-SPST-E and SS-SPST-E*fc* protocols, over the MAODV protocol. Holistically, SS-SPST-E and SS-SPST-E*fc* achieve at least 24% and 27% higher PDR per mJoules of energy expended, respectively, over both the MAODV and ODMRP protocols. Thus, SS-SPST-E*fc* strikes the maximum balance between the energy-reliability trade-off. Additionally, with the use of the AMO, SS-SPST-E*fc* achieves the required PDR within 80% to 100% of accuracy. The rest of the paper is organized as follows. Section 2 gives an overview of the related work in addressing energy efficiency in MANETs, and using self stabilization to provide multicast support. Section 3 provides basic assumptions, the system model, and background on related cost metrics. Section 4 gives a detailed description of the proposed scheme, including WECM, SS-SPST-E, SS-SPST-E*fc*, AMO and the Self Management mechanism. Section 5 presents the simulation study, and finally, Section 6 concludes the paper.

2. Related Work

This section presents the related efforts to address energy efficiency in MANETs, and the use of self-stabilizing protocols for multicasting.

2.1. *Energy efficiency*

There are two different types of energy-efficient protocols for MANETs: i) protocols to minimize total energy consumption at all the nodes, and ii) protocols to maximize the network lifetime (which is the volume of tasks to be completed with a given energy level in the distributed nodes [11,13,22]). We concentrate on the first type of protocols. There are further two major aspects in minimizing the total energy consumption for adaptive routing in MANETs: i) minimizing overhead energy (i.e. energy consumption of control message transmission to ensure adaptivity), and ii) minimizing energy consumption during data transmission. To reduce the overhead energy for route maintenance, many of the routing protocols are reactive in nature [44]. However, reactive approaches are not self-healing in nature because they do not take local autonomous actions. This leads to certain drawbacks in terms of high latency (due to the route re-construction delay whenever there is data to transmit) [44], low scalability (due to the routing information exchange across the network, which leads to flooding for a large number of destination nodes) [7, 30], and high energy consumption (for large data traffic in the network) [44]. A recent work [45] in this regard proposes an adaptive energy-aware hybrid protocol EAGER, which increases pro-activity depending on the rate of traffic and topological changes in a region of the network. However, energy-efficient route-construction (for minimizing energy consumption during data transmission) is not considered.

The energy consumption during data transmission can be reduced in various ways [21]. In [35], this is performed by turning-off unused transceivers. However, this technique can have a high overhead (due to the higher start-up energy during turning on the transceivers) when the rate of switching between the on and off state is very high [36]. Further, proper scheduling of transceiver wake-up cycles is required to ensure no loss of traffic due to transceiver shutdown. However, this leads to a high scheduling and synchronization overhead [46]. An efficient technique to ensure energy-reduction during data transmission is power control [8, 10, 25, 35, 40]. In this approach, a node can adjust its transmission power to a level which is sufficient to reach the receiving node. In this paper, we use the power control mechanism for enhancing energy efficiency.

Due to the fact that the problem of constructing an optimal energy-efficient broadcast/multicast tree using power control is NP-complete in WANET, several heuristic protocols for building energy-efficient multicast tree have been developed [10] [25], such as BIP/MIP/Dist-BIP [40], EWMA/Dist-EWMA [8], S-REMiT [39] and G-REMiT [38]. There are different types of cost metrics used in this regard to achieve power-control, e.g. transmission energy based metrics [35], and costliest (farthest) neighbor based metrics [8, 10, 25, 40]. The former only considers the transmission energy required per link. The latter considers transmission energy per node along with the reception energy of all the neighbors in the multicast tree. However, none of these protocols consider the possible energy wastage due to reception of unwanted broadcast/multicast packets at the non-intended (e.g. non-tree) nodes. Such overhearing energy has a high impact on the overall energy efficiency especially in case of group communication such as multicast [6]. Further, all these protocols require an explicit initialization phase, after which energy optimization is performed as a refinement process. As a result, any topological change due to mobility of nodes invalidates the energy optimization, making these protocols non-adaptive to topological changes. This paper uses self-stabilization for multicasting to enable such adaptivity based on local autonomous actions in the distributed nodes.

2.2. *Self-stabilizing Multicast*

Self-stabilization has been first utilized for multicast tree maintenance in MANETs in [19], where Self-Stabilizing Shortest Path Spanning Tree (SS-SPST), and Self-Stabilizing Minimum Spanning Tree (SS-MST) protocols have been proposed. However, energy efficiency has not been considered. This paper incorporates energy efficiency in self-stabilizing multicast. A preliminary version of the paper has been presented in [28]. The primary contributions of the preliminary version include: i) the WECM model, ii) the heuristic for energy-efficient tree generation in SS-SPST-E, and iii) simulation based comparative study of SS-SPST-E with SS-SPST, MAODV, and ODMRP protocols . However, the balance of energy-reliability trade-off was not studied for these protocols. Further, the overhead energy consumption due to the periodic beacon transmission was not considered in the design of the SS-SPST-E protocol. Also, self-stabilization can lead to high stabilization latency by allowing diffusion of local failures across the network [15, 16]. Local stabilization of the faults is therefore important to reduce the stabilization latency. This can further enable lowering of the beacon transmission frequency; and thus reduction in beacon overhead.

A recent work in this regard is LSRP [5], a local stabilization protocol for shortest path routing. This intends to bound the stabilization latency to the perturbation size in the network. However, it is developed for traditional infrastructure-based network models such as the Internet and is not applicable to MANETs. For instance, LSRP sends beacons with each and every change in the network. This is cumbersome in MANETs due to its highly dynamic nature. Therefore, using LSRP for MANETs may lead to high beacon transmission, with every change in the network, depleting energy in each node as well as hogging the bandwidth of the wireless channel. An efficient way to reduce stabilization latency is the fault-containment scheme [16], which enforces correct order of operation among the asynchronous nodes so that any change can be *locally contained* without any propagation. Fault-containment, unlike LSRP, does not have any specific assumptions on the network model for its application [16]. It only requires augmentation of some additional boolean flags in the periodic beacons [16], making it easily suitable for MANETs.

This paper enhances the previous work by – i) proposing the AMO model to optimize the overhead due to periodic beacon transmission; ii) applying fault-containment over the SS-SPST-E protocol to develop SS-SPST-E*fc* for reducing the stabilization latency; iii) developing a self-managing multicast suite for MANETs by adapting the SS-SPST-E*fc* and the optimum beacon periods to the changes in the network conditions and application requirements; and iv) simulation based comparative analysis of SS-SPST-E*fc* with SS-SPST-E, SS-SPST, MAODV, and ODMRP protocols.

3. Preliminaries

In this section, first we present the system model and assumptions for the MANETs followed by a brief overview on how the self-stabilizing multicast protocols operate in MANETs.

3.1. *System Model*

We consider a network of n mobile nodes, which are distributed randomly. We assume that each node in the MANET is identified by a unique identifier. When a node transmits (*broadcasts*) a message, the nodes in its *coverage area* can (almost) simultaneously hear the message. This assumption is valid because of the broadcast nature of the wireless medium. Nodes are equipped with IEEE 802.11 MAC wireless interface cards, which are the most popular wireless cards used in the mobile devices. An 802.11 MAC employs random access of the wireless channel through the RTS/CTS/DATA/ACK mechanism [2]. This can ensure reception of unicast packets in only the intended receivers. However, 802.11 MAC does not ensure the same for broadcast packets, in which case the sender simply performs CSMA/CA channel access [3] before broadcasting a DATA frame [2]. Due to the broadcast nature of the wireless medium, multicast packets are transmitted (broadcast) once to reach all the nodes in the transmission range (also known as *wireless multicast advantage*). Therefore even the non-intended nodes will expend energy in listening the multicast messages (called overhearing [6,11]) and then discard it. We define this wasted energy as the *wastage energy* and intend to minimize it while performing self-stabilization.

The MANET is modeled by an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of links between neighboring nodes. Further, there are n nodes in the network i.e. $|V| = n$. The set E is dynamic in nature due to node movements and the resulting link disconnections (and/or new link formations). Each node has distinct power levels for transmissions. Any pair of nodes i and j is said to be adjacent to each other if the link $(i, j) \in E$ when i transmits with highest transmission power level (C_i^{max}). If T_{ij} denotes the minimum transmission energy required for a packet to reach j from i, then the link $(i, j) \in E$ if $T_{ij} \leq C_i^{max}$. Further, all the nodes have power control capabilities to vary their transmission power level [23]. A node j is defined as the *costliest neighbor* of i if j can be reached with current power level of i but can not be reached with the next lower power level. We also assume that the reception energy is constant for all the nodes. This is a valid assumption in most of the cases as the significance of transmission power on the reception energy becomes noteworthy only if the transmitter and receiver are very close to each other [37]. Inclusion of transmission power dependent reception energy [37] would be a simple extension to the work presented in this paper. The cost C_i of each node i includes T_{ij} and the reception energy of the receivers. A detailed specification of node costs will be provided in Sections 3.4 and 4.1. Table 1 and 2 summarize the symbol definitions for global and local parameters respectively.

The multicast data packets are sent from source to the destinations (in the multicast group). Each node periodically transmits *beacon* messages (with β being the configured beacon interval in seconds i.e. the time interval between two consecutive *beacon* transmissions of a node). The beacon messages contain the transmitter node id to advertise transmitter's existence to the neighbor nodes. Further, the beacons are transmitted with maximum power level by each node (C_i^{max}) to advertise to all the possible neighbors. If a node does not receive any beacon from a neighbor for k ($k \geq 1$) number of consecutive beacon periods the corresponding link is assumed to be broken. The mobility assumptions are based on the work of link dynamics [33], which assumes a random way-point model and establishes that the link change inter-arrival time is exponentially distributed. The analysis in this paper assumes that the link change rate has a mean value of μ. The simulations use a random way-point mobility model.

3.2. *Self-stabilizing Multicast for MANETs*

Before proceeding any further, in this subsection we provide a brief overview on how self-stabilization works and its application operates in MANETs.

Self-stabilization enables local autonomous actions in the distributed nodes. Each action further consists of two parts: guard, and statement, and has the following form:

$$\langle guard \rangle \longrightarrow \langle statement \rangle$$

where *guard* is a boolean predicate over the protocol variables, and the *statement* updates zero or more protocol variables. The statements are executed only when the guarding predicates become true. There is a global *Legitimacy Condition (LC)* which each node has to abide by at any instance. The LC captures the validity of the global behavior. The guards are designed in such a way that they become true (triggering execution of the statements) at the appropriate nodes whenever the LC becomes false. A node is said to be stabilized at any time if it adheres to the LC at that instance (i.e. the guard is not true). For any self-stabilizing algorithm, a LC must be defined. Also, it must be proven that when the LC becomes false the local actions at each node converge to a global state where the LC becomes true.

Self-stabilizing multicast protocols for MANETs use hop metric to build multicast spanning tree. These algorithms construct source-based multicast trees (i.e. the multicast source is the root of the tree) in a best-effort manner where each node selects the best candidate among the neighborhood as the *parent*, i.e. the selected next hop for the path toward the root. For example, in SS-SPST [19], the best candidate is chosen as the neighbor with minimum number of hops from the root (source) node. At each node, the protocol consists of a finite set of variables and actions. The variables include the current estimation of the hop count of the node and its neighbors; the actions are performed based on these information to construct Shortest Path Spanning Tree (SPST). The guards are evaluated at each **round**, which is defined to be the time period in which each node in the system receives at least one beacon message from each of its neighbors. The LC for SPST ensures that all the nodes in the tree have the shortest route to the root.

The SPST [19] is constructed in a top-down manner where the root node stabilizes first, followed by the nodes in the next higher levels until the leaf nodes are stabilized to form the entire tree. The stabilization latency is determined by the number of rounds the protocol takes to stabilize to a valid SPST (i.e. all the nodes adhere to LC, and therefore guard is not true in any of the nodes). The next logical step is to prune the tree so that multicast messages are not forwarded to non-group members. This is achieved by maintaining a flag for each node. The flag becomes true if there is a group member downstream of the node. This information is gathered in a bottom-up manner from the leaf node to the root node. Flag becomes false when there is no member in the downstream and thus that portion of the tree is logically pruned. The implementation of pruning in this paper is described in Section 4.2.2.

Example 1 *Figure 2 shows the initial connections of the mobile network*[1]. *Since there is no multicast tree, the parents of all the nodes are $NULL$, and the nodes' hop-counts to the root are set to ∞. In the first round, the root node stabilizes by setting its hop-count to itself as* 0. *After the root node stabilizes, in the second round, all the root's neighbors (i.e. nodes 1, 2, 3, 4, and 6) stabilize by setting their parent to be the root node and the hop-count to the root to 1. Similarly, in the next round, all the nodes, which are two hops away from the root node, stabilize by selecting the neighboring nodes with an hop-count of* 1. *Figure 3 shows the stabilized SPST. SS-SPST protocol takes 3 rounds to stabilize in this case. Pruning is not shown in the figures and can be achieved easily by removing the links* $(4, 8)$, $(4, 9)$, *and* $(2, 7)$, *based on the aforementioned method.*

3.3. *The problem statement of energy-efficient routing*

Given the system model and the operations of the self-stabilizing multicast, we identify the following fundamental problems: 1) how to construct energy-efficient multi-cast tree in a self-stabilizing manner; and 2) how to minimize energy consumption due to periodic beacon messages. The problems can be specified in one as:

[1] Note here that when node 0 transmits at power level 2 it reaches the nodes 1, 2, 4, and 6; but does not reach node 3 which require a higher power level.

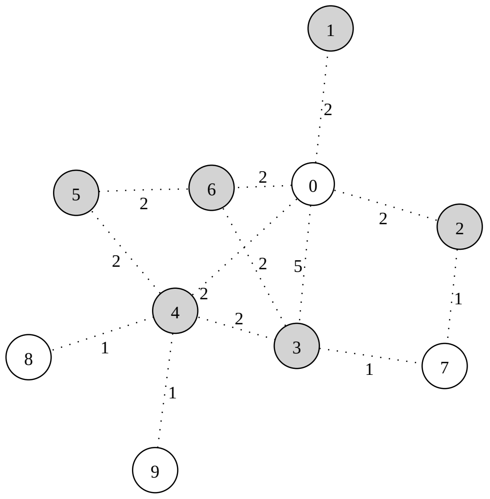


Fig. 2. Original Network : Neighboring nodes are connected using straight lines. Node 0 is the multicast source node. The edge weights denote the power level required for transmission between the connecting nodes (higher weight signifies higher power level). The colored nodes are the multicast group members.

