Proc. 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)
San Francisco, CA, April 10, 2016, pp. 846 - 851.

Trust Based Secure Routing in Delay Tolerant Networks

Thomas A. Babbitt
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180
Email: babbit@rpi.edu

Boleslaw K. Szymanski
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12180
Email: szymab@rpi.edu

***Abstract*—There is a growing need for secure and robust networks that function in austere environments or with nodes that have limited resources. The prime example is Delay Tolerant Networks that have military, first response, infrastructure failure and vehicle applications. This class of networks requires the modification of traditional routing and security protocols, such as TCP/IP or the use of certificate revocation servers, to effectively transmit a message from source to destination in a secure manner. There are multiple trust and routing schemes proposed for use in Delay Tolerant Networks. Here, we show how trust, usually defined in the context of network science, can be utilized for routing in Delay Tolerant Networks in which the fundamental challenge is balancing energy, delivery rate, delivery delay and security. Trust is a cornerstone of our approach to address this challenge. We propose a Trust Based Secure Routing Protocol, in which forwarding decisions are based on nodes trust, expected destination meeting time and the number of already created copies of the message.**

I. Introduction

In all Delay Tolerant Network (DTN) routing protocols, a node must decide which buffered messages to forward upon meeting with another node. This decision can be based on a count of already forwarded copies [1]; the likelihood of a node forwarding it to the destination based on it's "closeness" to the destination versus that of the current holder [2], [3]; trust value [4]–[6] or some combination of the three. In the first approach, the number of copies is bounded by some function. The larger the number of copies forwarded the more energy is used for delivery and the higher exposure to potentially bad nodes. At the same time, there are benefits of a higher delivery rate and shorter delivery delay. Extreme node copy values are 1 (source only forwarding directly to the destination) and unlimited (each met node receives a packet copy and shares with all met nodes, also called epidemic routing). The second approach uses the expected time to delivery of the two nodes that met and the node holding the packet(s) considers passing a copy only if its expected time to meet the destination is higher than that of the met node. This, of course, when done correctly, limits the number of copies being made without much increase in the delivery time or decrease in delivery rate. Finally, trust is used mainly as a means to limit exposure to bad nodes.

This work was supported in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies either expressed or implied of the Army Research Laboratory or the U.S. Government.

At the highest level of abstraction, novelty of our approach includes three aspects of routing in a DTN:

1) We use trust not only to limit exposure of a message to bad nodes but also to evaluate the predicted meeting time of another node. We accomplish this by modifying the reported time by a factor increasing with the inverse of the trust to the node reporting this time. Hence, the trust in our approach impacts two metrics, the destination meeting time, which is trust adjusted, and probability of exposure to malicious nodes.
2) We use parameters of our protocol as arguments of a control function that maps them into metrics of performance: (i) energy used for delivery, (ii) delay time to delivery, (iii) delivery probability, and (iv) the total exposure to bad nodes. Using multiple simulations, we obtain piece-wise linear approximation of this function (which we call "Situational Risk" (SR) mapping) for the specific DTN.
3) Using SR, packets flow with various priorities using different parameter setting, this enables optimal routing for each packet based on selected classification.

The network science aspects of our approach include considering the impact of routing on the dynamic process of trust convergence. To account for it, the protocol allows sending some messages via nodes with low trust. In future work, we plan also to study the impact of network structure imposed by different node mobility patterns on optimal settings of our protocol parameters.

The technical details of our approach are described in the subsequent sections. The rest of this section demonstrate that no routing protocol for DTNs in literature proposes such comprehensive and flexible framework for routing traffic of packets with different priorities.

Assume node k has a packet to deliver. Based on routing protocol, this node determines whether or not to forward to another met node i based on trust. The authors in [5] propose a single-copy routing protocol that forwards a packet if trust is above a given threshold and in the top Ω percentile of all known trusts. The authors in [4], [6] expand spray and wait, a multi-copy routing protocol, to use trusted nodes. In

the former, only trusted nodes receive a packet during the first spray period. Only if the packet is not delivered at the start of the second spray period less trusted nodes are used. For the latter, the number of copies of a message forwarded is modified based on trust. For known non-trusted nodes no copy is sent. For suspect nodes or those with unknown trust, one copy is sent. For more trusted ones additional copies are forwarded. Each of the aforementioned approaches uses trust to avoid exposure of the packet to a malicious node, but not to adjust information received from a node.

