# Trust Management in Delay Tolerant Networks Utilizing Erasure Coding

Proc. IEEE International Conference on Communication, ICC, London, UK, June 8-12, 2015, pp. 7959-65.

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 need for robust networks in all environments including austere ones. The prime example is Delay Tolerant Networks (DTN) that are subject to a growing body of research. These networks support applications used by the military and first responders, especially in emergency situations. Securing a DTN is not a trivial undertaking due to node mobility and the ADHOC method in which nodes communicate. Here, we propose a distributed trust management scheme to secure this class of networks with many of the underlying principles applicable to the larger class of Mobile ADHOC Networks (MANET). The scheme employs erasure coding not only to increase delivery rate in a DTN but also to infer the trustworthiness of the nodes along all paths that deliver a message segments to the destination. The proposed approach enables us to decide when the destination node should stop waiting for additional segments and instead request message retransmission. Moreover, even after the message is recreated successfully, additional segments received enable the destination to collect precious trust information about the nodes involved in delivery of these segments. We show how distributing this trust information identifies compromised nodes.**

## I. Introduction

There is a large and growing need for reliable and secure delay tolerant networks (DTN). These networks are important in a number of situations where well established networks either do not exist or are not functioning properly. The military preparing to deploy into hostile environments with austere communication infrastructure and emergency responsers called upon to provide assistance in an area with potentially unreliable communication infrastructures are examples in which use of a DTN is indispensable. DTN architecture and protocols allow for data transfer between communication systems and individuals in a mobile ad hoc fashion ideally suited for the chaotic environment of disaster relief and the battlefield.

There are numerous architectures and routing protocols proposed for use in a DTN [1]–[3]. They rightfully focus on the challenges associated with establishing and maintaining communication in a mobile ad hoc network where individual nodes cannot effectively maintain end-to-end routing. As nodes move through the network, a routing protocol attempts to quickly establish communication with those within broadcast range and transfer data packets in a manner that moves data through the network from source to destination. Ideally this would also provide some level of information assurance (IA).

This work was supported in part by the Army Research Laboratory under Cooperative Agreement Numbers W911NF-06-3-0001 and 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.

Because of DTN constraints, security protocols are lacking for this type of network. The constraints include bandwidth, memory size, battery life, processing power, and a lack of end-to-end routing tables. These constraints limit the utility of IA schemes and protocols used in traditional packet-switched networks. There are a number of different definitions and models for information assurance that involve a multitude of different attributes. One such IA model [4] highlights the key concepts of authentication, integrity, confidentiality, availability, and nonrepudiation. Key management and the ability to provide signatures is usually managed centrally in traditional networks. Being able to verify certificate revocation status allows for trust management of systems and users. It includes the ability to establish, update, and revoke trust. Due to topology changes over time and the constraint in a DTN, this process has limited use and predicates the need for a distributed trust management process. Trust decisions, in a DTN, are made by observed actions of other nodes and the aggregation of trust shared by interacting with other trusted nodes. Trust management in DTNs must take into account multiple trust properties including the fact that information propagation in a DTN is dynamic, subjective, context dependent, asymmetric, and not necessarily transitive [5].

There are numerous DTN attack vectors including eavesdropping, packet dropping, packet manipulation, denial of service, and node isolation. There are multiple adversarial models that a distributed trust management scheme must combat. For this paper the trust management scheme is meant to combat a threat that either permanently or intermittently modifies the payload of a message in an attempt to corrupt or manipulate a message between source and destination.

This paper proposes a trust management scheme for use in DTN that utilizes erasure coding and checksums not only to increase delivery ratio of packets to destination but also to establish, update, and revoke trust. Section II describes the proposed trust management scheme and shows mathematical bounds on its performance. Section III shows initial experimental results. Section IV proposes future work and conclusions.

## II. Erasure Coding Trust Management Scheme

The proposed trust management scheme uses erasure coding (see section II-A) and a checksum to determine the trustworthiness of neighbor nodes. If all message segments successfully arrive to recreate a complete message then the trustworthiness of those neighbors increases. If one or more of the segments are corrupt, maliciously or not, then by analyzing additionally received segments, bad actors can be determined. This is done by having the source node append a checksum onto every message sent prior to using erasure coding to segment the message and send it onto the network. The destination is then in a position to validate each message and make trust decisions. The scheme steps are listed below.

1) Node $x$ sends a message $M$ to node $y$.
2) Segment message $M$, with added checksum, using erasure coding such that $k$ segments recreate $M$.
3) When $k$ unique segments arrive at node $y$, attempt to recreate $M$.
4) If $M$ has a valid checksum, $y$ increases the trust for all nodes sending a valid segment and skips to 6, otherwise it continues to 5.
5) Node $y$ waits for each additional segment $m$ until recreating $M$ produces a valid checksum or based on cost it is better to request resending the message (see section II-B). If segment $m$ successfully recreates $M$, then nodes sending valid segments receive a trust increase and all others a trust decrease; move to 6.
6) The receiving node waits for time $T$ and accepts any addition segments for $M$, validity of each is checked against $k-1$ known good segments and trust along the relevant path is accordingly changed.