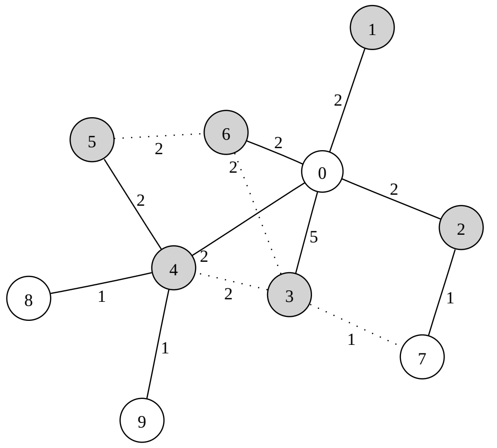


Fig. 3. SS-SPST : The tree is stabilized by minimizing the hop count from the source node

$$minimize \sum_{i=0}^{i=n} (\frac{1}{\Lambda} C_i + \frac{1}{\beta} C_i^{max}) \tag{1}$$

where n is the total number of nodes in the network, Λ is the rate of traffic arrival, and C_i^{max} is the cost at each node i when maximum power level is used for transmission. The connectivity of the multicast tree, along with the application PDR requirement acts as the constraints. As the nodes are capable of controlling the transmission power, there are two unknown variables in the problem—C_i, and β. We will see in the next section that C_i varies on the transmission range of i and is not co-related to β. Therefore, we solve the aforementioned problems by breaking the objective (Equation 1) in two parts. The first part ($\sum_{i=0}^{i=n} \frac{1}{\Lambda} C_i$) ensures energy-efficient tree construction. This is a NP-complete problem and we provide a heuristic solution. The second part of the objective ($\sum_{i=0}^{i=n} \frac{1}{\beta} C_i^{max}$) requires maximization of β and is solved based on standard non-linear optimization technique for a single variable[2] and the solution is applied to the proposed heuristics.

3.4. *Applying Available Cost Metrics to SS-SPST*

In this section, we give an overview of different types of cost-metric (to calculate C_i) in the literature and show how they can be applied to improve the SS-SPST algorithm for energy efficiency. In [26, 35, 41], the authors have proposed a link-based metric which captures the transmission energy spent by the node to maintain a link. By assigning transmission energy expended to the links, SS-SPST can build a tree which minimizes the total transmission energy. Let C_{ij} be the total energy cost of link between i and j. Then we have,

$$C_{ij}^T = T_{ij}. \tag{2}$$

We name the SS-SPST with the above metric to be SS-SPST-T, where 'T' denotes the transmission cost. It should be noted here that SS-SPST-T is similar to the distributed Minimum Spanning Tree (MST) problem and therefore we do not provide any specification for SS-SPST-T.

Example 2 *In the topology of Figure 2, when we apply SS-SPST-T algorithm, the resulting tree is constructed as in Figure 4. The protocol steps follow a similar pattern as in the Example 1 with the additional consideration of C_{ij} for*

[2] There is only one unknown variable, and therefore the optimization just boils down to reaching the extreme points of the constraints.

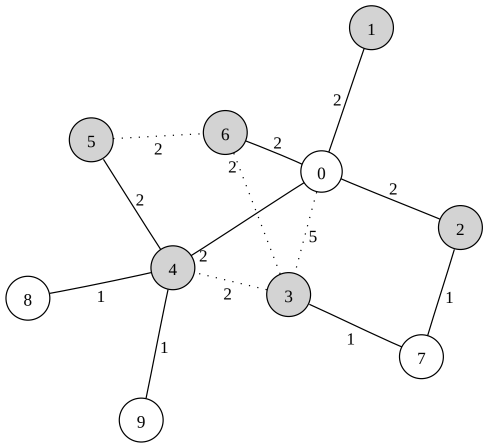


Fig. 4. SS-SPST-T - Transmission energy metric.

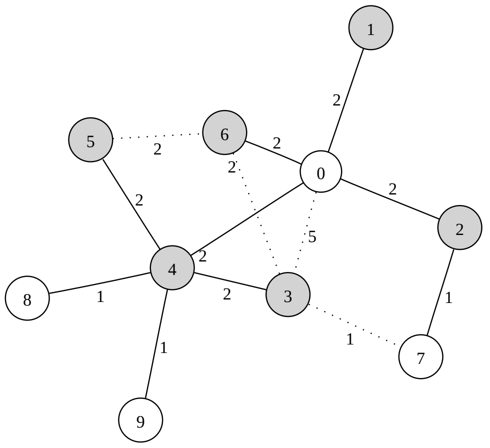


Fig. 5. SS-SPST-F - Costliest group node transmission energy.

each link at each node. According to the metric of Equation 2, it is more cost efficient if node 3 makes node 7 as its parent instead of node 0.

However, this metric does not capture the energy expended in real scenarios as each link is not required to spend separate transmission energy because of the multicast advantage of the wireless medium. Node based cost metric are considered in [9,40,42] where the authors propose schemes (such as BIP, and MIP) to build energy-efficient broadcast and multicast trees in wireless networks. These schemes take the maximum transmission energy of the transmitting node as the node metric. Let j be the node which is a neighbor of node i and is connected to node i by a link which requires node i to expend maximum transmission energy. We call node j to be the costliest neighbor of node i and the transmission energy for node i to reach node j is T_{ij}. Let t_i be the total number of tree neighbors of node i, R be the reception energy cost, and C_i be the total energy cost of node i. Then we characterize the cost of node i as,

$$C_i^F = T_{ij} + t_i \times R \tag{3}$$

where j is the costliest neighbor. We name the SS-SPST implementation with the above cost metric as SS-SPST-F, where 'F' signifies the costliest (farthest) node metric.

Example 3 *Figure 5 shows the tree constructed by applying the SS-SPST-F algorithm to the topology in Figure 1. The costliest neighbor of node 4 is node 5. So if node 3 makes node 4 as its parent the incremental cost at node 4 would increase only by the reception energy at node 3. Node 3 selects node 4 as its parent instead of node 0. After node 3, which is a former costliest child of node 0 (requiring a maximum transmission power level of 5), changes its parent in round 4, node 1 in the transmission range of node 0 becomes the costliest child (requiring a maximum transmission power level of 2). Therefore, taking node 4 as the parent of node 3 leads to significant reduction in the transmission energy at node 0 and incurs no additional transmission requirement at node 4 (which had already been transmitting at the power level of 2 to reach node 5).*

Node based metrics characterize the energy expended in a better way than a link based metric, however, none of the energy metrics in the literature consider the wastage energy discussed in Section 3.1.

Example 4 *Consider the example in Figure 6. Assume node X wants to join the multicast group and it has to make decision between node* 1 *and node* 2 *as its parent node in the multicast tree. All the edges have the same weights, therefore, all nodes transmit with the same transmission energy. In this example, total energy consumed will be reduced if node X chooses node* 2 *as its parent since node* 1 *has three non-group nodes as its neighbor and every time node* 1 *transmits, those three nodes will have to spend energy to receive the packet and discard it.*

4. The proposed self-managing multicasting scheme

This section provides a detailed description of the proposed self-managing multicasting support scheme and its components. It starts with a description of the WECM (Section 4.1) from our previous work [28], then a description of the SS-SPST-E along with proofs of correctness (Section 4.2). Section 4.3 describes the proposed optimization en-

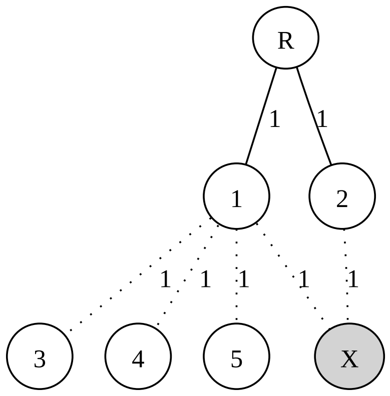


Fig. 6. Discard energy wastage.

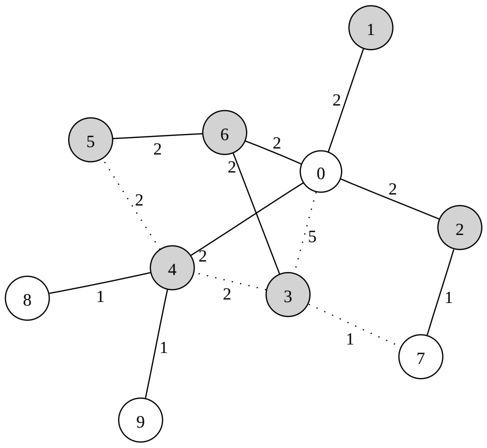


Fig. 7. SS-SPST-E - Costliest group node transmission energy with discarding energy.

hancements to the SS-SPST-E, including the fault-containing SS-SPST-E*fc* version. Lastly, Section 4.4 provides a discussion of the mechanisms that allow for autonomic and self-managed operation of the multicasting protocol.

4.1. *The Wastage Energy Cost Metric (WECM)*

To take into account wastage energy, we propose WECM by modifying the aforementioned cost metric (Equation 3) as follows. Let, L_i be the wastage energy spent by nodes to discard a packet when node i transmits to its costliest neighbor node j. Let N_i be the total number of neighbors of node i. Now, L_i can be presented as follows

$$L_i = R \times (N_i - t_i) \tag{4}$$

We can now represent our energy metric C_i as follows

$$C_i = C_i^E = T_{ij} + L_i + t_i \times R \tag{5}$$

where j is i's costliest child. Replacing L_i from Equation 4 to Equation 5, we get the following equation:

$$C_i = T_{ij} + N_i \times R \tag{6}$$

where j is i's costliest child. The above equation shows that to capture the wastage energy of the non-tree (i.e. non-intended) neighbors, reception energy at all the neighbors need to be considered instead of just the tree-neighbors (as in Equation 3). We name the protocol with the proposed WECM, given above, as SS-SPST-E.
Example 5 *Figure 7 shows the multicast tree with SS-SPST-E for the topology of Figure 2. It builds a tree similar to SS-SPST-F based on costliest neighbor while trying to minimize the discard energy expended by non-group neighbors. In Figure 5, whenever node 4 transmits a data packet, nodes 8 and 9 listen to it and discard it. In terms of energy consumption, it will be better for nodes 5 and 3 to join node 6 instead of node 4.*

4.2. *The SS-SPST-E Protocol (SS-SPST with WECM)*

This section formalizes the SS-SPST-E protocol. It first describes the basic local variables and protocol operations needed to achieve overall correct functionality,

4.2.1. *Local Variables and Protocol Operations*

The goal of the protocol is to construct and maintain energy-aware source-based multicast tree in a self-stabilizing manner based on local actions in each node. The protocol works in a best-effort manner, where each node locally selects the best parent to the root in the tree. Table 2 presents that symbol definitions for local parameters at each node. Each node $i \in V$ locally maintains the following basic protocol variables:

Table 1
Global Symbols and Definitions.

Symbol	Definition
λ	the average rate of Poisson distributed message arrival
α	the average number of packets per message
τ	the constant time between consecutive packets inside each message
Λ	the average rate of Bulk-Poisson packet arrival (given by Equation 10)
Γ	the PDR required by the application
G	Undirected graph modeling MANET ($G = (V, E)$)
V	set of nodes (mobile devices)
E	set of links (edges) between neighboring nodes
n	number of nodes in the network ($n = \|V\|$)
r	root (source) node for the tree
D	the diameter (in number of hops) of the network
c	average number of children for a node in the tree
μ	the average rate of change of every link in the network
η_p	the probability of packet loss over a single link in a route between any two nodes
k	the number of consecutive beacon intervals after which link disconnection is detected if no beacons are received during the time
δ	the worst case delay to re-establish valid route from any link failure
δ_{stab}	stabilization latency of SS-SPST-E
Ω	the ratio of the probability of packet loss (η_p) to the worst-case route reconstruction delay (δ) due to link failure (this is a function of λ, α, τ, Λ, and μ as shown in Equation 11)
β	average beacon interval
β_{opt}	optimum value of average beacon interval

(i) C_i, denoting the current energy cost of i (calculated as per Equation 6).
(ii) $H_i(r)$, denoting the hop count from i to the root node r along the multicast tree ($H_r(r) = 0$, and $H_i(r) = \infty$ if i is not a part of the tree).
(iii) P_i, denoting the current parent of i toward r along the multicast tree ($P_r = NULL$, and $P_i = NULL$ if i is not a part of the tree).

In addition, each node i maintains the set Adj(i) of all nodes adjacent to it (i.e. $N_i = |\text{Adj}(i)|$ in Equation 6). Note here that for all $j \in \text{Adj}(i)$, the link $(i, j) \in E$ at any given instance. A valid value of the variable P_i points to a node which is in the set Adj(i). Due to the dynamic nature of the network, the sets Adj(i) (for all $i \in V$) and E are not constant and change with node movements, link breakages, and new link formations. Each node i is involved in periodic beacon transmission to maintain information about Adj(i) (Section 3.1). The adjacent nodes also exchange their local variables through this periodic beaconing (see description of the beacon message content in Section 4.4). Upon reception of beacon messages, each node i updates its local information according to the received neighbor information through beacons. In this regard, the node i ($\forall i \in V$) performs the following two primary operations.

(i) **Determining the set of potential parents in Adj(i)**: The basic requirement for any adjacent node to possibly become a parent is contingent on the existence of a path along the multicast tree from the adjacent node to the root of the tree. Therefore, it is required to check if the adjacent node is 1) in the tree, and 2) not involved in any loop (or cycle) [20]. In this regard, we define the set

$$\mathcal{VP}(i) = \{j \mid j \in \text{Adj}(i) \land i \neq r \land H_j(r) < n\} \tag{7}$$

which characterizes the subset of nodes in Adj(i) that can be selected as a parent (based on the aforementioned requirements) of node i. Note here that always $\mathcal{VP}(r) = \emptyset$ as r is the root of the tree and therefore does not

Table 2
Local Symbols and Definitions at each node $i \in V$.