II. Trust Based Secure Routing

The proposed Trust Based Secure Routing Protocol (TBSR) for use in Delay Tolerant Networks uses node trust, estimated time to meet the destination and number of copies of a message to make routing/forwarding decisions. TBSR is designed to work with any trust management scheme that converges and maintains a valid distributed trust that is dynamically updated. The trust input is a value between 0.0 and 1.0 where 0.0 is complete lack of trust and 1.0 is implicit trust. The lower the value the higher the likelihood of a node being isolated or bypassed when forwarding decisions are made. Any trust management scheme satisfying the above conditions can be applied to TBSR. The authors in [5] and [7] have proposed and shown different methods adaptable to TBSR.

In a DTN when nodes i and k meet, they first conduct a handshake to establish a communication link. While TBSR does not put limitations on protocol used to establish this link, most implementations creates a TCP or UDP connection. Each node party to the meeting decides which, if any, messages to forward. Typically, the trust management scheme (e.g., [7]) adds the final part of the handshake by trading trust information, in the form of a trust vector. This consists of node i's trust value for all nodes in the network and vice-verse. We extend the trust vector to a $n \times 2$ matrix, designated tm^i, which includes the estimated times at which node i will meet all other nodes in the network. An individual time estimate given by node i is t_j^i for a given node j in the DTN. With a valid trust and an estimated next destination meeting time, node k can make a decision on whether or not to forward a given message m to node i.

Once node k completes the handshake with node i, it then must decide which buffered messages to forward. If node k has multiple messages to forward, it might decide to forward all or some subset of them. We chose to use estimated destination meeting time in our risk calculations. Another valid approach that merits additional research is the modification of that time based on social interaction or network structure, for example using community affiliation tags.

Node k's decision to forward a message m to node i is based on the following risk assessment: node k calculates the "Situational Risk" (SR) (Section II-A) and determines based on SR if it is better to forward or hold m (Section II-B) (ii) node i determines if it is better to buffer or drop m (Section II-C). If node k has low trust for node i, then it should only forward a message if node i will meet the destination significantly faster or if the message has a low security classification. Conversely, if node i has a low trust of node k, it should limit the number of messages it accepts from node k.

Since node i is ignorant of node k's buffer, it is difficult to falsify time estimates credibly. Reconstructing network topology [8], using time estimates, might allow nodes to verify mobility model and identify inconsistencies of time estimates. It is a direction for future research and was a consideration in the construction of our SR function.

A. Determine Situation Risk

1) Overview: Node k determines the "Situational Risk" (SR) associated with sending a message to node i using Equation 3. The results for SR are a value in the range $0.0 \leq sr^k_{(i,j)}(m) \leq 1.0$. In all cases, trust that node k has for node i is adjusted based on the number of copies of the message node k previously sent, plus one to account for sending a copy to node i ($nc+1$). The number of copies (nc) is multiplied by a weight $0.0 \leq w_{nc} \leq 1.0$ that adjusts the effect of the number of copies on SR. Node k's trust for node i adjusted by the number of copies is our first dimensional use of trust (probability of using a malicious node).

A value $tm^k_{(i,j)}$, denoting how many times faster or slower node i versus node k meets the destination, determines if further adjustments are required (Equation 1). This value is determined based on the following equation:

$$tm^k_{(i,j)} = \frac{t^k_j - t^k_{adj(i,j)}}{min\left(t^k_j,\ t^k_{adj(i,j)}\right)} \tag{1}$$

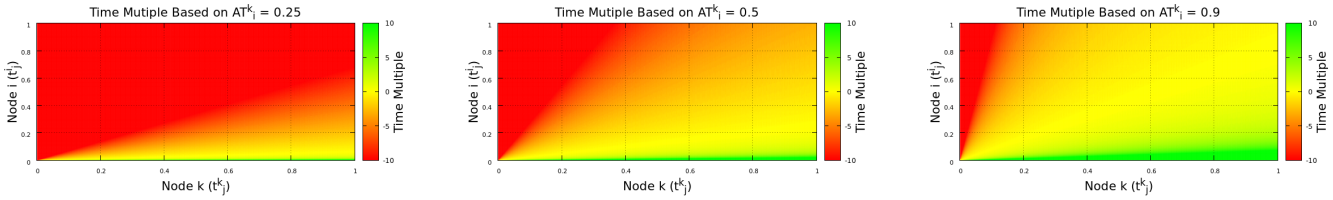
that takes into account the time node k is expected to meet the destination (t^k_j) and an adjusted time, based on trust (AT^k_i), for when node i is expected to meet the destination ($t^k_{adj(i,j)}$). The adjusted time is calculated using the following equation:

$$t^k_{adj(i,j)} = \frac{t^i_j}{{AT^k_i}^2} \tag{2}$$

that modifies the expected destination meeting time node i provides based on node k's trust in node i. Equation 2 represents the second dimension in which we use trust (trust adjusted next destination meeting time).

