Spring 2007

Energy-aware and secure routing with trust levels for wireless ad hoc and sensor networks

Eyad Taqieddin

Department: Electrical and Computer Engineering

ENERGY-AWARE AND SECURE ROUTING WITH TRUST LEVELS FOR WIRELESS AD HOC AND SENSOR NETWORKS

by

EYAD SALAH TAQIEDDIN

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2007

A. Miller, Co-advisor	J. Sarangapani, Co-advisor
S. Ali	R. J. Stanley

S. Madria

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following articles that have been submitted for publication as follows:

Pages 1-31 were accepted for publication in the INTERNATIONAL JOURNAL OF NETWORK SECURITY

Pages 32-53 are intended for submission to the INTERNATIONAL JOURNAL OF RELIABILITY, QUALITY AND SAFETY ENGINEERING

Pages 54-78 are under review INTERNATIONAL JOURNAL OF INFORMATION TECHNOLOGY

Pages 79 – 97 are intended for submission to the IEEE WIRELESS COMMUNICATOINS

ABSTRACT

This dissertation focuses on the development of routing algorithms for secure and trusted routing in wireless ad hoc and sensor network.

The first paper presents the Trust Level Routing (TLR) protocol, an extension of the optimized energy-delay routing (OEDR) protocol, focusing on the integrity, reliability and survivability of the wireless network. TLR calculates the link costs based on a composite metric (delay incurred, energy available at the neighbor node, energy spent during transmission and the number of packets sent on each link) for the selection of MPR nodes.

The second paper analyzes both OLSR and TLR in terms of survivability and reliability to emphasize the improved performance of the network in terms of lifetime and proper delivery of data. The goal is to explain analytically the achieved improvements in the energy consumption and its effect on the survivability and reliability.

The third paper proposes a statistical reputation model that uses the watchdog mechanism to observe the cooperation of the neighboring nodes. This model uses the Beta distribution as a basis for calculating/updating the trust value by applying the moment matching method for parameter estimation.

The last paper presents the results of the hardware implementation of Energy-Efficient Hybrid Key Management. This study serves two purposes; first, it identifies the effects of hardware constraints on the overall performance of the protocol. Secondly, it gives a foundation for the improvement of the simulators.

ACKNOWLEDGMENTS

رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ وَعَلَى وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا تَرْضَاهُ وَأَصْلِحْ لِي فِي ذُرِّيَّتِي إِنِّي تُبْتُ إِلَيْكَ وَإِنِّي مِنَ الْمُسْلِمِينَ

First and foremost, I thank Almighty Allah (God) for providing me with health, patience, and knowledge to complete this work.

It is a pleasure to thank the many people who made this dissertation possible. I would like to express my gratitude to my supervisors Dr. Ann Miller and Dr. Jagannathan Sarangapani for their guidance and encouragement throughout my graduate study as well as in completing this research. Their keen interest, guidance, advice and inspiration initiated and made this research possible.

To my committee members, Dr. R. Joe Stanley, Dr. Shoukat Ali, and Dr. Sanjay Madria, I extend my deepest appreciation for their time and effort in serving as committee members and reviewing this dissertation.

I would like to express my deepest thanks to Dr. Maciej Zawodniok for his invaluable contribution and suggestions that helped in completing this work. Special thanks go to Mr. James Fonda for all his help in setting up the hardware tests. I thank all my friends here in Rolla for all their support and encouragement. I spent great days with them and will cherish all the happy moments we shared.

I thank Jordan University of Science and Technology for financially supporting my doctoral study.

Special thanks go to my siblings, Ranya, Ehab, and Lara for all their support during the years. They always pushed me forward and for that I will always be grateful. Thanks go to my niece of 11 months, baby Jana, for bringing joy into our family.

Lastly, and most importantly, I am indebted to my parents for their patience with me during my endeavors. Without their support, I would not have been able to achieve success. To them, I dedicate this dissertation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PAPER IV

LIST OF TABLES

PAPER

I. Optimal Energy-Delay Routing Protocol with Trust Levels for Wireless Ad Hoc Networks

Eyad Taqieddin, S Jagannathan, Ann Miller

Department of Electrical and Computer Engineering
University of Missouri – Rolla
Rolla, Missouri 65409
{eyad,sarangap,milleran}@umr.edu

Abstract

This paper presents the trust level routing (TLR) protocol, an extension of the optimized energy-delay routing (OEDR) protocol [1], focusing on the integrity, reliability and survivability of the wireless network. TLR is similar to OEDR in that they both are link state routing protocols that run in a proactive mode and adopt the concept of multi-point relay (MPR) nodes. However, TLR aims at incorporating trust levels into routing by frequently changing the MPR nodes as well as authenticating the source node and contents of control packets. TLR calculates the link costs based on a composite metric (delay incurred, energy available at the neighbor node, energy spent during transmission and the number of packets sent on each link) for the selection of MPR nodes. We highlight the vulnerabilities in OEDR and show ways to counter the possible attacks by using authentication and traffic partition as a basis for mitigating the effects of malicious activity. Network simulator NS2 results show that TLR delivers the packets with a noticeable decrease in the average end-to-end delay with a small increase in the power consumed due to the additional computational overhead attributed to the security extension.

Keywords*—*Authentication, Optimal Route, Energy, Delay.

1. Introduction

A Mobile Ad-hoc Network (MANET) is a group of wireless mobile nodes that form a dynamic network topology without any centralized administration or fixed infrastructure. The nodes mobility requires establishing and breaking connections whenever needed. Each node communicates directly with the nodes within its wireless range. However, the nodes need to collaborate together to deliver the information between nodes that are beyond the wireless range of the source. With this approach, in terms of transmission, each node operates in two modes; source or router. Source nodes generate the traffic on the network whereas routing nodes receive the packets and forward them to the intended destination.

A routing protocol is used to detect the topology of the network and to enable each node to have a path to any of its intended destinations. The nodes use Link State Update (LSU) packets to share information among each other to build their respective routing tables and report any changes in network topology. The routing protocol should focus on energy conservation to increase the lifetime of the nodes while choosing routes with the least delay, jitter and congestion. Occasionally, the best routes for several sources, in terms of delay, go through the same node whose energy gets consumed at a higher rate compared to other nodes. This, eventually, leads to a premature loss of the battery of the node. A more efficient approach is to route packets through paths that may have higher delays but with more energy resources in order to extend the life time of the network. Another important factor to be considered is the security of the communication among nodes. The routing protocol should detect any attempt to change the LSUs in transit, reject fabricated

routing messages, avoid the creation of routing loops that lead to denial of service attacks and exclude all unauthorized nodes from the routing process.

The optimized link state routing protocol (OLSR) protocol [2] was proposed with the goal of reducing the flooding of routing messages in a network by designating specific nodes to act as multi point relay nodes (MPR). The selection is based on the hop count. The main drawback of OLSR is that it is not suitable for the dynamic link characteristics. OEDR [1] targeted the energy-delay optimization in OLSR and resulted in better performance in terms of end-to-end delay and energy efficiency. Both protocols, however, are prone to various security threats that could impede their proper operation.

Security in ad hoc networks has been extensively studied and several security extensions have been proposed for both reactive and proactive routing protocols. Some examples of reactive routing include SAODV [10] which is an extension of AODV that verifies Route Requests and Route Replies using digital signatures. Also, [15] present two approaches to improve the security of DSR. The first approach is based on Public Key Infrastructure (PKI) and the second is the Neighbor Set Detection (NSD). Each approach has an advantage over the other. The PKI can prevent the use of fabricated routing messages. On the other hand, the NSD does not depend on preexisting security mechanisms nor does it depend on resource-expensive encryption and decryption operations. Simulation results indicate a 95% probability of detecting a single attacker. As for proactive routing, the work in [4] presented a basis for identifying various threats and suggested methods for solving them using a key distribution mechanism. The idea is to use signatures and timestamps with routing messages to avoid replay attacks. Another variation of secure routing was given in [5], the secure link state routing (SLSP) is robust against individual

Byzantine attackers but remains vulnerable to colluding attackers. The work in [17] presents CSS-OLSR which ensures that the nodes in the network cooperate with each other. The core of this scheme is the penalty/reward system in which a cooperating node gets a higher rating whereas the nodes that refuse to cooperate receive a lower probability of being selected as MPR nodes. In [11], a signature system for OLSR is proposed to overcome the compromise of trusted nodes. When a trusted node is compromised, it can inject false routing messages into the network while correctly signing them. The suggested solution is to use the ADVSIG message which includes a timestamp and a signature. When the ADVSIG is received, its contents are stored in a Certiproof table for comparison with subsequent messages. A technique to mitigate the wormhole attacks is presented in [12]. Here, the nodes advertise the hashes of the packets received within the previous k intervals. A misbehaving node may be detected by comparing the packet losses with a set threshold. Another algorithm to thwart wormhole attacks is given in [13]. In this work, the nodes' geographical location is embedded in a SIGNATURE message. The extension is further fortified by the use of directional antennae instead of omniscient ones to verify the direction from which the packet was received. However, the disclosure of the nodes geographical location may prove harmful in situations where such information needs to be concealed, e.g. military operations. One suggested solution is ANODR [16], which addresses route anonymity and location privacy to prevent intruders from detecting the identities of transmitting nodes or to trace a packet flow. ANODR employs a loose definition of anonymity in which the identity of the destination is disclosed to the intermediate nodes as well as the number of hops between any intermediate node and the source. To overcome the effects of such loose definition, ASR

[14] was proposed with the goal of providing identity anonymity and strong location privacy. ASR provides identity privacy to both of the source and destination as well as the ability to secure against multiple-to-one DoS attacks when compared with ANODR.

In this work, we extend the current definition of OEDR to include trust level. This is done by adding authentication and traffic partition. With proper authentication, most of the malicious packets can be ignored without affecting the proper routing. Traffic partition is used to send the traffic in different paths with the aim of denying an intruder the chance to capture the whole stream of communication.

The paper is organized as follows, in section two we present some of the security threats in current link state routing protocols followed by a discussion of the operation of TLR in Section 3. Section 4 details the implementation of trust levels. Sections 5 and 6 detail the security and optimality analyses of TLR, respectively. Simulation results are given in Section 7 and the paper is concluded in Section 8.

2. Security Threats

A major focus is the security of the ad hoc network where the integrity of data is essential. Due to the absence of a central authority for authentication, simple network functions, such as packet forwarding, become susceptible to attacks as they are executed by the nodes on the network instead of trusted centralized routers.

To introduce trust levels, a node must examine the trustworthiness of the nodes with which it communicates before adding them to its routing table. The lack of authentication can be a serious hazard to the proper operation of the routing protocol. OEDR does not provide any security measures to guarantee the confidentiality, integrity, availability and authenticity of the data and proper routing. Because of this, malicious nodes can perform

a variety of attacks to obstruct the communication on the network. It should be emphasized, however, that the following vulnerabilities are inherent in all link state routing protocols and do not represent faults in the initial design of OEDR.

2.1 Passive Attacks

In this form of attack, a node resides within the communication range of another node to capture all the information sent. This vulnerability is especially harmful if the intruder is within the range of the original source of traffic and can only be solved using encryption. If, however, the eavesdropper resides within the range of an MPR and not the original sender then the effect can be reduced by fragmenting the traffic into different paths.

One approach for traffic partition is to use the number of packets sent through each MPR as a factor in calculating the cost of the link. As a result, with every packet sent, the cost of the link will increase until a point is reached where another link has a lower cost and the routing tables are updated to use a different MPR.

2.2 Active Attacks

These attacks can be categorized as fabrication, identity spoofing, modification, or replay. Since routing functions are performed by the nodes within the network, carrying out such attacks is easier which will result in worse consequences on the overall performance of the protocol. Moreover, the node mobility adds complexity to the design of a secure protocol. Following is an explanation of these attacks and their effect on the routing protocol.

2.2.1 Fabrication

In this form of attack, a node generates false HELLO or TC packets to cause changes in the routing tables of the nodes. This could lead to routing loops or denial of service. For the recipient, there is no way to verify the correctness of the data received.

Examples of such an attack include generating TC packets that contain either an incomplete list of the MPR or a list of imaginary nodes. Another example involves the false advertisement of bi-directional links to the nodes in its neighborhood which may results in having it selected as the MPR. At that point, the intruder can either drop or selectively forward the packets to the MPR selector. Finally, since the OEDR link cost calculation depends on the reciprocal of remaining energy in the MPR candidate, an adversary can sends a Hello packet showing that it has a large amount of energy remaining in its battery. This misleads the recipient node to calculate a low cost for the link and thus selects the malicious node as its MPR. At this point, all the traffic will be routed through this malicious node which can either drop (denial of service), selectively forward or change the contents of the packets.

2.2.2 Identity spoofing

Since no authentication takes place, a malicious node can masquerade as another node (identity spoofing). When the neighboring nodes receive its Hello packets, they will be misled to believe that the claimed node is within their range. Later on, the deluded nodes will advertise themselves as the last hop to the intruder (which they mistakenly believe to be the legitimate node). This would result in conflicting information in the network as well as denial of service.

2.2.3 Modification

MPR nodes are responsible for forwarding the packets to other nodes. While doing that task, a malicious MPR may change the payload or even change the destination field before transmitting the packet to the next hop. Another form of modification is to change the packet sequence number to match one that was previously used, resulting in a packet drop.

2.2.4 Replay

A malicious node may hold copies of LSU packets that were sent earlier and retransmit them at a later time to poison the routing tables of the recipients with incorrect routing information. Although the destination nodes check the sequence number of any received packet to avoid duplicates, replay attacks can succeed by simply changing the packet sequence number to a higher value.

These attacks can render the OLSR and OEDR protocols ineffective. Security extensions are necessary in order to ensure safe transfer of data which is discussed next.

3. Trust Level Routing

TLR is a proactive link state routing protocol. Its operation is table driven through periodically exchanging topology information with other nodes in the network. The objective of the protocol is to give the same functionality of OEDR as well as providing guarantees of the integrity, timeliness and authenticity of the packets. The proposed protocol follows the lines of OEDR. However, the routing criteria differ for selecting the MPR nodes. There are several metrics to be considered:

Energy consumed per packet: For this metric, the best path is selected based on the least consumed total energy. Any packet going from source n_1 to destination n_k through some intermediate nodes will consume

$$E_t = \sum_{i=1}^{k-1} e(i, i+1) \tag{1}$$

where E_t is the total energy consumed and $\mathbf{e}_{(i,\ i+1)}$ is the energy consumed to send the packet from n_i to n_{i+1}.

Delay per packet: Similar to the energy metric, the goal is to find the path with the least total delay. For a packet going from node n_1 to node n_k through some intermediate nodes, the total delay incurred is given by

$$D_t = \sum_{i=1}^{k-1} d(i, i+1) \tag{2}$$

where D_t is the total delay and $d_{(i,\ i+1)}$ is the time that starts when a packet enters the queue of node n_i until it reaches the queue of n_{i+1}.

It is worth mentioning that a trade off between the two metrics exists. For example, in the network shown in Figure 1, assuming that the energy consumed per each link is equal, and that node B is heavily congested such that

$$d_{A,D} + d_{D,E} + d_{E,C} < d_{A,B} + d_{B,C} \tag{3}$$

Then according to the delay metric the route A, D, E, and C will be taken instead of A, B, C. However, according to the energy metric

$$e_{A,D} + e_{D,E} + e_{E,C} > e_{A,B} + e_{B,C} \tag{4}$$

Which implies that route A, B, C should be chosen.

Furthermore, by selecting only one of the two metrics, the paths will tend to be always the same (assuming no mobility). As a result, some nodes will have high energy

consumption while others will retain their energy. Such variance in node energy can result in network partition. Thus we consider a third metric.

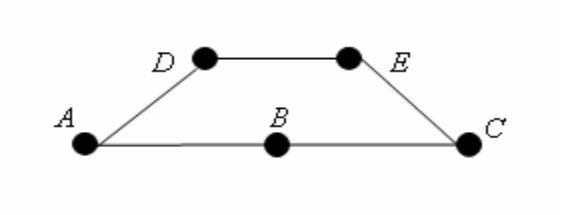

Figure 1. A simple ad hoc network consisting of 5 nodes.

Residual energy levels: This metric is used to guarantee that all nodes will have approximately equal rates of consumption by using the nodes with the highest energy levels. That is, the lower the remaining available energy in a node, the higher the cost of routing through it. The cost in this case can be taken as the reciprocal of the residual energy.

Traffic partition: This metric serves multiple purposes. First, less congestion and delay will be incurred through the intermediate nodes. Second, a higher throughput will be achieved because data packets are going through different paths. More importantly, the traffic will be fragmented into multiple paths which can potentially reduce the ability of a malicious node to capture the whole stream of traffic.

TLR depends on the following steps for proper operation: neighbor sensing, cost calculation, MPR selection, broadcast of costs and routing table calculation.

The following notation will be used:

N: set of nodes in network

s: source node

d: destination node

N_1(s): One-hop neighbors of node s

$N_2(s)$: Two-hop neighbors of node s

MPR(s): The set of nodes selected as MPRs by node s ($MPR(s) \in N_1(s)$)

$P_{x,y}$: Number of packets sent from *x* through MPR *y*.

$C_{x,y}$: cost of link between nodes x and y, where

$$C_{x,y} = w_1 * (Energy_{x \to y} * Delay_{x \to y}) + w_2 * P_{x,y} \quad (5)$$

E_x: Available energy of node *x*

3.1 Neighbor Sensing

Each node maintains a table called the neighbor table that stores information about the link status between the node and all its immediate neighbors. Information about the delay, energy consumption on each link, two-hop neighbors accessible through the immediate neighbors, the set of selected MPRs and the number of packets sent through each MPR are also stored in the table. The table is populated using the HELLO messages. Every node in the network periodically sends HELLO packets to all the nodes within its transmission range. Each message contains a list of neighbors of the originator, transmission time, energy level, and the amount of energy used for transmission.

When the message is received, an entry is added to the table (if one does not already exist). The delay is calculated as the difference between the time of reception and time of transmission (we make the assumption that the clocks are synchronized). Furthermore, the energy consumption on the link is calculated in a similar manner by comparing the energy level of the signal at the receiver to the level stamped in the HELLO message.

If the originator of the HELLO message has no entries in the neighbor table of the recipient (i.e. it just moved into the vicinity of the recipient), then a reply will be sent

back with information about all authentic nodes and the information associated with each (as will be discussed in Section 4).

3.2 Cost Calculation

Since TLR bases the selection of MPRs on a composite metric that differs from that of OLSR or OEDR, the MPR set chosen does not necessarily have to be identical for the same network topology and conditions. Moreover, if the nodes are static, the MPR set for nodes running OLSR will always be the same until one or more nodes lose their battery power or the topology changes due to node mobility. In TLR, the MPR set is dynamically changed more frequently compared to OEDR.

