Energy Based Computation In Wireless Sensor Network Using S-Remit Algorithm

Ms. Sonali Raut [1] and Ms. Hemlata Dakhore [2]
[1]Department Computer Science & Engineering, GHRIETW, RTMNU University, Nagpur

[2]Department Computer Science & Engineering, GHRIETW, RTMNU University, Nagpur

ABSTRACT

Wireless Sensor Networks (WSN) are formed by a large
number of networked sensing nodes. It is rather complex, or
even unfeasible, to model analytically a WSN and it usually
leads to oversimplified analysis with limited confidence. Besides, deploying test-beds supposes a huge effort. Therefore, simulation is essential to study WSN. However, it requires a suitable model based on solid assumptions and an appropriate framework to ease implementation. enough to capture the real behavior of a WSN, thus, jeopardizing the credibility of results. However, detailed models yields to scalability and performance issues, due to the large number of nodes. Trust and reputation models research and development
for distributed systems such as Wireless Sensor Networks (WSNs) has arisen and taken importance in the last recent years among the international research community. In this paper we propose a distributed algorithm called S-REMiT for building an energy-efficient multicast tree in a wireless sensor network. Our simulations show that it performs better than BIP/MIP and EWMA algorithms.

RELATED WORK

The energy-efficient broadcasting/multicasting tree problem
is presented in [5]. Wieselthier et al. have proposed a "nodebased"elastic model for wireless multicast and the concept of *wireless multicast advantage* [5]. Because the problem of constructing the optimal energy-efficient broadcast/multicast tree is NP-hard, several heuristic algorithms for building a source-based energy-efficient broadcast/multicast tree have been developed recently [6]. Wieselthier et al. presented BIP/MIP algorithm which is a centralized source-based broadcast/ multicast tree building centralized algorithm [5]. They also presented two distributed version of BIP algorithm (Dist-BIP-A,Dist-BIP-G), but these two distributed algorithms have slightly worse performance than centralized version [2]. Banerjee et al. have presented the reliability issues and energy cost metrics suitable for energy-efficient source-based broadcast/multicast tree [7]. Cagalj et al. have presented an Embedded Wireless Multicast Advantage (EWMA) algorithm to enhance energy efficiency of source-based broadcast tree by refining a MST [3]. They also described a distributed version of EWMA algorithm. We propose a distributed algorithm called S-REMiT which is a part of a suite of algorithms called REMiT (Refining Energy efficiency of Multicast Trees) which we are designing to achieve various energy-efficiency goals related to multicasting in WSNs. REMiT algorithms are distributed algorithms which refine the energy-efficiency of a pre-existing multicast tree using local knowledge at each node. The REMiT algorithms can be categorized along energy metric dimension (minimizing energy-consumption or maximizing lifetime) and multicast-tree type dimension (source based or group-shared tree). For example, we have previously presented G-

REMiT [4] which minimizes energy-consumption for group-shared trees and L-REMiT [8] which maximizeslifetime for source-based trees, respectively. Both S-REMiT and EWMA algorithm refine an existing initial tree to an energy-efficient tree. EWMA is not extensible to energy efficient group-shared tree. However, S-REMiT can be easily extend to group-shared tree by incorporating multicast message generation rate in node metric [4].

INTRODUCTION

In contrast to wired network, availability of limited energy
at nodes of a wireless sensor network (WSNs) has an impact on the design of multicast protocols. For example, the set of network links and their capacities in WSNs is not pre-determined but depends on factors such as distance between nodes, transmission power, hardware implementation and environmental noise. Thus in WSNs, there is a tradeoff between the long "reach" of one-transmission (but received simultaneously by several nodes in the transmission range) and interference effects it creates in its communication neighborhood [1]. We assume that the transmission power level can be dynamically varied between specified lower and upper bound [2][3]. Therefore, there also exists a trade-off between reaching more nodes in a single hop by using more power and reaching fewer nodes in a single hop by using less power but requiring multiple hops for reaching all the nodes in the multicast group [1]. Hence new approaches are needed to
account for these characteristics.

In this paper, we focus on source initiated multicasting of data in WSNs. Our main objective is to construct a *minimum-energy multicast tree* rooted at the source node. We explore the following two problems related to energy-efficient multicasting in WSNs using a source-based multicast tree:

1) How to reduce the total energy cost for multicasting in
a source-based tree?
2) How to build an energy-efficient multicast tree in a distributed manner?

In this paper, we study these two problems and propose S-REMiT (An algorithm for Refining Energy-Efficient Source-based Multicast Tree) for building an existing multicast tree into a more energy efficient multicast tree. As a distributed algorithm, S-REMiT uses *minimum-weight spanning tree* (MST) or *single-source shortest path tree* (SSSPT) as the initial solution and improves the multicast tree energy efficiency by switching some tree nodes from their respective parent nodes to new corresponding parent nodes. The selection of the initial tree is dependent on the energy model used.