$$sr^k_{(i,j)}(m) = \begin{cases} \left(AT^k_i{}^{w_{nc} \cdot nc+1}\right)^{\frac{1}{tm^k_{(i,j)}}} & tm^k_{(i,j)} > 1 \\ \left(AT^k_i{}^{w_{nc} \cdot nc+1}\right)^{|tm^k_{(i,j)}|} & tm^k_{(i,j)} < -1 \\ AT^k_i{}^{w_{nc} \cdot nc+1} & \text{Otherwise} \end{cases} \tag{3}$$

There are three cases for SR in Equation 3. The first case occurs when, based on time difference, it is significantly better to forward, $tm^k_{(i,j)} > 1$. This will increase adjusted trust to a higher value, but smaller than 1.0. The second case occurs when $tm^k_{(i,j)} < -1$ and it is significantly better to wait causing $sr^k_{(i,j)}(m)$ to be a small fraction of the adjusted trust and it tends to zero as trust decreases. The final case occurs when



(a) $AT_i^k = 0.25$ (b) $AT_i^k = 0.5$ (c) $AT_i^k = 0.9$

Fig. 1. Time Multiple for Node k Using Varying Trust of Node i (AT_i^k)

$-1 \leq tm_{(i,j)}^k \leq 1$. In this case, there is less than one time interval difference between the trust adjusted and expected time that node i or node k will meet the destination, resulting in no modification.

2) Analysis: There are four variables, in Equation 3, that adjust SR and determine whether node k forwards a message to node i. These include node k's trust of node i (AT_i^k), the number of times node k has forwarded a message (nc), the weight given to the number of copies (w_{nc}), and the number of time intervals by which the time delay increases or decreases by holding versus forwarding a message segment (Equation 1). Valid values for w_{nc} are $0.0 \leq w_{nc} \leq 1.0$; the smaller the value of w_{nc} the less weight is given to the number of copies. When $w_{nc} = 0.0$ then the number of copies has no effect on SR. Unless otherwise stated, $w_{nc} = 1.0$. Node k stores trust and nc in it's buffers and, while they ultimately effect the SR, Equation 1 yields significantly different values based on the times sent during the handshake.

Figure 1 shows a comparison of results for Equation 1 where node k has various trust levels for node i. The x-axis for each heat map is the node k estimate of the time by which it will next meet the destination node j, and the y-axis is node i's estimate of such time. The color is the z-axis that represents time corresponding to multiple values of $tm_{(i,j)}^k$ computed from Equation 1. Subfigures 1a, 1b and 1c show $tm_{(i,j)}^k$ with $AT_i^k = \{0.25,\ 0.50,\ 0.90\}$ respectively. The area in each subfigure that is shaded red represents values that will make $sr_{(i,j)}^k$ smaller, while the green shaded area contains values that do the opposite. The area shaded yellow covers values close to $tm_{(i,j)}^k = 0$ that have minimal effect on $sr_{(i,j)}^k$.

Any chosen metric that guides forwarding must have the property that the lower a given nodes trust, the bigger advantage it must have in reduction of delivery time to the destination to be consider for forwarding. There are a number of reasons for this, that include that a low trusted node is more likely to lie, or that the link is faulty increasing the chance that a message will be lost. Equation 3 is designed to have these properties that become evident when node k's trust for node i is low ($AT_i^k < 0.5$). Subfigure 1a is mostly red and since the trust is low, this ultimately lowers the SR and essentially eliminates nodes with trust values from this area from contention for forwarding. Even the trust values from the green area are unlikely to result in forwarding, because node i has to expect approximately 1000 times faster delivery than node k for forwarding to happen. The motivation for requiring such vast improvement is that a node with low trust is more likely to lie to get preferential treatment. Subfigures 1b and 1c have a larger area that are yellow and only when a node is significantly faster or slower in meeting the destination is the forwarding decision affected by the value of $sr_{(i,j)}^k$. While not shown here, by adjusting w_{nc}, the range that is shaded yellow can be made smaller. This is the expectation for efficient routing that should forward data to trustworthy nodes, excluding those that are not likely to meet the destination more quickly than the sender.