### A. Erasure Coding Background

Erasure coding, as a network protocol, has been studied for use in a DTN [6], [7]. It works by breaking a message into a set of message segments. When a sufficiently large subset of message segments are received the original message can be reconstructed. Specifically, erasure coding starts with a message of size $M$ and the the total size of information $I = M(1+\epsilon)$ needed for message recreation. $\epsilon$ is a small constant that depends on the exact encoding algorithm used. Then, the minimum number of segments, $k$ is selected, such that $I$ is divided evenly by $k$. Finally, the total number of segments, $s > k$, is chosen and the encoded message is broken into that many segments. The value $r = (1+\epsilon)s/k > 1$ is called a replication factor as it defines how much more information is sent to transfer $M$ bytes of a message. For the purposes of this paper the exact encoding algorithm is not important.

When using erasure coding, the key aspect is the replication factor $r$. To recreate a message only $s/r$ of the message segments must arrive at the destination. In order to transmit the message segments over multiple different paths, an algorithm similar to *srep* [7] can be used. In that variation of erasure coding, the generated message segments are split between $s = kr/(1+\epsilon)$ relays. For example if $r = 5$, then in order to send a message of $M$ bytes, $5 \times M$ bytes of data is transmitted, and if $\epsilon = 0.25$ and $k = 3$ then the number of nodes that receive segments are $s = kr/(1+\epsilon) = 12$. Furthermore, to reconstruct the original message, $k$ segments must arrive at the destination. A lower replication factor or multiplier $k$ reduce the number of separate message segments that must be transmitted through the network.

### B. Utility of Waiting For One More Segment

If the first $k$ message segments are intact then all additional segments $m$ can be used to determine if the sending node of the $(k+m)^{th}$ segment is truthful, see step 6 in section II above. This requires one message creation operation using $k-1$ known good segments and the unknown $(k+m)^{th}$ segment. If the recreated message is good then the sending nodes trust is increased, otherwise it is decreased. The issue is when the first $k$ segments do not recreate the original message.

*1) Approximate Path Probabilities:* The scheme described above is distributed and each node maintains a table with a trust value for all other nodes in the network. The probability $p_{nx}$ is the average trust value of all other nodes in the network at a given point in time for node $x$. Let $p_{nx}$ denote the percentage of good nodes in the network and $p$ represent the likelihood that the next segment to arrive at the destination is good. Probability $p$ is required to calculate the cost of waiting for an additional segment and is equal to the probability along all likely paths from source to destination. The rest of this section discusses how we approximate $p$ using the following equation

$$p = \sum_{h=1}^{n-1} f_h^{(n)} \times (p_{nx})^h \tag{1}$$

where $n$ is the number of nodes in the given DTN and $f_h^{(n)}$ is the fraction of paths with $h$ hops from the source to destination a segment likely will take. The essential for $p$ value of $f_h^{(n)}$ is routing protocol dependent; however, using a simplified and restrictive routing protocol allows for a reasonable approximation for $p$. In this simple protocol, each node maintains a value, $t_{a,b}$ from 0.0 to 1.0 that equates to the frequency of inter-meetings between nodes $a$ and $b$. We assume that $t_{a,b} = t_{b,a}$. Without loss of generality, we assume below that the source is node 1 and the destination node $n$. The node currently holding the message segment, $c$ will pass it to met node $d$ only if $d$ is the destination $(d = n)$ or if $t_{d,n} > t_{c,n}$. We also assume that our network is fully connected, and if node $d$ does not meet the destination then $t_{d,n} = 0$, so it will never be selected as an intermediate node. This protocol approximates well also the number of hops expected in epidemic routing as it trades number of hops for time of delivery. The fastest packet reaching the destination in epidemic routing is likely to traverse the route that segments in our protocol travel.

Let edge $(d,n)$, where $1 \leq d < n$ has weight $0 \leq x_d \leq 1$. According to the routing protocol, an intermediate node $1 < d < n$ is eligible for passing a message segment (in short, eligible), if $x_d > x_c$, where $c$ is the node currently holding the message segment. Below outlines our approximation method for determining the probability of a path having the given hop count and the number of intermediate nodes $n-2$ using the simple routing protocol above. This is used to determine $f_h^{(n)}$ in equation 1.

Let $p_h^{(n)}$ denote the probability that after making $h-1$ hops, the message segment will be delivered to the destination in $h$-th hop. Of course, $0 < h < n$ and $p_{n-1}^{(n)} = 1$ because no message segment is passed to the same node twice. Hence, we need to consider only $n > 2$. It is easy to show by induction that the value of interest, $f_h^{(n)}$ is defined by the relevant probabilities as

$$f_h^{(n)} = p_h^{(n)} \prod_{j=1}^{h-1} (1 - p_j^{(n)}) = p_h^{(n)} (1 - \sum_{j=1}^{h-1} f_j^{(n)}) \qquad (2)$$

Consequently, once $p_h^{(n)} \approx 1$ the fraction of paths longer than $h$ is negligible.