3.3 MPR Selection Algorithm

- Initially the MPR set MPR(s) is empty.
- First, find all the nodes in $N_2(s)$ that have a single neighbor in $N_1(s)$. Add these nodes of $N_1(s)$ to the MPR set if they are not already in MPR(s). (Because there are no other MPR candidates).
- While there exists a node in $N_2(s)$ for which MPR node is not selected, do:

  For each node in N2(s), with multiple neighbors from $N_1(s)$, select a neighbor from $N_1(s)$ as multipoint relay node which results in minimum cost from s to the node in $N_2(s)$, C_{MPR} according to (5), and add it to the MPR set if it is not already in MPR(s).

3.4 MPR and Costs Declaration

Every selected MPR will transmit LSU packets called the Topology Control (TC) packets that contain information about the MPR node's selector set (i.e. the nodes that have selected the originator of the TC message as their MPR). The TC messages, which

include the link costs between the MPR node and its selectors, are forwarded throughout the network through MPR nodes only. When a node receives a TC message, it can use the information to build a 'topology table', in which it stores the information about the topology of the network and the associated link costs. An entry in the topology table consists of the address of a destination (an MPR selector in the received TC message), address of the last-hop node to that destination (originator of the TC message), and the cost of the link between the destination and its last hop. It implies that the destination node can be reached in the last hop through this last-hop node at the given cost. [3]

In Section 4, a deeper discussion of the contents of TC packets and how to maintain their integrity is given.

3.5 Routing Table Calculation

For each node, a routing table is maintained to route packets to their destinations. Each entry contains the destination address, next-hop address, estimated distance to destination (in hops) and the total cost of the path from the source to the destination. For every known destination, an entry is added to the routing table listing the next hops to be taken. Our proposed protocol uses the least cost spanning tree method.

4. TLR Implementation

With the vulnerabilities listed in Section 2, it is clear that the operation of OLSR and OEDR would become ineffective in the presence of malicious nodes. The main requirements of security, i.e. origin authentication, timeliness and ordering, and data integrity are missing in the original implementations of both protocols. Every LSU packet should hold enough information to clearly prove that it originated from the claimed source. Furthermore, late control packets should be dropped to avoid replay attacks.

Finally, a mechanism to detect any alteration of LSU en route is needed. In this work, we propose two methods for adding trust levels in the routing protocol: traffic partition and authentication.

4.1 Traffic Partition

In this method, the traffic transmitted from source A to destination B is routed through different paths by continuously switching between different MPR candidates. This provides confidentiality and forbids eavesdropping in the presence of a single eavesdropping node. With this approach, each path contains only partial information of the data stream. Consequently, an intruder will not be able to reconstruct the whole flow. From (5), it is evident that as the number of data packets increases on a certain link, the cost of using that link will increase until the total cost becomes higher than that of another link. At this point, the MPR list of the source will be updated to force a route change.

This scheme would not be sufficient, however, in the presence of multiple eavesdropping nodes that may monitor the different selected paths and combine the data for analysis.

4.2 Authentication and Timestamps

Traffic partition provides a reasonable level of security when combined with a lightweight encryption algorithm. It is helpful in reducing the risk of passive attacks by limiting the ability of one intruder to analyze the data stream completely and in a timely manner. However, it has little potential in overcoming or even detecting active attacks. Authentication of the LSU source can be used to counter active attacks. Assuming that the authentic nodes in the network have a mechanism for sharing the encryption keys, node A uses the shared key and the contents of the LSU to calculate a Message

Authentication Code (MAC) which is appended to the LSU packet. When node B receives the packet, it calculates the MAC again using the same secret key and the body of the received packet (excluding the MAC). Node B accepts the packet if the resulting MAC equals the MAC field of the packet. If, however, a malicious node alters the message but does not alter the MAC (because it does not have the secret key), then node B can easily detect the changes in the packet and drop it.

This guarantees that the packet was sent by one of the authentic nodes because only they have the shared key and that the payload of the control packet was not changed en route. The scheme above, by itself, is not enough for authentication. Consider a scenario where node A sends control messages to node B. Assuming that a malicious node M intercepts the control message, it could wait for a random period of time and then retransmit the same packet. When node B receives the replayed message, it checks the hash code and accepts the packet as authentic. This will cause inconsistencies in the routing table of node B.

Timestamps can be used to overcome this problem. Whenever a node sends a packet, it adds the time of transmission as a field and includes that in the MAC calculation. This enables the recipients to check the time of transmission.

We propose the use of one way hash functions to create a hash chain analogous to that given in [6]. The main characteristic of a hash function is that it is easy to compute in the forward direction but computationally infeasible to find its inverse. That is, given g = h(y) where h is the secure hash function whose input is y, then it is easy to find g given y. However, the reverse process of finding ý such that h(ý) = g is too expensive to be

practical. A third condition to be satisfied is it is difficult to find two values x and y such that h(x) = h(y), but x ≠ y.

In this method, every node selects a random seed value S and computes $g_1 = h(S)$ then it computes the next value in the chain as $g_2 = h(g_1)$. The whole sequence of hash values is computed iteratively until the last value of the chain is given.

$$g_1 = h(S), g_i = h(g_{i-1}) \quad (6)$$

where $2 < i \leq m$ and m is the length of the sequence.

The source node, A, starts using the hash chain in the backward direction (i.e. it starts with g_m followed by g_{m-1}, g_{m-2} … etc). If a receiver knows the value of g_i (where $2 < i \leq m$) and has guarantees that it is authentic then the next packet coming from node A can be checked for authenticity because it must be stamped by g_{i-1}. The receiver needs to check that $h(g_{i-1}) = g_i$ as a proof of authenticity.

4.3 Secure Broadcast of g_m

We assume that every node in the network has a mechanism to verify the public keys of all other nodes. We further assume that all the clocks of the nodes are synchronized (this assumption is needed for the proper calculation of delays in the OEDR protocol). It is worth mentioning that the use of public key cryptography is only applied on the control packets which comprise a smaller part of the whole packets being communicated.

The broadcast of the last calculated value in the chain g_m is done using a new packet called the C*hain tip packet*. The format of the packet is shown in Figure 2.

The packet has four fields; the identifier of the sending node, a timestamp, the current iteration (CI) field (which must be initially set to zero), and the last field that holds the

encrypted value of the concatenation of the previous three fields and g_m. When the packet is received, it will be decrypted using the public key of the sender.

$$\mathbf{E_{KU}\,[C] = E_{KU}\,[E_{KR}[\ Node\ ID\ ||\ timestamp\ ||\ Chain\ Length\ (m)||\ g_m\ ]}$$

$$\mathbf{= Node\ ID\ ||\ timestamp\ ||\ Chain\ length\ (m)\ ||\ g_m}$$

Node ID
Timestamp
Current iteration (Zero)
$C = E_{KR}[$ Node ID $\|\|$ timestamp $\|\|$ Chain Length (m) $\|\|$ g_m $]$

Figure 2. Chain tip packet format.

After decryption, the Node ID field will be compared with the decrypted value to make sure that it originated from the claimed source. The timestamp field proves that the message is "fresh" and that it is not a replayed message. The chain length field informs all recipient nodes about how many iterations of the hash function had been calculated to construct the chain, thus allowing them to know when the last value in the chain has been reached. Finally, g_m will be stored in the memory along with its associated Node ID.

Note, however, that any node that moves into the neighborhood after g_m has been delivered will not be able to authenticate the received packets. This can be solved by introducing a *Hello_response* packet that is sent back from every node that receives the HELLO packet. (Refer to Section 3.1).

The Hello_response should contain the same information given in the Chain tip packet with the only difference that the CI field will be set to the number of chain elements already received. For example, if a node received g_m, g_{m-1}, …, g_{m-i} , then it will set the CI field to hold the value of i before transmitting the Hello_response packet. Thus, the

originator of the HELLO packet will receive the same information that it would have received through a chain tip packet and will also have proof that it was indeed sent from the claimed source (No other node can forge it). Furthermore, by knowing the CI field, the node can check for the authenticity of the any TC packet it receives by calculating the hash function on the received chain value i+1 times.

$$h^{i+1}(g_{m-i-1}) = g_m \tag{7}$$

After that, the new node will have all the information that is available to all other nodes.

To avoid multiple copies of Hello_response packets, each node waits for a random time before replying back with its Hello_response while overhearing the responses of other nodes. If any matching Hello_response packet is overheard then the node backs off and does not transmit, otherwise it sends its own version of the Hello_response.

4.4 Broadcasting TC Packets

The main purpose of using authentication is to guarantee that the TC packet was generated by the claimed source and that the contents were not changed in transit. Following is a description of how the hash chain can be applied.

- Select a new random seed and calculate the hash chain.
- Broadcast g_m
- For $1 \leq i < m$
- Find the message digest of the concatenation of the node ID, timestamp, message, and g_{m-i}
- Transmit the node ID, timestamp, message and the calculated hash value.

- Wait for a set period of time to make sure that the TC message reached all destinations in the network then transmit g_{m-i} to update the current hash value being used.
- Increment i
- If (i = m) then the chain has been consumed and a new chain has to be created (step a.).
- Otherwise, repeat step c. for the next TC packet.

When a node receives the TC packet, it has no way to calculate the message digest because g_{m-i} is not known yet. The node must wait until it receives the value of g_{m-i} to verify the authenticity and integrity of the latest received TC packet.

Note that this update packet must be received within a protocol specific period of time after the TC packet to guarantee that none of the intermediate nodes held the TC packet until g_{m-i} was released. This is an essential condition without which any malicious node can modify the contents of the packet and calculate a new message digest before sending the TC packet followed by the value of g_{m-i}. The recipient, in this case, will not be able to detect the changes.

For that reason, the recipient node compares the current time with the timestamp. If it finds that updating with the value of g_{m-i} took more time than expected then it drops the TC packet.

5. Security Analysis of TLR

In this section, we analyze the ability of TLR to limit various attacks.

5.1 Replay Attacks

An adversary may hold old copies of TC packets to transmit them at a later instance of time. This would result in conflicting information in the routing tables since either the topology or the MPR nodes would have changed. TLR mitigates this threat with the use of a timestamp in packets which is further enforced by the hash chain.

5.2 Identity Spoofing and Link Spoofing

Identity spoofing involves a node using an ID that does not belong to it whereas link spoofing attacks occur when a node sends out incomplete or forged information about its links. The presence of the mechanism for verifying the keys of other nodes limits the ability of an attacker to attempt identity spoofing. Furthermore, in normal cases, only the packets signed by trusted nodes are accepted and all others are rejected thus a malicious node can not run a link spoofing attack since it packets will not be accepted.

5.3 Modification Attacks

An adversary may change the contents of TC packets in an attempt to add, delete or alter the entries of the routing tables. This may result in routing loops or dropped packets due to incomplete routes. In TLR, modifying the contents of a TC packet will be detected since the intruder has no access to the held hash value. A malicious node trying to relay a modified TC packet will need to wait for that unknown hash value which would make it too late for it to transmit its modifications since that packet would be dropped.

5.4 Passive Attacks

As mentioned earlier, a node may listen in to capture the data stream. If the node is overhearing the source itself, then only encryption may protect the data. However, if the

eavesdropper is positioned around one of the MPR nodes then TLR can be helpful by partitioning the data stream through different paths. By doing so, the intruder will only have a part of the data stream. With multiple cooperating intruders, the stream may be gathered in full and the protection of encryption is the last line of defense.

It is implicitly assumed that all the nodes that were selected as MPRs would cooperate in relaying the packets to the destination. In the case of a compromised node, some packets may be dropped instead of being relayed. TLR can not detect this but due to its frequent topology updates it may limit this attack by switching to different MPR nodes. The work in [17] provides a protocol that addresses this problem in specific.

We give a comparison between TLR, SLSP [5], and CSS-OLSR [17] in Table 1. The comparison is based on the ability of each of the respective protocols to successfully limit the effects of an attack.

Table 1. Comparison between SLSP, CSS-OLSR, and TLR.

	SLSP	CSS-OLSR	TLR
Replay	NO	YES	YES
Identity Spoofing	YES	YES	YES
Link Spoofing	YES	YES	YES
Modification	YES	YES	YES
Traffic Relay Refusal	NO	YES	YES
Evesdropping	NO	NO	YES

6. Optimality Analysis of TLR

Theorem 1: Only authentic nodes can be selected as MPR nodes.

Each packet received is checked for authenticity of source and content. Non-authentic packets are dropped by the recipient and no entries are made in the neighbor table. Hence, non-authentic nodes do not qualify as MPR candidates.

Theorem 2: The MPR selection will result in a trusted optimal route between the source and destination with added trust levels only if there are multiple MPR candidates.

Case I: If a node in $N_2(s)$ has only one neighbor from $N_1(s)$, then that single neighbor will be selected as the MPR. This MPR will be selected always in an optimal route but there will be no traffic partition.

Case II: If a node in $N_2(s)$ has multiple neighbors in $N_1(s)$, the MPR selection will follow the cost function to determine the path with the least cost. Since the costs are dynamic due to the nature of the network and traffic sent, the paths selected will be dynamic to allow for traffic partition in addition to the energy-delay considerations.

Assume that a source node s that has multiple one-hop neighbors in $N_1(s)$ needs to reach a node d in $N_2(s)$ that has multiple neighbor nodes n_1 , n_2 , …, n_k ($k>1$) belonging to $N_1(s)$. Let the cost to reach any of these neighbors from s be $C_{s,ni}$ ($i=1,2,\ldots k$) and the cost to reach d from n_i is given by $C_{ni,d.}$ The MPR node between s and d is selected as the node n_i with the minimum cost $\text{Min}\{(C_{s,n1} + C_{n1,d}), (C_{s,n2} + C_{n2,d}), \ldots , (C_{s,nk} + C_{nk,d})\}$. The cost values of $C_{s,ni}$ change frequently and a different MPR is selected. Consequently, the MPR selection of TLR will result in trusted optimal routes with trust levels from s to its two-hop neighbors in $N_2(s)$.

Lemma 1: All intermediate nodes on the trusted optimal path are selected as multipoint relays by the preceding nodes on the path.

Proof: To be selected as an MPR, a node has to prove its authenticity, provide

connection between the source node and its two-hop neighbors and have the lowest link cost.

Case I: The node in $N_1(s)$ of the source node s, does not provide connection to any node in $N_2(s)$.

In Figure 3, node n_2 has no direct connection to node d. The two possible paths from s to d are s-> n_1 ->d and s-> n_2-> n_1 ->d.

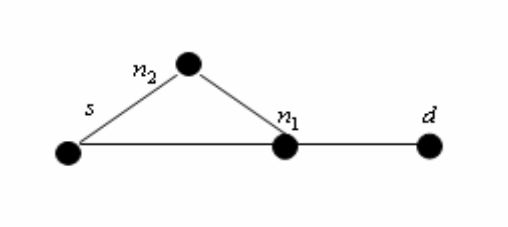

Figure 3. Case I.

Considering the added delay and energy consumption, it is clear that n_2 is not on the optimal path from s to d.

Case II: The node n_2 in $N_1(s)$ of the source s does not provide proof of authenticity to any node in $N_2(s)$. Any packet received from it by any node in the $N_2(s)$ will be dropped.

Case III: There is a trusted optimal path from source to destination such that all the intermediate nodes on the path are selected as MPRs by their previous nodes on the same path.

In Figure 4, suppose that in an optimal path, s -> n_1 -> n_2 -> … -> n_k -> n_{k+1} .-> … -> d, both MPR and non-MPR nodes exist. Also, based on the result of Cases I and II, we suppose that for each node on the path, its next node on the path is its one-hop neighbor, and the node two hops away from it is its two-hop neighbor.

1) Suppose that on the optimal route, the first intermediate node n_1 does not meet the criteria for MPR selection by source s. However, n_2 is the two-hop neighbor of s. Based

on the basic idea of MPR selection, every two-hop neighbor of s must be covered by its MPR set, then s must select another neighbor as its MPR. In this case, n_1' is selected as the MPR to cover node n_2.

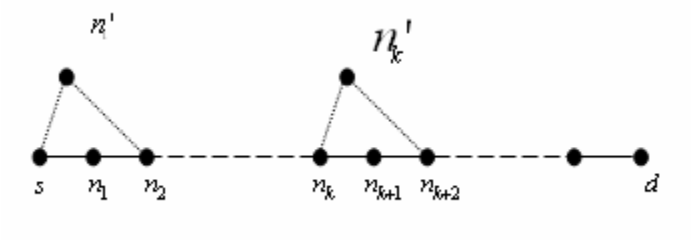

Figure 4. Case III.

Since route s -> n_1 -> n_2 -> … -> d is a trusted optimal path then s -> n_1' -> n_2 -> … -> d is also a trusted optimal path. Thus, the MPR is a part of the optimal path.

2) Assume that on the optimal route s -> n_1 -> n_2 -> … -> n_k -> n_{k+1} .-> … -> d, all the nodes on segment n_1 -> … -> n_k -> are chosen as MPR by their previous node, we now prove that the next hop node of n_k is on the optimal route is an MPR.

Suppose that n_{k+1} is not an MPR of n_k. Same as in the previous situation, n_{k+2} is the two-hop neighbor of n_k, so it must have another neighbor n_{k+1}' which covers n_{k+2}. Since route s -> n_1 -> n_2 -> … -> n_k -> n_{k+1} .-> … -> d is an optimal path the s -> n_1 -> n_2 -> … -> n_k -> n_{k+1}' .-> … -> d is also an optimal path because it has a lower cost.

This implies that in an optimal route, the $(k)^{th}$ intermediate node selects it $(k+1)^{th}$ as the MPR.

Based on I and II, all the intermediate nodes of an optimal path are MPRs of the previous node.

Theorem 3: For all pairs of nodes s and d, if s transmits a broadcast packet P, d will receive a copy of that packet.

Proof: The proof follows on similar lines to [1]. Let k be the number of hops to d from which a copy of packet P has been retransmitted. We shall prove that there exists a minimum k=1, i.e., a one-hop neighbor of d which eventually forwards the packet.

Assume that n_k ($k \geq 2$) forwards the packet P to node d. Assume there exists a path n_k -> n_{k-1} -> n_{k-2} -> …-> n_2 -> n_1 -> d.

Based on Lemma 1, any packet received by n_{k-1} from n_k must be relayed to n_{k-2}. Similarly, when n_{k-2} receives the packet, it forwards it to n_{k-3}. This repeat until node n_1 receives the packet where it is automatically forwarded to node d.