B. Risk Assessment and Forwarding Decision

The previous section provides analysis of the SR associated with forwarding a given message from node k to node i. The higher the value, the lower the risk of forwarding the message. Nodes with low trust are excluded unless they are expected to deliver a message to the destination significantly faster than the current message holder. Nodes with high trust are likely to receive messages for forwarding unless their expected destination meeting time is larger then that of the sender. This leads to a number of potential algorithms to decide on whether node k forwards a message to node i. In all cases, the node computes $sr_{(i,j)}^k(m)$ using Equation 3 first. It then forwards the message if either (i) a number uniformly randomly picked between 0 and 1 is less then $sr_{(i,j)}^k(m)$ or (ii) if both $sr_{(i,j)}^k(m)$ and node k's trust for node i (AT_i^k) are above a threshold. Alternatively, a combination of 1 and 2 may be used.

The first proposal allows for all nodes to have a chance to forward a message and does not completely "blacklist" a node. This allows for the distributed trust management schemes to continue to update trust for all nodes. The second solution forwards messages to all nodes with trust above a threshold that also are expected to meet the destination soon. The downside is that any node below a set threshold is "blacklisted" and does not have an opportunity to modify it's trust. This can have a significant effect on nodes that are isolated for a time period. Trust decays over time and that could cause a given node to go below the trust threshold.

Figure 2 shows an algorithm for the final solution and allows for both a threshold and nodes with lower trust to forward message segments. This is the approach used in TBSR. If the SR is above a set threshold (Tr_r), and the trust that node k has

```
1: procedure FORWARDDECISION(t_j^i, t_j^k, m)
2:     Compute sr_(i,j)^k(m)                          ▷ Equation 3
3:     if sr_(i,j)^k(m) ≥ Tr_r and AT_i^k ≥ Tr_r then
4:         return forward = TRUE                      ▷ Forward
5:     else
6:         r = random(0.0..1.0)
7:         if sr_(i,j)^k(m) ≥ Tr_r and r ≤ Tr_a then
8:             return forward = TRUE                  ▷ Forward
9:         else
10:            return forward = FALSE                 ▷ Skip
11:        end if
12:    end if
13: end procedure
```

Fig. 2. Algorithm: Node k Forwarding Decision (TBSR)

for node i is above that same threshold, then the message is forwarded. If the SR is above the threshold because node i will see the destination soon, but node k's trust in node i is low, then depending on the adaptive threshold (Tr_a) the message is forwarded or not. This gives a node with low trust a chance to participate to increase trust, but it does so in a manner that limits the overall network risk. Based on message classification and user requirements, Tr_r and Tr_a can be modified to accept the appropriate risk. The former lowers the threshold for what is considered a good versus bad node and the latter modifies the chance of forwarding to a malicious node.

C. Assessing Risk in Decision to Buffer a Message

So far, the focus has been on TBSR making the best forwarding decision. In addition to forwarding, a receiving node has to decide whether or not to buffer a received message. If the sending node broadcasts a message then the receiving node will, at a minimum, processes it. Buffering a corrupt message can negatively effect a nodes trust and uses limited resources (e.g. battery power) processing and forwarding. If there is a high enough chance of corruption, then a message probably should be ignored and not buffered.

There are a number of possible solutions: use a threshold and only accept messages from a node with a trust above that threshold; randomly choose which nodes to accept messages from based on trust; or a combination of both. The first approach creates a good base. Any node below a certain trust threshold should, in most instances, be ignored. The downside is that it "blacklists" all nodes with lower trust from not only forwarding message but also creating and sending messages. These trust values can be due to malicious activity or hardware/isolation. In order to maintain dynamic trust, some nodes with lower trust need to interact and be allowed to forward a message. The purely random approach drops messages from trusted nodes and, hence, was not seriously considered.