Symbol	Definition
Adj(i)	set of all nodes adjacent to node i
N_i	number of neighbors of node i ($N_i = \|Adj(i)\|$)
t_i	total number of tree neighbors of node i
C_i^{max}	highest transmission power level of node i
T_{ij}	transmission power level required to reach of node i ($T_{ij} \leq C_i^{max}$)
R	reception power of a node (constant for all the nodes)
C_i	current power cost of node i (calculated as per Equations 3 or 6)
$H_i(r)$	hop count node i to r
P_i	parent of node i to reach r along the multicast tree
$\mathcal{VP}(i)$	set of nodes in $Adj(i)$, that can be selected as parents
$O_i(j)$	overhead energy of node $j \in \mathcal{VP}(i)$, if i selects j as parent
O_i	minimum overhead energy among node i's adjacent nodes (Equation 8)
$\mathcal{N}(i)$	set of neighboring nodes of i which are on currently estimated energy-efficient paths from node i to r (Section 4.2.2)
f_i	flag of whether there is at least one node in node i's branch that is member of the multicast group
m_i	flag of whether the node i is a member of the multicast group

have any parent. It can further be verified that if a node $j \in$ Adj(i) ($i \in V - \{r\}$) is not part of the tree i.e. $H_j(r) = \infty$, then node $j \notin \mathcal{VP}(i)$. The significance of the value of $H_j(r)$ to be less than n in $\mathcal{VP}(i)$ is attributed to the requirement of selecting a parent, which is not part of any cycle. For example, in case a node $j \in$ Adj(i) is part of a cycle (such that j is the parent of its parent, i.e. $P_{P_j} = j$), $H_j(r)$ gets updated as $H_{P_j}(r) + 1$ and $H_{P_j}(r)$ gets updated as $H_{P_{P_j}}(r) + 1 = H_j(r) + 1$. This process would keep continuing after every round and eventually $H_j(r)$ would become greater than or equal to n. A known result in graph theory [20] is that the maximum hop distance from any node to the root along a tree can not be greater than $n - 1$. Therefore the value of $H_j(r)$ has to be less than n for node j to be a possible parent (which is not a part of any cycle) of node i. Although a node j can be a part of cycle while $H_j < n$, its H_j value would eventually reach a value greater than n, at which point it is removed from $\mathcal{VP}(i)$. Now, with the precise definition of the potential parents of node i, the parent selection is narrowed down from $P_i \in$ Adj(i) to $P_i \in \mathcal{VP}(i) \subseteq$ Adj(i).

(ii) **Selection of parent with minimum overhead energy cost**: Each node i further calculates the overhead energy cost of the potential parents and selects the node with the minimum overhead energy as its parent. For this purpose, the following auxiliary variables are maintained at each node i in $V - \{r\}$:
  - $O_i(j)$ - denotes the overhead energy cost of node i joining node j ($\forall j \in \mathcal{VP}(i)$).

  $O_i(j)$ is calculated as $O_i(j) = C_j^{iIS\,child} - C_j^{iNOT\,child}$, where $C_j^{iIS\,child}$ represents the energy cost (according to Equation 6) of j when i is its child and $C_j^{iNOT\,child}$ is the energy cost (according to Equation 6) of j when i is NOT its child. Now, based on these auxiliary variables, each node i, maintains another auxiliary variable O_i, which denotes the minimum overhead energy cost among node i's adjacent nodes, i.e.

$$O_i = \min_{\forall j \in \mathcal{VP}(i)} (O_i(j)) \tag{8}$$

Remark 6 *In a connected network of at least three nodes and at least two nodes in the multicast group, the minimum possible value for $O_i(j)$ is $O_{min} = T_{min} + R$, where T_{min} is the minimum transmission power supported by the wireless interface card and R is the reception power needed; while the maximum possible value for $O_i(j)$ is $O_{max} = T_{max} + (n - 2)R$, where T_{max} is the maximum transmission power supported by the wireless interface card, n is the number of nodes in the network ($n = |V|$). Moreover, if we have knowledge of the diameter of the network, D, the maximum possible value for $O_i(j)$ becomes $n - D + 2$.*

Remark 7 *For all i, O_i is determined by the topology of the (network) graph and group information.*

Note that always $O_r = 0$, because $\mathcal{VP}(r) = \emptyset$. After the parent selection is performed, the variables P_i and $H_i(r)$ are updated accordingly.

4.2.2. *Protocol Specification*

In this subsection, we specify the SS-SPST-E protocol. For notational simplicity, we further define the following set for any node i in the graph:

• $\mathcal{N}(i) = \{j \mid j \in \mathcal{VP}(i) \wedge O_i(j) = \min_{k \in \mathcal{VP}(i)}(O_i(k))\}$

The set $\mathcal{N}(i)$ contains the neighbors of i which have a path to r and have minimum overhead energy if taken as parent. Note that $\mathcal{N}(r) = \emptyset$. Formally we can describe SS-SPST-E as follows:

```
SS-SPST-E Action Specification at each node i:

1. $(i = r \wedge (H_i \neq 0 \vee P_i \neq NULL)) \longrightarrow H_i = 0; P_i = NULL; O_i = 0;$
2. $\left(i \neq r \wedge (O_i \neq \min_{\forall j \in \mathcal{VP}(i)}(O_i(j)) \vee P_i \notin \mathcal{N}(i) \vee H_i \neq H_{P_i} + 1 \vee H_i > n)\right) \longrightarrow$
                    $O_i = \min_{\forall j \in \mathcal{VP}(i)}(O_i(j)); P_i = j, j \in \mathcal{N}(i); H_i = H_j + 1;$
```

The pruning is performed in a bottom-up manner from the leaf to the root node. Each node i maintains a boolean variable m_i, which denotes the multicast group membership. Nodes in the multicast group set the variable m_i to *true*; otherwise it is set to $false$. Apart from this, a flag f_i is maintained which denotes the existence of a multicast member in the downstream of node i. The value of f_i becomes 1 if there is a group member in the downstream of the node. A leaf node sets f_i to 1 if it is a group member i.e. m_i is true, otherwise f_i is set to 0. In the stabilized SS-SPST-E, each node i determines if it is a leaf node if it has no neighbor j such that $i = P_j$. Any intermediate node sets f_i to 1 if any of its child j has either m_j set to true (i.e. a group member) or f_j set to 1 (i.e. has a group member in the downstream). In this way, the flag gets stabilized in a bottom-up manner. During multicast operations, a node i with the value of f_i set to 0 does not forward any packet, resulting in a logical pruning of the portion of the tree downstream to i. The pruning procedure of the tree built by SS-SPST-E is similar to the SS-SPST protocol (Section 3) and takes $n - 1$ round to stabilize [19].

4.2.3. *Protocol Execution*

We will now demonstrate the execution of the SS-SPST-E protocol on the example network of Figure 2. The root node changes its local variables based on line 1 of the SS-SPST-E algorithm, and all the other nodes change their local variables according to line 2 of the algorithm. For simplicity, here we assume that initially there is no tree and that H_i is infinity, and P_i is NULL for all nodes $i \in V$. In reality, SS-SPST-E can converge to an energy-efficient state from any state (ensuring adaptivity) and the execution of the algorithm can be easily verified from any other initial state. After the first round (Figure 8), the root node changes its local variables. As a result, the $\mathcal{VP}$ sets for nodes 1, 2, 3, 4, and 6 change and contain only node r. Therefore, in the second round (Figure 9), all these node take r as their parent and change their local variables accordingly.

In the third round (Figure 10), nodes 8 and 9 take node 4 as parent as that is the only node in their $\mathcal{VP}$ sets. For node 5, however, it is beneficial to take node 6 as parent instead of 4. This is because $O_5(4) = T_{45} + 5R$, whereas $O_5(6) = T_{65} + 3R$. As both the links $(4, 5)$ and $(6, 5)$ require same power level for transmission (i.e. $T_{45} = T_{65}$), this means that $O_5(4) > O_5(6)$. Therefore node 6 is selected as parent instead of 4. Node 7 has same overhead for selecting 2 or 3 as parent and 2 is selected in this case at random. For node 3, the $\mathcal{VP}$ set also changes from round 1 to round 2. Now, $O_3(6) = T_{36} + 3R < O_3(4) = T_{34} + 5R < O_3(r) = T_{r3} + 5R - T_{r2} - 4R = T_{r3} - T_{r2} + R$. Therefore, $O_3(6)$ has the minimum value and as a result node 3 changes its parent in round 3. After the third round the predicates (line 2 in the SS-SPST-E algorithm) for all the nodes become false and therefore the tree is stabilized. Next, in the pruning phase, the links $(4, 9)$, $(4, 8)$, and $(2, 7)$ are removed (Figure 11) from the multicast tree similar to SS-SPST (Section 3). We have demonstrated here how the local actions at each node stabilize to a valid tree. In the next subsection, we will theoretically prove this property in a more general way.

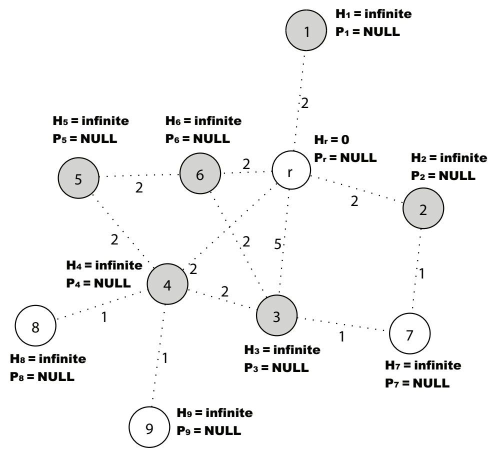


Fig. 8. Node status after round 1. Here $\mathcal{VP}(1) = \mathcal{VP}(2) = \mathcal{VP}(3) = \mathcal{VP}(4) = \mathcal{VP}(6) = \{r\}$, and $\mathcal{N}(1) = \mathcal{N}(2) = \mathcal{N}(3) = \mathcal{N}(4) = \mathcal{N}(5) = \{r\}$.

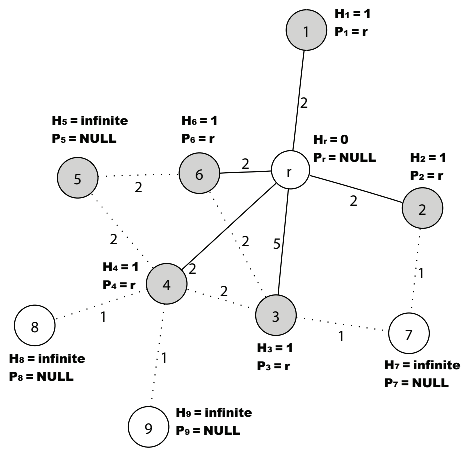


Fig. 9. Node status after round 2. The sets $\mathcal{VP}(1)$ and $\mathcal{VP}(2)$ are same as in Figure 8 (round 1). Here $\mathcal{VP}(3) = \{r, 4, 6\}$, $\mathcal{VP}(4) = \mathcal{VP}(6) = \{r, 3\}$, $\mathcal{VP}(5) = \{4, 6\}$, $\mathcal{VP}(7) = \{2, 3\}$, and $\mathcal{VP}(8) = \mathcal{VP}(9) = \{4\}$.

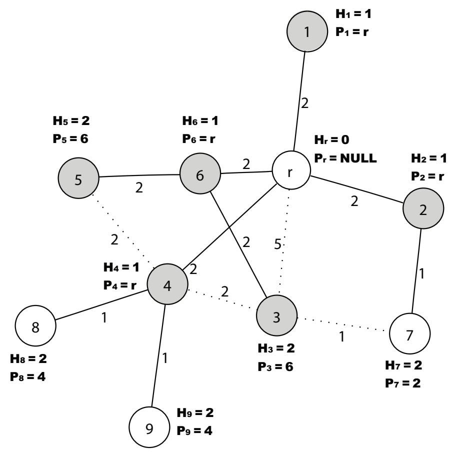


Fig. 10. Node status after round 3. The sets $\mathcal{VP}(1)$, $\mathcal{VP}(5)$, $\mathcal{VP}(7)$, $\mathcal{VP}(8)$, and $\mathcal{VP}(9)$ are same as in Figure 9 (round 2). $\mathcal{VP}(2) = \{r, 7\}$, $\mathcal{VP}(3) = \{r, 4, 6, 7\}$, $\mathcal{VP}(4) = \{r, 3, 5, 8, 9\}$, $\mathcal{VP}(6) = \{r, 3, 5\}$

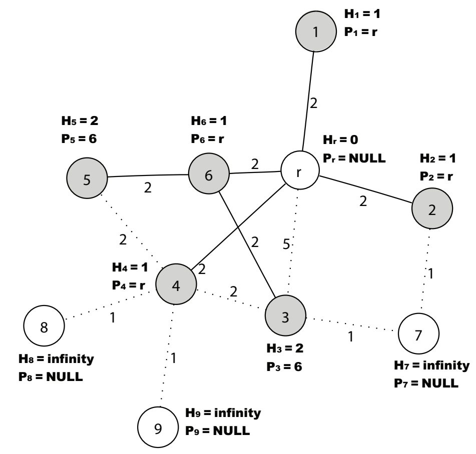


Fig. 11. Node status after round 4. The tree is pruned in a bottom-up manner.

4.2.4. *Global Properties and Protocol Correctness*

The total energy cost of a tree is given by the sum of energy cost of all nodes in the tree. We introduce a constant $FLOOR$, which gives the minimum possible value for the total energy cost of the tree (when best-effort local parent selection is performed) for a given network topology. When all the nodes select their respective parents with the minimum overhead (using the WECM model) the total energy cost of the tree should be equal to the value of $FLOOR$. For the aforementioned example (Figure 10), $FLOOR = T_{r1} + T_{27} + T_{48} + T_{65} + 10R$ (by adding the energy consumption at each node). The $FLOOR$ constant will be used later to show that, after the tree is stabilized the local actions does not incur any fault (thus increasing the total energy consumption over $FLOOR$). Each node in the network, when it is not connected to the tree has a energy cost of $EMax$, which is greater than the maximum possible energy cost of the tree which is nothing but the energy required for the root node to reach all the nodes if we assume only one hop communication. To control the maximum number of hops of any node from the root, we fix the maximum number of hops to be the total number of nodes n as described previously.

Any system S is said to be self-stabilizing with respect to predicate P, defined on global system states, if it satisfies the following two properties:

(i) **Convergence**: Starting from any arbitrary global state, S is guaranteed to reach a global state, satisfying P in a finite number of state transitions.

(ii) **Closure**: Once S reaches a global state satisfying P, it cannot be falsified.