7. Simulation Results

To simulate the protocol modifications, the NS2 implementation of OEDR was extended to reflect the changes in cost calculation. A new table was added in each node to hold the number of packets sent on each link. The OEDR implementation in NS2 was modified by adding definitions for the malicious nodes and providing a mechanism for authentication. Every packet received is checked in the MAC layer, and any packet from a non-authentic source is dropped. This means that no entries will be added in the one-hop or two hop tables.

The simulation scenarios were based on networks of 50 and 200 nodes. The data rates varied from 128 kbps to 4096 kbps with a packet size of 512 bytes. The nodes were stationary in an area of 1000 x 1000 meters, their locations and flows were randomly generated. The performance of the OLSR, OEDR and TLR was compared based on the end-to-end delay and the energy-delay product.

Figures 5 and 6 display the average end-to-end delay for data packets on networks of 50 and 200 nodes, respectively. The delay in TLR is similar to that of OEDR. In some cases,

however, it is less and that is due to the implicit congestion avoidance in TLR. When a sequence of packets is sent through a path, the congestion in the intermediate nodes will increase. Since TLR sends data through different paths, it avoids causing congestion in the intermediate nodes, thus reducing the average delay.

The figures also show that the delay for OLSR is higher; this is because OEDR and TLR consider the delay as a factor in choosing the routes whereas OLSR just selects the smallest set of possible MPRs.

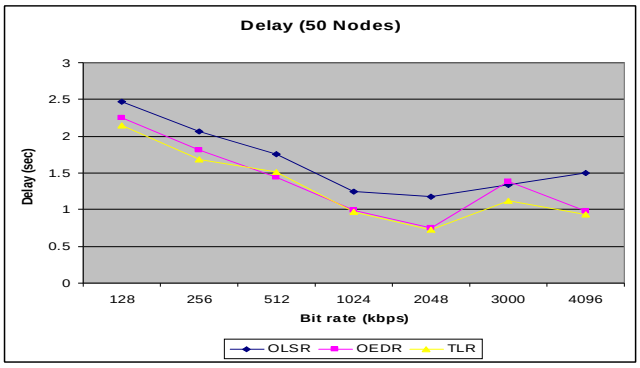

Figure 5. Delay for the three routing protocols in a 50 node network.

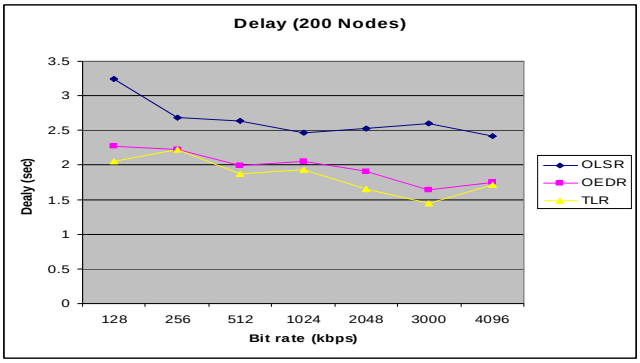

Figure 6. Delay for the three routing protocols in a 200 node network.

In Figures 7 and 8, the energy-delay per packet is given for networks of 50 and 200 nodes, respectively. OLSR always has a higher energy-delay product compared to OEDR and TLR. From the figures, we also note that OEDR performs better in terms of

this metric. This is explained by the introduction of the packet count as a metric which forces the nodes to select some MPR nodes that are not optimal in terms of energy consumption and link delay.

By comparing Figure 7 with Figure 8, we find that the energy-delay metric increases as the number of nodes increases. This is due to the higher amount of traffic flowing in the network and the use of different paths with more intermediate hops (i.e. more energy consumption).

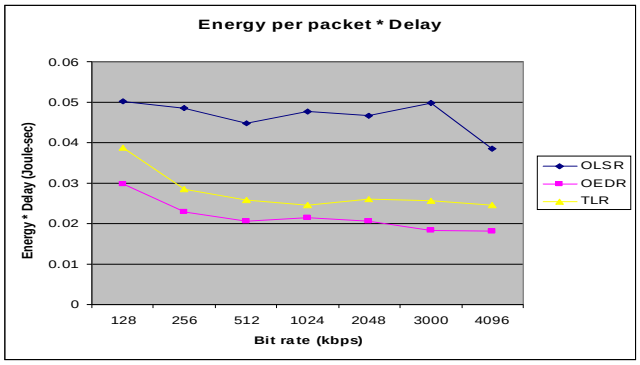

Figure 7. (Energy* Delay) for the three routing protocols in a 50 node network.

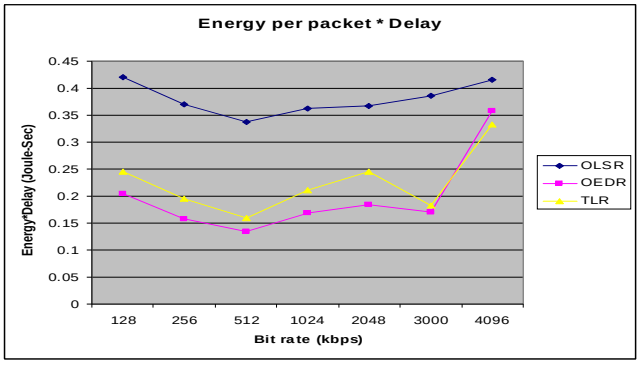

Figure 8. (Energy* Delay) for the three routing protocols in a 200 node network.

To demonstrate the effect of changing the weights in (5) on the delay and the (Energy * Delay), we simulated the same network of 50 nodes running the TLR protocol for three cases ({w1,w2} = {1.0, 0.0},{0.5,0.5},{0.25,0.75}). From Figure 9, when w1 decreases

from 1.0 to 0.5 we notice that the delay decreases, this is because of the inherent congestion avoidance as we mentioned earlier. However, when w1 decreases from 0.5 to 0.25 the effect of that is an increase in the delay because the paths being selected are not necessarily the optimal paths in terms of delay, rather they are mostly selected based on the number of packet flowing through each MPR.

Figure 10 shows that the energy times delay factor will increase as the values of w1 decreases. This is because the MPRs will be selected mainly based on the number of packets going through them.

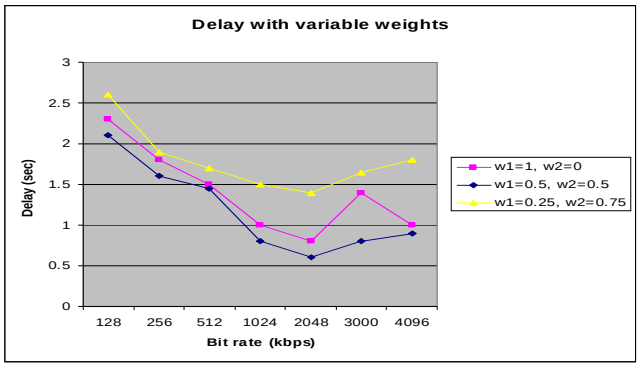

Figure 9. Delay for the TLR with variable weights in a 50 node network.

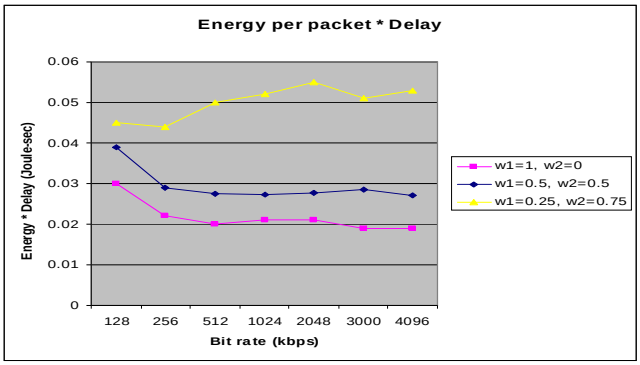

Figure 10. (Energy* Delay) for the TLR with variable weights in a 50 node network.

8. Conclusions and Future Directions

With the rapid deployment of wireless networks, security of the routing protocols is essential for reliable operation. The threats presented in this paper indicate that more work is needed to guarantee the privacy and integrity of the data. This is especially important in military and safety critical environments.

TLR, an extension of the OEDR protocol, resulted in better management of route selection for security purposes. The simulation results indicate that TLR delivered the packets with a noticeable decrease in the average end-to-end delay. This, however, increased the power consumed when longer routes were selected.

The addition of the authentication model in NS2 demonstrated how the TLR protocol dropped non-authentic control packets. Nevertheless, more work needs to be done to improve the model to enable the analysis of the computational overhead involved in computing the hash fields as well as the bandwidth utilized for the additional bytes inserted into the control packet in the form of the hash code.

Another modification to be investigated is the weight calculation given in (5). A dynamic model that allows assigning different weights to each factor would be more suitable in cases where, for example, delay is given higher priority than energy consumption.

9. References

[1] N. Regatte, and J. Sarangapani, “Optimized Energy-Delay Routing in Ad Hoc Wireless Networks,” Proceedings of World Wireless Conference, San Francisco, CA, May, 2005.

[2] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," pp. 62–68, IEEE International Multi Topic Conference on Technology for the 21st Century, IEEE INMIC'01, December, 2001.

[3] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," pp. 3866 – 3875, 35th Annual Hawaii International Conference on System Sciences, HICSS, January, 2002.

[4] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, and D. Raffo, "Securing the OLSR Protocol," Proceedings of Med-Hoc-Net, Mahdia, Tunisia, June, 2003.

[5] P. Papadimitratos, and Z. Haas, "Secure Link State Routing for Mobile Ad Hoc Networks," IEEE Workshop on Security and Assurance in Ad Hoc Networks, 2003.

[6] L. Lamport, "Password Authentication with Insecure Communication," Communications of the ACM, 24(11):770-772, November 1981.

[7] P. Michiardi, and R. Molve, "Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks," European Wireless Conference, 2002.

[8] K. Rawat, and G. Massiha, "Secure Data Transmission Over Wireless Networks: Issues and Challenges," IEEE Region 5, 2003 Annual Technical Conference, 2003.

[9] B. Dahill, B. Levine, E. Royer, and C. Shields, "A Secure Routing Protocol for Ad Hoc Networks," Technical Report UM-CS-2001-037, Electrical Engineering and Computer Science, University of Michigan, August, 2001.

[10] M. Zapata, "Secure Ad Hoc On-demand Distance Vector (SAODV) Routing," Internet Draft, draft-guerrero-manet-saodv-00.txt, October, 2002.

[11] D. Raffo, C. Adjih, T. Clausen, and P. Muhlethaler, "An Advanced Signature System for OLSR," Proceedings of the 2nd ACM workshop for security on ad hoc and sensor networks, New York, USA, 2004.

[12] C. Adjih, D. Raffo, and P. Muhlethaler, "Attacks Against OLSR: Distributed Key Management for Security," 2005 OLSR Interop and Workshop, Palaiseau, France, July, 2005.

[13] D. Raffo, C. Adjih, T. Clausen, and P. Muhlethaler, "Securing OLSR Using Node Locations," Proceedings of 2005 European Wireless (EW 2005), Nicosia, Cyprus, April, 2005.

[14] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng, "Anonymous Secure Routing in Mobile Ad Hoc Networks," Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN 2004), pages 102–108. IEEE, 2004.

[15] B. Sun, K., Wu, and W. Pooch, "Secure Routing against Black-hole Attack in Mobile Ad Hoc Networks," Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN02), Cambridge, USA, Nov., 2002.

[16] J. Kong, and X. Hong, "ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad Hoc Networks," Proceedings of the fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '03), Annapolis, Maryland, USA, June, 2003.

[17] J. Vilela and J. Barros, "A Cooperative Security Scheme for Optimized Link State Routing in Mobile Ad-hoc Networks," Proceedings of the 15th IST Mobile and Wireless Communications Summit, Mykonos, Greece, June, 2006.

II. Survivability and Reliability Analysis of the Trusted Link State Routing Protocol for Wireless Ad Hoc Networks

Eyad Taqieddin, Ann Miller, S. Jagannathan

Department of Electrical and Computer Engineering
University of Missouri – Rolla
Rolla, Missouri 65409
{eyad, milleran, sarangap}@umr.edu

ABSTRACT

The nodes in a wireless network are responsible for both sending their traffic as well as relaying the traffic of other nodes in the network. This form of collaboration between the nodes is essential for the proper delivery of data. Without fair participation of all nodes in the routing process, some nodes may lose their energy reserves at a high rate compared to other nodes in the network. However, bandwidth and energy are not the only issues in wireless networks; survivability and reliability are critical as well. Our focus in this work is on two link state routing protocols; OLSR and TLR. We study the effect of using these protocols on the survivability and the reliability of wireless networks. Both analytical and simulation work show that TLR results in better performance due to the inherent energy aware approach and the traffic partition used to reduce congestions in the network.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design – *wireless communication;* C.2.2 [**Computer Communication Networks**]: Network Protocols – *routing protocols.*

General Terms

Performance, Reliability, Security.

Keywords

Reliability, Survivability, Energy, Delay, Wireless Networks.

1. INTRODUCTION

In mobile ad-hoc networks, a set of nodes collaborate with each other to guarantee proper communication between any pair of nodes in the network. This is done with no centralized entities to monitor the operation. Due to the dynamic topology, the links in the network are established and broken continuously depending on the velocity, direction and transmission range of the nodes. Unlike wired networks, the nodes in an ad hoc network have no fixed infrastructure and are limited in communication range. Intermediate nodes need to forward the packets in the case where the destination node is not within the transmission range of the source.

Efficient routing in ad hoc networks is needed to properly deliver the packets to their intended destinations with minimal delay and to extend the operational lifetime of the network through energy efficient selection of the routes.

Many energy aware routing algorithms were proposed with the aim of improving the survivability of the network by routing packets from their sources to the destinations with the minimal amount of energy. The Minimum Battery Cost Routing (MBCR) is presented in [8]. This algorithm selects the routes with the maximum total capacity. As the name implies, the routes with the least battery cost among all candidate routes are selected. Another algorithm is Simple Energy Aware Dynamic Source Routing (SEADSR) [3]. This is an extension of DSR in which a node, upon receiving the RREQ, incorporates a

delay that is proportional to its energy level. This is done because DSR, by design, does not use cost functions in route selection. The authors in [5] present the Optimal Energy-Delay Routing protocol (OEDR), an algorithm based on finding a tradeoff between delay and energy consumption in LSR and compare that to OLSR. Paths are selected based on the least incurred delay and energy consumed. The Power Aware Multi-Access with Signaling (PAMAS) [6] is a protocol that efficiently uses energy by powering off nodes that are not sending or receiving packets. In [1], a detailed analysis of the network behavior both in steady and transient states is given. When the routing nodes move out of the range, then the packets in transit will either be dropped or delayed and this affects the performance of the network. Another analysis that is given in [9] studies the effect of mobility on the availability of the paths. Both [1] and [9] use a the Continuous Time Markov Model (CTMC) to model the connection availability for a two hop ad hoc network with respect to node mobility and failures.

In [7], we proposed a Link State Routing protocol based on the Optimized Link State Routing (OLSR) [4]. Our proposed protocol, the Trust Levels Routing protocol (TLR), adds security measures to counter the vulnerabilities in OLSR and to better consume the energy resources of the nodes in the network. In this work, we will analyze both OLSR and TLR in terms of survivability and reliability to emphasize the improved performance of the network in terms of lifetime and proper delivery of data. Our focus is to explain analytically the achieved improvements in the energy consumption and its effect on the survivability and reliability. The analysis is further supported by simulation scenarios.

The paper is organized as follows, in Section 2 we present an overview of OLSR and TLR. Sections 3 gives a discussion of the survivability analysis of TLR followed by the

reliability analysis in Section 4. Simulation results are given in Section 5 and the paper is concluded in Section 6.

2. OVERVIEW OF OLSR AND TLR

The routing protocol should focus on energy conservation to increase the lifetime of the nodes while choosing routes with the least delay, jitter and congestion. Occasionally, the best routes for several sources, in terms of delay, go through the same node whose energy gets consumed at a higher rate compared to other nodes. This, eventually, leads to a premature loss of the battery of the node. A more efficient approach is to route packets through paths that may have higher delays but with more energy resources in order to extend the lifetime of the network.

OLSR and TLR are proactive link state routing protocols. Their operation is table driven through periodically exchanging topology information with other nodes in the network. The cornerstone in the operation of both protocols is the use of Multi Point Relay nodes which limit the flooding in the network to a specific set of nodes that are chosen according to the dynamics of the network. In OLSR, the algorithm searches for the smallest set of MPR nodes, i.e. it selects the nodes that cover the most two hop neighbors. As for TLR, the MPR set is selected based on a tradeoff between the delay incurred, energy consumed and load balancing as will be explained later. In addition to energy conservation, TLR provides guarantees of the integrity, timeliness and authenticity of the packets. Our proposed protocol uses the following metrics when considering the selection of the MPR nodes.

Energy consumed per packet: For this metric, the best path is selected based on the least consumed total energy. Any packet going from source n_1 to destination n_k through some intermediate nodes will consume

$$E_t = \sum_{i=1}^{k-1} (e^t_{(i,i+1)} + e^r_{(i,i+1)}) \quad (1)$$

where E_t is the total energy consumed and $\mathbf{e}^t_{(i,\ i+1)}$ is the energy consumed to send the packet from n_i to n_{i+1} and $\mathbf{e}^r_{(i,\ i+1)}$ is the energy consumed by n_{i+1} when receiving a packet from n_i .

Delay per packet: Similar to the energy metric, the goal is to find the path with the least total delay. For a packet going from node n_1 to node n_k through some intermediate nodes, the total delay incurred is given by

$$D_t = \sum_{i=1}^{k-1} d_{(i,\ i+1)} \quad (2)$$

where D_t is the total delay and $d_{(i,\ i+1)}$ is the time that starts when a packet enters the queue of node n_i until it reaches the queue of n_{i+1}. Note that a trade off between the two metrics exists.

Residual energy levels: This metric is used to guarantee that all nodes will have approximately equal rates of consumption by using the nodes with the highest energy levels. That is, the lower the remaining available energy in a node, the higher the cost of routing through it. The cost in this case can be taken as the reciprocal of the residual energy.

Traffic partition: This metric serves multiple purposes. First, less congestion and delay will be incurred through the intermediate nodes. Second, a higher throughput will be

achieved because data packets are going through different paths. More importantly, the traffic will be fragmented into multiple paths which can potentially reduce the ability of a malicious node to capture the whole stream of traffic.

Since TLR bases the selection of MPRs on a composite metric that differs from that of OLSR, the MPR set chosen does not necessarily have to be identical for the same network topology and conditions. Moreover, if the nodes are stationary, the MPR set for nodes running OLSR will always be the same until one or more nodes lose their battery power. In TLR, the MPR set is dynamically changed more frequently based on the MPR routing criteria.