SYSTEM MODEL AND ASSUMPTIONS

We make the following assumptions in our model:
1) Nodes are stationary in the WSNs.
2) Each node in the WSNs uses omni-directional antennas.
3) Each node knows the distance between itself and its neighboring nodes using distance estimation schemes

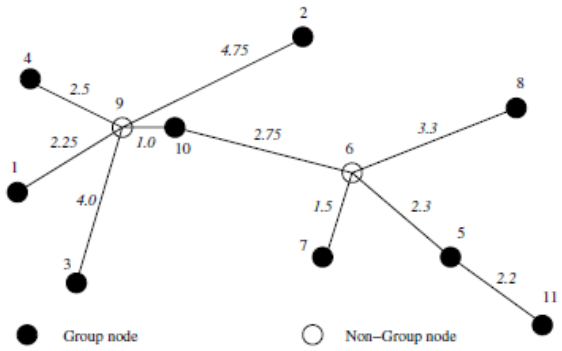

Figure 1 Distribution of nodes

such as [9] and [10]. We use wireless communication model in [11]. The connectivity of network depends on the transmission power. Each node can choose its power level p, where $0 \le p \le pmax$. A node may use different power level for each multicast tree in which it participates. Let Ei,j be the minimum energy cost

(per bit) needed at node *i* on the link between nodes *i* and *j* in a packet transmission. Then, $Ei,j = ET + K(ri,j)\alpha$, (1) where ri,j is the Euclidean distance between *i* and *j*, *ET* is a distant-independent constant that accounts for real-world overheads of electronics and digital processing, *K* is constant dependent upon the properties of the antenna and α is a constant which is dependent on the propagation losses in the medium [5][1]. Some of the related work in this area, such as [5][3], did not consider *ET* . However, *ET* is not negligible especially for short range radios, since *ET* can substantially exceed the maximum value of the $K(ri,j)\alpha$ [11]. Compared to wired networks, WSNs have "wireless multicast advantage" which means that all nodes within communication range of a transmitting node can receive a multicast message with only one transmission if they all use omni directional antennas [5].

In our model, every node (say node *i*) has two kinds of coverage area. One is Control coveRage area (*CRi*), another

is Data coveRage area (*DRi*) such that $DRi \subseteq CRi$. **Neighbors** of node *i* are the nodes within *CRi*. We use *Vi*, $Vi \subseteq CRi$, to denote the set of **tree neighbors** of node *i*, i.e, those neighbors of node *i* which also belong to the multicast tree *T*. A **connected tree neighbor** *j* of a node *i* is a tree neighbor of node *i* which is connected to the node by a *branch*, i.e., link $(i, j) \in T$. A **non-connected tree neighbor** *j* of a node *i* is a tree neighbor of node *i* which is connected to the node *i* by more than one branch in *T*, i.e. the length of the unique path between *i* and *j* in *T* is greater than 1. We denote the set of connected and non-connected tree neighbors of node *i* as *CTNi* and *NCTNi*, respectively. Note that $NCTNi = Vi - CTNi$. since S-REMiT ignores other links. Branch labels denote the Euclidean distance between their endpoints.

MULTICAST ENERGY EFFICIENCY METRIC

The energy consumption (per bit) at every tree node is determined by the distance between the children nodes.

We calculate $Ei(T, s)$, the **energy cost metric** of node *i* on the multicast tree *T* in node *s*'s source-based multicast tree, as follows:

$$E_i(T, s) = \begin{cases} E^T + Kd_i^{\alpha} & \text{if i is the source node;} \\ E^T + Kd_i^{\alpha} + E^R & \text{if i is neither the source nor a leaf node in T;} \\ E^R & \text{if i is a leaf node in T.} \end{cases} \quad (2)$$

We use $TEC(T, s)$ to denote the Total Energy Cost of all the nodes in the multicast tree *T* in node *s*'s source-based multicast tree. So $TEC(T, s)$ in *s*'s source-based multicast tree as:

$$TEC(T, s) = \sum_{i \epsilon T}^{n} Ei(T, s) \quad (3)$$

So the *problem of minimizing the energy consumption of*

multicast tree becomes the problem of minimizing the energy

cost (per bit) at every node in the multicast tree as much as

possible.

S-REMIT ALGORITHM

S-REMiT tries to minimize *TEC* of the initial multicast tree by changing a node's parent to another tree node so that the tree's *TEC* is reduced. We use MST or SSSPT as the initial tree because these two trees perform quite well for our problem based on our experimental results. These two trees are used for different scenarios: when nodes use long range radios, MST is the initial tree, and when nodes use short range radios, SSSPT is the initial tree. We use $Change_i^{x, j}$ to refer to the refinement step in which node *i* switches from node *x* to node *j*. Let *T* be the initial

multicast tree, and *T'* be the resulting graph after refinement $Change_i^{x,j}$ is applied to *T*. The following lemmas, presented here without proof, guarantees that *T'* is a tree and identify which node's energy cost change due to refinement:

Lemma 1: If node *j* is not a descendant of node *i* in tree *T*, then the tree remains connected after $Change_i^{x,j}$

Lemma 2: Nodes *j* and *x* are the only nodes in the tree whose energy cost may be affected by $Change_i^{x,j}$.