The final approach was implemented and it is a combination of the previous two (Figure 3). Using this approach a node automatically accepts messages from more trusted nodes and,

```
1: procedure BUFFERDECISION(m)
2:     r = random(0.0..1.0)
3:     if AT_k^i ≥ T_r_r or r ≤ T_r_a then
4:         return buffer = TRUE                       ▷ Store in Buffer
5:     else
6:         return buffer = False                      ▷ Delete From Buffer
7:     end if
8: end procedure
```

Fig. 3. Algorithm: Node i Message Buffer Decision (TBSR)

like in the previous section, allows suspected bad nodes to be utilized based on Tr_a and a uniformly random selected number between 0.0 and 1.0. There could be situations where the only path from source to destination is through a node with lower trust.

III. Simulation Comparisons

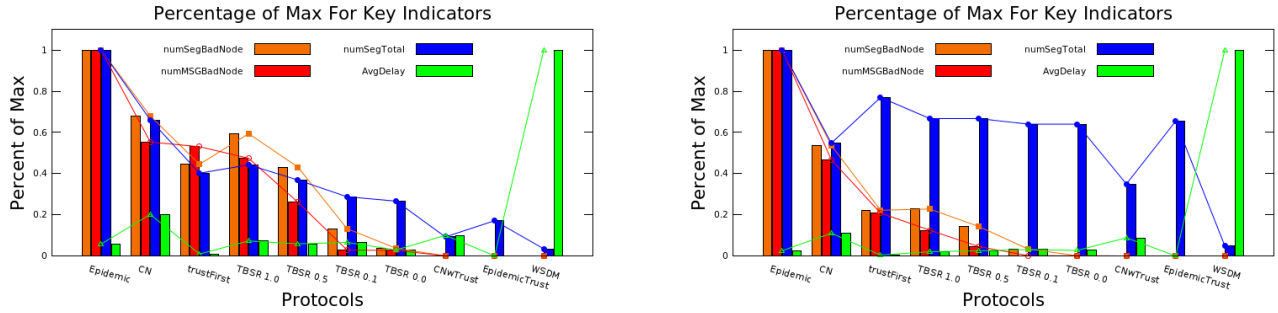
In order to demonstrate the benefits of TBSR, a number of simulations were conducted to determine improvements in security and the cost of doing so. Using an epidemic routing scheme is expected to produce the lowest delay. Since every node met receives a copy of the message, the fastest path is used, but the risk of data compromise is maximized. Reducing the number of copies of a given message will increase message delay, but lower the risk of data compromise. As long as that delay does not hinder an application, it is better to be selective in sharing a message with met nodes.

Risk increases with each message received by a bad node as it can be used for malicious purposes. Once the adversary has the message, even if encrypted, it can potentially break the encryption or glean useful information just knowing about the source and destination and how often they communicate. Our goal is to limit the delay, while reducing the number of message segments a bad node receives (this also decreases the number of broadcasts used in message transmission lowering the energy used for communication, a positive side effect of this solution). Hence, the solution seeks a trade-off between security but large average delays due to limited message sharing, and speed but low security and high energy use of more promiscuous sharing.

A. Simulator Overview/Execution

We built a discrete event simulator using the principles outlined in [9] in order to show the value of our approach. The simulator creates a complete graph based on the user defined network, if arbitrary, size n. The edge weight is the inverse of the average inter-meeting time between any two nodes. If this value is 0.0 then the connected nodes never meet while with 1.0 they are in constant broadcast range all the time. By uniformly randomly selecting the edge weight we simulate a random mobility pattern.

There are three event types that are managed by the simulation event queue: meeting, trust update and message initiation. A meeting event occurs when node i and node k are within broadcast range, conduct a handshake and as protocol dictates



(a) Percentage of Good Nodes is 60%

(b) Percentage of Good Nodes is 90%

Fig. 4. Protocol Comparisons

trade message segments. A node trust update event occurs every Δt time period. This is user defined and follows the rules outlined in [7]. A message initiation event occurs when a node places a message in its buffer for transmission to another node. The user can designate how often a node creates a message. By default that message is sent randomly to another node in the network.

Each node is initialized with a vector consisting of the Aggregate Trust (AT) for all other nodes in the network as a value between 0.0 and 1.0. The user defines the fraction of good nodes in the network. To simplify tracking and illustration, we use the highest node id numbers for bad nodes. Since our simulation represent a more mature network where trust has converged due to previous network activity, during simulation initialization the AT for nodes designated as bad start with a trust of 0.25 and those as good with a 0.75 trust.