TLR is enforced with security mechanisms to overcome some of the security threats that may impede the operation of the network. The control packets are signed and the data stream is encrypted with using a lightweight encryption algorithm.

2.1 MPR Selection Algorithm

The following notation will be used

N: set of nodes in network

s: source node

d: destination node

$N_1(s)$: One-hop neighbors of node s

$N_2(s)$: Two-hop neighbors of node s

MPR(s): The set of nodes selected as MPRs by node s ($MPR(s) \in N_1(s)$)

$P_{x,y}$: Number of packets sent from *x* through MPR *y*.

$C_{x,y}$: cost of link between nodes x and y, where

$$C_{x,y} = w_1 * (Energy_{x \to y} * Delay_{x \to y}) + w_2 * P_{x,y} \quad (3)$$

Initially the MPR set MPR(s) is empty.

1. First, find all the nodes in $N_2(s)$ that have a single neighbor in $N_1(s)$. Add these nodes of $N_1(s)$ to the MPR set if they are not already in MPR(s). (Because there are no other MPR candidates).

2. While there exists a node in $N_2(s)$ for which MPR node is not selected, do:

    a. TLR: for each node in N2(s), with multiple neighbors from $N_1(s)$, select a neighbor from $N_1(s)$ as multipoint relay node which results in minimum cost from s to the node in $N_2(s)$, C_{MPR} according to (3), and add it to the MPR set if is not already in MPR(s)

    b. OLSR: for each node in N2(s), with multiple neighbors from $N_1(s)$, select a neighbor from $N_1(s)$ as multipoint relay node which covers that maximum number of two hop neighbors in $N_2(s)$ and add it to the MPR set if is not already in MPR(s)

3. SURVIVABILITY ANALYSIS OF TLR

The survivability of the network is directly related to the lifetime of nodes in the network. Clearly, the energy in every node in the network has to be properly utilized in order to avoid premature depletion of energy of the nodes and to minimize the variance in energy levels between them. The power consumption is due to 1) transmission of a packet; 2) reception of a packet 3) retransmission of packets due to congestion 4) power used when a node is idle.

Given a network modeled as a graph G = (N,L) where N is the set of nodes and L is the set of links (i,j) ∈ N. A link (i,j) exists if and only if j ∈ N_1(i). The lifetime of a node in the network running the OLSR protocol is given as

$$LT_i = \frac{Ei}{(1+\gamma)\sum_{k \in N1(i)} e_{tx} r_{i->k} + \sum_{j:i \in N1(j)} e_{rx} r_{j->i} + \varepsilon} \quad (4)$$

where,

E_i Energy level of node i at t = 0 (Joule)

e_{tx} Energy consumed to transmit data (Joule/bit)

e_{rx} Energy consumed to receive data (Joule/bit)

$r_{i->j}$ Data rate between source i and destination j (bit/sec)

ε Energy consumed during the idle time (Joule/sec)

γ Factor representing the needed retransmissions

As for a node running TLR, the lifetime is

$$LT_i = \frac{Ei}{(\alpha+\beta\gamma)\sum_{k \in N1(i)} e_{tx} r_{i->k} + \sum_{j:i \in N1(j)} e_{rx} r_{j->i} + \varepsilon + \mu} \quad (5)$$

with

μ Energy used for overhead operations in TLR (encryption functionality).

The total energy used for transmission of packets and the total energy to compensate for repeated packets due to congestion are multiplied by factors α and β, respectively. This is a result of the load balancing in TLR which results in a node being used less frequently compared to the case of OLSR. Moreover, load balancing results in less congestion

leading to fewer dropped packets. Note that the factors α and β are inversely proportional to the number of available MPR candidates.

TLR outperforms OLSR in terms of energy conservation by distributing the load on various intermediate nodes. This leads to a relatively smaller variance in the battery levels and longer lifetime for the whole network.

To illustrate, consider the network given in Figure 1 where nodes S1 and S2 send traffic through nodes A, B and C to the destinations D1 and D2, respectively. Assuming that each of nodes S1 and S2 has 50 packets to send and also assume that each of nodes A, B and C has sufficient energy to receive and forward 100 packets only. With OLSR, the node that covers the maximum number of hops will be selected as the MPR. In this case, node C will be selected because it covers 2 nodes (D1 and D2) whereas nodes A and B only cover D1 and D2, respectively. Thus, both nodes S1 and S2 will send all their traffic through C which will result in consuming all it energy resources after sending 50 each (100 in total). After that, S1 will select A as the MPR whereas S2 will select B. In this scenario, node C lost all its energy prematurely whereas nodes A and B still maintained their initial energy reserves. If either node A or B moves out range then network will be partitioned due to the lack of available paths.

On the other hand, when TLR is used with the same network topology using the initial link costs given in Figure 1, node C will be initially selected as the MPR because it has the least cost for both S1 and S2. After forwarding 20 packets, the link cost for S1-C and S2-C will be 3.5. Thus, on the next topology update, S1 will switch to node A for forwarding whereas S2 will maintain the same link S2-C. After transmitting 20 more packets, the updated link costs will be (S1-A = 4.33, S1-C = 2.86, S2-C =3.86, S2-B =

3.75. As a result, node S1 will use the link S1-C and node S2 will use S2-B. Table 1 details the energy levels of the nodes throughout the above mentioned scenario. Note that node C still retains 40% of its energy whereas it consumed all its energy when OLSR was used.

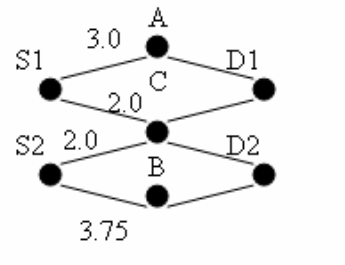

Figure 1. Ad hoc network with seven nodes.

Table 1. Remaining energy levels of the three routing nodes using TLR.

# Packets Sent	Link Costs				Active Link		Remaining Energy		
	S1-A	S1-C	S2-C	S2-B	S1	S2	A	B	C
0	3.0	2.0	2.0	3.75	S1-C	S2-C	100%	100%	100%
20	3.0	3.5	3.5	3.75	S1-A	S2-C	100%	100%	80%
40	4.33	2.86	3.86	3.75	S1-C	S2-B	90%	100%	70%
60	3.33	4.333	3.33	5.17	S1-A	S2-C	90%	90%	60%
80	4.75	4.0	5.0	4.17	S1-C	S2-B	80%	90%	50%
100	3.75	6.0	5.0	5.69	S1-A	S2-C	80%	80%	40%

4. RELIABILITY ANALYSIS OF TLR

The reliability of a system is defined as the conditional probability that the system is operational at time t given that it was operational at time t_0. Similarly, the reliability of a routing protocol can be given as the conditional probability that the network will properly

deliver the packets from the source to the destination given that the routing paths were available at time t_0.

Considering that a network is a system consisting of nodes and links then the reliability of the network will be directly dependent on the reliability of the components. Any failure to either of the nodes or links may lead to disruption of the service of the network. The service may remain available due to the redundancy of paths through various nodes. In this section, we analyze the reliability of OLSR and TLR for stationary nodes using Markov analysis.

4.1 Markov Analysis

Markov analysis provides a way of studying the sequence of events that a system may execute. This sequence of random and related events is useful in determining the reliability and availability of the system since it reflects the transition probabilities between the system states.

A Markov model is a function of two random variables; the time of observation and the current state, irrespective of the previous sequence that led to the current state.

The main advantages using Markov analysis are the simplicity of constructing the model, availability of transition probabilities between states and the incorporation of component redundancy. However, one drawback of this form of analysis is that the complexity increases with the number of states.

A Continuous Time Markov Chain (CTMC) can be used to model a system with a finite number of states to obtain a closed form solution of the reliability or availability of the system. For simplicity, we will assume that 1) component failures are mutually

independent; 2) failures occur at an exponential rate; 3) equal repair rate for all component failures and 4) only one failure can occur during transition between states. The last condition is to guarantee that no two routers fail during a switching delay period.

Consider the Markov reliability model given in Figure 3. The probability of transition from state S_0 to S_1 in time Δt is equal to $\lambda_0 \Delta t$ and the probability of remaining the same state is $(1- \lambda_0)\Delta t$. We can find the probabilities of the system being in either of the states at time $t = t+ \Delta t$ using the difference equations.

$$P_{S0}(t + \Delta t) = (1 - \lambda_0 \Delta t) P_{S0}(t) \tag{6}$$

$$P_{S1}(t + \Delta t) = (\lambda_0 \Delta t) P_{S0}(t) + (1 - \lambda_1 \Delta t) P_{S1}(t) \tag{7}$$

$$P_{S2}(t + \Delta t) = (\lambda_1 \Delta t) P_{S1}(t) + (1 - \lambda_2 \Delta t) P_{S2}(t) \tag{8}$$

$$P_{S3}(t + \Delta t) = (\lambda_2 \Delta t) P_{S2}(t) + P_{S3}(t) \tag{9}$$

Rearranging the equations:

$$\frac{P_{S0}(t + \Delta t) - P_{S0}(t)}{\Delta t} = - \lambda_0 P_{S0}(t) \tag{10}$$

$$\frac{P_{S1}(t + \Delta t) - P_{S1}(t)}{\Delta t} = \lambda_0 P_{S0}(t) - \lambda_1 P_{S1}(t) \tag{11}$$

$$\frac{P_{S2}(t + \Delta t) - P_{S2}(t)}{\Delta t} = \lambda_1 P_{S1}(t) - \lambda_2 P_{S2}(t) \tag{12}$$

$$\frac{P_{S3}(t + \Delta t) - P_{S3}(t)}{\Delta t} = \lambda_2 P_{S2}(t) \tag{13}$$

When $\Delta t \to 0$

$$\frac{dP_{S3}(t)}{dt} = \lambda_2 P_{S2}(t) \tag{14}$$

$$\frac{dP_{S2}(t)}{dt} = \lambda_1 P_{S1}(t) - \lambda_2 P_{S2}(t) \tag{15}$$

$$\frac{dP_{S1}(t)}{dt} = \lambda_0 P_{S0}(t) - \lambda_1 P_{S1}(t) \tag{16}$$

$$\frac{dP_{S0}(t)}{dt} = -\lambda_0 P_{S0}(t) \tag{17}$$

The set of differential equations (14)-(17) can be solved to find the overall reliability as will be shown in Section 4.2.

The analysis will use the CTMC to find the reliability of a network consisting of stationary nodes with no repair of dead nodes.

4.2 Reliability Analysis

The reliability of any ad hoc network depends on the reliability of the nodes. Thus, the failure of the network will be proportional to the expected lifetime of the nodes in the network. The network reliability and survivability calculation, however, is not simple; at some point, given the topology and the number/location of failed nodes, a network will fail. This will be case/state specific. However, we can make the following generalizations. The failure rate, λ, is defined as the number of failures per unit of time. One of the advantages of TLR is that it increases the average lifetime of all the nodes and decreases the variance of energy levels between the nodes compared to OLSR. In effect, TLR decreases the failure rate of the nodes which results in higher reliability.

By design, OLSR runs the nodes in a standby mode where only one node is responsible of routing and all other nodes remain in idle mode. When the routing node dies, another node will be selected to perform the routing. On the other hand, TLR runs the nodes in a parallel mode where one node handles the routing functionality for some time then another node takes over. The routing nodes in such a scenario will have a longer lifetime as was shown in (5).

Consider the network given in Figure 2, in which S1 has no direct access to its two hop neighbor D1. Instead, nodes A, B and C will be used to route the traffic.

For OLSR, the Markov Reliability Model is given in Figure 3. The system may be in one of the four states $\{S_i \mid i = 0,1,2,3\}$. State S_0 represents the initial state of the network at time t = 0 and all three nodes will be functional whereas S_3 is the state when all three routing nodes fail.

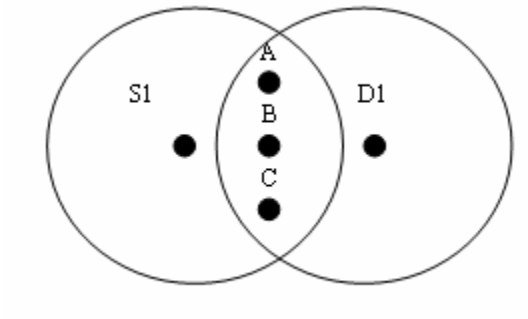

Figure 2. Ad hoc network with five nodes.

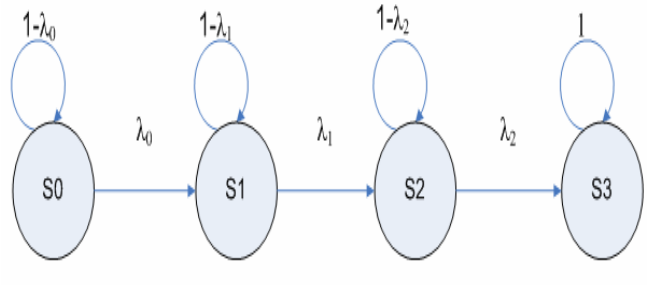

Figure 3. Markov Reliability Model for the network running OLSR.

The probabilities of each of the states can be calculated using equations. (14) –(17) with the initial conditions $P_{S0}(0) = 1$, $P_{S1}(0) = P_{S2}(0) = P_{S3}(0) = 0$. Taking the Laplace transform of the above equations yields

$$sP_{S3}(s) = \lambda_2 P_{S2}(s) \tag{18}$$

$$sP_{S2}(s) = \lambda_1 P_{S1}(s) - \lambda_2 P_{S2}(s) \tag{19}$$

$$sP_{S1}(s) = \lambda_0 P_{S0}(s) - \lambda_1 P_{S1}(s) \tag{20}$$

$$sP_{S0}(s) - 1 = -\lambda_0 P_{S0}(s) \tag{21}$$

Rearranging the equations

$$P_{S0}(s) = \frac{1}{s + \lambda_0} \tag{22}$$

$$P_{S1}(s) = \frac{\lambda_0}{(s + \lambda_0)(s + \lambda_1)} \tag{23}$$

$$P_{S1}(s) = \frac{\lambda_0 \lambda_1}{(s + \lambda_0)(s + \lambda_1)(s + \lambda_2)} \tag{24}$$

Converting back to the time domain

$$P_{S0}(t) = e^{-\lambda_0 t} \tag{25}$$

$$P_{S1}(t) = \frac{\lambda_0}{(\lambda_1 - \lambda_0)}\left[e^{-\lambda_0 t} - e^{-\lambda_1 t}\right] \tag{26}$$

$$P_{S2}(t) = \lambda_0 \lambda_1 \left[\frac{e^{-\lambda_0 t}}{(\lambda_1 - \lambda_0)(\lambda_2 - \lambda_0)} + \frac{e^{-\lambda_1 t}}{(\lambda_0 - \lambda_1)(\lambda_2 - \lambda_1)} + \frac{e^{-\lambda_2 t}}{(\lambda_0 - \lambda_2)(\lambda_1 - \lambda_2)}\right] \tag{27}$$

If $\lambda_0 = \lambda_1 = \lambda_2 = \lambda$ then we can find $P_{S1}(t)$ and $P_{S2}(t)$ using L'Hospital's rule (ordinary substitution would not work because the equations would yield 0/0).

$$P_{S0}(t) = e^{-\lambda t} \tag{28}$$

$$P_{S1}(t) = \lambda t e^{-\lambda t} \tag{29}$$

$$P_{S2}(t) = \frac{(\lambda t)^2}{2} e^{-\lambda t} \tag{30}$$

The network will remain operational as long as it remains in S_0, S_1, or S_2. Thus, the reliability of the network will be

$$R(t) = P_{S0}(t) + P_{S1}(t) + P_{S2}(t) = e^{-\lambda t} + \lambda t e^{-\lambda t} + \frac{(\lambda t)^2}{2} e^{-\lambda t} \tag{31}$$

An interesting observation is that the terms in R(t) are the first three terms in the Poisson distribution. It can be read as the probability of zero failures in time t plus the probability of one failure in time t plus the probability of 2 failures in time t. Extending the model to n nodes would result in

$$R(t) = \sum_{i=0}^{n-1} \frac{(\lambda t)^i}{i!} e^{-\lambda t} \tag{32}$$

Figure 4 shows the Markov Reliability Model for the TLR protocol. Here, the transition probability from state S_0 to S_1 is $3\lambda\Delta t$ because all three nodes are functional and anyone of them could fail. The same logic can be applied to the other transitions in the figure.

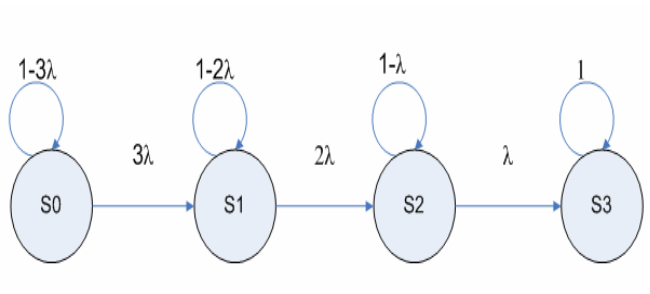

Figure 4. Markov Reliability Model for the network running TLR.

The reliability can be either calculated in a similar manner to that done for OLSR or we can use the fact that all the components are working in parallel and the system will be functional if either one of them is working.

$$\mathrm{R(t)=P(A+B+C)} = 3e^{-\lambda t} - 3e^{-2\lambda t} + e^{-3\lambda t} \tag{33}$$

The reliability of a network with n routing nodes can be give as

$$R(t) = \sum_{i=1}^{n} P(A_i) - \sum_{i=1}^{n} \sum_{j=i+1}^{n} P(A_i A_j) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} P(A_i A_j A_k) - \ldots + (-1)^{n-1} [P(A_1 A_2 \ldots A_n)] \quad (34)$$

Looking back at (5), we see the lifetime of a node will increase as the factors α and β decrease. Evidently, with more MPR candidate these factors must decrease since the node will be forwarding or retransmitting fewer packets. When only one MPR is available then α and β will be equal to 1.0 as the packets must follow the same path.

The relation between the failure rates for nodes running TLR (λ_{TLR}) and that of OLSR (λ_{OLSR})is

$$\lambda_{TLR} = \left[\frac{(\alpha + \beta\gamma) \sum_{k \in N1(i)} e_{tx} r_{i->k} + \sum_{j:i \in N1(j)} e_{rx} r_{j->i} + \varepsilon + \mu}{(1+\gamma) \sum_{k \in N1(i)} e_{tx} r_{i->k} + \sum_{j:i \in N1(j)} e_{rx} r_{j->i} + \varepsilon} \right] \lambda_{OLSR} \quad (35)$$

To illustrate, assume that the failure rate of nodes resulting from running the OLSR and TLR protocol are λ_{OLSR}=0.05 and λ_{TLR}=0.025, respectively. The reliability functions of the network for both protocols are given in Figure 5.