Criterion for Switching Parent

The *TEC* value of the multicast tree may change as a result of performing a refinement. In our heuristic, we call the change in the tree's *TEC* due to refinement $Change_i^{x,j}$ as *gain* in the tree's *TEC*, i.e. $gain = TEC(T, s) - TEC(T', s)$. S-REMiT uses *gain* as the criterion for changing the parent of a node: the refinement $Change_i^{x,j}$ is performed only if it is expected that $gain > 0$.

For example, consider node 10's source-based multicast tree in Figure-1. We consider how node 2 decides to change its parent from node 9, to node 6. We refer to this change event as $Change_2^{9,6}$. To simplify the following explanation, we assume that:

$$K = 1,\ \alpha = 2,\ ET = 0, \text{ and } E^R = 0.$$

Using Equation (2), node 2 will estimate the change in the energy cost at nodes 2, 9 and 6 if it makes $Change_2^{9,6}$ First, node 2 will estimate the current energy consumed at nodes 2, 6 and 9:

$E_6(T, 10) = r_2^{6,8} = 10.89$, and $E_9(T, 10) = r_2^{9,2} = 22.56$.

Similarly, node 2 can estimate the new energy cost at nodes 9 and 6 (based on *Lemma* 2, node 2's energy cost will not changed by $Change_2^{9,6}$) after $Change_2^{9,6}$, i.e $E_6(T', 10)$ and $E_9(T', 10)$ respectively:

$E_6(T', 10) = r^2_{6,2} = 12.96$, and $E_9(T', 10) = r^2_{9,3} = 16.0$.

The *gain* ($g_2^{9,6}$) obtained by switching at node 2 from node 9 to node 6 is:

$g_2^{9,6} = (E_9(T, 10) + E_6(T, 10)) - (E_9(T', 10) + E_6(T', 10)) = 33.45 - 28.96 = 4.49$.

Likewise node 2 can compute the gain in energy cost if it switches to node 10 and node 8:

$g_2^{9,10} = (E_9(T, 10)+E_{10}(T, 10))-(E_9(T', 10)+E_{10}(T', 10)) = 30.12 - 32 = -1.88$.

$g_2^{9,8} = (E_9(T, 10) + E_8(T, 10)) - (E_9(T', 10) + E_8(T', 10)) = 22.56 - 30.44 = -7.88$.

By comparing the gains, node 2 selects a node with the highest positive gain as the new parent which is node 6. Node 2 will disconnect from node 9 and connect to node 6 as its new parent node. So in Figure 1, tree link between nodes 2 and 9 will be deleted, and tree link between nodes 2 and 6 will be added to the multicast tree. Because DR_9 does not need to cover node 2 anymore, radius of DR_9 will decrease to $r_{9,3}$. DR_6 should be increased to cover node 2, hence radius of DR_6 will increase to $r_{6,2}$.

Local Data Structure and Messages Types

Before describing a node's local data structure and message types used by our distributed protocol, we introduce the following notation. Let *d'i* be the second longest link between *i* and its children. We denote the two-tuple (*di*, *d'i*), as *li*. Further, let node *j* be a neighbor of *i*, $j \in Vi$. We will use the notation $Data_i$ to denote the data associated with node *k*:

• *Ei*(*T*, *s*): energy cost (per bit) of node *i* on the tree *T* in node *s*'s source-based multicast tree;

• *CTNTi*: a list of records of the type (k, l_k), $\forall k \in CTNi$;

• *NCTNTi*: a list of records of the type (k, l_k), $\forall k \in NCTNi$.

S-REMiT uses the following message types:

• *TOKEN*(*i*, *flag*): source node *s* uses Depth-First Search (DFS) to pass token to every node on the multicast tree along the tree branches. Node *i* is the next hop node in DFS order. *flag* is a boolean value to represent the refinement was successful or not in the

DFS. This message is important and used throughout the second phase of S-REMiT.

• *JOIN REQ*(*i, j*): sent by node *i* to node *j* requesting *j* to become its parent. This message is used in Step 2 by node *i* to make $Change_i^{x,j}$.

• *JOIN REP*(*i, j*): sent by *j* to reply node *i*'s *JOIN REQ*(*i, j*). This message is used in Step 2 by node *j* to make $Change_i^{x,j}$.

• *LEAVE*(*i, x*): sent by node *i* to leave parent node *x*.

This message is used in Step 2 by node *i* to make $Change_i^{x,j}$ and in Step 5 by node *i* to leave the tree when *i* is a leaf node and non-group node.

• *NEIGHBOR UPDATE*(*i, x, j*): sent by node *i* to nodes in *Vi* notifying $Change_i^{x,j}$. This message is used in Step 3 by node *i*. S-REMiT needs reliable passing these messages between nodes.

Distributed Algorithm Description

S-REMiT consists of two phases:
1) multicast tree construction and
2) multicast tree refinement.