To initialize the network, the event queue is initialized and each node determines when it will meet all other nodes using a Poisson distribution and Equation 4 [7]. The edge weight between node i and node k is designated $w_{i,k}$ and $mTime_{i,k}$ is the current expected meeting time of nodes i and k. For simulation initialization $mTime_{i,k} = 0$. Once the meeting time is calculated, the meeting events are added to the event queue. For the initial node update event, each node will calculate $\Delta t * U(0,1)$. The user selected message send time t_{ms} divided in half becomes the average time in which each node will send a message. Each node will send its first message at $t_{ms} * U(0,1)$.

$$mTime_{i,k} = mTime_{i,k} - \left(\frac{1}{w_{i,k}} \times ln([0,1])\right) \quad (4)$$

Once the simulated network is initialized, each event is pulled from the event queue and executed in chronological order. Node update events use the vector approximation approach outlined in [7] to track and update trust. Message initialization events add another message to a randomly selected destination at $ct + t_{ms} * U(0,1)$ (ct is current simulation time). Meeting events check to see if either node is busy; determine broadcast time; trade messages; process new messages; and determine next meeting. While our simulator abstracts away many network details, a given node can only broadcast and trade messages with one node at a time and is barred from additional meetings during the time it would take to trade messages. We assumes a 100Mbps connection, $(100 + 4 \cdot n)$ Byte handshake, and 1000 Bytes per message traded.

B. Simulation Results

Using the simulator discussed above, we ran a number of simulation sets. The results consist of the average of ten iterations for each configuration. Each simulation runs for 1000 time units with trust initialized based on a nodes status as either good or bad as described above. Each node sends a message randomly on average every 50 time units. Trust is updated using the trust management scheme in [7] and messaging utilizes erasure coding. As a threat model, malicious nodes will modify received packets prior to forwarding. While this is limited, the goal is to minimize messages forwarded to bad nodes. For each simulation we report the following: (i) Number of Bad Segment (*NumBadSeg*) counting unmodified message segments that arrive at a bad node; (ii) Number of Bad Messages (*NumMSGBad* that a bad node would have been able to recreate during the simulation; (iii) Total number of Segments (*TotalSeg*) that are broadcast; (iv) Average Delivery Delay (*AvgDelay*) measuring the average time taken from when a message is created at the source until it is delivered to the destination.

Figure 4 shows our initial results. There are multiple protocols shown that include the following: epidemic; any node closer to destination (CN); any node closer to the destination with a higher trust (CNwTrust); wait and send it to the destination (WSDM); two spread periods (trustFirst) [4]; and TBSR with various values for Tr_a. For each subfigure they are sorted left to right using *NumMSGBad*. As expected, epidemic is on the far left with the highest security issues (*NumBadSeg* and *NumMSGBad*) and energy use (*TotalSeg* maximized). It has one of the lowest average delays (*AvgDelay*). WSDM is on the far right with the lowest security issues and energy use but the highest average delay. The values for CN and CNwTrust show some of this trade-off. Using only closer nodes adds additional security risks and energy. Adding in trust reduces security risk and energy, but does not allow for

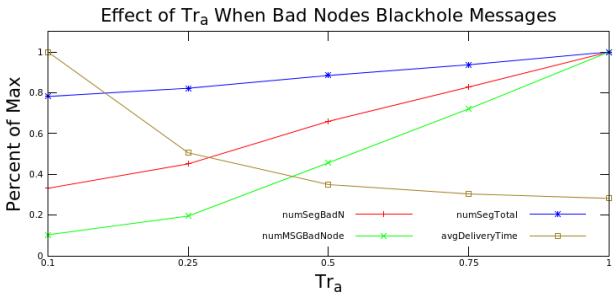


Fig. 5. Percent of Nodes Bad 60%, Bad Nodes "Blackhole" Attack

dynamic changes in trust. The middle results for trustFirst and TBSR try and better balance the needs of security/energy versus requirements for delivery time.

1) Fraction of Good Nodes = 60%: When the network is highly polluted, with fraction of good nodes 60%, TBSR outperforms trustFirst for all security and energy metrics (*NumBadSeg*, *NumMSGBad* and *TotalSeg*) but has a larger *AvgDelay*. Subfigure 4a shows this inverse relationship where the trade off between energy use, security and delivery delay are seen. There is a 20% decrease in number of bad segments, 24.6% decrease in messages to bad nodes and the total number of segments sent is 44.6% lower. These will decrease the energy use and increase security, but it comes with a 40.1% increase in average delivery time; however, the resulting delay of 0.5 time units might be acceptable to the application in view of the increased energy savings and lower risk. As Tr_a increases the metrics converge toward trustFirst and when $Tr_a = 1.0$ it is slightly worse.