All global states in which the predicate P holds are called *legitimate states*. The predicate P is defined by the Legitimacy Condition (LC) (Section 3.2). The legitimate state for a self-stabilizing algorithm depends on the corresponding Legitimacy Condition (LC). As described in Section 3.2, a self-stabilizing algorithm has to define the LC and prove the correctness of the local actions in terms of the aforementioned properties. The LC for SS-SPST-E is defined as:

Definition 8 *Legitimacy Condition (LC) for SS-SPST-E is given by the following boolean expression*
$\forall i \in V : H_i \leq n \wedge ((i = r \wedge P_i = NULL \wedge H_i = 0 \wedge O_i = 0) \vee (i \neq r \wedge P_i \in \mathcal{N}(i) \wedge H_i = H_{P_i} + 1 \wedge O_i = \min_{j \in \mathcal{VP}(i)} \{O_i(j)\}))$

An informal description of the LC follows. For root the node (i.e. $i = r$), the parent (P_i) has to be $NULL$, and the hop count to the root (H_i) has to be 0 in the legitimate state. Since there is no parent for the root node, the overhead to the parent (O_i) is also 0. For the non-root nodes (i.e. $i \neq r$), the parent (P_i) must be selected from the set $\mathcal{N}(i)$ such that the hop count (H_i) is one greater than the hop count of the parent (i.e. $H_i = H_{P_i} + 1$). Further, the overhead of the selected parent has to be minimum among all the virtual parents (i.e. $O_i = \min_{j \in \mathcal{VP}(i)}\{O_i(j)\}$). Any state which does not satisfy the LC is an illegitimate state. SS-SPST-E algorithm tries to bring an illegitimate state to a legitimate state thereby reducing the total energy consumption. It should be noted here that the guard of at least one node $i \in V$ becomes true when LC is not satisfied. Each node looks at the states of its neighbors and if the node is not locally legitimate (i.e. the guard is true), it triggers local actions and tries to bring the whole system to a legitimate state. The correctness of the SS-SPST-E algorithm involves the aforementioned correctness and closure properties. Further, the validity of the stabilized tree needs to be proved. This is done by proving that the stabilized structure is a acyclic connected graph (i.e. a tree [20]). The following lemmas prove the correctness of SS-SPST-E.

Lemma 9 *Acyclicity: If the legitimacy condition holds, then the P_i values define no cycle in $G(V, E)$.*

Proof: This is proved by contradiction. Assume that there is a cycle defined by the P_i values: $\langle P_1, P_2, \ldots, P_k, P_1 \rangle$. Looking at the values H_i, let $H_1 = l$, then $H_2 = l + 1$ and so on so forth until $H_k = l + k - 1$ and then again $H_i = l + k$ which is impossible. Therefore there cannot be any cycles. □

Lemma 10 *Connectivity: If the legitimacy condition holds, then from any node the P_i values define a route to the root that is connected and consistent with $G(V, E)$.*

Proof: This is done by constructive proof. We arbitrarily select a node s (i.e. "start") and with the node set $\mathcal{S}^{(1)} = \{s\}$. Then according to the acyclic property of the P_i values, P_i will point to a node in $V - \{s\}$, i.e. $V - \mathcal{S}^{(1)}$, and through a valid edge in G, i.e. $(s, P_s) \in E$. If that node is the root r, we stop. Otherwise, we add P_s to $\mathcal{S}$, i.e. $\mathcal{S}^{(2)} = \mathcal{S}^{(1)} \cup \{P_s\}$. We could inductively show that, at any iteration, the set $\mathcal{S}^{(i)} = \{s, P_s, P_{P_s}, \ldots\}$ is finite and that the last-added node will have a parent in $V - \mathcal{S}^{(i)}$ thanks to the acyclic property. The set $\mathcal{S}$ cannot exceed V, at which point there is a path to the root. Additionally, every P_i corresponds to an existing edge in the graph $G(V, E)$, therefore the path is valid. □

Corollary 11 *From Lemmas 9 and 10 we can deduce that if the legitimacy condition holds, then the P_i values define a connected tree in $G(V, E)$.*

Lemma 12 *Convergence: Starting from a given illegitimate state and while there are no further faults, the total energy cost of the graph reduces after every round till all the nodes in the system are stabilized.*

Proof: We prove the above lemma using induction. When the node is not connected to the tree it has a energy cost of $EMax$. After the first round, root node stabilizes and sets its cost variable. Since $EMax > C_r$, the total graph energy cost reduces after first round. Now let us assume the energy cost of the graph after k rounds is TC_k. We now have to prove that if TC_{k+1} is the energy cost of the graph after $k + 1$ rounds then $TC_{k+1} < TC_k$. There are two possibilities,

Case 1: There exists at least one node that is not connected to the tree. By our assumption, the graph is connected. So there will be at least one tree neighbor of the not connected nodes and in the $k + 1^{st}$ round, it will be connected to the tree and its energy cost will reduce from $EMax$ to a lesser value. Hence, $TC_{k+1} < TC_k$.

Case 2: All nodes are connected to the tree. After k rounds, each node will have a parent in the tree and will have a energy cost associated with itself. In the $k + 1^{st}$ round, when a node changes its parent then the overhead associated with that node reduces. The reduction of energy cost of the old parent will be greater than the increase of energy cost of the new parent. Hence, $TC_{k+1} < TC_k$.

Thus the energy cost of the graph will keep reducing till it reaches a minimum value, which in our case is

FLOOR. □

Lemma 13 *Closure: Once the energy cost of the graph equals FLOOR, it stays there until further faults occur.*

Proof: If the total energy cost of the graph decreases after it attains the $FLOOR$ value, then our assumption that $FLOOR$ is the minimum possible energy tree value does not hold true. Hence the total energy cost of the graph cannot reduce further than $FLOOR$ value. The total energy cost of the graph will never increase after a round as shown by Lemma 12. Hence, the total energy cost of the graph will neither decrease nor increase after it attains $FLOOR$. □

Lemma 14 *Count-to-infinity: Any cycle formed will be resolved in a finite number of steps.*

Proof: There is a possibility that a cycle may be created during the tree formation. As soon as a cycle forms the hop count value $H_j(r)$ for each node j involved in the cycle will increase with every round and soon it will exceed the value n and hence the nodes will be removed from the sets $\mathcal{VP}(i)$ and $\mathcal{N}(i)$. Thus, they will not be considered as the parent node by any node. In this way, the cycle will be detected and necessary stabilizing actions will repair it. Since by our assumption all the nodes are connected in the graph, the resultant tree will also remain connected as per Lemma 12. □

4.2.5. *Protocol Evaluation*

We evaluate SS-SPST-E in terms of i) its energy consumption in delivering every single packet to the destination, and ii) the beacon overhead energy. To address the first criteria we compare energy consumption of SS-SPST-E for random topologies with other energy-efficient protocols such as EWMA, and Dist-EWMA [8]. EWMA and Dist-EWMA are shown to have less energy consumption [8] compared to other energy-efficient protocols such as BIP/MIP [40]. The maximum transmission power (C_i^{max}) was set to 10 J for each node. We assigned asymmetric transmission power thresholds for the two arcs connecting the same nodes in opposite directions. We randomly selected $T_{ij} = d^2$ or $T_{ij} = d^2/2$ independent upon T_{ji}, where d (in meters) is the Euclidean distance between i and j [11]. Table 3 gives the energy consumption per node for delivering one data packet to the destinations. EWMA is a centralized protocol and therefore has the minimum energy consumption to deliver data packets. It can be verified that the tree generated by SS-SPST-E leads to 1–8% higher energy consumption than the distributed version of EWMA (Dist-EWMA) protocol for a given network topology. However, unlike Dist-EWMA, SS-SPST-E is designed to adapt to the topological changes in an autonomic manner.

Overhead energy consumption is a function of the beacon interval of the SS-SPST-E protocol (i.e. the term $\sum_{i=0}^{i=n} \frac{1}{\beta} C_i^{max}$ in Equation 1). *Increasing the beacon interval reduces the beacon overhead with the cost of slow response to topological changes (due to high stabilization delay in recovering from these changes).* The following theorem gives the worst case stabilization delay for SS-SPST-E.

Theorem 15 *Starting from a given change in the link state and assuming that there are no further perturbations in the network during the recovery process the multicast tree in SS-SPST-E converges to a stable state with a worst case of O(n) steps with a stabilization delay of $\delta_{stab} = 3\beta n$ where β is the beacon interval and n is the number of nodes in the network.*

Proof: The proof is divided into three parts:

(i) It takes O(n) steps for a fault to propagate before a node realizes that the tree is not stabilized.
A fault, in the worst case, makes a node assign its link metric to its assumed parent to be O_{min} (see Remark 6). This may cause a "chain reaction" of nodes being "deluded" and potentially changing their weight to O_{min} and starting pointing toward the faulty node, with a depth being at most $D-1$, where D is the diameter of the network. At the same time, the faulty node and and its assumed parent will be involved in a count-to-infinity process, and after $n-1$ steps the initial faulty node will realize the count-to-infinity and set its distance to D and its cost to O_{max}.

(ii) It takes O(n) steps for the fault effect to be negated.
The fault realization will take up to $D-1$ steps to propagate to the deluded nodes and make them set their distance and WECM values to the respective maxima. Since the diameter can be as large as n, then this can take up to n steps.

(iii) It takes O(n) steps to stabilize once the fault effect is negated.
We prove this part by induction on the height of the tree (here we assume that the height of the root node is 0). The base case for the induction is a tree with the height of 1 (i.e. $H = 1$). As all the adjacent nodes of the source

Table 3
Average Energy Consumption Per Node to Deliver Single Packet

Protocols	20 Destinations	60 Destinations	100 Destinations
SS-SPST-E	487.36 mJ	543.40 mJ	625.1 mJ
EWMA	310.07 mJ	366.24 mJ	457.50 mJ
Dist-EWMA	480.23 mJ	527.80 mJ	583.9 mJ

would know their adjacency to the source after 1 round.

Now, let us assume that the theorem holds if the height of the routing structure is m where m is any positive integer. Now, if the height is m+1, the first m levels of the routing structure take a m rounds to stabilize according to the induction hypothesis. Now, all the nodes which become one-hop downstream neighbors of a node at level m, have to wait for one more round before getting a beacon. Therefore, a network of height H. The height of the tree can be as long as the diameter of the network, which in turn can be as large as n. Therefore it can take up to n rounds to reach a legitimate state.

So, in overall, it takes O(n) steps to stabilize from the first occurrence of a fault, with a stabilization time of $\delta_{stab} = 3\beta n$. □

4.3. *Optimizing the routing mechanism*

4.3.1. *Optimizing The Beacon Interval*

In the previous section we developed SS-SPST-E protocol for energy efficiency during data transmission. In this section, we focus on minimizing the beacon overhead of SS-SPST-E.

Analytical Model for beacon Overhead (AMO) optimization. As mentioned in Section 4.2.5, stabilization delay of SS-SPST-E increases with the beacon interval. Intuitively, it is required to reduce the beacon interval (increasing the beacon overhead) so that the multicast tree is recovered quickly and the loss of multicast packets during the recovery (stabilization) phase is minimized. On the other hand, in order to minimize the overhead, the beacon interval needs to be as low as possible. We optimize the beacon frequency depending on the following three parameters: i) the rate of change in the link states (exponentially distributed with mean μ as described in Section 3.1), ii) the rate of multicast traffic, and iii) the application reliability requirement (measured by PDR). The last two application dependent parameters determine the need for the multicast tree. First, we provide the application model used for the analysis, followed by a characterization of dependence of stabilization latency δ_{stab} on the aforementioned parameters.

Application Model. We assume the Bulk Poisson traffic arrival model [3] [4, 33] for our analysis, where the message arrival from source to destination (and thereby at each intermediate node) is exponentially distributed with the average λ messages per unit time. Each message contains on average α data packets. Packets inside a message arrive after every constant time τ. This traffic model characterizes many real-time applications (such as voice, video and multimedia) [34] which we believe would be the major MANET targets in many situations such as military applications, disaster rescue, etc. Further, these applications can tolerate packet losses within a threshold [34]. We assume that, on average, a PDR of Γ is required by the application.

Relationship between Stabilization Latency (δ_{stab}), PDR (Γ), traffic rate (λ) and link change rate (μ). Before proceeding any further, we analyze the average probability of packet loss at each link. This analysis is required to quantify the average PDR achieved by the self-stabilizing protocols. If η_p denotes the probability of packet loss over a single link in the route, then probability of successful packet delivery from the source to destination is given by $(1-\eta_p)^D$, D being the average network diameter in number of hops. If Γ is the PDR required by the application then we have the following constraint:

$$(1-\eta_p)^D \geq \Gamma \tag{9}$$

[3] Although in this paper we apply Bulk Poisson traffic model to illustrate our analysis, the analytical methods can be applied to other traffic models.

Note here that η_p is dependent on the rate of link change, the traffic rate and the route reconstruction delay. Further, if η_p is one, i.e. if all the packets are lost, the PDR becomes zero, which is clearly unacceptable for any application. In the following paragraphs, we analyze η_p in terms of μ and λ assuming that η_p is not one.

Packet Loss Analysis. The average rate of packet arrival includes the constant data packet arrival in a message and exponential message arrival. Under the same traffic model, it has been proved [4] that if, on average, there are α packets per message where the arrival of messages is distributed exponentially with mean rate λ and if the packets in a message arrive after every constant time τ, then the average number of packets per unit time is given by,

$$\Lambda = \frac{\lambda\alpha}{1 - \lambda\alpha\tau(1 + \lambda\alpha - \frac{\lambda+1}{\alpha})e^{-\lambda\alpha\tau}} \tag{10}$$

It can be easily verified from the above equation that if α is one then the rate of packet arrival degenerates to λ. Now, we can quantify the loss of packets due to a single link failure as follows:

Theorem 16 *If δ is the worst-case delay in recovering the route in the occurrence of disconnections, then the average probability of packet loss due to each single link failure is given by,*

$$\eta_p = \frac{\lambda\alpha + \Lambda}{\lambda\alpha + \mu}\mu\delta - \left(\frac{1}{\alpha} - \frac{\lambda}{\alpha(\lambda + \mu)} - \frac{\mu}{\lambda\alpha + \mu}\right)(\mu - \Lambda)\delta e^{-(\lambda\alpha+\mu)\tau} \tag{11}$$

The proof of the theorem is provided in Appendix A. Replacing η_p in the Equation 9, we get

$$\delta \leq \frac{1 - \Gamma^{\frac{1}{D}}}{\Omega} \tag{12}$$

where $\Omega = \frac{\lambda\alpha+\Lambda}{\lambda\alpha+\mu}\mu + \left(\frac{1}{\alpha} - \frac{\lambda}{\alpha(\lambda+\mu)} - \frac{\mu}{\lambda\alpha+\mu}\right)(\Lambda - \mu)e^{-(\lambda\alpha+\mu)\tau}$.