In the **first phase**, if nodes use long range radios, all nodes run a distributed algorithm proposed by Gallager et al. [12] to build a MST tree; if nodes use short range radios, all nodes run a distributed algorithm proposed by Chandy et al. [13] to build a SSSPT tree. We require that at the end of the first phase, node *I* ($i \in T$, where *T* is the multicast tree) has all local information *lk*, $\forall k \in Vi$. Nodes obtain *lk* by hearing *k*'s one-hop local broadcasting within *CRk*. In the **second phase**, the difficulty in this distributed environment is when and how to terminate the refinement. We organize the second phase in rounds. Each round of the second phase is led by the multicast source *s*. It terminates S-REMiT algorithm when there is no energy *gains* by switching any node in the multicast tree. In each round, S-REMiT token is passed to the nodes one by one in DFS order. The S-REMiT token gives the permission to a node to do refinement. The node holding the S-REMiT token can do refinement, other nodes only can respond to the node with S-REMiT token. When *i* obtains the S-REMiT token, it does the following steps to refine the tree. We use *Ej*(*T_, s*) and *Ex*(*T_, s*) to denote the energy cost at *j* and *x* after $Change_i^{x,j}$, respectively. *JOIN REQ*, *JOIN REP* and *LEAV E* messages are used by nodes *i, x,* and *j* to make $Change_i^{x,j}$. Following are the steps of the second phase in S-REMiT algorithm (see Figure 2 for illustrations of these steps):

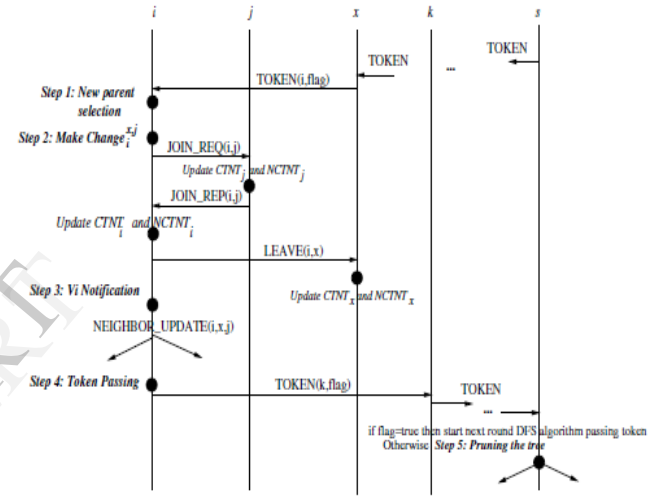

Figure-2 Second Phase of S-REMiT at node *i*. Node *k* is the next hop nodeof *i* in DFS algorithm

1) New parent selection: Select a new parent candidate

j with the highest positive *gain* ($gx,j\ i := (Ex(T, s) + Ej(T, s)) - (Ex(T_, s) + Ej(T_, s))$), which will not result in tree disconnection if node *i* makes $Changex,j\ i$. If there is no such node *j* available, then it constructs token as *TOKEN*(−, *false*). **Make** $Changex,j\ i$ **:** Node *i* makes $Changex,j\ i$ by *JOIN REQ* and *JOIN REP* negotiation with node *j*. Node *j* sends *JOIN REP* back to node *i*. If node *i* gets *JOIN REP* message, it will change *CTNTi* and *NCTNTi*, send *LEAV E* message to node *x*, constructs token as *TOKEN*(−, *true*) and go to next step. Otherwise, it will go back to step 1 to select a new parent *j*.

3) *Vi* **Notification:** Node *i* notifies nodes in *Vi* about the *Changex,j i* .

4) **Token Passing:** Node *i* passes the token to next hop node

according to DFS algorithm.

5) **Pruning the tree:** If node *s* gets back the token with *flag = false*, which means that no energy *gains* in this DFS round, *s* will request all of the tree node to prune the redundant transmissions that are not needed to reach the members of the multicast group from the tree.

1) calculates *gains* as explained previously in Step 1 and finds out *g* is the highest positive value.

2) now sends *JOIN REQ* to node . When node responds to other node with *JOIN REP* message, node will move node from *NCTNT*2 to *CTNT*2 and it will send *LEAV E* message to node . Then node will remove node from *CTNT*2 to add it to *NCTNT*2.

3) Node will send *NEIGHBOR UPDATE* to nodes in *V*2 about *Change*.

4) Finally, node will pass the token *TOKEN*) to next node according to the DFS algorithm.

Complexity of S-REMiT algorithm for minimizing sourcebased multicast tree

The message complexity of each node changing parent is $O(1)$. Hence the message complexity of one round in which each node performs at most one parent changing is $O(N\delta max)$, where N is the number of nodes in the tree, and δmax is the maximum number of neighbor any node has in the tree. The computational complexity of one parent changing is $O(\delta max)$. Therefore the computational complexity of one round is $O(N\delta max)$. The space complexity of S-REMiT for each node is $O(\delta max)$ since the size of V is $O(\delta max)$.