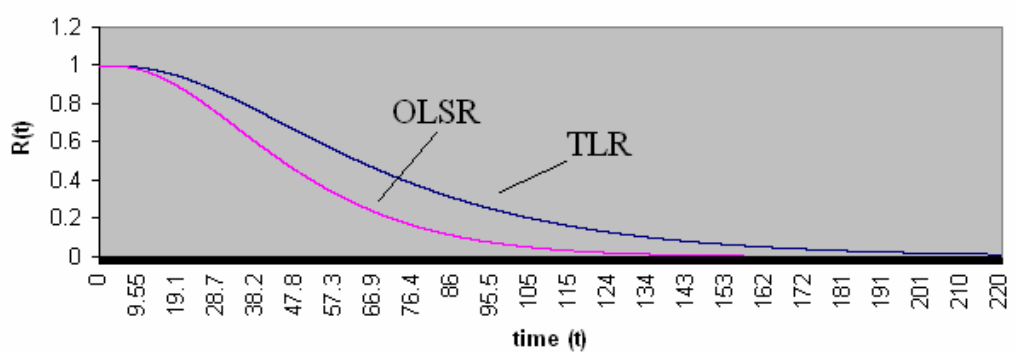

Figure 5. Reliability curves for OLSR and TLR.

If the parameters of the network such as the congestion and data rates are known then we can determine the encryption algorithm to be used based on the value of μ, according to the following condition

$$\mu < ((1-\alpha)+\gamma(1-\beta)) \sum_{k \in \mathrm{N1(i)}} e_{tx} r_{i->k} \qquad (36)$$

5. SIMULATION RESULTS

We used the network simulator NS2 to study the effect of TLR on the survivability of the nodes in the network. We implemented the algorithm as an extension to the existing OLSR code by modifying the selection of the MPR nodes in NS2. The energy values for transmission were increased to simulate the extra energy needed for encryption. The simulations are conducted in a 670 x 670 area with 100 nodes. The nodes communicate using Constant Bit Rate (CBR) sources that are connected to the nodes in a random fashion. The data rates of 64, 128, 256 and 512 kbps were used and the results of the runs were averaged. Each run had a period of 100 seconds.

The first objective of the simulation was to show the distribution of the energy in the nodes after running for the whole simulation period. After each run, the remaining energy of each node was sorted into one of 10 categories based on the percentage of the remaining energy compared to the energy level at the beginning of the simulation.

We note from Figure 6 that the percentage of remaining energy in the nodes running OLSR is not distributed optimally throughout the network. On one hand, we notice that a high percentage of the nodes hold a high percentages of their initial energy which means they were not actively involved in the routing process. This might seem to be a positive

feature but it comes at a cost. Namely, another high percentage of nodes have little remaining energy remaining. Only a few nodes have moderate consumption of their energy during the simulation.

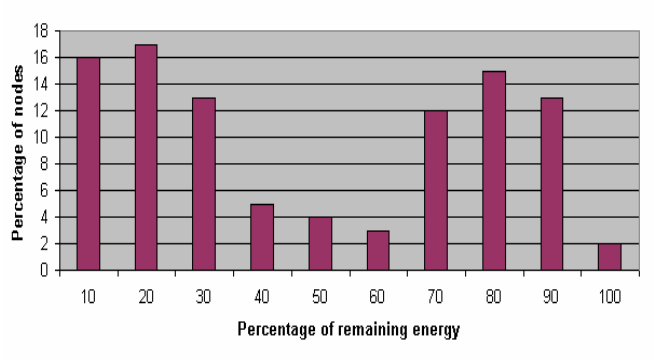

Figure 6. Energy distribution in nodes running OLSR.

This distribution is not desirable because it shows that around 46% of the nodes had energy levels of 30% or less.

A more desirable distribution of energy is shown in Figure 7. In this figure it is evident that a very small percentage of nodes have 30% of less of their initial energy. Also, the percentage of nodes with the higher energy reserves, 80% and above, is smaller than what was shown in the case of OLSR. Most of the energy is distributed among the nodes equally which leads to an extended lifetime of the nodes in the network.

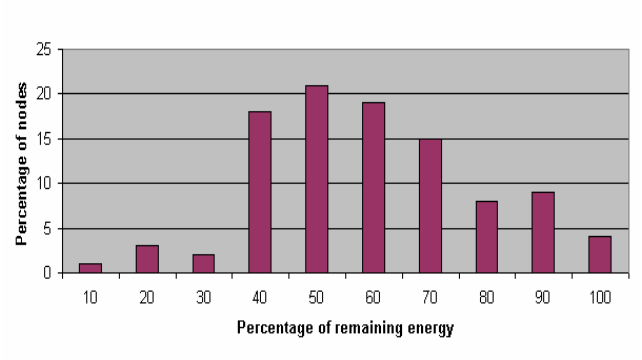

Figure 7. Energy distribution in nodes running TLR.

The second objective of the simulation was to study the average energy reserves of all the nodes in the network throughout the period of the simulation. This was done by using the energy levels given in the trace files of NS2. The sampling of the energy was done in 10 second intervals.

From Figure 8, it is evident that TLR leads to a higher average percentage of remaining energy. This is attributed to the decrease in dropped packets in the network. We see that TLR will give a longer lifetime of the network.

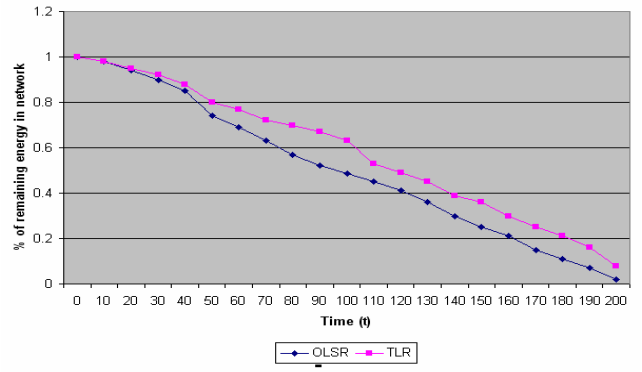

Figure 8. Percentage of remaining energy for networks running OLSR and TLR.

6. CONCLUSIONS

Two link state routing protocols were compared in terms of survivability and reliability. Markov models were used to find the reliability function and it was shown how the reliability of TLR is higher than that of OLSR. The results of this work show the advantage of using TLR instead of OLSR. Notably, TLR results in an extended lifetime of the nodes which may translate to a higher availability of paths. Future extension of this work includes the analysis of the routing protocols in the case of mobility. In such a case, the intermediate node may move in and out of the communication range.

7. REFERENCES

[1] Chen, D., Garg, S., and Trivedi, K. Network Survivability Performance Evaluation: A Quantitative Approach with Applications in Wireless Ad Hoc Networks. In *Proceedings of the Fifth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Atlanta, USA, 2002.

[2] Chen, H., Mineno, H. and Mizuno, T. An Energy-Aware Routing Scheme with Node Relay Willingness in Wireless Sensor Networks. In *Proceedings of the First International Conference on Innovative Computing,* Beijing, China, 2006.

[3] Domingo, M., Remondo, D. and Leon, O. A simple Routing Scheme for Improving Ad Hoc Network Survivability. In *Proceedings of IEEE GLOBECOM'03,* San Francisco, USA, December 1-5, 2003, 718–723.

[4] Jacquet, P. Muhlethaler, P. Clausen, T. Laouiti, A. Qayyum, A., and Viennot, L. Optimized Link State Routing Protocol for Ad Hoc Networks. In *Proceedings of the IEEE International Multi Topic Conference on Technology for the 21st Century, (INMIC'01)*, December, 2001, 62 – 68.

[5] Regatte, N., and Sarangapani, J. Optimized Energy-Delay Routing in Ad Hoc Wireless Networks. In *Proceedings of World Wireless Conference,* San Francisco, USA. *May, 2005.*

[6] Singh, S. and Raghavendra, C.S. PAMAS – Power Aware Multi-Access Protocol with Signaling for Ad Hoc Networks. *ACM Computer Communications Review,28, 3* (July 1998), 5 - 26.

[7] Taqieddin, E., Sarangapani, J. and Miller, A. Optimal Energy-Delay Routing Protocol with Trust levels for Wireless Ad Hoc Networks. In *Proceedings of International Telemetry Conference,(ITC'05)*, Las Vegas, Nevada, Oct 2005.

[8] Toh, C-K, Cobb, H., and Scott, D. Performance Evaluation of Battery-Life-Aware Routing Schemes for Wireless Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications (IEEE ICC)*, Helsinki, Finland, June 2001.

[9] Trajanov, D., Filiposka, S., Efnuseva, M. and Grnarov, A. Ad Hoc Networks Connection Availability Modeling. In *Proceedings of the 1st International ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, Venezia, Italy, 2004, 56 - 60.

[10] Zhou, Y., Laurenson, D., and McLaughlin, S. High Survival Probability Routing in Power-aware Mobile Ad Hoc Networks. *IEE Electronics Letters, Vol. 40, No. 22*, (Oct. 2004), 1424-1426.

III. A Reputation Based System to Avoid Packet Dropping in MANET

Eyad Taqieddin, Ann Miller, S. Jagannathan

Department of Electrical and Computer Engineering
University of Missouri – Rolla
Rolla, Missouri 65409
{eyad, milleran, sarangap}@umr.edu

Abstract

The routing of packets in ad hoc networks is based on the assumption that all neighbors are trustworthy and willing to cooperate with other nodes. This inherent trust, however, can lead to undesirable consequences in the presence of selfish or malicious intermediate nodes that drop part of or all the packets to be routed through them. In this paper, we highlight this vulnerability and propose a reputation based model, TRUST (Trusted Routing Under Security Threats), to exclude non-cooperative nodes from the routing tables. TRUST is a statistical model based on the Beta probability distribution function (PDF). Each node monitors the activity of its neighbors and calculates a value based on the ratio of forwarded packets to the sent packets. This information is shared by all the nodes in the network to infer the trust levels by calculating the expected value of the PDF using the method of moment matching. As an example, we apply TRUST to OLSR and OEDR. Through simulation, we show that it improves the packet delivery rate and reduces the end-to-end delay and the number of dropped packets.

Keywords—Reputation, Trust, Energy, Delay.

I. INTRODUCTION

Wireless ad hoc networks are formed by a set of nodes that collaborate together to achieve proper delivery of data packets. Every node depends on its neighbors to forward its packets to the destination. At the same time, the node must relay packets not intended to it to their destination. Proper routing depends on the assumption that all the nodes are cooperative and willing to relay the packets. In reality, this can not be guaranteed and successful communication in the network can be put at risk in the presence of selfish or malicious nodes that do not perform the forwarding operation.

A selfish node may not forward packets due to its limited power reserves whereas a malicious node misleads its victims to route their traffic through it and later drops all or part of the packet stream. Current MANET routing protocols do not have a mechanism to detect this threat of denial of service. We propose a reputation based model that encourages cooperation between nodes and isolates misbehaving nodes based on the percentage of their successful forwarding and their reputation amongst other nodes in the network.

Generally speaking, reputation models are used to improve cooperation between entities by implicitly forcing them to work together. This is done through giving incentives in the form of higher rating for cooperation and penalties of lower rating for disobliging entities. Any entity that does not cooperate will risk getting a lower reputation and thus losing the asset of its reputation. In this work, we study how nodes adhere to the condition of forwarding packets from sources other than themselves and how to translate such cooperation into a numerical value representing the trust value.

Various reputation based models for packet forwarding have been proposed. In [1], Baras and Mehta suggested an algorithm based on swarm intelligence for discovering and maintaining paths in MANET. CORE [2] and CONFIDANT [3] are reputation based models that monitor the packet loss in the neighborhood by running the nodes in a promiscuous mode to determine the ratio of forwarded packets to the total packets received. SAFE [4] is another system that monitors the neighborhood. However, unlike CONFIDANT and CORE, SAFE relies on monitoring the sequence numbers of packets for its calculations. Marti et al. [5] proposed the use of Watchdog and Pathrater to determine the routing path based on the knowledge of misbehaving nodes and link reliability. Buchegger and Le Boudec [6] and Ganeriwal and Srivastava [7] proposed reputation systems that are based on using the Beta distribution as the conjugate prior for Bayesian analysis to update the trust values based on the observations. Both references have similar concepts except that the former deals with ad hoc networks while the latter is concerned with sensor network. Jøsang and Ismail [8] proposed the Beta reputation system for centralized environments. Their work, which can be extended to non-centralized environments, discusses methods for merging, discounting and forgetting the estimated opinions.

In this work, we propose a statistical reputation model that uses the watchdog mechanism to observe the cooperation of the neighboring nodes. Similar to [6], [7], and [8], our model uses the Beta distribution as a basis for calculating/updating the trust value. However, it differs in that it does not use Bayesian analysis. Instead, it uses self observed values along with the observations of all other nodes to estimate the parameters of the PDF using the method of moment matching.

The remainder of this paper is organized as follows: Section 2 introduces the reputation based model and the parameter estimation. Section 3 gives a brief overview of the operation of OLSR and OEDR and discusses the algorithm used for updating reputation values. In Section 4, we present the simulation results followed by the conclusions in Section 5.

II. BACKGROUND

A. The Beta Distribution Function

The Beta distribution is a continuous PDF defined on the interval [0,1] as

$$f(x;\alpha,\beta)=\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1} \tag{1}$$

where $\alpha > 0$, $\beta > 0$ and $0 \le x \le 1$ with the constraint that $x \neq 0$ if $\alpha < 1$ and $x \neq 1$ if $\beta < 1$.

The parameters α and β are called the shape parameters and Γ(.) is the Gamma function defined as

$$\Gamma(y)=\int_0^\infty t^{y-1}e^{-t}dt \tag{2}$$

This can be integrated by parts to yield

$$\Gamma(y)=(y-1)\Gamma(y-1) \tag{3}$$

A special case of the Beta distribution occurs when $\alpha=\beta=1$. In this case the PDF will represent the Uniform distribution.

The moments of the Beta distribution are calculated according to

$$\mu_i=\prod_{j=1}^{i}\frac{(\alpha+j-1)}{(\alpha+\beta+j-1)} \tag{4}$$

Thus,

$$E[X] = \mu_1 = \frac{\alpha}{\alpha + \beta} \tag{5}$$

$$E[X^2] = \mu_2 = \frac{\alpha(\alpha + 1)}{(\alpha + \beta)(\alpha + \beta + 1)} \tag{6}$$

From (5) and (6), the variance is calculated as

$$Var[X] = E[X^2] - (E[X])^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)} \tag{7}$$

Figure 1 shows the effect of the changes in the shape factors. As α and β increase, the peak of the curve becomes higher and the width of the curve decreases indicating more confidence in the accumulated observations.

The Beta distribution provides much flexibility in the case of trust. This is a result of it being bounded in the interval [0,1] which can be considered as a range of no trust to full trust. It also gives a level of confidence in the reported value. For example, in the curves we observe that the mean for the PDFs for α = β is equal to 0.5 according to (5), however the variance decreases from 1/12 (when α = β = 1) to 1/68 (when α = β =8). In such a case, although both curves give the same expected value, we get more confidence in the latter due to low variance. This coincides with the belief that a larger sample should give a more accurate result.

If we consider the Beta distribution, represented by its shape parameters, as the reputation of a node, then we can quantify that reputation by taking the expected value in (5) as the trust value.

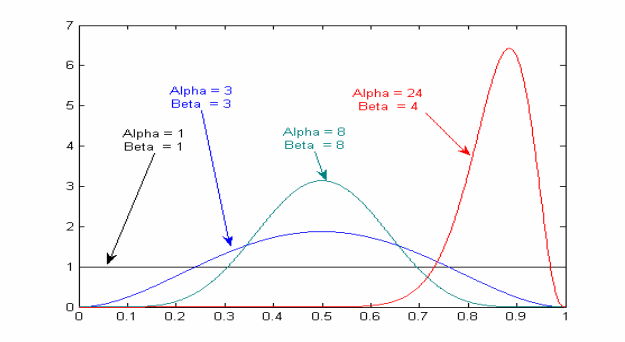

Figure 1. Different shapes of the Beta distribution function

B. Bayesian Analysis

Bayesian analysis is used in [6] and [7] for forming opinions about the reputation of other nodes. Basically, the goal is to estimate the probability that a node will not act maliciously. Baye's theorem is given as

$$P(Belief \mid Observation) = \frac{P(Observation \mid Belief)P(Belief)}{Normalization} \tag{8}$$

where P(Belief | Observation) is the posterior probability which reflects the updating probability of the belief given the added observations. P(Observation | Belief) represents the new observations and P(Belief) represents the prior belief.

One interesting property of the Beta distribution is that it is a conjugate prior for the Binomial distribution. This means that if the prior is a beta distribution and the latest observations are binomially distributed then the posterior will be a beta distribution. Transactions with only two results, such as satisfaction and dissatisfaction, can be considered binomially distributed.

The shape parameters α and β represent the satisfaction and dissatisfaction about transactions between two entities, respectively. Initially, with no prior experience, α=β=1

which represents the Uniform distribution. If a node has s + f transactions of which s are successful and f are failures then the new posterior is given as [7]

$$P(x) = \frac{Bin(s+f, s) * Beta(1,1)}{Normalizaton} = Beta(s+1, f+1) \tag{9}$$

The use of Bayesian analysis results in a flexible scheme for updating the reputation value by continuously modifying the shape parameters according to

$$\alpha = \lambda\alpha + s \tag{10}$$

And

$$\beta = \lambda\beta + f \tag{11}$$

where λ is the aging factor used for giving higher weight to recent values

C. Parameter Estimation Using Moment Matching

In Section 2.1, it was demonstrated how the curve changed according to the changes in the shape factors which indirectly represent the trust value to be assigned. By continuously incrementing α and β with each successful or failed transaction, the curves will tend to have higher peaks and lower standard deviations.

In normal circumstances, a cooperative node in a network should uphold its reputation unless it gets compromised. The same can be said regarding malicious nodes whose uncooperative nature would be detected through its high failed transactions. Thus, it is safe to say that after a relatively long period of network activity, malicious nodes will have high β values and cooperative nodes will have high α values which translate to low trust and high trust, respectively.