Upper Bound on the Stabilization Latency: It should be noted that δ is determined by the worst case time to detect any change in the link status ($k\beta$ as any disconnection can be detected after k beacon intervals), and the worst case stabilization latency. Therefore, $\delta = k\beta + \delta_{stab}$, and replacing this in Equation 12, we get the upper bound on the stabilization latency for self-stabilizing protocols.

$$k\beta + \delta_{stab} = \frac{1 - \Gamma^{\frac{1}{D}}}{\Omega} \tag{13}$$

4.3.2. *Optimum Beacon Interval for SS-SPST-E*

Replacing δ_{stab} (in Equation 13) for SS-SPST-E (from Theorem 15), we get β_{opt} as follows:

$$\beta_{opt} = \frac{1 - \Gamma^{\frac{1}{D}}}{(k + 3n)\,\Omega} \tag{14}$$

It should be noted that for either $\mu \to 0$ or $\lambda \to 0$, β tends to ∞, i.e., the periodic beaconing is unnecessary - in essence the protocol behavior become more and more reactive in nature. For any other values of μ and λ, however, if the stabilization delay of SS-SPST-E protocol can be reduced, we can decrease the denominator of Equation 14, thereby increasing β_{opt}. For this purpose we apply fault-containment [15] over SS-SPST-E in the next subsection.

4.3.3. *Reducing Stabilization Delay*

Applying fault-containment over self-stabilization eradicates the dependence of the stabilization delay on the size of the network in case of limited transient faults in the MANETs. This results in a stabilization delay which linearly varies with β [15, 16] in such cases. Intuitively it can be said that if the tree information of a node is changed it should be sufficient to take corrective measures only in the faulty node instead of several other nodes. Figure 12 shows a tree generated by SS-SPST-E (for simplicity, the transmission ranges among the nodes are not shown in the figure). Here, both C and G are in the range of D initially. When C moves out of the range of node D, it is sufficient to change the parent information of node D from C to G. However, if node E takes action before D (taking I as parent) then the fault propagates down the tree as all the downstream of D would have to change the respective H_i values. This leads to unnecessarily long time and energy to stabilize the system.

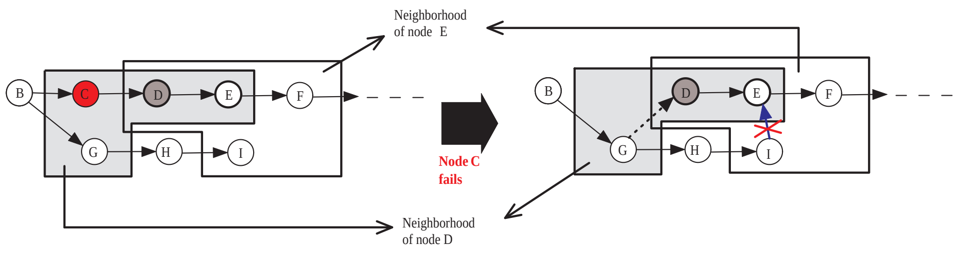


Fig. 12. Fault propagation: when node C fails it is enough to change the parent of D. But if E takes action before D, it is possible that E changes its parent (and level) and propagates this change in level down the tree before D changes its parent and E reverts to D as parent.

In [16] and [15], proper sequence of execution among the nodes is enforced. The authors have developed a fault-containment scheme over self-stabilization to prevent propagation of limited transient faults. Appendix B.3 summarizes the scheme to help understand its use over self-stabilizing protocols. The scheme generalizes the structure of the self-stabilizing protocols so that it can be applied over any such protocols. We apply the fault-containment method proposed in [16] to our SS-SPST-E algorithm (a brief overview on how to apply fault-containment is provided in Appendix B.3). It should be noted here, that the only difference between the original SS-SPST-E and SS-SPST-E*fc* (i.e. SS-SPST-E after the application of fault-containment) is the reduction in the stabilization delay and consequently the protocol overhead. For a given topology, both SS-SPST-E and SS-SPST-E*fc* converge to the same multicast tree. As a result, the energy consumption to deliver a data packet by SS-SPST-E*fc* has the same results as SS-SPST-E (Section 4.2.5).

On the other hand, the SS-SPST-E*fc* algorithm reduces the stabilization delay of SS-SPST-E from $\beta \sum_{i=0}^{i=H} c^i$ to β in case of single transient faults. If f is the fraction of single transient faults in the MANETs, the stabilization delay for SS-SPST-E*fc* can be given by $f\beta + (1 - f)\beta \sum_{i=0}^{i=H} c^i$. As a result, the β_{opt} for SS-SPST-E*fc* is given by,

$$\beta_{opt} = \frac{1 - \Gamma^{\frac{1}{H}}}{(k + f + 3(1 - f)n)\,\Omega} \tag{15}$$

Clearly, this is greater than or equal to β_{opt} for SS-SPST-E (Equation 14) as $f \geq 0$. The fraction f is dependent on the concurrent mobility of the nodes in the network and reduces with the increase in rate of topological changes (μ) [16]. The simulation results in Section 5.4) will show the reduction in energy consumption for SS-SPST-E*fc*.

4.4. *Autonomic and self-managing behavior*

The previous subsections defined a distributed self-stabilizing protocol, along with analytical and algorithmic optimizations, to support energy-efficient multicast routing. This subsection describes how these are used to develop a self-managing multicast suite (Algorithm 1). There are two major aspects in developing the self-managing multicast suite: i) the emergence of an energy-efficient multicast tree from any arbitrary initial state; and ii) autonomic calculation of the optimum beacon interval to minimize overhead. It is proved in the Sections 4.2.4 and 4.2.5 how SS-SPST-E converges (emerges) to an energy-efficient tree from any arbitrary state within a finite time based only on local execution at each node. SS-SPST-E*fc* is an extension of SS-SPST-E and converges to an energy-efficient tree in even less time (Section 4.3.3). However, the optimization of the beacon intervals in both the protocols is a static process (Section 4.3.1) and depends on the link change and the data traffic intensity. In order to make the optimization autonomic, each node first estimates these parameters, and disseminates them to the root as a convergecast using the multicast tree (developed by the SS-SPST-E*fc* protocol). The root performs the AMO optimization and uses the current multicast tree to disseminate the optimum beacon interval back to each of the nodes. The optimization relies on the knowledge of various input parameters to operate; in specific, these parameters are:

- the parameters λ, α, τ, that describe the data traffic pattern. Section 4.4.1 describes the monitoring and estimation mechanism for the data traffic.
- The parameter μ, that describes the overall average intensity of link breakage. Section 4.4.2 describes the monitoring and estimation mechanism for the μ value needed by the AMO. Since the AMO is a centralized computation

process, it is chosen to be calculated by the root node. The same subsection also explains how the root is updated in a distributed manner with the μ from the rest of the network.
– The parameter Γ, that describes the PDR requirement by the application. Our proposed solution is: if there is an API that allows the application to declare the Γ value, the optimization uses it (as the application source and the AMO are collocated at the root); otherwise it is assumed to be 1.

Once the estimation of input parameters for the AMO optimization is performed, it is disseminated to the root using the beacon packets in a bottom-up manner from the leaves to the root using the current multicast tree. The updated beacon interval (β), which is calculated by the AMO optimization at the root is then disseminated to all the nodes. Again, the value of β is announced to the network through the beacon packet. Consequently, the beacon packet contains the following fields:

– i: the node id
– $O_j(i)$: the overhead energy of node i if j selects node i as parent.
– $H_i(r)$: the current hop count to r from node i.
– P_i: the current parent of node i.
– $\beta_{opt}(i)$: the optimum value of beacon interval as estimated at node i.
– μ_i: the cumulative estimate of μ at node i and its branch.
– n_i: the number of nodes in node i's branch.
– f_i: the pruning flag representing the existence of a multicast member in the branch.

4.4.1. *Monitoring and Estimation of the application process at the network layer*

The network layer only observes the arrival of the application packets without any further knowledge of the Bulk Poisson packet generation process. The principal challenge in developing a per node optimization at the network layer is the proper knowledge of the different parameters (τ, α, and λ) of the process. There are two different steps in performing an estimation; these include: i) distinguishing the packet bursts from the set of packets received (to derive τ, and α), and ii) estimating the mean of the Poisson process (λ) of the burst arrival. The first step is achieved through monitoring the inter arrival time of the consecutive data packets. Packets in a single burst have a constant inter-arrival time (τ). A change in the inter-arrival time of the data packets signifies the end of a burst and the beginning of a new burst.The parameter α is the total number of counted packets between the beginning and the end of a burst.

A new Poisson event is detected with the beginning of a new burst. A Poisson event count is maintained to keep track of the number of such events from the start of the packet reception. Given the distinct Poisson events (burst of packets), the mean of the Poisson process (i.e. λ) is estimated using Bayesian sequential estimation [29]. This approach makes use of the prior information on the intensity of the Poisson process to reduce the computational overhead. This mechanism is explained by the pseudocode of the procedure EstimateBulkPoissonMessageParameters in Algorithm 1 (see Appendix C).

4.4.2. *Monitoring and Estimation of the link breakage rate*

The links are being tested for connectivity using the beaconing process at the link layer initiated by the network layer. As discussed in Section 3.1, the beacons are used to identify the nodes to their neighbors. Each node constructs a fresh list of neighbors Adj(i) with the beacons period. Nodes that are removed from Adj(i) are considered to be disconnected. The network layer keeps a counter κ_i of disconnections, which is incremented with every disconnection detected. Every $(H_i{+}n_i)\beta$ time, the localized view of link breakage rate is revised with the following formula:

$$\mu_i^{'} = 0.5\kappa_i + 0.5\mu_i,$$

and the counter κ_i is reset to zero. The choice of the re-calculation period to be $(H_i{+}n_i)\beta$ is that to make more or less all nodes to report a localized view of μ that reflects to the same time window.

4.4.3. *Aggregation of μ and dispersion of β values*

AMO is centrally performed by the root node and requires aggregation of μ values from all the non-root nodes in the multicast tree. We take advantage of the multicast tree itself to *converge-cast* the link breakage rate to the root. With every beacon, each *leaf* node includes its μ_i value to it. Intermediate nodes receive the beacons and add the μ values together with their own μ_i value and report it to their respective parents with the next beacon packet. Finally that information reaches the root. The root adds all the received values to its own local value and *halves* the sum, because every disconnection is observed by both ends. The root performs the AMO and announces the new β with the next beacon.

5. Simulation Results

We perform simulation based analysis to verify the performance of SS-SPST-E and SS-SPST-E*fc* (SS-SPST-E*fc* is an enhancement over SS-SPST-E) protocols. The impact of WECM and fault-containment is evaluated by comparing SS-SPST-E and SS-SPST-E*fc* with SS-SPST, SS-SPST-T, and SS-SPST-F. We further varied the beacon interval to see its impact on the performance of the protocols. The results verify the theoretical foundations of AMO (Section 4.3.1). *We further compared these protocols with reactive multicast protocols such as MAODV and ODMRP to compare the energy efficiency, adaptivity, and scalability in diverse scenarios.*

5.1. *Performance Metrics*

We measure the performance of the protocols based on the following metrics.

(i) **Packet delivery ratio (PDR)**: The ratio of the number of data packets actually delivered to the receivers and the number of data packets supposed to be received by the receivers. This metric shows the effectiveness of a protocol to deliver data packets to multiple recipients.

(ii) **Energy consumed per packet delivered**: We measure the energy consumed per packet delivered by dividing the total energy consumed by all the nodes in the network by the total number of data packets delivered.

(iii) **PDR achieved per unit of energy expended to deliver a packet**: This metric is measured by dividing the first metric (i.e. the PDR) with the second metric (i.e. the energy consumed per packet delivered). This metric captures the trade-off between the achieved PDR and the energy consumption. The higher the PDR achieved per unit of energy expended by any protocol, the better is its performance.

5.2. *Simulation Model*

Given the performance metrics to evaluate, this section explains the models used for the simulation followed by the experiments performed in Section 5.4.

5.2.1. *Simulator, node deployment, radio propagation, and MAC protocol*

Network simulator (ns-2) is used for the simulation. A 750 m $\times$ 750 m simulation area is modeled with 50 nodes deployed at random positions. Two-ray ground reflection model is used as the radio propagation model with the maximum range of transmission set to 250m for each node. This model gives an accurate prediction of propagation range at a long distance. Each node is assumed to be equipped with an omni-directional antenna which can dynamically vary the transmission power. 802.11 MAC protocol is used with the ns default bandwidth setting of 2 Mbps.

5.2.2. *Transport model*

One node was designated as the multicast source. UDP was used as the transport protocol because UDP does not employ packet retransmissions for end-to-end reliability at the transport layer. The calculation of the PDR in the simulation study is only based on the reliability at the network layer where the protocols — ODMRP, AODV, SS-SPST, SS-SPST-T, SS-SPST-F, SS-SPST-E, and SS-SPST-E*fc* — reside.

5.3. *Experiments performed*

Experiments were performed by varying – i) the node velocity, ii) beacon interval, iii) the multicast group size, iv) application PDR requirements, and v) application data traffic model. Sections 5.3.1, 5.3.2, 5.3.3, 5.3.4 and 5.3.5 provide the rationale behind these experiments, respectively. For each experiment with a specific set of parameters, ten different random scenario files were generated (in *tcl*). Each scenario file specified the initial co-ordinate locations of the 50 nodes and the time instances during which they start moving to reach a end location in the simulated 750×750 simulation area. For the ten different random scenarios, the initial node positions, the time instances of node movements, and the end positions were all different. Experiments were performed for the ten scenarios separately. Each simulation ran for 1800 seconds of simulation time. The mean and the standard deviation of the performance metrics (resulting from the ten different scenarios) were calculated. The results are shown by plotting the mean of the performance metrics with respect to the variations of certain parameters (as described in the Sections 5.3.1, 5.3.2, and 5.3.3). The standard deviation for each point in the plot is shown as the average error of the plotted value.

5.3.1. *Node Velocity*

Node velocity was varied to study the performance of the protocol under different network dynamics. We used the random way point mobility model with variation of the maximum velocity of nodes from 1 m/s (around 2.25 miles per hour) to 20 m/s (45 miles per hour). This range of node velocity simulates movement from normal human walking to city driving, which we believe is a reasonable observable range for MANET applications. As pointed out by Noble et al. [43] one has to guard against the velocity-decay problem of the random way point model. The use of the random-way point model in our simulation conforms to the fix suggested by Noble et al. Specifically, the settings of simulation parameters ensures that the nodes use non-zero minimum velocity.