PERFORMANCE EVALUATION

We used simulations to evaluate the performance of SREMiT algorithms. We compared our algorithm with BIP/MIP algorithm and EWMA algorithm distributed version (EWMADist). Because EWMA-Dist algorithm is used for building broadcast tree, we extend EWMA-Dist algorithm for multicasting by pruned the redundant transmission from the broadcast tree produced by EWMA-Dist algorithm. The simulations performed using networks of four different sizes: 10, 40, 70, and 100. The distribution of the nodes in the networks are randomly generated. Every node is within the maximum transmission range (*rmax*) of at least one other node in the network. In other words, the network is connected. We use two different *ET* values to represent the long range radios and short range radios. Based on the experiment data in [11], we decide to use $ET = 0$ to represent long range radios

and $ET = 4 * K(rmax)2$ to represent short range radios. We

ran 100 simulations for each simulation setup consisting of a

network of a specified size to obtain average *TEC* with 95%

confidence, the propagation loss exponent α is varied. And the

source node *s* is randomly selected for every network setup.

Performance Metric

The performance metric used is *TEC*. We use *TEC* of multicast tree to define *Normalized TEC* with algorithm *alg*

is: *TECalg TECbest* , where *TECbest* = min*{TECalg}*, $alg \in A$,*A* = {S-REMiT, MIP or EWMA-Dist*}*.

Simulation Results

For short and long range radios, the performance is shown in Figures 3 and 4 respectively. We can see the average *normalized TEC* (show on the vertical axis) achieved by the algorithms on networks of different size (the horizontal axis). The figure show that the solutions for multicast tree obtained by S-

REMiT have, on the average, lower *normalized TEC* than the solutions of and EWMA-Dist when 50% of the nodes are group members . S-REMiT and EWMA-Dist have very close performance, when 100% nodes are group members. In other words, performance difference between S-REMiT and EWMA-Dist becomes larger as the group becomes sparse (This is also true for other scenarios). This is because the greedy nature of EWMA-Dist, the algorithm trying to reduce the number of downstream transmitting nodes as many as possible when there is a chance to reduce the total energy consumption of the multicast tree.

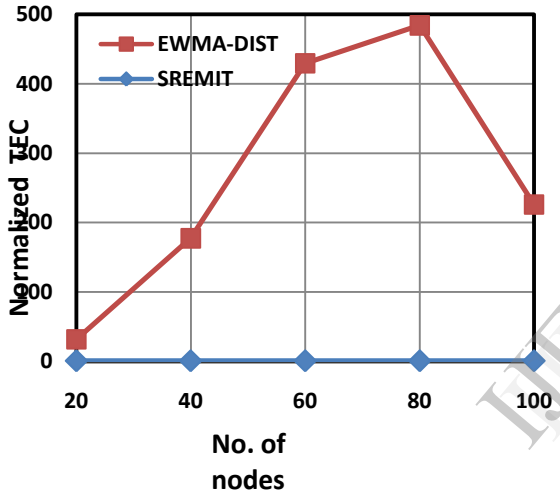

Figure-3. Normalized *TEC* for short range radios in multicast group.

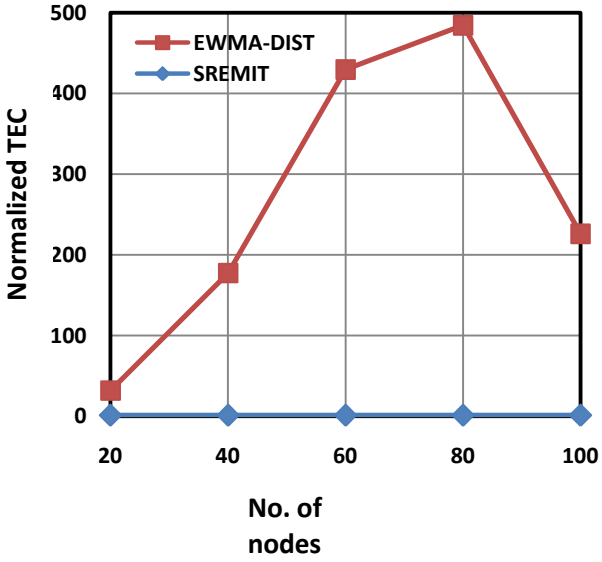

Figure-4. Normalized *TEC* for long range radios in multicast group.

EWMA-Dist has more unnecessary coverage to non-group nodes than S-REMiT. Although these non-group nodes which are leaf nodes will be pruned from the multicast tree, the greedy effect cannot be reimbursed in EWMA-Dist algorithm., we can see that the multicast trees produced by S-REMiT algorithm have, on the average, lower *normalized TEC* than those obtained by the EWMA-Dist. Because of the space limitation, we do not present all of the results. Our results show that for various scenarios the average *normalized TEC* of EWMA-Dist is between 1.0 and 3.8, and the average *normalized TEC* of S-REMiT is between 1.0 and 1.1, respectively. Also our simulation results show that energy overhead of S-REMiT is always below 1.5% of total energy cost of the multicast tree when source node send out 1MBytes data to the all of group members. Based on our simulation results, we find that S-REMiT has better performance than EWMA-Dist for various scenarios.