When a trusted node with a high α value gets compromised, its ratings will start to decline due to the aging factor in (10) and the β value will increase because of detected

failed transmissions. However, the reputation updates will take a long time to reach correct values that correctly represent the status of the compromised node depending on the value of λ. This means that a compromised node will not be detected for a relatively long period of time.

A solution for fast convergence is to use self-observations along with the updates of other nodes in the network to back the observations. This will result in higher confidence as the sample becomes larger. One method is to use all the reported trust values to estimate the shape parameters of the PDF that would represent the overall trust value. In other words, instead of having the shape parameters and trying to find the probability within [0,1], we will have a set of samples (reported trust values from others) and try to find the best values of α and β that represent the curve.

Moment matching is a popular method in statistics used for parameter estimation in large samples. It usually results in a small loss of precision. The main idea of moment matching is to calculate a number of moments equal to the number of parameters to be estimated. In the case of the Beta distribution, we have two shape parameters to estimate, and for that we will calculate the first two central moments.

For a sample of n observations, the data mean, $\bar{x}$, and the variance, σ^2, are calculated as

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{12}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{(\sum_{i=1}^{n} x_i)^2}{n^2} \tag{13}$$

Thus, by equating the calculated values to those given in (5) and (7)

$$\bar{x} = \frac{\hat{\alpha}}{\hat{\alpha} + \hat{\beta}} \tag{14}$$

And

$$\sigma^2 = \frac{\hat{\alpha}\,\hat{\beta}}{(\hat{\alpha} + \hat{\beta})^2 (\hat{\alpha} + \hat{\beta} + 1)} \tag{15}$$

where $\hat{\alpha}$ *and* $\hat{\beta}$ are the *estimated* shape parameters. With some algebra we get

$$\hat{\beta} = \frac{(1 - \bar{x})}{\sigma^2}\left[\bar{x}(1 - \bar{x}) - \sigma^2\right] \tag{16}$$

$$\hat{\alpha} = \frac{\bar{x}\,\hat{\beta}}{(1 - \bar{x})} \tag{17}$$

By using (16) and (17), we can determine the estimated shape parameters for the overall sample which can be used for calculating a numerical trust value based on (5).

III. TRUSTED ROUTING UNDER SECURITY THREATS (TRUST)

We show in this section an example of how the proposed reputation system can be applied to MANET routing protocols. The protocols selected are Optimal Link State Routing (OLSR) [9] and Optimal Energy Delay Routing (OEDR) [10] due to the fact that they both involve making critical decisions about which nodes to select for forwarding their traffic as will be explained shortly. It should be emphasized that this system is not solely applicable to the chosen protocols and that it can be applied to other existing protocols.

A. Overview of OLSR and OEDR

OLSR and OEDR are table driven protocols with each node maintaining multiple tables for proper routing. Each node holds information about the immediate neighbors, two-hop neighbors, topology and routes. The cornerstone of their operation is the concept of Multi-Point Relay (MPR) nodes. In brief, an MPR is a node that has a bidirectional link with its selector and is responsible of forwarding control packets from and data packets to that selector. The use of MPR nodes results in reducing the packets flooded in the network since broadcast packets will be forwarded through them only.

In [10], it is suggested that all the intermediate nodes on a path must be selected as MPRs by the preceding node on the path. This highlights the importance of choosing dependable MPRs to properly deliver the packets from the source to the destination. The role of TRUST in this example will be to enhance the selection of MPRs such that nodes with higher trust values are chosen.

In the case of OLSR, each node selects its MPR nodes from its one hop neighbors based on the maximum coverage of two hop neighbors. On the other hand, OEDR uses the energy-delay product and the remaining energy levels as factors in determining its MPRs. This essential information is usually communicated through control packets.

There are two types of control packets; HELLO and Topology Control (TC) packets. A node sends HELLO packets to its immediate neighbors informing them about its presence in their vicinity and detailing its neighbors. If a recipient detects its ID in the list of neighbors of the source then a bidirectional link does exist and the node becomes an MPR candidate. HELLO packets in OEDR serve an additional purpose of passing information about the energy and delay requirements in the neighborhood.

TC packets, on the other hand, detail the current information that a node possesses regarding the network topology. A TC packet is flooded in the network through MPR nodes only, and each node that receives it uses the information along with its own view of the network to update its network topology tables.

The routes are determined according to the information in the immediate neighbors and topology tables. Routing tables are calculated and updated whenever a change in the network topology is detected. OLSR protocol uses a shortest path algorithm while OEDR employs a minimum cost spanning tree in constructing the routes between a source and destination pair.

In the next sub-section, we discuss the vulnerabilities in these protocols and show why a reputation model is needed to ensure proper routing of data in the network.

B. Vulnerabilities in OLSR and OEDR

Due to the inherent trust in these protocols, they become vulnerable to malicious/selfish activity. When a node selects its MPRs, it assumes that the neighbor will forward the packet in a best effort to its intended destination without any modifications in the contents of the packet. This assumption, unfortunately, is not always valid. Attackers can exploit this exposure to disrupt the operation of network. Following are two examples of attacks.

Modification Attack: An intruder can change the header or payload of the packet. In the former, it either alters the destination address, source address, time-to-live, or the sequence number to match a previously used number. Thus, the packet will either be dropped prematurely or be delivered to a wrong destination.

This kind of attack can be avoided using *Message Authentication* Codes to guarantee that it was not changed in transit. Another approach is for the source to overhear its next hops transmission and compare the forwarded packet with the one that it has actually sent. If an anomaly is detected, it feeds that information to the reputation system and lowers its level of trust for that node.

Packet Dropping: This happens in various scenarios. 1) a given node with limited power reserves may choose not to forward the traffic it receives from its MPR selectors to conserve its energy. 2) a given malicious node broadcasts false TC information to convince others that it is a suitable MPR. Once it's selected as an MPR, it starts dropping packets instead of forwarding them.

It's worth mentioning that a node does not necessarily drop all the packets routed through it. It may selectively drop packets from a node while forwarding packets from others. Both vulnerabilities mentioned above highlight the importance of having a mechanism to rate the level of cooperation of MPR nodes. Timestamps and authentication can be used to overcome the modification problem but the packet dropping vulnerability needs to be addressed using a reputation system to guarantee that selected MPRs are, indeed, forwarding packets for their selectors.

IV. TRUST CORE

TRUST is based on running the nodes in promiscuous mode to capture all the packets transmitted by immediate neighbors. The idea is that whenever a node sends a packet, it keeps a copy of it in a repository for future comparison. Since the node shares a bidirectional link with the MPR, it can monitor the next hop to determine whether the packet was forwarded or not.

To illustrate how the model works, suppose that node 0 in Figure 2 uses node 1 as its MPR to deliver a broadcast packet. Node 0 keeps a copy in a list of previously sent packets and increments the number of packets sent to node 1. Within a specified interval, node 0 monitors the transmissions of node 1 to detect whether the packet was forwarded. If the packet is detected and the content has not been changed then node 0 increments the number of packets forwarded by node 1. If, on the other hand, the contents were changed or the timer reached zero without detecting a forwarded copy then the number of failed forwards for node 1 is incremented.

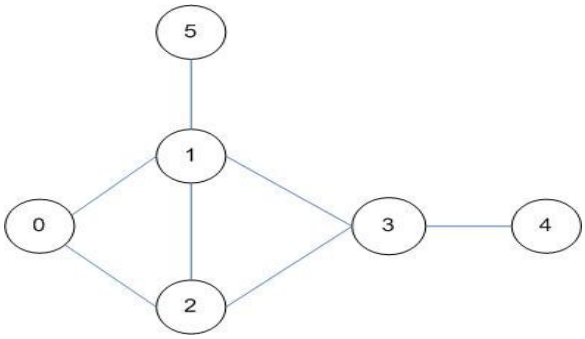

Figure 2. Ad hoc network with six nodes

The ides is further extended to cover nodes that are not part of the route. For example, node 2 is an immediate neighbor of node 0 but it is not selected as an MPR. Thus, when it receives the packet from node 0, it does not forward it. However, node 2 uses the information to monitor node 1 (another immediate neighbor of 2) to verify its cooperation with node 0. This approach gives node 1 an added incentive to cooperate rather than risk getting lower ratings from both nodes 0 and 2 which would, later on, affect its overall reputation in the network.

Each node maintains two values of reputation; local and global. Both values range from 0.0 (no packets forwarded) to 1.0 (all packets were forwarded). All nodes initially assign a reputation level of 0.5 to both local and global values, which is equivalent to Beta(1,1). These levels change according to the observed forwarding operations in the node being monitored. Both the local and global reputation values are used in the overall calculation of a numeric trust value.

The overall trust value is calculated by combining both the local trust and the reported trust from other neighbors.

$$T_{i,j} = \delta\ L_{i,j} + \eta\ G_{i,j} \tag{18}$$

where,

$T_{i,j}$: The overall trust value for node j at node i.

δ, η : weigh factors

$L_{i,j}$: The locally observed trust value for node j at node i.

$G_{i,j}$: The globally reported trust value for node j at node i.

A. Monitoring Cooperation

A local reputation value reflects the observation of node *i* regarding the cooperation of its neighbor *j*. By referring again to Figure 2, suppose that node 1 is selected by nodes 0 and 2 to forward their traffic to nodes 4 and 5, respectively. The local reputation of node 1 at node 0, $L_{0,1}$, depends on the observation of node 0 regarding its own traffic in addition to its observation regarding the traffic generated by node 2.

Thus, the local trust value of node *j* at node *i* can be calculated as follows

$$L_{i,j} = w_1\ \frac{F_i}{S_i} + w_2\ \frac{F_k}{S_k} \tag{19}$$

where,

F_i : Number of packets forwarded by *j* on behalf of *i*

S_i : Number of packets sent from node *i* to node *j*

F_k : Number of packets forwarded by *j* on behalf of *k*, (k≠i)

S_k : Number of packets sent from node *k* to node *j*.

The weights ω_1 and ω_2 are assigned based on the nature of traffic going through the monitored node. If a node just forwards the packets generated by the monitor then (ω_1 =1) and (ω_2 = 0). If the traffic is generated by nodes other than the monitoring node then (ω_1 = 0) and (ω_2 = 1). If both types of sources are available, then we assign values to ω_1 and ω_2 such that $\omega_1 + \omega_2 = 1$.

B. Reputation Algorithm Pseudo Code

For every packet received, do

- If unique ID does not exist in table
  - (i) If the packet is destined to monitoring node
    - ∗ Accept and move packet to upper layers
    - ∗ Return
  - (ii) Else, packet not destined to monitoring node
    - ∗ If monitoring node is the MPR
      - i) If next hop is the final destination
        - ◊ Forward the packet
        - ◊ Return
      - ii) Else, next hop is not the final destination

- ◊ Forward the packet
- ◊ Increment the number of packets sent from monitoring node.
- ◊ Return

* Else, monitoring node is not the MPR

i) If the MPR is a neighbor of the monitoring node with a bidirectional link

- ◊ Increment the number of packets sent from neighbor
- ◊ Return

- Else, unique ID exists in table

(i) If monitoring node is the original source of the message

* Increment the number of packets forwarded for monitoring node.
* Jump to (iii)

(ii) Else, monitoring node is not the original source

* Increment the number of packets forwarded for others.

(iii) Erase the entry of the packet from the list.

(iv) Return

C. Global Reputation

In general, a node i assigns a global trust value to node j, $G_{i,j}$, by using the reported trust values of all the nodes in the network other than j. Then it assigns weights to the reported trust values based on its current observed reputation of each reporting node. This, in effect, assigns lower values to the reported samples of low-trust nodes. Each reported value from node k to node i about node j, $T_{k,j}$, will be modified according to

$$T_{i,j} = T_{k,j}.T_{i,k} \quad \forall k \neq j \tag{20}$$

The updated reports are then used as the sample for estimating the global reputation value using the method of moments in (15) and (16) to get the estimated shape parameters, $\hat{\alpha}, \hat{\beta}$ and then to calculate the global trust value

$$G_{i,j} = \frac{\hat{\alpha}}{\hat{\alpha} + \hat{\beta}} \tag{21}$$

With the global reputation values, each node has an idea of the history of cooperation of every other node in the network.

To support the reputation model in a changing environment, the network members need to communicate their local trust values to all the nodes in the network. This is to inform non-neighbors about the current observations a node has regarding its neighborhood. For example, if node 0 determines that node 1 is not cooperative it should broadcast that information to make the observation known to all nodes (especially non-neighbors that may become neighbors of node 1 in the future).

To allow proper dissemination of reputation values, a third type of control packet is added. Reputation Control (RC) packets are sent through all the nodes in the network instead of just through the selected MPR as is currently done with TC packets. RC packets are sent periodically to update other nodes or whenever a local value for a node goes below a specified threshold.

We reason that the added overhead is justifiable since the main purpose of such a flooding is, in fact, to guarantee that multiple copies of the same RC packet are received and compared together to detect any modifications. Furthermore, forwarding RC packets through the selected MPR only may pose a threat if the MPR drops the RC packets.

Normally, malicious/selfish behavior must be detected by more than one neighbor resulting in various transmissions of RC packets.

D. Overhead Analysis

The protocol control overhead results from the exchange of HELLO packets locally and the broadcast of TC packets. According to [11], a network running with N nodes and periodic control packets will send

$$hN + O_p tN^2 \text{ (packets/s)} \tag{22}$$

where,

h: rate of sending HELLO packets.

t: rate of sending TC packets

O_p: Optimization broadcast factor ($1/\Delta \leq O_p \leq 1$)

Δ: Average number of neighbors.

The optimization broadcast factor represents the usage of MPR nodes. A value of (O_p = 1) means that all the nodes are broadcasting the packet whereas a value of ($O_p = 1/\Delta$) means that on average one MPR is used.

The ratio of control packets to the total bandwidth is given as

$$(hNH + O_p tN^2 T)/B \tag{23}$$

where,

H: average size of HELLO packets.

T: average size of TC packets.

B: available bandwidth.

The overhead given above applies to OLSR and OEDR only. For TRUST, the overhead is different due to the usage of RC packets which are sent out either periodically in a set

interval to ensure proper updates of reputation in the whole network or randomly depending on the detected changes in the reputation values.

Thus, the number of control packets sent per second in TRUST is

$$hN + O_p tN^2 + (r + \mu)N^2 \text{ (packets/s)} \tag{24}$$

where,

r: rate of RC packets

μ: average number of detected anomalies.

Note that O_p in the case of RC packets is equal to 1 since all the packets broadcast through all the nodes and not just the MPRs. The ratio of control packets to the total bandwidth is given as

$$(hNH + O_p tN^2 T + (r + \mu)N^2 R)/B \tag{25}$$

V. SIMULATION RESULTS

We used the network simulator NS2 to study the effect of the reputation model on the routing process. We implemented the algorithm as an extension to the existing OLSR and OEDR codes by modifying the nodes in NS2 to simulate the behavior of a malicious/selfish node. The simulations are conducted in a 670 x 670 area with 50 mobile nodes. The nodes move in the network with a maximum speed of 20 m/s following the random waypoint mobility model. The nodes communicate using Constant Bit Rate (CBR) sources that are connected to the nodes in a random fashion. Each run had a period of 100 seconds. Finally, the network has a maximum of 15 malicious nodes.

Our first objective was to study the effect of the number of uncooperative nodes on the overall delivery of packets. We ran the simulation with 0, 1, 2, 3, 5, 10 and 15 malicious nodes that were chosen at random.

Figure 3 shows that the packet delivery ratio decreased as the number of malicious nodes increased. This is a direct result of the excessive dropping that takes place as more selfish/malicious nodes join the network. With 2 nodes, we observe that at least 90% of the packets are delivered for all protocols. However, with 15 nodes, OLSR delivery ratio is around 30% whereas TRUST has a ratio of about 70%. This difference occurred because the source nodes avoided the malicious nodes when doing their routing. Finally, we see that OEDR is more reliable compared to OLSR. This can be attributed to the fact that OEDR selects the MPRs in a dynamic fashion depending on the energy-delay product. At one point, a node updates its MPR set due to changes in the residual energy of its previous MPR. This results in switching to a cooperative node, thus unintentionally avoiding a malicious node.

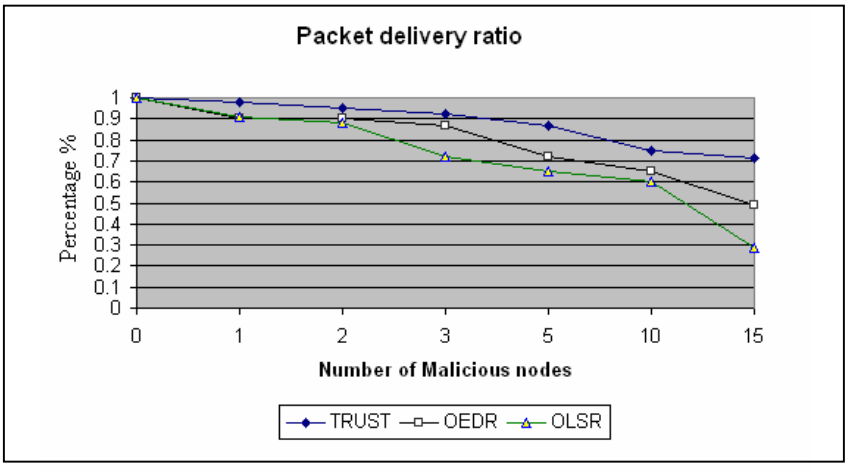

Figure 3. Packet delivery ratio

Next, we examined the effect of malicious nodes on the average end-to-end delay. The results are shown in Figure 4. We define the average end-to-end delay as the average time it takes a packet to reach its intended destination. If a packet is dropped and another copy is retransmitted, then the time is considered starting from the transmission of the initial packet to the time a copy reaches the intended destination.

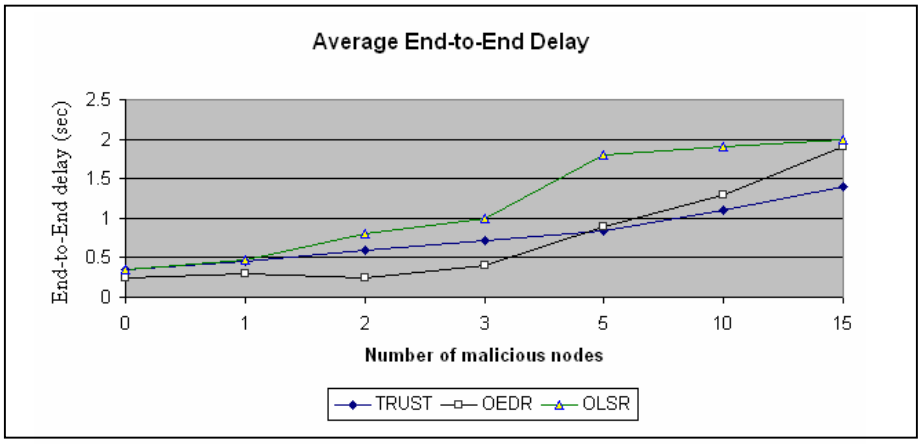

Figure 4. Average end-to-end delay

Our first observation is that the end-to-end delay increases in proportion with the number of malicious nodes. Furthermore, notice that OEDR performs better than TRUST when the number of malicious nodes is low. This is due to the nature of OEDR where more emphasis is given to the delay when selecting the routes. Thus, the routes that are intruder free decrease the overall end-to-end delay. However, when the number of intruders increases, we see that TRUST performs better since it avoids routes that do not forward its packets. Finally, OLSR has high delay compared to its counterparts due to its static nature of selecting its MPRs. This difference, nevertheless, becomes marginal when the number of intruders reaches 15 as the nodes have fewer options for routing the packets around the uncooperative nodes.