5.3.2. *Beacon Interval*

An important parameter in evaluating SS-SPST, SS-SPST-E, and SS-SPST-E*fc* is the beacon interval. To analyze the impact of the beacon interval on the protocol performance we varied the beacon interval from 1 to 4 seconds. Although AMO optimization determines the beacon interval for SS-SPST-E and SS-SPST-E*fc* protocols, the variation in beacon interval without the AMO optimization can verify the requirement of fault-containment to allow higher beacon interval (thus saving energy) as argued in Section 4.3.3.

5.3.3. *Multicast Group Size*

The number of multicast group nodes was varied from 10 to 50. A multicast group size of 50 leads to broadcast of the packets. This set of experiments is used to compare the protocols' scalability with respect to the number of receivers.

5.3.4. *Application PDR requirements*

The end-to-end reliability requirement by the application is specified as the end-to-end PDR requirement (Γ) by the application. The PDR requirement (Γ) can be communicated at the network layer: (i) through piggybacking on the data packets, thus requiring modification of the network layer packet, (ii) through the control packets, thus requiring modification of control packets, or (iii) through an adjunct control protocol that disseminates the PDR requirement information. In this work, we choose the first option. Before the first packet transmission, the initially assumed PDR is one. The forwarder nodes locally store the PDR requirement before forwarding the data packets.The PDR requirement is varied from 0.5 to 0.9 in the experiments performed to analyze the effect of the PDR requirement over the total energy consumption in SS-SPST-E*fc*.

5.3.5. *Application Data Traffic Model*

Two different traffic models were used for the experiments. The Constant Bit Rate (CBR) traffic model is used to—i) compare the different cost metrics (presented in the Sections 3.4 and 4.1) in Section 5.4.1, ii) validate the effect of the beacon interval on the end-to-end PDR and energy consumption in Section 5.4.2, and iii) compare the performance of SS-SPST-E*fc* with MAODV and ODMRP. Bulk Poisson traffic model is further used to show the conformance to the end-to-end PDR requirements in SS-SPST-E*fc*; thus verifying the analysis Section 4.3.1. Both CBR and Bulk Poisson data packets originate from the selected source at the rate of 64Kbps.

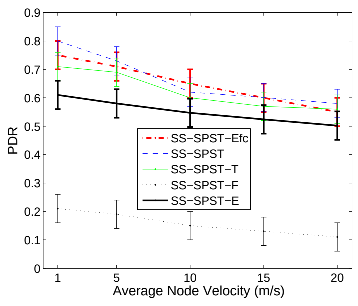


Fig. 13. Packet Delivery Ratio vs. Velocity. The error bars show the standard deviations.

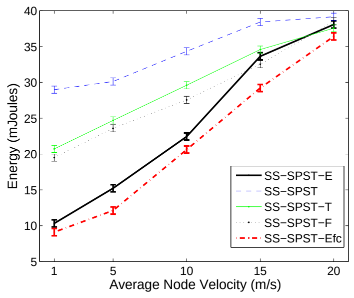


Fig. 14. Energy Consumption vs. Velocity. The error bars show the standard deviations.

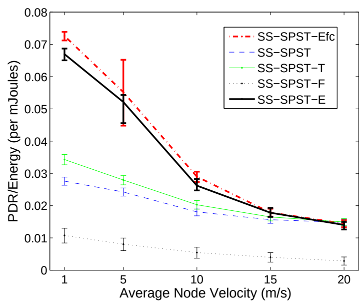


Fig. 15. PDR per unit of energy consumption vs. Velocity. The error bars show the standard deviations.

5.4. *Simulation Results*

This sections presents the simulation results. First, SS-SPST-E and SS-SPST-E*fc* are compared with the SS-SPST, SS-SPST-T, and SS-SPST-F protocols in Section 5.4.1 to evaluate the effectiveness of the WECM model. Second, as described in Section 5.3.2, the impact of varying beacon intervals over SS-SPST-E and SS-SPST-E*fc* are compared in section 5.4.2. For this set of results we fixed the maximum mobility speed to be 5 m/s. Lastly, in Section 5.4.3, SS-SPST, SS-SPST-E, and SS-SPST-E*fc* protocols are compared with MAODV and ODMRP protocols.

5.4.1. *Comparison of different cost metrics*

1. PDR: Figure 13 shows that, with increase in the node velocity, the PDR drops for all the protocols. This is because with increase in node velocity the topological changes occur at a higher rate than the stabilization delay and hence the PDR for all the protocols decreases.

The WECM model forces the SS-SPST-E protocol to choose an energy-efficient path available. SS-SPST maintains the shortest path tree (in terms of number of hops from the root) unlike SS-SPST-E. Thus, from any node to the root (source) node, SS-SPST-E may lead to paths of same length as SS-SPST or higher length than SS-SPST. However, the increase in number of hops incurs higher accumulated probability of packet losses at each link. This reason contributes to reduction in PDR for SS-SPST-E protocol when compared to that of SS-SPST. SS-SPST-E*fc* varies similarly to SS-SPST-E with varying mobility. However, it has higher PDR since the multicast tree gets recovered faster in SS-SPST-E*fc* to allow for more data packets to be delivered. The reason SS-SPST-F has a very low PDR is because of its dynamic nature which causes instability. As the protocols perform power control for energy efficiency, from the graph we see that PDR value of SS-SPST-T is slightly better than SS-SPST-E because changes to the tree structure cause changes to the existing cost and hence the protocol needs more time to stabilize.

2. Energy Consumption: As we see from Figure 14, the energy consumed per node in the network increases as the velocity increases. As the velocity increases, the number of faults that occur in the system increases. This causes more beacon packets to stabilize the system after which a data packet can be delivered to the nodes. So more control packet overhead causes increase in energy consumption per packet.

Further from the graph, we see that SS-SPST has the highest energy consumption and SS-SPST-E has the lowest. The other two protocols SS-SPST-F and SS-SPST-T have lesser energy consumption than SS-SPST but they do not give the most energy-efficient solution to build the multicast tree. Thus our simulation results coincide with our initial analysis. When the mobility speed is relatively slow, the rate at which the fault occurs because of mobility is also slow and hence the SS-SPST-E protocol transmits most of its packets through energy-efficient paths. As the mobility speed increases, the fault rate increases and hence energy saving of SS-SPST-E compared to that of SS-SPST decreases. At high mobility speeds both the protocols consume almost the same amount of energy per data packet delivered. As expected both SS-SPST-E and SS-SPST-E*fc* have similar trends in terms of energy consumption. However, SS-SPST-E*fc* can reduce the overhead energy consumption by increasing the beacon intervals (Section 4.3.3). As a result, the overall energy consumption is less in SS-SPST-E*fc*.

3. PDR achieved per unit of average energy expended to deliver a packet: Figure 15 shows the PDR achieved

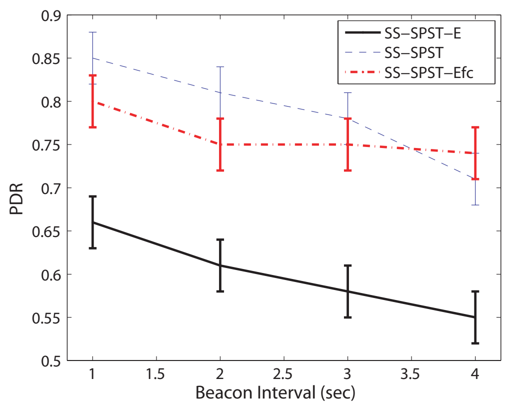


Fig. 16. PDR vs. Beacon Interval. The error bars show the standard deviations.

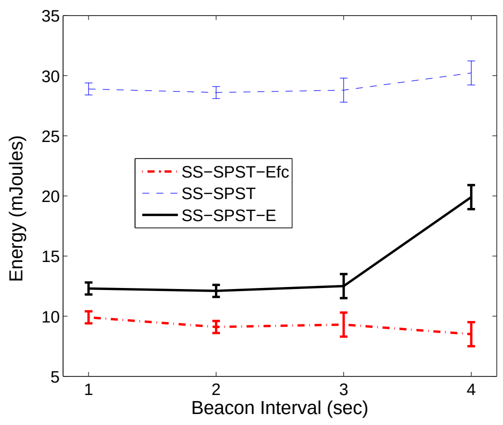


Fig. 17. Energy Consumption vs. Beacon Interval. The error bars show the standard deviations.

per mJoules of energy expended with respect to node velocity. With the increase in node velocity the PDR achieved per mJoules of energy expended deteriorates for all the self-stabilizing protocols because of the higher route instability leading to lower PDR (Figure 13) and higher energy consumption (Figure 14). As expected, SS-SPST-E*fc* outperforms all the other self-stabilizing protocols by achieving the best balance between energy-reliability trade-off. SS-SPST-E achieves lower trade-off than SS-SPST-E*fc* due to the high stabilization delay (Section 4.3.3). SS-SPST-T and SS-SPST-F leads to higher energy consumption and lower PDR, respectively, thus achieving lower PDR per mJoules of energy expended than the SS-SPST-E and SS-SPST-E*fc* protocols.

From the above discussion we see that SS-SPST-E is the most energy-efficient among the other protocols. Though incorporation of WECM causes a slight drop in the PDR, the amount of energy saved per packet is higher than SS-SPST, SS-SPST-T, and SS-SPST-F protocols. The PDR can be further improved in SS-SPST-E*fc* with not much compromise in the energy consumption.

5.4.2. *Beacon Interval*

1. PDR: Figure 16 shows the variation of PDR with the beacon interval. As the beacon interval increases the PDR value drops for all the protocols. This drop is due to the fact that time to realize that a fault has occurred in the system increases with the beacon interval. With low beacon interval the faults are identified and corrected quicker and hence we see a higher PDR. Note however, SS-SPST-E*fc* does not allow the drop of PDR with higher values of beacon interval as it allows fast recovery leading to higher tolerance to the increase in the beacon interval.

2. Energy Consumption: Figure 17 shows the variation of energy consumption with the beacon interval. Energy consumption per packet delivered value is altered by total energy consumption and the number of packets delivered. As the beacon interval increases, one expects the energy consumption to decrease as the beacon overhead sent will decrease. From the graph we see that the energy consumed per packet delivered value decreases for a while and then increases. This increase is attributed to the reduction in the number of packets delivered with increase in beacon interval as explained by Figure 16. At higher velocities the rate of decrease in PDR is more than the rate of increase in energy saving thus energy expended per packet increases beyond certain velocity. However, as SS-SPST-E*fc* does not allow the PDR to drop at high beacon intervals, it means that it reduces the energy consumption due to less beacon transmission.

In general, use of short beacon intervals increases the number of control messages and use of long beacon intervals decreases the PDR. However, to achieve a minimum allowable PDR (0.5 in the results in the following sections), the beacon interval is increased based on the AMO model described in Section 4.3.1. As expected, SS-SPST-E*fc* achieves higher energy efficiency with high beacon interval.

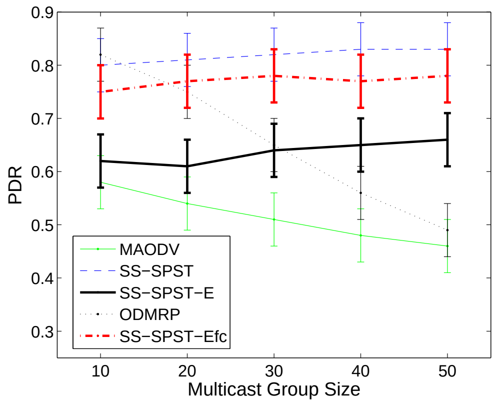


Fig. 18. PDR vs. Multicast Group Size. The error bars show the standard deviations.

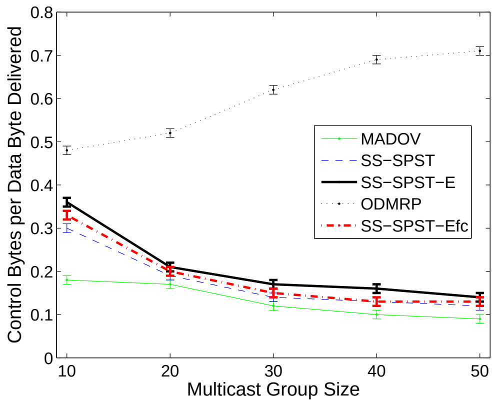


Fig. 19. Control Byte Overhead vs. Multicast Group Size. The error bars show the standard deviations.

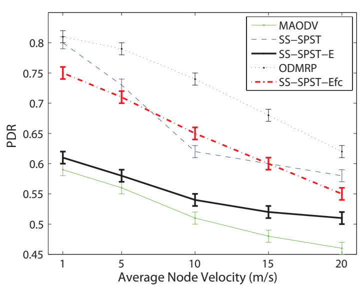


Fig. 20. PDR vs. Velocity (CBR traffic). The error bars show the standard deviations.

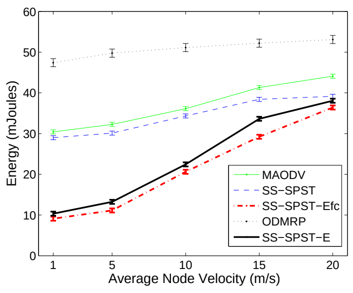


Fig. 21. Energy Consumption per Packet Delivered vs. Velocity (CBR traffic). The error bars show the standard deviations.

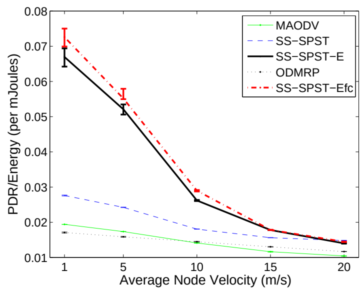


Fig. 22. PDR per unit of energy consumption vs. Velocity (CBR traffic). The error bars show the standard deviations.

5.4.3. *Comparison with other multicast routing protocols*

In this section we compare SS-SPST, SS-SPST-E, and SS-SPST-E*fc* protocols with other multicast routing protocols like MAODV and ODMRP.

1. PDR: From the graph (Figure 18) we see that self-stabilizing protocols are group-scalable i.e., when the multicast group nodes are more, the self-stabilizing protocols perform as well as they do when the multicast group nodes are less. ODMRP outperforms self-stabilizing protocols when the number of multicast nodes is less. It does not scale well with the increase in the multicast group nodes. As we see from the graph the PDR drops drastically as the number of multicast nodes increases. This decrease is attributed to the increase in overhead caused by redundant paths.