GENERIC TRUST AND REPUTATION MODEL

Each trust and reputation model has its own specific characteristics and particularities. However, most of them share the same abstract schema or pattern about what steps have to be given in order to complete a whole transaction in a distributed system making use of a trust and/or reputation model. Therefore, one of the main targets followed by our work was to design and provide a trust and reputation models interface as generic as possible. So first of all, we identified the four main steps to be done in most of this kind of models [12], [13]. Figure 1 shows these steps.

Figure 5 Generic Trust and Reputation Model Scheme

We have developed an abstract Java class called TRModel_WSN containing one attribute: a set of generic parameters for trust and reputation models

(abstract class TRMParameters containing the name of the parameters file and the parameters themselves in the form <parameter=value>). In order to add a

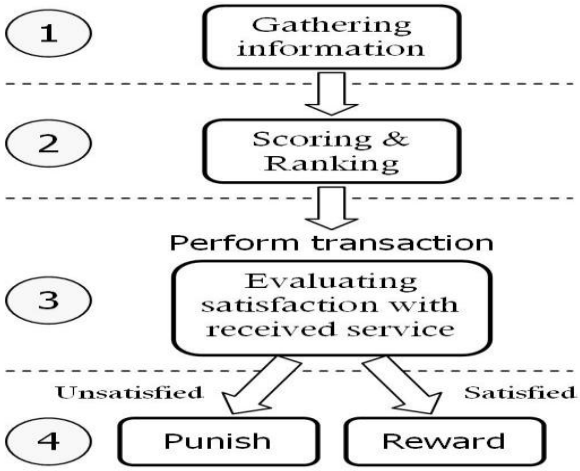

Figure 5 Generic Trust and Reputation Model

new trust and/or reputation model to the simulator both subclasses of TRModel_WSN and TRMParameters have to be implemented. A subclass of Service class could be also defined in order to specify more details or characteristics (such as associated costs or quality parameters, for instance) of a certain service. Additionally, class TRModel_WSN defines the five public abstract methods shown in table I in order to accomplish the steps illustrated in figure 1.

The first method, gatherInformation, is responsible for collecting or gathering the necessary information from other nodes in the network (indirect experiences, recommendations, reputation values, etc.) if we are dealing with a pure reputation model, direct experiences or pre-trusted nodes, if what we have is a pure trust model, or a combination of both, which is the most common case.

Its first parameter is the Client who is requesting thecdesired service and, therefore, needs the application of the

trust and reputation model in order to find the most trust worthy or reputable server offering the Service given as a second parameter. It returns a GatheredInformation object. Currently this class only contains the paths leading to those servers which are candidates to be selected as service providers. Each model can create a subclass of this one including the specific information needed to work. The second method, scoreAndRanking, receives the gathered information from the previous one and scores each path leading to a server, returning either a sorted collection of these servers (according to the score received) or the path leading directly to the most trustworthy server found. The third abstract method belonging to class TRModel_WSN, called performTransaction, receives as a parameter the path found in the previous step, so it can actually apply for the required service to the server selected as most trustworthy or most reputable by the implemented model . Then the server, according to its goodness, will provide exactly the same service it has been asked for, a worse one or even a better one, in some cases. Once the client receives the service, it assesses its satisfaction and returns its value In an Outcome object (necessary to perform some statistics in order to evaluate the accuracy of the model). Some models would store in this step that transaction satisfaction as a direct experience.

Finally, the last two methods, reward and punish, carry

out the fourth step pointed out in the scheme shown in figure. That is, they perform the reward and punishment, respectively, to the server who has been selected to have the transaction with. Depending on the satisfaction of the client with the supplied service, one or the other will be applied. They both receive two parameters: the path leading to the most trustworthy or reputable server found in the second step

and the outcome got in the third one, containing, among other

things the satisfaction or dissatisfaction of the client with the

received service. It is worth mentioning that there are, however, some trust and reputation models which do

not apply any additional punishment and/or reward to those nodes the interaction has been carried out with. Thus, these two methods may not have any particular code associated depending on the particular trust and reputation model being implemented and addapted to the TRMSim-WSN proposed architecture. Regarding the parameters needed for the trust and reputation model, abstract class TRMParameters defines several protected methods, used to store and retrieve generic parametersof any of the primitive types.

TRMSIM-WSN

In this section we will formally present and describe our proposal of Trust and Reputation Models Simulator for Wireless Sensor Networks, called TRMSim-WSN [7]. A screenshot of the main window of TRMSim-WSN can be observed in figure 6.