Next, we considered the case of mobility and the benefit of using RC packets on the performance of TRUST. We show the results in Figure 5 where five nodes were randomly selected as malicious nodes. Note that the delay increases when the velocity increases. This is due to the rapid changes in the network topology and the construction of new routes. One interesting observation is that most of the added delay in TRUST is a result of mobility only. OEDR and OLSR experience added delays due to the fact that when a malicious node moves around the network, it can be selected as an MPR by unsuspecting neighbors and thus their packets will be dropped. In the case of TRUST, on the other hand, the reputation of the node is global and any attempt for it to become an MPR candidate will be rejected based on its previous cooperation levels observed by others.

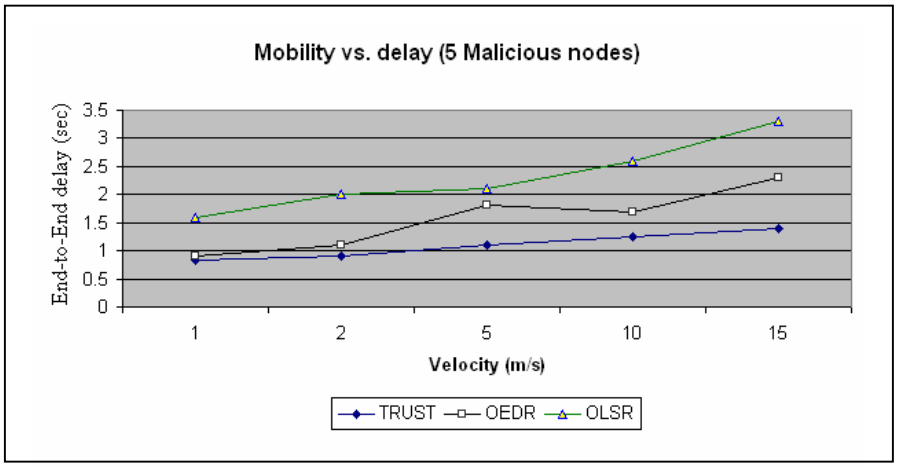

Figure 5. Mobility vs. delay

The next simulation was to compare the overhead in the three protocols. We took the ratio of control packets to the sent data packets in our simulation.

It is obvious from Figure 6 that the overhead for OLSR and OEDR fluctuate around a constant value with a higher average for OEDR. This is because of the added content to

the control packets in OEDR (energy and delay information) as well as the more frequent topology updates in OEDR. Moreover, we notice that the overhead for TRUST is much higher than that for the other two protocols. This is a direct result of the increased transmission of RC packets throughout the network as well as the TC packets detailing the new topology updates.

The results in Figure 6 would give a false impression that TRUST is underperforming. However, when this information is taken in conjunction with the packet delivery ratio in Figure 3 we can see that the added overhead was the price for sustaining a higher throughput in TRUST compared to the other two protocols.

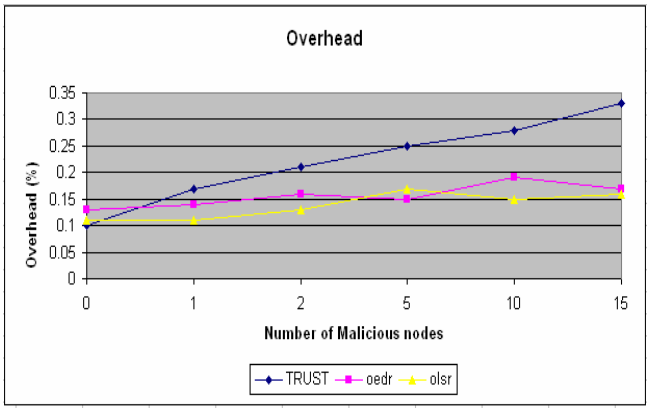

Figure 6. Control packets overhead

VI. CONCLUSION

We presented a reputation model based on the Beta distribution function. The goal of this work was to provide a method for MPR selection based on monitoring the traffic activity in the neighborhood and using reputation reports given by other nodes in the network. Experiments in ns2 show that a higher delivery rate was achieved using

TRUST. Moreover, lower delays in both static and dynamic networks were observed. The application of TRUST takes advantage of MPR usage just like in OLSR and OEDR. However, the way the MPRs are selected in TRUST makes it more defiant to packet dropping attempts.

VII. REFERENCES

[1] J. Baras, and H. Mehta, "A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks," *Proceedings of WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-antipolis, France, March 3-5, 2003.

[2] P. Michiardi, and R. Molva, "CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," *Proceedings of the IFIP conference, Vol 228*, Portoroz, Slovenia, September 26-27, 2002, pp. 107-121.

[3] S. Bucheggar, and J. Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes-Fairness In Dynamic Ad Hoc NeTworks," *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*. Lausanne, Switzerland, June, 2002.

[4] Y. Rebahi, V. E. Mujica-V, C. Simons, and D. Sisalem "SAFE: Securing pAcket Forwarding in ad hoc nEtworks," *Proceedings of the 5th workshop on Applications and Services in Wireless Networks (ASWN 2005)*, Paris, France, June 29 – July 1st, 2005.

[5] S. Marti, T.J. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, Boston, MA, August 2000, pp. 255–265.

[6] S. Bucheggar, and J. Y. Le Boudec, "The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks," *Proceedings of WiOpt 2003 Modeling of Optimization in Mobile, Ad Hoc and Wireless Networks,* Sophia-Antipolis, France, March 2003.

[7] S. Ganeriwal, and M. B. Srivastava, "Reputation-based Framework for High Integrity Sensor Networks," *Proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks,* New York, NY, USA, 2004, pp. 66 – 77.

[8] A. Jøsang, and R. Ismail, "The Beta reputation system," *Proceedings of the 15th Bled Electronic Commerce Conference*, Bled, Slovenia, June 2002.

[9] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," *IEEE International Multi Topic Conference on Technology for the 21st Century, IEEE INMIC'01*, Lahore University of Management Sciences, Pakistan, December 28-30, 2001, pp. 62 – 68.

[10] N. Regatte, and S. Jagannathan, "Optimized Energy-Delay Routing in Ad Hoc Wireless Networks," *Proceedings of World Wireless Conference*, San Francisco, CA, May, 2005.

[11] L. Viennot, P. Jacquet, and T. Clausen, "Analyzing Control Traffic Overhead versus Mobility and Data Traffic Activity in Mobile Ad-Hoc Network Protocols," Wireless Networks Vol.10, Kluwer Academic Publishers, 2004, pp. 447–455.

IV. Hardware Implementation of the Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks

Eyad Taqieddin, Maciej Zawodniok, S. Jagannathan, Ann Miller
Department of Electrical and Computer Engineering
University of Missouri - Rolla
Rolla, MO 65401, USA
{eyad, mjzx9c, sarangap, milleran@umr.edu}

ABSTRACT

Key distribution in wireless sensor networks is challenging due to the lack of a trusted third party. Several approaches for assigning the encryption keys to the nodes were proposed ranging from using the same key throughout the network to assigning unique keys to each pair of nodes. The former approach may lead to the compromise of the whole network, whereas the latter results in the depletion of the energy, processing, and memory resources of the network. Dynamic key distribution has been proposed as an alternative for the pre-deployment of keys. In this approach, the keys are generated through the collaboration of the nodes in the network. Each participating node sends a partial key to a Head Cluster Head that later calculates the sub-network key and distributes it to the participating nodes. The advantage of such an approach is that the keys change frequently and adapt to the changes in the created clusters within the network. Furthermore, the changes in the dynamic keys mean that the malicious disclosure of the key will have limited effect on the network. In previous work, Energy-efficient Hybrid Key Management (EHKM) was proposed for dynamic key distribution. The protocols' validity was shown through simulation. In this work, we implemented EHKM and an extended version of it EEHKM on the UMR/SLU motes to get a better understanding of how the protocol would behave under the actual constraints of hardware that are not shown by simulation.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design – *wireless communication;* C.2.2 [**Computer Communication Networks**]: Network Protocols – *routing protocols.*

General Terms

Security, Energy, Key Distribution, Hardware, Sensor Networks.

Keywords

Energy Efficient, Delay, Security, Key Distribution.

1. INTRODUCTION

The numerous applications of Wireless Sensor Networks (WSN) have encouraged the research in this promising field. Some examples of such applications are military operations, humanitarian relief, and medical services. A WSN consists of a set of nodes that collaborate with each other to guarantee proper communication between any node in the network and the Base Station (BS).

Unlike wired networks, the nodes in a sensor network have no fixed infrastructure and are limited in communication range. Moreover, they lack the abundant resources available in wired networks. This dictates the use of energy efficient and resource friendly routing algorithms.

The information in WSN is transmitted using RF channels which may pose a security threat. This calls for security features to guarantee the confidentiality and authenticity of

the data delivered to the BS. This is especially needed in situations where the nodes are deployed in hostile environments.

The corner stone of security is the proper distribution and safe handling of the cryptographic keys. Any compromise of a node or the way the keys are distributed can affect part or the whole network. One approach for key distribution is to use public/private key pairs. Although this option would provide reasonable security, it does not serve well in WSN due to the lack of a trusted certification authority, the high energy consumption, and computational expensiveness.

The use of shared keys can provide the needed security while maintaining the limited energy and computational resources. One approach is to load the encryption keys into the nodes' memory before deployment. This may be done by using a global key shared by all the nodes in the network [1], [2]. Alternatively, the nodes can be pre-deployed with unique key pairs for every pair of nodes in the network. The former approach may be defeated if one or more nodes are compromised which gives the attacker access to the global key. The latter, on the other hand, is not susceptible to such a threat (a node compromise can only affect the traffic of that node) but it is resource consuming. Each node in a network of n nodes should store (n-1) keys for the pair-wise communications. In [4], it is shown that the number of keys increases exponentially with the size of the network. A tradeoff between the two approaches is to load the nodes with key rings consisting of sets of random keys selected from a larger pool of keys. The ides is that any two nodes in the network will share the same key with a certain probability p that depends on the number of keys in the key ring and the total keys in the pool [3], [5].

The use of dynamic key agreement is also proposed in the literature [12],[13]. Dynamic keys have the advantage of reduced storage requirements compared to pre-deployed keys. Moreover, the frequent changes of the keys limit the access of an intruder to them. In [13], a protocol for dynamic key generation is presented in which a cluster key is computed starting from the leaf nodes up to the Cluster Head. Every node in the group uses the partial keys of its children as inputs of a function $\alpha^{k_1 \oplus k_2} \bmod p$ where p is a prime number, α is the primitive root of p and k_1, k_2 are the partial input keys of the function. The CH computes the cluster key and securely broadcasts it to the group.

In [12], a mechanism called Energy-efficient Hybrid Key Management (EHKM) is given in which separate keys with different purposes are maintained by each node. These keys are either pre-deployed or dynamically generated using a method similar to that given in [13]. The goal of EHKM is to diversify the levels of security in the network since not all the data on the network need to be handled with the same level of security. EHKM enables the nodes in the network to operate in a low-security energy-efficient mode using static keys, while dynamically creating keys for high security sub-networks.

Most of the key management approaches mentioned above rely on simulations using NS2 or GloMoSim. This may be useful in cases where the performance metrics of two protocols are compared against each other. Nevertheless, these simulations do not give a fair assessment of the behavior of the protocols when run on hardware. Energy consumption, memory constraints, collisions and topology can be modeled in a simulator but the value of simulating them would be only as good as the model given to the simulator. The motivation of this research is to implement and evaluate the performance of EHKM and EEHKM on hardware. This study serves two purposes. First, it identifies

the effects of hardware constraints on the overall performance of the protocol. Secondly, it gives a foundation for the improvement of the simulators. The actual testing results can be fedback into the simulator through a modified network model that closely resembles that of the hardware in terms of energy consumption and delay. This paper presents the results of the hardware implementation of EHKM and a future paper will show the modifications on the simulator and how it better maps to the actual hardware results.

2. OVERVIEW OF EHKM

The hybrid nature of EHKM is a result of running two different key management schemes to handle the different keys along with their varying levels of security requirements. Both dynamic and group wide distribution approaches are used.

EHKM assumes that the nodes will be safe from being compromised for a time T_{min} after deployment. Furthermore, it is assumed that the nodes form a sub-network whenever a sensing event occurs and that the nodes know the number of nodes that are members of the same sub-tree or the same level in the tree. Finally, it is assumed that the nodes employ a self-organizing protocol for selecting the CH nodes such as OEDSR.

In EHKM, three different types of keys are used to support the various levels of security and network applications.

Group-wide keys: Two group-wide keys are pre-deployed into each node. One key is used for group wide communication between nodes that are not involved in the sub-network (K_1). The other key, K_2, is used for the exchange of pair-wise keys as will be explained later.

Individual key: This is a unique key shared between the node and the BS for private communications (K_3). This key is also pre-deployed.

Sub-network key: This key is dynamically calculated whenever the nodes in the network form a sub-network.

At the beginning of the operation of the network, the nodes start the process of sharing a new random pair-wise key (K_r) to be used instead of K_2 that is common to all nodes. This needs to be replaced since the whole network may be compromised if a successful attack on at least one node takes place.

To share the K_r, the nodes transmit *KEY_HELLO* messages that contains the node ID, the new key encrypted using K_2 and a MAC of <node ID, new_key>. Upon reception of the KEY_HELLO message, a node stores the node ID and the new key in a table for future use. Thus, at the end of this process each node will hold the key to be used to encrypt the packets by simply referring to the key table using the destination node ID. After the period of T_{min} each node must clear K_2 from its memory.

The sub-network key management protocol of EHKM is designed in a way to maximize the efficiency of the energy consumption in the network. Here, the nodes collaborate with each other to calculate a sub-network key. The main idea is that all the CH nodes collaborate to select the CH with the highest average energy reserves in its cluster, i.e, each CH calculates the average energy of all the nodes that belong to its sub-tree according to

$$\text{avg\_energy}(i) = \sum(E_{ij}) / n_i \qquad (1)$$

Where i is the cluster index and j is the node index with cluster i.

These values are advertised to all other CH to select the one with the highest average energy as the Head Cluster Head (HCH) according to

$$HCH = CH(\max(avg\_energy(i))) \quad (2)$$

The HCH in cooperation with the other CHs generates a dynamic key using partial keys from the nodes in the network. The HCH checks whether there are sufficient nodes in its sub-tree to generate the desired key. In case there are less than m nodes then the HCH has to use keys produced by the members in the sub-tree of the CH with the second highest average energy. This process is repeated until m nodes are chosen to provide the needed partial keys.

The next step is to initiate the creation of the partial keys using a *START_ALGORITHM* message sent from the CH to the nodes in the sub-tree. The message contains the cluster ID and the depth metric which dictates the level that the message should reach within the sub-tree before a partial key is created. If all the nodes in the sub-tree are required to generate a partial key then the CH sets the depth field to -1. This is actually done in all sub-trees contributing the m keys except for the last one in which the depth field is set to

$$d = m - \sum n_{other\_clusters} \quad (3)$$

Each node that receives the START_ALGORITHM message checks whether the depth field is -1 or not. If it is -1 then it sends the packet to the leaf nodes. Once the leaf nodes receive the message, they generate their partial keys and send them up the hierarchy to finally reach the HCH. For the nodes in the last sub-tree in which the depth (d) is set to some positive value, the node checks how many other nodes are at the same level within the cluster and subtracts that number from d. If new value of d is negative then it does not

need any more contributions from the nodes below in the hierarchy and sends its partial key. Otherwise, it updates the value of d in the START_ALGORITHM message and forwards it.

After the HCH gets all the m partial keys from the participating clusters, it calculates the sub-network key and sends it out to the nodes after encrypting it using the new key (K_r) of the destination node. This key will be transmitted m times and in each time it will be encrypted with the key K_r of one of the nodes that participated with a partial key.

Following is a pseudo-code of the dynamic key calculation [12]

I. After creation of a subnetwork

Set cluster_heads = {Each cluster head}

For each cluster head

set avg_energy(cluster) = average(energy_in_nodes_in_cluster)

set $n_{cluster}$ *= number of nodes in cluster*

Set HCH = {Cluster head: max(avg_energy(cluster_heads)}

Set chosen_cluster_heads = {HCH}

cluster_heads = {cluster_heads} – {HCH}

Set m_temp = m - n_{HCH}

while m_temp ≥ 0

CH_temp = {Cluster head: max(avg_energy(cluster_heads)}

chosen_cluster_heads = {chosen cluster heads} $\cup$ *{CH_temp}*

m_temp = m_temp – n_{CH_temp}

Set n_chosen = 0

Set depth = -1

while |chosen_cluster_heads| > 1

CH_temp = first_element(chosen_cluster_heads)

CH_temp broadcasts start_algorithm {Cluster_ID || depth}

n_chosen = n_chosen + n_{CH_temp}

chosen_cluster_heads = {chosen_cluster_heads} – {CH_temp}

depth = m – n_chosen

CH_temp = only_element(chosen_cluster_heads)

CH_temp broadcasts start_algorithm{Cluster_ID || depth}

II. After hearing a *start_algorithm* message

If node is a leaf

Start_algorithm( ); return

Else

depth = received_depth – n_{ij} (n_{ij} is number of nodes in cluster i on level j)

If depth ≤ 0

Start_algorithm( ); return

Else

Broadcast start_algorithm{Cluster_ID || depth}

3. HARDWARE IMPLEMENTATION OF EHKM

This section gives an overview of the hardware implementation of EHKM. We present a description of the capabilities, limitations and support for networking applications as well as the actual implementation details on the BS and the UMR/SLU motes.