Of all the four protocols, PDR of MAODV is the least. Though PDR is not affected too much by the increase in number of multicast nodes, it is less when compared to other protocols. The graph (Figure 19) depicts the cost involved in multicasting using different protocols. This figure represents the control byte overhead when the data is sent continuously. Since ODMRP is a mesh-based protocol, it uses more control bytes for building the multicast tree. ODMRP constructs more than one route from source to destination and hence uses many control messages. As the number of group members increase, ODMRP behaves similar to a *flooding* algorithm and hence the control byte overhead increases with increase in the number of group members.

We further varied the mobility speed of the nodes in the network from 1 m/s to 20 m/s while fixing the multicast group size at 20. Mobility against PDR graph (Figure 20) shows the performance of the multicasting protocols in presence of mobility. We see that PDR in all the protocols drops as the mobility of the nodes increases. By providing alternate paths, ODMRP performs better than other protocols even under high mobility. MAODV, which is a tree-based protocol performs in a similar fashion to SS-SPST. As the mobility speed increases the PDR decreases.

2. Energy Consumption per packet delivered: By varying the mobility speed we captured the energy spent to successfully deliver a packet to the receiver in all the protocols. Our aim was to show how energy-efficient are SS-

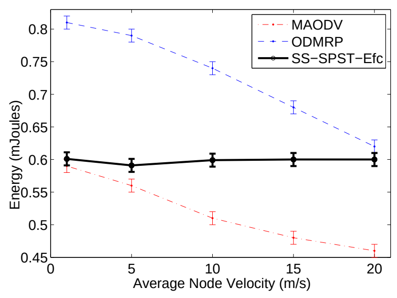


Fig. 23. PDR vs. Velocity (Bulk Poisson traffic). The required PDR is 0.6. The error bars show the standard deviations. The PDR is achieved with at least 85% accuracy.

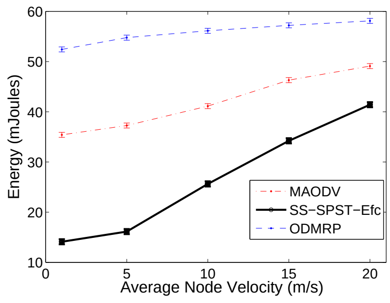


Fig. 24. Energy Consumption per Packet Delivered vs. Velocity (Bulk Poisson traffic). The required PDR is 0.6. The error bars show the standard deviations.

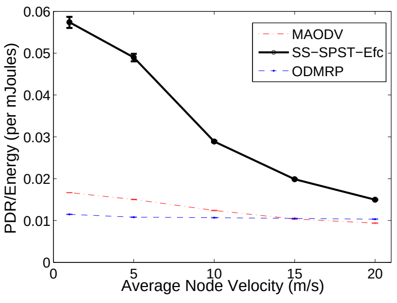


Fig. 25. PDR per unit of energy consumption vs. Velocity (Bulk Poisson traffic). The required PDR is 0.6. The error bars show the standard deviations.

SPST-E and SS-SPST-E*fc* when compared to other protocols. From Figure 21, we see that ODMRP has the highest energy consumption per packet and SS-SPST-E*fc* has the least. Energy is expended on transmission of control messages to build the multicast structure and transport data messages along the route. ODMRP constructs more than one path to deliver the data message to the receiver and hence consumes maximum energy among all the other protocols. MAODV and SS-SPST construct tree structure to deliver multicast messages. Although SS-SPST has more control overhead than MAODV, the PDR of SS-SPST is also more. Hence, SS-SPST spends less energy per packet delivered than MAODV.

3. PDR achieved per unit of average energy expended to deliver a packet: Figure 22 shows the variation in PDR achieved per mJoules of expended for all the protocols with respect to the variation in node velocity. ODMRP leads to the lowest PDR per energy expended even though the end-to-end PDR is much higher (Figure 20). This is due to to the high energy expended by ODMRP (Figure 21) to achieve the high PDR. In the other extreme, MAODV achieves low PDR (Figure 20) by expending low energy (Figure 21). MAODV achieves higher PDR achieved per unit of average energy expended over ODMRP for low node velocity. For high node velocity, however, ODMRP achieves higher PDR achieved per unit of average energy expended–principally because of the duplicate routes maintained by ODMRP. SS-SPST-E and SS-SPST-E*fc* intend to achieve the optimal balance for the energy-reliability tradeoff achieving higher PDR per unit of energy expended over both MAODV and ODMRP. To be specific, the balance is achieved by at least 18% and 23% energy reduction in SS-SPST-E and SS-SPST-E*fc*, respectively, over the ODMRP protocol while increasing the PDR by at least 5% and 25%, respectively, over the MAODV protocol. SS-SPST-E*fc* achieves higher PDR per unit of energy expended over SS-SPST-E by incorporating lower stabilization time. Holistically, SS-SPST-E and SS-SPST-E*fc* achieve at least 24% and 27% higher PDR per mJoules of energy expended, respectively, over both MAODV and ODMRP protocols.

5.4.4. *Conformance of the end-to-end PDR requirements in SS-SPST-Efc*

In the previous results, CBR traffic traffic was used as explained in Section 5.3.5. This section verifies the conformance to the PDR requirement by the SS-SPST-E*fc* protocol when Bulk Poisson traffic model is used (thus verifying the analysis in Section 4.3.1). Figure 23 shows that SS-SPST-E*fc* achieves the required PDR with at least 85% accuracy. ODMRP and MAODV however overshoots and undershoots the required PDR, respectively. This leads to higher energy consumption in ODMRP (Figure 24), while MAODV consumes lower energy than SS-SPST-E*fc* (Figure 24). The PDR achieved per unit of energy consumption is however higher in SS-SPST-E*fc* than both MAODV and ODMRP protocols (Figure 25) reflecting the optimum balance between the end-to-end PDR and energy consumption. Figure 26 shows the achieved PDR by the SS-SPST-E*fc* protocol for varying PDR requirements and for different rate of node mobility. The accuracy to which the required PDR is achieved varies from 80% to 100%. The energy consumption however increases to achieve higher PDR requirements (Figure 27) as predicted by the analysis in Section 4.3.1.

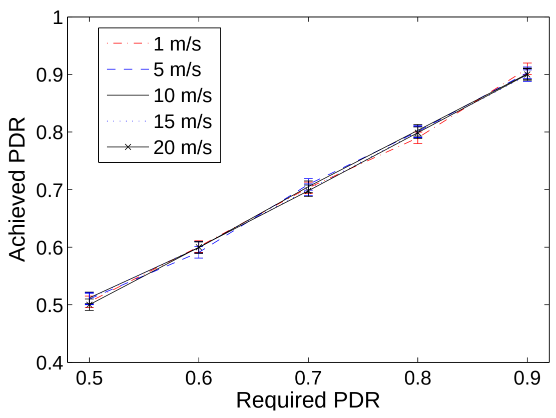


Fig. 26. Achieved PDR for SS-SPST-E*fc* vs. the required PDR. SS-SPST-E*fc* achieves the required PDR for different node mobility. The error bars show the standard deviations.

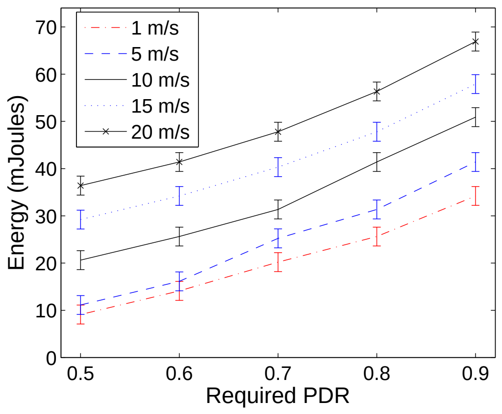


Fig. 27. Energy expended by SS-SPST-E*fc* to achieve the different PDR required. The energy consumption increases for both increase in the PDR requirement and the node mobility. The error bars show the standard deviations.

6. Conclusions

In this paper, we focused on developing a self-managing, self-healing, energy-aware multicast tree for MANETs which tolerates topological changes due to network dynamics. We used the self-stabilization paradigm from the distributed computing domain for this purpose. Self-stabilization enables local autonomous actions in the distributed nodes for any topological changes. Using self-stabilization SS-SPST-E protocol was developed, which extended the SS-SPST protocol to construct and maintain an energy-efficient multicast tree while ensuring self-healing to topological changes. Secondly, to ensure optimized overhead due to periodic beacon messages in the SS-SPST-E, an analytical model named AMO was developed, which determined the optimum beacon interval, β_{opt}, for SS-SPST-E. β_{opt} was further increased by reducing the stabilization delay of SS-SPST-E. We incorporated fault-containment over SS-SPST-E to produce SS-SPST-E*fc*, which reduces the stabilization latency of SS-SPST-E. SS-SPST, SS-SPST-E, and SS-SPST-E*fc* were compared with other traditional non-self-stabilizing protocols such as MAODV and ODMRP based on simulation study. A general conclusion is that, self-stabilizing protocols are group-scalable and have a better PDR even when the number of multicast group nodes is large. Further, results show at least 18% and 23% energy reduction in SS-SPST-E and SS-SPST-E*fc*, respectively, over the ODMRP protocol while increasing the reliability in packet delivery by at least 5% and 25%, respectively, over the MAODV protocol. The balance between the energy-reliability trade-off, measured in terms of the PDR per millijoules of energy expended, is at least 24% and 27% higher in the SS-SPST-E and SS-SPST-E*fc* protocols, respectively, when compared to both the MAODV and ODMRP protocols. Additionally, SS-SPST-E*fc* achieves the required PDR within 80% to 100% of accuracy.

Such increased reliability in the packet delivery ratio is a direct consequence of the self-healing nature of SS-SPST-E and SS-SPST-E*fc*, which enables automatic detection, diagnoses, and repairs of the changes in the network in a localized manner. Such behavior is particularly important for large and complex networks such as MANETs for disaster rescue operations. However, previous approaches to develop self-healing network protocols based on the self-stabilization technique have not been particularly useful due to their inherent problem of high energy consumption. Consideration of energy efficiency in SS-SPST-E and SS-SPST-E*fc* in this paper makes these protocols appropriate for MANETs while maintaining the self-healing behavior.

Acknowledgments

We are thankful to Ganesh Sridharan for his technical contribution, the anonymous reviewers for helpful suggestions for improving the quality of this paper, and NSF for providing support (grant numbers ANI-0196156 and CNS-0834797).

References

[1] ANA: Autonomic network architecture. `http://www.ana-project.org/`.

[2] The IEEE inc. part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specification.

[3] Internet engineering task force (IETF) mobile ad hoc networks (MANET) working group charter.

[4] I. D. Aron and S. K. S. Gupta. On the scalability of on-demand routing protocols for mobile ad hoc networks: An analytical study. *Journal of Interconnection Networks*, 2(1):5–29, March 2001.

[5] A. Arora and H. Zhang. LSRP: local stabilization in shortest path routing. *IEEE/ACM Transactions on Networking*, 14(3):520–531, June 2006.

[6] P. Basu and J. Redi. Effect of overhearing transmissions on energy-efficiency in dense sensor networks. In *Proc. ISPN'04*, pages 196–204, Apr. 2004.

[7] B. Bellur and R. G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *IEEE INFOCOM*, pages 178–186, Mar. 1999.

[8] M. Cagalj, J. P. Hubaux, and C. Enz. Minimum-energy broadcast in all-wireless networks: NP-Completeness and distribution issues. In *Proceedings of ACM MobiCom 2002*, pages 172–182, Atlanta, Georgia, Sept. 2002.

[9] J.-H. Chang and L.Tassiulas. Energy conserving routing in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM 2000*, pages 22–31, 2000.

[10] A. E. F. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. On the complexity of computing minimum energy consumption broadcast subgraphs. In *Proc. of 18th Annual Theoretical Aspects of Comp. Sc. (STACS)*, volume 2010, pages 121–131. Springer-Verlag, 2001.

[11] G. Deng and S. K. S. Gupta. On maximizing network lifetime of broadcast in wanets under an overhearing cost model. In *LNCS 4308/2006 (Proceedings ICDCN'06)*, pages 215–226, Dec. 2006.

[12] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, pages 643–644, Nov. 1974.

[13] P. Floréen, P. Kaski, J. Kohonen, and P. Orponen. Lifetime maximization for multicasting in energy-constrained wireless networks. *IEEE Journal on Selected Areas in Communications*, 23(1):117–126, Jan. 2005.

[14] M. Gerla, S.-J. Lee, and C.-C. Chang. On-demand multicast routing protocol (ODMRP) for ad hoc networks. In *IEEE Wireless Communications and Networking Conference 1999*, Los Angeles, California, Sept. 1999.

[15] S. Ghosh, A. Gupta, T. Herman, and S. V. Pemmaraju. Fault-containing self-stabilizing algorithms. In *15th Annual ACM Symposium on Principles of Distributed Computing*, pages 45–54, 1996.

[16] S. Ghosh, A. Gupta, and S. V. Pemmaraju. Fault-containing network protocols. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 431–437, San Jose, California, Apr. 1997.

[17] S. K. S. Gupta, A. Bouabdallah, and P. K. Srimani. Self-stabilizing protocol for shortest path tree for multi-cast routing in mobile networks (research note). In *Lecture Notes in Computer Science: Proceedings of the Euro-Par 2000*, volume 1900, pages 600–604, Munich, Germany, May 2000.

[18] S. K. S. Gupta and P. K. Srimani. Cored-based tree with forwarding regions (CBT-FR): A protocol for reliable multicasting in mobile ad hoc networks. *Journal on Parallel and Distributed Computing*, 61(9):1249–1277, Sept. 2001.

[19] S. K. S. Gupta and P. K. Srimani. Self-stabilizing multicast protocols for ad hoc networks. *Journal of Parallel and Distributed Computing*, 63(1):87–96, 2003.

[20] F. Harary. *Graph Theory*. Westview Press, Jan. 2001.

[21] C. E. Jones, K. M. Sivalingam, P. Agrawal, , and J. C. Chen. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7(4):343–358, 2001.

[22] I. Kang and R. Poovendran. Maximizing network lifetime of broadcasting over wireless stationary ad hoc networks. *ACM/Kluwer Mobile Networks; Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks*, 10(6):879–896, Dec. 2005.

[23] P. Karn. MACA – a new channel access method for packet radio. In *Proc. of the 9th ARRL Computer Networking Conference*, Canada, 1990.

[24] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, pages 41–50, Jan. 2003.