Network settings

The very first step to be carried out when using our simulator is to create a new WSN. To do that, there are two fields where we can establish the maximum and the minimum number of sensors we want our networks to have, as well as a slide bar to set the wireless range of every sensor. Those three parameters will determine the links density of the network (i.e., the neighborhood of every node). Additionally, we can select which percentage of the nodes we want to act as clients requiring a default service. The rest of them will act, therefore, as servers. We can also say which percentage of those servers will not offer the required service and will then only act as relay nodes. Finally, regarding the servers who actually offer the desired service it is possible to determine the percentage of them who will be malicious ones, that is, they will not provide the service these are actually offering, but a worse one or even any service.

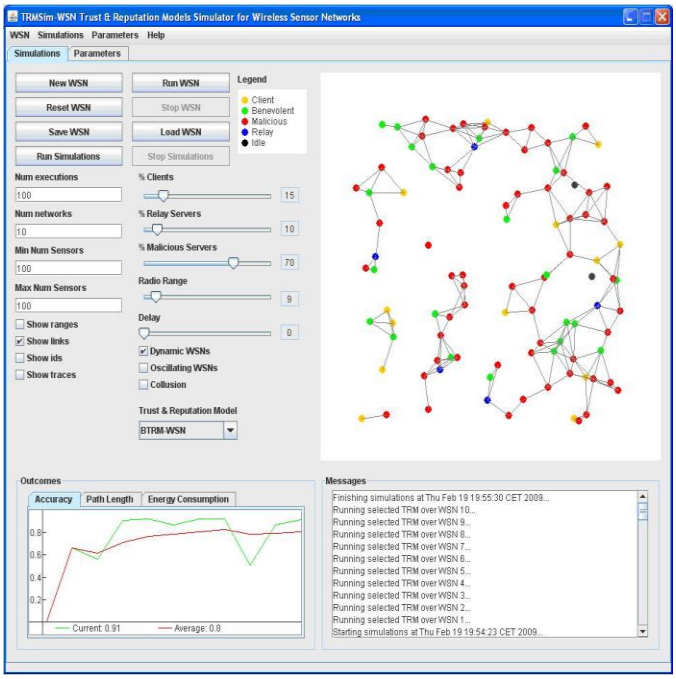

Figure 6 TRMSim-WSN Trust and Reputation Models Simulator for Wireless Sensor Network.

Once we have set all those parameters according to our needs, a new random WSN can be created just by pushing the bottom labelled “New WSN”. It is also possible to load a WSN from a XML file by pushing “Load WSN” button, and to save the current one into a XML file through the “Save WSN” button.

If we want to evaluate the WSN we currently have, but with different links density, we can change the wireless range parameter and push “Reset WSN” button.

Simulation settings

The next thing to configure are the simulation settings. First

we can determine the number of executions we want for our

simulations, that is, the number of times every client in the

network will ask for its default service, making use of the

selected trust and reputation model. We can set the number

of different random WSNs we want as well, according to the
settings described in the previous subsection. We can take some decisions regarding the visual or graphic presentation of the networks to be tested in our simulations. For instance, we can decide whether we want the wireless ranges to be shown or not, as well as the links connecting sensors or the identifier of each one of them. TRMSim-WSN is initially released with two trust and reputation models: BTRM-WSN [3] and Peer Trust [14]. Furthermore, the parameters panel allows us to set the input parameters file, or to manually specify the value of each parameter needed by the current selected trust and reputation model. Since one of the main characteristics of WSNs are their constraints about battery and energy consumption, a dynamic WSN can also be simulated, where some sensors swap into an idle state for awhile if they do not receive any request within a certain period of time. A sensor in an idle state does not receive nor transmit any message or packet. After a certain timeout they wake up again.

Once we have established all the previous settings, we are ready to start our simulations. If we want to run a simulation only over current network, we should press “Run WSN”. TRMSim-WSN. Trust and Reputation Models Simulator for Wireless Sensor Networks button. “Stop WSN” button allows us to force the finishing of that simulation. Otherwise, if what we want is to run a simulation over a given number of random WSNs, then we have to push button labelled “Run Simulations”. We can stop that simulation whenever we want by pressing “Stop Simulations” button, and current outcomes will be shown. Finally we can also add some delay between each simulated network, if we need to check the topology of every tested WSN. The maximum value corresponds to one second.

Oscillating behavior and collusion

In order to test the accuracy of every simulated trust and
reputation model we have included two security threats [15] to
our simulator. First one has to do with the oscillating behavior
of the servers offering the requested service. Therefore, if that option is selected, after every 20 executions (i.e. transactions or interactions), each malicious server becomes benevolent. Then the same percentage of previous malicious servers are randomly chosen to be now malicious(note that with a scheme like this a malicious server could remain malicious after 20 executions). The second security threat introduced consists of the possibility for the malicious servers to form a collusion among themselves. That implies that every malicious sensor will give the maximum rating for every other malicious sensor, and the minimum rating for every benevolent one.

A good trust and reputation model should quickly react against these behavioral changes and collusions and readapt
itself in order to prevent selecting a malicious node as the
most trustworthy or reputable one.