2) Fraction of Good Nodes = 90%: When a network has a high fraction of good nodes the adaptive approach works even better than when the fraction of good nodes is lower. There is a decrease in bad segments and message numbers as well as average delay; however there is an increase in the total number of segments. When $Tr_a = 0.1$ there is a 41.5% decrease in number of bad segments, 50.0% decrease in bad messages and a 17.9% decrease in delay. The only increase is in the total number of segments sent. This is due to nodes with higher trust having a better chance of receiving a message as the number of copies increase. This happens often in a network with a higher number of good nodes.

3) Extended Threat Model: Bad Nodes Conduct "Blackhole" Attack: In this section, the threat model is extended such that all bad nodes "blackhole" all received messages; bad nodes still send messages on average every 50 time units. Figure 5 shows preliminary results when the percentage of bad nodes is 60%. There is an inverse relationship between energy *numTotalSeg* and security (*NumBadSeg* and *NumMSGBad*) versus delivery time as Tr_a increases. Security risk and energy use are maximized when $Tr_a = 1.0$ while delivery time is minimized. Since we assume some risk ($Tr_a \geq 0.1$), energy and security do not go to zero. This trade-off allows for routing differentiating between message categories. Some messages require maximum security while others are more time sensitive and payload discovery presents a minimal risk. In addition, nodes with intermittent connections, or those new to the network, might have lower trust without being malicious. Message classification allows for all nodes to forward/receive messages and modify trust.

IV. Future Work and Conclusion

Here, we present a novel approach to making forwarding decision in a DTN. The Trust Based Secure Routing Protocol allows for different risk levels to be associated with different message categories. It limits the number of copies of a message and preforms well in reducing both the exposure (number of segments sent to bad nodes) and the number of total messages sent (energy saving) while limiting the resulting message delay time increase. TBSR works with any distributed trust management scheme that provides a valid trust value between 0.0 and 1.0. It also performs well under the extended threat model in which all bad nodes "blackhole" all received messages.

Trade-offs between energy, average delay time, and security, suggest a number of additional research directions. These include refinement of the SR formula, conducting a more complete comparison of TBSR to other DTN routing protocols, using additional threat models, exploring different mobility patterns, further analysis on the weight given to the number of message copies, message classification with key SR parameter proposals, use of trust patterns and the reconstruction of topology, and using the traded proposed time estimates as a clue for mobility modeling and trust analysis.

References

[1] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE/ACM Trans. Netw.*, vol. 16, no. 1, pp. 77–90, Jan. 2008.

[2] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 387–401, Jan. 2013.

[3] T. Le, H. Kalantarian, and M. Gerla, "A novel social contact graph based routing strategy for delay tolerant networks," in *2015 Int. Conf. Wireless Commun. and Mobile Computing*, Aug. 2015, pp. 13–18.

[4] E. Bulut and B. Szymanski, "Secure multi-copy routing in compromised delay tolerant networks," *Wireless Personal Commun.*, vol. 73, no. 1, pp. 149–168, Jan. 2013.

[5] I.-R. Chen, F. Bao, M. Chang, and J.-H. Cho, "Dynamic trust management for delay tolerant networks and its application to secure routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1200–1210, May 2014.

[6] A. Al-Hinai, H. Zhang, Y. Chen, and Y. Li, "Tb-snw: Trust-based spray-and-wait routing for delay-tolerant networks," *The J. of Supercomputing*, vol. 69, no. 2, pp. 593–609, Feb. 2014.

[7] T. A. Babbitt and B. Szymanski, "Trust metric integration in resource constrained networks via data fusion," in *Proc. 18th Int. Conf. Inform. Fusion (Fusion '15)*, Washington, DC, Jul. 2015, pp. 582–589.

[8] J. Baumes, M. Goldberg, M. Magdon-Ismail, and W. A. Wallace, "Discovering hidden groups in communication networks," in *Intell. and Security Informatics: Proc. 2d Symp. Intell. and Security Informatics, ISI 2004, Tucson, AZ, June 10-11*, H. Chen, R. Moore, D. D. Zeng, and J. Leavitt, Eds. Berlin, Germany: Springer, 2004, pp. 378–389.

[9] J. Gross and M. Günes, "Introduction," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Günes, and J. Gross, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 1–11.