[25] F. Li and I. Nikolaidis. On minimum-energy broadcasting in all-wireless networks. In *Proc. of the 26th Annual IEEE Conference on Local Computer Networks (LCN 2001)*, pages 193–202, Tampa, Florida, Nov. 2001.

[26] A. Misra and S. Banerjee. Minimum energy paths for reliable communication in multi-hop wireless networks. Technical Report CS-TR 4315, Department of Computer Science, University of Maryland, College Park, Dec. 2001.

[27] T. Mukherjee, S. K. S. Gupta, and G. Varsamopoulos. Analytical model for optimizing periodic route maintenance in proactive routing for MANETs. In *ACM International Modeling of Wireless and Mobile Networks Symposium (MSWiM)*, pages 201–208, Chania, Greece, Oct. 2007.

[28] T. Mukherjee, G. Sridharan, and S. K. S. Gupta. Energy-aware self stabilization in mobile ad hoc networks : A multicasting case study. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–10, Long Beach, CA, Mar. 2007.

[29] B. Novic. Bayes sequential estimation of a Poisson rate: A discrete time approach. *The Annals of Statistics*, 8(4):840–844, July 1980.

[30] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[31] S. Ramanathan. Multicast tree generation in networks with asymmetric links. *IEEE/ACM Transaction on Networking*, 4(4):558–568, Nov. 1996.

[32] E. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM/IEEE MobiCom'99*, pages 207–218, Aug. 1999.

[33] P. Samar and S. B. Wicker. Link dynamics and protocol design in a multihop mobile environment. *IEEE Transactions on Mobile Computing*, 5(9):1156–1172, 2006.

[34] S. Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176 – 1188, Sept. 1995.

[35] S. Singh and C. S. Raghavendra. Pamas-power aware multi-access protocol with signaling for ad hoc networks. *ACM Communications Review*, July 1998.

[36] K. M. Sivalingam, J.-C. Chen, P. Agrawal, and M. B. Srivastava. Design and analysis of low-power access protocols for wireless and mobile ATM networks. *Wireless Networks*, 6(1):73–87, 2000.

[37] S. Vasudevan, C. Zhang, D. Goeckel, and D. Towsley. Optimal power allocation in wireless networks with transmitter-receiver power tradeoff. In *Proceedings of INFOCOM'06*, pages 1–11, Apr. 2006.

[38] B. Wang and S. K. S. Gupta. G-REMiT: An algorithm for building energy efficient multicast trees in wireless ad hoc networks. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA-03)*, pages 265–272, Cambridge, Massachusetts, Apr. 2003.

[39] B. Wang and S. K. S. Gupta. S-REMiT: An algorithm for enhancing energy-efficiency of multicast trees in wireless ad hoc networks. In *Proceedings of the IEEE 2003 Global Communication Conference (GLOBECOM'03)*, volume 6, pages 3519–3524, San Francisco, CA, Dec. 2003.

[40] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the IEEE INFOCOM 2000*, pages 585–594, 2000.

[41] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Algorithms for energy-efficient multicasting in static ad-hoc wireless networks. *Mobile Networks and Applications*, 6:251–263, 2001.

[42] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. Resource management in energy-limited, bandwidth-limited, transceiver-limited wireless networks for session-based multicasting. *Computer Networks*, 39(2):113–131, June 2002.

[43] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedigns of the IEEE INFOCOM'03*, pages 1312–1321, 2003.

[44] Q. Zhao and L. Tong. Energy efficiency of large-scale wireless networks: Proactive versus reactive networking. *IEEE Journal on Selected Areas in Communications.*, 23(5):1100–1112, 2005.

[45] Q. Zhao, L. Tong, and D. Counsil. Energy-aware adaptive routing for large-scale ad hoc networks: Protocol and performance analysis. *IEEE Transactions on Mobile Computing*, 6(9):1048–1059, Sept. 2007.

[46] J. Zhu, C. Qiao, and X. Wang. A comprehensive minimum energy routing scheme for wireless ad hoc networks. In *IEEE INFOCOM*, pages 1437–1445, 2004.

Appendix A. Proof of Theorem 16

There are two possible cases. When the link change rate is higher than the packet arrival rate the probability of packet loss in each single link is η_1 and when the packet arrival is higher the probability is η_2.

Case I: One or more packets are generated before a link breakage. δ is the worst-case delay to discover the route path in the occurrence of link changes. During the time of route recovery, the transmitted packets do not reach the destination. Thus, the number of packets that does not get delivered during this period is $\delta\Lambda$ as Λ is the rate of packet arrival. Moreover, between two consecutive route disconnections there are Λ/μ number of packets from the source. Let us assume that the total number of packets transmitted is t'. There is a total of $\mu t'/\Lambda$ number of route disconnections during the period in which t' packets are transmitted. So, the total number of packets lost is $(\mu t'/\Lambda)(\delta\Lambda)$. The probability of packet loss is thus given by

$$\eta_1 = \frac{\frac{\mu t'}{\Lambda}(\delta\Lambda)}{t'} = \mu\delta \tag{A.1}$$

Case II: One or more link breakage happen before a packet arrival Similar argument as above would give the probability of packet loss as,

$$\eta_2 = \frac{t'\Lambda\delta}{t'} = \Lambda\delta \tag{A.2}$$

Now, the probability of Case II, under same set of assumptions, have been calculated in [4], and is given by the following equation.

$$P_m = \left[\frac{1}{\alpha} - \frac{\lambda}{\alpha(\lambda+\mu)} - \frac{\mu}{\lambda\alpha+\mu}\right] e^{-(\lambda\alpha+\mu)\tau} + \frac{\mu}{\lambda\alpha+\mu} \tag{A.3}$$

The probability of Case I is simply $P_t = 1 - P_m$. Now the average probability of packet loss can be given as follows:

$$\eta_p = P_t\eta_1 + P_m\eta_2 \tag{A.4}$$

Replacing P_m, P_t, η_1 and η_2 we get the result.

Appendix B. Fault-containment over self-stabilization

B.1. *Execution Model*

At each node, the multicast protocol consists of a finite set of variables and actions. The variables include the current estimation of cost of the node and its neighbors based on the aforementioned cost model; the actions are performed based on these information to construct energy-aware multicast structure. Each action further consists of two parts: guard, and statement, and has the following form:

$\langle guard \rangle \longrightarrow \langle statement \rangle$

where *guard* is a boolean predicate over the protocol variables, and the *statement* updates zero or more protocol variables. The statements are executed only when the guarding predicates become true.

B.2. *Fault Model*

In a MANET, nodes and links can fail-stop when they function correctly. This can happen due to battery drain-out in the mobile nodes. Further, the state of the node, i.e. the values of all the variables of the node, can be corrupted. However, the protocol actions of a node can not be corrupted. When beacon is not received from a node, all the neighboring nodes sense a link failure or disconnection of the node. In essence, any disconnection in the tree is treated as a fault in the system.

B.3. *Local Fault-confinement*

In this section, we concentrate on reducing stabilization overhead by reducing its latency [16]. One of the efficient ways to achieve this reduction is through fault-containment [16] [15] which intends to contain the effects of single faults. In section 4.3.3, we presented a single transient fault scenario (Figure 12) for which fault-containment can reduce stabilization latency over SS-SPST. In this section we will briefly revisit the concept of fault-containment for self-stabilizing algorithms.

Fault-containment: As per our discussion in Section 4.3.3, the principal problem of self-stabilizing algorithms is the propagation of single transient faults across the nodes due to the wrong sequence of local execution in the asynchronous nodes. Fault containment intends to enforce proper sequence of execution in the distributed nodes.

Self-stabilizing algorithms, including SS-SPST and SS-SPST-E, have a common structure of execution at any particular node i as given below:

$$\begin{aligned} self\_stabilize(i) \equiv\ & G_1(i) \rightarrow A_1(i) \\ & G_2(i) \rightarrow A_2(i) \end{aligned}$$

where $G_1(i)$, $G_2(i)$ are the guarding predicates at node i that becomes true when the system is in an illegitimate state; and $A_1(i)$, $A_2(i)$ are the corresponding local actions taken in node i to stabilize the system. For SS-SPST-E algorithm

•$G_1(i) \equiv i = root \wedge (H_i \neq 0 \vee P_i \neq NULL)$

•$A_1(i) \equiv H_i = 0; P_i = NULL; O_i = 0;$

•$G_2(i) \equiv i \neq r \wedge (O_i \neq \min_{\forall j \in \mathcal{VP}(i)} (O_i(j)) \vee P_i \notin \mathcal{N}(i) \vee H_i \neq H_{P_i} + 1)$

•$A_2(i) \equiv O_i = \min_{\forall j \in \mathcal{VP}(i)} (O_i(j)); P_i = j, j \in \mathcal{N}(i); H_i = H_j + 1;$

Further, for the ease of representation, the following notations are defined [16]:

– N_i : Neighboring nodes of i.

– $G_p(i) = G_1(i) \vee G_2(i)$

– $stabilize(i) \equiv$if $G_2(i)$ then $A_2(i)$ else $A_1(i)$

Here, $G_p(i)$ is a boolean predicate which signifies if any fault has occurred from the perspective of node i; and $stabilize(i)$ is a function similar to $self\_stabilize(i)$ signifying the local actions. In addition, another boolean function $can\_stabilize(i)$ is defined which is used to ascertain if the local actions ($stabilize(i)$) can indeed eradicate the faults in case $G_p(i)$ is true [16]. Formally, this function can be defined as follows:

$$can\_stabilize(i) \equiv G_p(i) \wedge (\text{execution of } stabilize(i) \Rightarrow (\sim G_p(i) \wedge (\forall j \in N_i :\sim G_p(j)))$$

Now, based on these definitions, fault-containing self-stabilizing algorithms can be developed as follows:

```
1. if $G_p(i) \wedge can\_stabilize(i)$
2. then $stabilize(i)$
3. else if $G_p(i) \wedge \sim can\_stabilize(i) \wedge$
                $\forall j \in N_i :\sim can\_stabilize(j)$
4. then $self\_stabilize(i)$
```

The question is to find out if the local action can totally stabilize the system. If it is so, then local actions are taken which will stabilize the system. So, the fault is not propagated. Let us consider the scenario in Figure 12. Here G_p of any non-root node i signifies if the selected parent is not $NULL$ and leads to shortest path to the root. When C fails, the *can_stabilize* function in node E returns false, as execution of $stabilize(i)$ can not eradicate the fault (i.e. make G_p true) in D which is a neighbor of E. So it will not take any local actions (as per lines 1 and 2 in the fault-containing algorithm). However, the *can_stabilize* in D returns true as execution of $stabilize(i)$ in D would make G_p false for both D and E. D would therefor execute $stabilize(i)$ as per lines 1 and 2 of the algorithm. Further, the check in line 3 ensures that node E would refrain from taking self-stabilizing action. Therefore, the fault-containment over self-stabilizing algorithms ensure a sequence of execution among the nodes eradicating possible propagation of single transient faults during stabilization (as described in section 4.3.3).

Algorithm 1. Self-managing multicast suit for MANETs

```
procedure LocalActionsAtEachRound
    Update the neighbor set $Adj(i)$ and the disconnection counter
    call PerformActionsForSS-SPST-E$fc$
    Update power level for data transmission
    PerformAMO
end procedure

procedure PerformAMO
    if $i = r$ then  ▷ Root performs the optimization and disseminates $\beta_{Opt}$ to the child using beacon messages
        call EstimateBulkPoissonMessageParameters
        call EstimateLinkChangeIntensity
        call CalculateOptimumBeaconInterval
        update optimum beacon interval to be transmitted with the next beacon.
    else  ▷ Non-root nodes transmits the link change intensity to the root through beacon messages and gets $\beta_{Opt}$ from the parent
        call EstimateLocalLinkChangeIntensity
        transmit local link change intensity to parent
        set $\beta_{Opt}$ as the value from the parent
    end if
end procedure

procedure EstimateBulkPoissonMessageParameters
    Initialize timer for estimating $\lambda$
    while Packet received do
        calculate $t_{last}$ i.e. the time from the last packet received
        if $t_{last} = t_{old}$ then  ▷ No change in the inter-arrival time, packet is within the same burst
            select $\tau$ as $t_{last}$
            increase packet count within a burst
        else  ▷ Change in the inter-arrival time, new burst starting
            calculate $\alpha$ as the total packet count in the last burst
            initialize packet count for the new burst
            increment Poisson event count
            estimate $\lambda$ from the number of Poisson events using Bayes sequential estimation
        end if
    end while
end procedure

procedure EstimateLocalLinkChangeIntensity  ▷ at each non-root node
    Estimate $\mu_i$ based on exponential weighted average at every $1/\lambda$ time
    Aggregate the local $\mu_i$ with the $\mu_i$s from the children
end procedure

procedure EstimateLinkChangeIntensity  ▷ at the root node
    call EstimateLocalLinkChangeIntensity
    set $\mu$ by halving the aggregate $\mu_r$
end procedure

procedure CalculateOptimumBeaconInterval
    Replace the left hand side of Equation 13 with $k\beta + f\beta + 3(1 - f)\beta$ (Section 4.3.3)
    Derive $\beta_{opt}$ (Equation 15) by taking the equality of the resulting equation
end procedure

procedure PerformActionsForSS-SPST-E
    if $(i = r \wedge (H_i \neq 0 \vee P_i \neq NULL))$ then  ▷ Guard 1 (root node)
        $H_i = 0; P_i = NULL; O_i = 0;$  ▷ Actions triggered by Guard 1
    end if
    if $\Big(i \neq r \wedge (O_i \neq \min_{\forall j \in \mathcal{VP}(i)}(O_i(j)) \vee P_i \notin \mathcal{N}(i) \vee H_i \neq H_{P_i} + 1 \vee H_i > n)\Big)$ then  ▷ Guard 2 (non-root nodes)
        $O_i = \min_{\forall j \in \mathcal{VP}(i)}(O_i(j)); P_i = j, j \in \mathcal{N}(i); H_i = H_j + 1;$  ▷ Actions triggered by Guard 2
    end if
end procedure

procedure PerformActionsForSS-SPST-E$fc$
    if either Guard 1 or Guard 2 is true in SS-SPST-E then
        if local actions at any neighbor node falsifies the now true guard then
                                                                        ▷ skip actions
        else
            call PerformActionsForSS-SPST-E
        end if
    end if
end procedure
```

Appendix C. Comprehensive pseudocode for the multicast

Algorithm 1 gives the pseudocode of SS-SPST-E*fc* with all the optimizations and self-managing enhancements.