Outcomes and messages

Finally, two panels help us to know what has happened or is
currently happening in the simulator, and which are the results
of the last simulation. In the messages panel, for instance, several messages are shown containing useful information like the instant when the last simulation started or finished, or which is the current WSN being tested. Moreover, every action such as creating a new WSN, loading or saving current one or showing ranges, identifiers and links, among others, are also recorded and shown there. On the other hand, the outcomes panel lets us know the results of the

current simulated network, or the average outcomes for a whole simulation. Three important values can be observed here: the accuracy of the model, the average length of all the paths found by every client of every simulated network, and the energy consumed by the model (for future work).

Additional panels can be easily added if required in order to

show more details about the experiments. The average satisfaction is computed collecting the satisfaction of every client belonging to each one of the tested WSNs. However, clients who can not reach any benevolent server are not taken into account for computing these outcomes (since any trust and reputation model is useful in that situation). In figure we can observe that a simulation over 10 random dynamic WSNs (with 100 sensors each one) has been carried out using BTRM-WSN model. There were a 15% of clients, an 8.5% (85% · 10%) of relay sensors, a 53.55% (85% · 90% ·70%) of malicious servers and a 22.95% (85% · 90% · 30%) of benevolent ones. The average number of hops needed to reach the most trustworthy server was 6.04 and the average percentage of times that the model selected a benevolent server as the most trustworthy one was 80%.

CONCLUSIONS

In this paper, we proposed a distributed algorithm (S-REMiT)

for building an energy-efficient multicast tree in a WSNS. Further, S-REMiT employs a more realistic energy consumption model for wireless communication which takes

into account the energy losses not only due to radio propagation but also the energy losses in the transceiver electronics. This enables S-REMiT to adapt a given multicast tree to a wide variety of wireless networks irrespective of whether they use long-range radios or short-range radios.

We show that this algorithm outperforms two most famous

proposals in the literature, BIP/MIP and Distributed version of

EWMA. And we find that the energy consumption overhead

of the algorithm itself is very small compared with the total

energy consumption of the tree.

REFERENCES

[1] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Resource-limited energy-efficient wireless multicast of session traffic," in *34th Annual Hawaii Int'l Conf. System Sciences*, Maui, Hawaii, Jan. 2001, pp. 3460–3469.

[2] "S-REMiT: A Distributed Algorithm for Source-based Energy Efficient Multicasting in Wireless Ad Hoc Networks"
Bin Wang and Sandeep K. S. Gupta Department of Computer Science and Engineering Arizona State University

[3] M. Cagalj, J. P. Hubaux, and C. Enz, "Minimum-energy broadcast in allwireless networks: NP-Completeness and distribution issues," in *Proc. ACM MobiCom 2002*, Atlanta, Georgia, Sept. 2002, pp. 172–182.

[4] "TRMSim-WSN, Trust and Reputation Models Simulator for Wireless Sensor Networks" F´elix G´omez M´armol and Gregorio Mart´ınez P´erez Departamento de ngenier´ıa de la Informaci´on y las Comunicaciones University of urcia
Facultad de Inform´atica, Campus de Espinardo, s/n 30.071, Murcia, Spain

[5] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "On the construction of energy-efficient

broadcast and multicast tree in wireless networks," in *Proc. IEEE INFOCOM 2000*, Tel Aviv, ISRAEL, Mar. 2000, pp. 585–594.

[6] 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. 18th Annual Theoretical Aspects of Comp. Sc. (STACS)*, vol. 2010, Springer-Verlag, 2001, pp. 121–131.

[7] S. Banerjee, A. Misra, J. Yeo, and A. Agrawala, "Energy-efficient broadcast and multicast trees for reliable wireless communication," in *IEEE Wireless Communications and Networking Conf. (WCNC)*, New Orleans, Louisiana, Mar. 2003.

[8] B. Wang and S. K. S. Gupta, "On maximizing lifetime of multicast trees in wireless ad hoc networks," in *International Conference On Parallel Processing (ICPP-03) (to appear)*, Kaohsiung, Taiwan, China, Oct. 2003.

[9] W. C. Y. Lee, *Mobile Communication Engineering*. McGraw-Hall, 1993.

[10] R. Steele, *Mobile Radio Communications*. PentechPress, 1992. [11] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 1548–1557.

[12] R. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum weight spanning trees," *ACM Trans. Programming Lang. & Systems*, vol. 5, no. 1, pp. 66–77, Jan. 1983.

[13] K. Chandy and J. Misra, "Distributed computation on graphs: Shortest path algorithms," *Communications of the ACM*, vol. 25, no. 11, pp. 833–837, Nov. 1982.

[14] C.-C. Chiang, M. Gerla, and L. Zhang, "Adaptive shared tree multicast in mobile wireless networks," in *Proceedings of IEEE Globecom'98*, Sydney, Australia, Nov. 1998, pp. 1817–1822.

[15] S. J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A performance comparison study of ad hoc wireless multicast protocols," in *Proceedings of the IEEE INFOCOM 2000*, Tel Aviv, ISRAEL, Mar. 2000, pp. 565–574.