3.1 Overview of Hardware and Associated Limitations

The hardware used was chosen to be energy-conservative, performance-oriented and of small form-factor. For a low-power consumption, fast 8-bit processing and ease of interface to peripheral hardware components, we chose the Silicon Laboratories® 8051 variant. With the external RAM, UART interface and A/D sensing, the microcontroller is capable of performing various tasks done by sensor nodes. The Maxstream XBee™ RF module was used for communication.

To implement the algorithm on hardware, many limitations had to be taken into consideration. The speed, precision and storage requirement all become factors in making a decision on which microcontroller to use. For EHKM, the initial design goals were to limit the energy consumption while providing a reasonable level of security. This was a factor in our hardware implementation as the options of low-power consumption and powerful 8-bit processor architectures are limited. Other factors that were considered are the limitations on the available memory (since EHKM stores different types of keys) and processing speed (encryption/decryption can be time consuming in some cases).

3.2 Sensor Node Hardware

The Generation-4 Smart Sensor Node (G4-SSN) was used to perform the functionality of the sensor nodes in our hardware model. G4-SSN was originally developed at UMR and updated afterwards at St. Louis University (shown in Figure 1). This node has several abilities for sensing such as strain gauges, accelerometers, thermocouples, and general A/D sensing. It is also capable of performing processing tasks such as analog filtering, Compact Flash memory interfacing and 8-bit data processing at a maximum of 100 MIPS. Table 1 gives a summary of the specifications of the G4-SSN [9],[10].

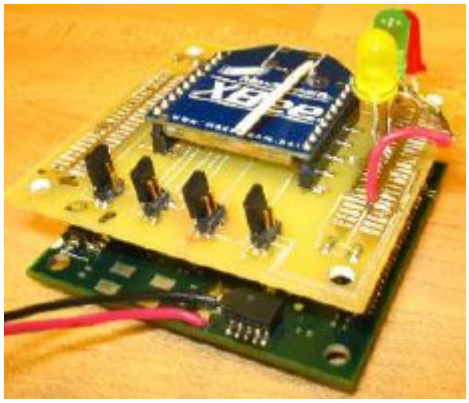

Figure 1. UMR/SLU G4-SSN

Table 1. G4-SSN specifications

	I_c @ 3.3 V [mA]	Flash Memory [bytes]	RAM [bytes]	ADC Sampling Rate [kHz]	Form-Factor	MIPS
G4-SSN	35	128k	8448 bytes	100@10/12-bit	100-pin LQFP	100

3.3 Implementation Details

EHKM extends the software architecture that was developed in [10] by modifying the behavior of the nodes and the control packets they receive from the BS.

Figure 2 shows the software architecture of our hardware implementation which illustrates the three layer approach that is used in this development. The various layers provide flexibility since the layer specific details may be changed with minimal effect on the overall system. The Application layer is responsible of handling all sensor data processing. The Physical layer sets up the serial interface between the microcontroller and the radio module. The layer in between is responsible of the queuing, scheduling, routing, and message abstraction. Note that EHKM falls within the same layer but runs in parallel to the sub-layers.

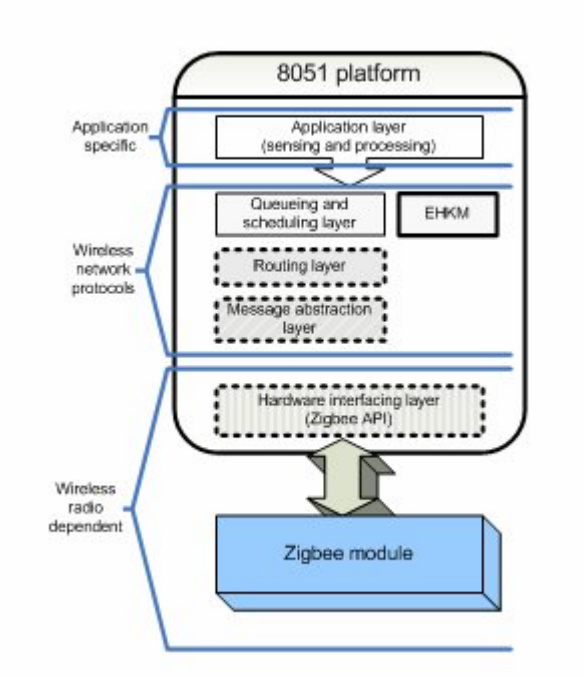

Figure 2. Software architecture of EHKM

Before implementation, we considered the possibility of further enhancements on the protocol. Our first observation is that it is resource consuming to distribute the sub-network key using the unicast pair-wise approach described above. Instead, we propose that a node encrypts the message containing the sub-network key using its own K_r. This message can then be sent as a broadcast message and all the nodes in the sub-network will be able to decrypt it using the key K_r of the HCH.

In addition to the existing packets in the routing protocol implementation in [10], we added several packets that are sent either from the BS to the nodes or among the nodes themselves. These packets are essential for the proper exchange of the various keys in the network. Following is an explanation of the purpose of each packet.

START_KEY_EXCHANGE: The BS initiates the key exchange between the nodes at the beginning of the operational lifetime of the network. This will mark the beginning of the

period running for T_{min} seconds. The nodes proceed with their key exchange upon receiving this packet by sending a KEY_HELLO message.

KEY_HELLO: During the interval that starts when a START_KEY_EXCHANGE is received and ends T_{min} seconds later, the nodes in the network generate their new key K_r to replace the old pair-wise key K_2. The message will be sent to all one hop neighbors in a broadcast mode in the following format.

$$N_i \rightarrow \text{all one hop neighbor: } E_{K2}(\text{Node\_ID}, K_r)$$

During T_{min}, the key K_2 is known by all the one hop neighbors and can be used to decrypt the message. After T_{min} the nodes will delete K_2 from the memory and use K_r Instead.

START_ALGORITHM: This triggers the partial key exchange discussed earlier.

PARTIAL_KEY_EXCHANGE: Each node generates a random partial key and transmits it to the node in the level above it in the sub-tree until it reaches the HCH that calculates the final sub-network key.

SUB_NETWORK_KEY: After the final key is calculated it is sent to the sub-network members in an encrypted packet. The message in EHKM will differ from that of our Enhanced EHKM (EEHKM). For EHKM, the packet is encrypted m times using the key K_r of the intended destination then transmits a unicast to node i (i = 0, 1, 2 …., m). EEHKM, on the other hand, encrypts the packet once using its own K_r and broadcasts it. The recipients of the message are responsible of decrypting it using K_r of the HCH.

Figure 3 gives a graphical explanation of how EHKM and EEHKM handle the various packets for their operation.

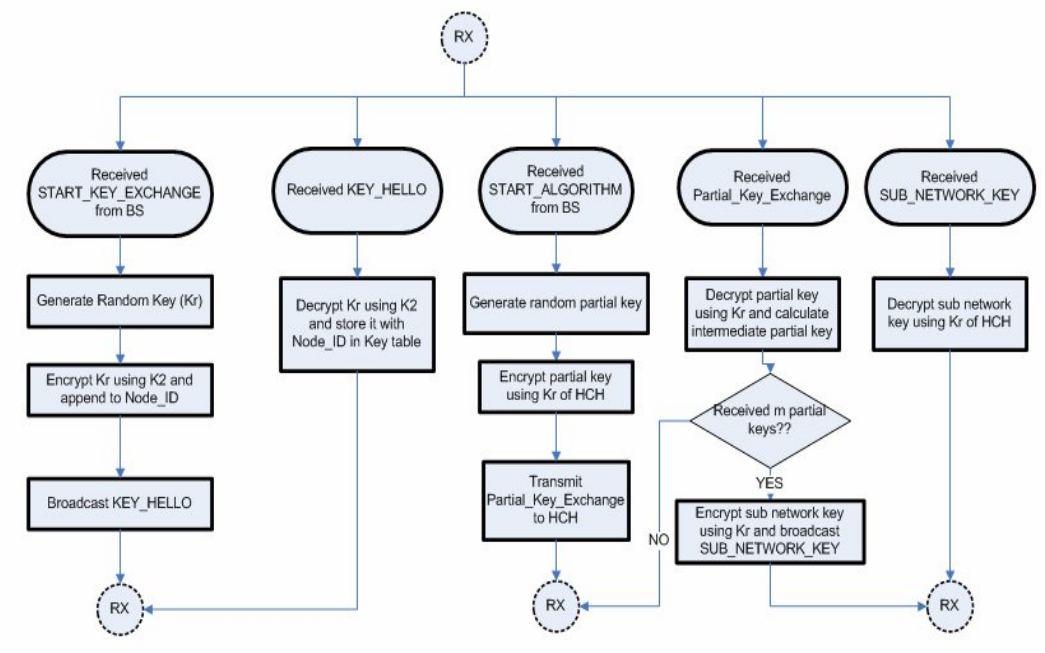

Figure 3. Packet handling in EHKM implementation

Table 2. Average % of routing energy and bit rate for the 3 cases

	OEDSR w/o EHKM	OEDSR with EHKM	OEDSR with EEHKM
Average % Routing Energy	22.5	32.4	29.1
Average kbps rate	6.28	5.84	6.15

4. HARDWARE TEST SETUP AND RESULTS

We ran the Optimal Energy Delay Senor Routing (OEDSR) on the network under test. The choice of the protocol has no effect on the operation of the key management technique and is only responsible of routing the various packets within the network. The sensor nodes employ 802.15.4 modules that transmit at data rate of 250 kpbs and

interfaces to the node processor at 38.4 kbps. These modules run at a low-power of 1-mW whereas the module connected to the BS transmits at 100-mW to extend its range.

The experiments were conducted on a network consisting of 6 nodes. The nodes generated around 3.2 packets/sec with a packet length of 100 bytes of which 12 were header bytes. The experimental cases were

1) Nodes using OEDSR with no key management.
2) Nodes using OEDSR with EHKM.
3) Nodes using OEDSR with EEHKM.

We measured the performance of the network in terms of end-to-end delay, throughput, and percentage of overhead energy consumption.

The first experiment was run 10 times for each of the 3 cases to find the throughput and the change in the energy consumption.

From Table 2 we notice the increase in the overhead energy consumption. This is attributed to the added overhead of packets for the functionality of the key management schemes. The other contributing factor is the encryption of the packets. We note that EEHKM results in better energy management and this is attributed to the reduction of the repeated encryptions and transmissions that were eliminated from the original EHKM.

As for the throughput, we studied the bit-per-second of data that were sent out. We see that OEDSR with no key management gives a higher data rate but EEHKM is relatively close to it while EHKM has the least throughput. This decrease in the throughput may be attributed to the added overhead and the bottlenecks around the CH nodes.

Next, we studied the end-to-end delay to determine the effect of applying the EEHKM. The results are given in seconds in Table 3. We notice that the application of the encryption algorithm increased the average end-to-end delay by about 15%. This is reasonable given the computations required with an algorithm such as the Advanced Encryption Standard (AES). The choice of a lower cost encryption scheme such as the Tiny Encryption Algorithm (TEA) can reduce the change in the delay.

Table 3. Average end-to-end delay comparison

Test	OEDSR w/o EHKM	OEDSR with EEHKM
Test # 1	0.1728	0.2013
Test # 2	0.1537	0.2711
Test # 3	0.1674	0.1792
Test # 4	0.1456	0.1628
Test # 5	0.1911	0.1813
Test # 6	0.1731	0.1623
Average	0.1673	0.1930

5. CONCLUSION AND FUTURE WORK

Two key management protocols were implemented on hardware for performance analysis. The study aimed at investigating the behavior of the protocol on hardware. The results of this work can be applied to the existing simulation model for simulations of different scenarios where a wide area or a high number of nodes is needed.

There is always a tradeoff between applying security in the network and the energy, delay, and throughput. This was the case in this study as we noticed a drop in the throughput and an increase in the end-to-end delay and overhead energy consumption. An increase of about 7% on the energy consumption and around 15% on the delay were observed. These values are relatively low and may be acceptable given the added security in the network.

The nodes demonstrated different behavior than that of what was seen in simulation. For example, the effect of contention was clearly apparent in some initial test when all the nodes started transmitting at the same time. We overcame that problem by setting a random timer.

Our future direction with this work is to improve the random key generation. Currently, our model runs a simple random generator which is clearly not a safe option from a cryptographic point of view. We will investigate the use of an advanced random generator such as the Pseudo Random Number Generator such as Fortuna. Another aspect to be studied is the priorities given to key management packets. Currently, all the packet flow through a queue while we believe that a more suitable implementation is to assign higher priorities for such packets. Finally, we will develop attack scenarios and traffic analysis techniques to study the effectiveness of this protocol.

6. ACKNOWLEDGMENT

We express our sincere gratitude to Mr. James Fonda for his contribution in the hardware tests and analysis scripts. His invaluable assistance is appreciated.

7. REFERENCES

[1] Anderson, R., and Kuhn, M., Tamper Resistance – A Cautionary Note. In *Proceedings of the second USENIX Workshop on Electronic Commerce,* Oakland, CA, USA, November, 1996.

[2] Basagni, S., Herrin, K., Rosti, E., and Bruschi, D., Secure Pebblenets. In *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001),* Long Beach, CA, USA, October, 2001, 156 – 163.

[3] Blundo, C., Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., and Yung, M. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology, Proceedings of CRYPTO'92*. LNCS 740, 1993, 471–486.

[4] Carman, D., Kruus, P., and Matt, B. Constraints and Approaches for Distributed Sensor Network Security. NAI Labs Technical Report #00-010, September 2000.

[5] Chan, H., Perrig, A., Song, D. Random Key Predistribution Schemes for Sensor Netoworks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)*, Oakland, CA, USA, May, 2003.

[6] Culler, D., Estrin, D., and Srivastave, M. Overview of Sensor Netowrks, In *IEEE Computer, 37, 8 (2004), 41 – 49.*

[7] Du, W., Deng, J., Han, Y., Chen, S., and Varshney, P. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. In *Proceeding of the 23rd Conference of the IEEE Communications Society (INFOCOM '04*), Hong Kong, March, 2004.

[8] Eschenauer, L, and Gligor, V. A Key-Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communication Security,* November, 2002.

[9] Fonda, J., Zawodniok, M., Jagannathan, S., and Watkins, S. Adaptive Distributed Fair Scheduling and Its Implementation in Wireless Sensor Networks, In *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, Taipei, Taiwan, 2006, 3382-3387.

[10] Fonda, J., Zawodniok, M., Jagannathan, S., and Watkins, S. Development and Implementation of Optimized Energy-Delay Sub-network Routing Protocol for Wireless Sensor Networks, *In Proceedings of the IEEE International Symposium on Intelligent Control*, Munich, Germany, 2006, 119-124.

[11] Kim, Y., Perrig, A., Tsudick, G. Tree-Based Group Key Agreement. ACM Transactions on Information and System Security, 7(1): 60 – 96, February, 2004.

[12] Landstra, T., Zawodniok, M., Jagannathan, S. Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks. In *Proceedings of Workshop on Distributed Systems and Networks,* Bern, Switzerland, May, 2006.

[13] Panja, B., Madria, S. and Bhargava, B. Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks. In *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006)*, Taichung, Taiwan, June, 2006.

[14] Price, A., Kosaka, K., Chatterjee, S. A Secure Key Management Scheme for Sensor Networks. In *Proceedings of the Tenth Americas Conference on Information Systems,* New York, NY, August, 2004.

[15] Zhu, S., Setia, S., and Jajodia, S. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington D.C., October, 2003.

APPENDIX

CONTRIBUTION AND FUTURE WORK

This work was motivated by the lack of sufficient studies dealing with link state routing in wireless ad hoc networks. OLSR was taken as a case study and we found several drawbacks in the initial proposal in terms of its energy awareness, end-to-end delay reduction and sustainable service.

The goals of this research were set as follows:

1. Study the security risks and their effect on the performance of a network running OLSR and OEDR.
2. Introduce an extension of the current protocols that involves trust levels to mitigate these risks.
3. Analyze the reliability and survivability of the suggested protocol and its overall effect on the network performance.
4. Implement a key distribution algorithm for wireless sensor networks.

In chapter I, we introduced TLR which is an extension that involves trust levels for trusted routing in an ad hoc network. The chapter contributes with a better understanding of the security threats facing a network designer and also suggests several methods to overcome them. The chapter shows both analytically and through simulation that the suggested extension results in an improved performance of the network in addition to the added security.

Chapter II presented an analysis of TLR in comparison to OLSR. This analysis was based on reliability and survivability. We applied the Markov analysis to the network and

modeled the system with Markov graphs. This chapter gives the reader a deeper appreciation of the enhancements achieved by using TLR.

Chapter III gave a different approach for handling trust in terms of the selection of relay nodes in a network. It borrowed methods from statistical analysis and showed how they can be used in running a network without any Trusted Third Party.

Finally in chapter IV, we implemented a key distribution algorithm for wireless network on hardware. The G4-SSN motes that were initially developed at UMR were used for testing purposes. We extended an existing key management scheme to efficiently use the limited energy and memory resources.

The work in the four chapters of this dissertation has led to even more interesting research ideas to be investigated. A few examples of these ideas include

1. A study of a dynamic approach for weight selection in the TLR cost calculation. Currently, the model runs on two preset values that do not adapt to the network conditions. A combination of the work in chapters I and III can lead to a better identification of trusted paths which can help in dynamically changing the weights based on the presence, or lack of, trusted routes.
2. The results achieved in chapter two can be fortified with real world data and hardware tests using the hardware benchmark that was developed for chapter IV. This form of testing can be used for practical comparison between the two protocols.
3. The work in chapter IV will be used for network parameters measurement for simulation enhancement. For example, by running the nodes and measuring real-time data such as energy consumption, delay, congestion, and packet drops, we

can use that information to infer a better model to represent the nodes in the simulator instead of the default model of NS2. This should help in having more realistic simulation results and also can be used in cases when networks of large size are to be tested. In such a case, the hardware implementation may not be practical due to limited space, lack of sensor nodes for testing or difficulty in setting up the test.

VITA

Eyad Taqieddin was born on Dec 26, 1976 in Amman, Jordan. After receiving his primary and secondary education in Irbid-Jordan, he enrolled in Jordan University of Science and Technology (JUST) to receive his Bachelor of Science (B.S.) in Electrical Engineering in June 1999. In that same year, he was employed by JUST as a web developer. He enrolled in the department of Electrical and Computer Engineering at the University of Missouri-Rolla (UMR) in Fall 2000 where he received his M.S in Computer Engineering in August 2002. He later enrolled in the Ph.D. program at UMR. During his tenure as a graduate student, he worked as a Graduate Research Assistant and Graduate Teaching Assistant in the Department of Electrical and Computer Engineering. He received his PhD in May 2007. Currently, he is a faculty member at the Computer Engineering Department at JUST.