**Exact solution for $h = 1$:** We have $n - 2$ intermediate nodes each with an edge to the destination with the weight randomly distributed with the same probability distribution. Hence, probability that we have $j$ eligible nodes is $\sum_{j=0}^{n-2} \binom{n-2}{j} (1 - x_1)^j x_1^{n-2-j}$ and probability that the destination will be chosen with this number of eligible nodes is $\frac{x}{x + \sum_{i=1}^{k} y_i}$ where $y_i$ denotes the weight of edge from $i$-th eligible node to the source. Hence, the following $n-1$ integrals is the solution:

$$p_1^{(n)} = \int_0^1 \ldots \int_0^1 \sum_{j=0}^{n-2} \binom{n-2}{j} \frac{(1-x_1)^j x_1^{n-1-j}}{x_1 + \sum_{i=1}^{j} y_i} dx_1 \ldots dy_{n-2}. \qquad (3)$$

Algebraic solution for $n = 3, 4$ is simple, and yields:

$$p_1^{(3)} = \int_0^1 \int_0^1 \frac{(1-x_1)x_1}{x_1 + y_1} dy_1 + x_1 dx_1 \approx 0.7046 \qquad (4)$$

$$\begin{aligned} p_1^{(4)} &= \int_0^1 \int_0^1 \int_0^1 \frac{(1-x_1)^2 x_1}{x_1 + y_1 + y_2} dy_2 + \frac{2(1-x_1)x_1^2}{x_1 + y_1} dy_1 \\ &+ x_1^2 dx_1 \approx 0.5763 \end{aligned} \qquad (5)$$

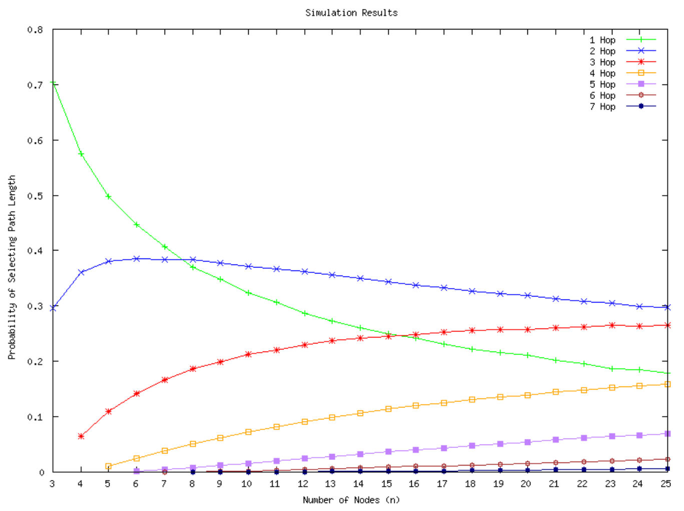
For larger $n$'s the numerical integration can be used to get values for validating simulation results.

**Approximation for general case of $0 < h < n - 1$:** After $h - 1$ hops, there is $n - h - 1$ intermediate nodes left and each has probability $1 - x_h$ to be eligible, where $x_h$ denotes the edge weight of the current segment holder to the destination. The expected number of eligible nodes is $(1 - x_h)(n - h - 1)$ each with the average edge weight to the source being $1/2$. Since each hop on average hits the middle of the previous range, the size of the $h$ range, for weights of eligible nodes, is $(1 - x_1)/2^{h-1}$ so the value of $x_h$ is $(x_1 + 2^{h-1} - 1)/2^{h-1}$. To get a good approximation, we will compute the expected value over each half of this range which yields the result:

$$\begin{aligned} p_h^{(n)} &= -\frac{2}{(n-h-3)} + \frac{2^h(n-h-1)}{(n-h-3)^2} \qquad (6) \\ &\times \ln\left[\left(1 + \frac{1.5(n-h-3)}{2^h}\right)^{\frac{1}{3}} \left(1 + \frac{0.5(n-h-3)}{2^h}\right)\right] \\ &\approx \frac{2^h * (n-h-1)}{(n-h-3)^2} \ln(1 + \frac{n-h-3}{2^h}) - \frac{2}{(n-h-3)} \end{aligned}$$

The critical value for this function is $h_c = log_2(n)$. For $h > h_c$, we have $\ln(1 + (n-h-3)/2^h) \approx (n-h-3)/2^h$ so the result is nearly 1. For example, it is greater than $15/16$ for $h > h_c + 1$. For $h = h_c - h' < h_c$ an approximate value is $1/2^{h'} * h' * \ln(2) - 2/n$ close to 0. For example this value is less than 0.1 for $h' > 5$.

Fig. 1: Probability of using a Path with $h$ Hops



It is easy to check that the peak of fraction of paths happens around $log2(n) - 3$ and these fractions are significant only for $h$'s from $h_c - 5$ to $h_c + 1$. Approximating again, we can use $h_c - 3$ as the value for the most common path length and then we get the following from equation 6.

$$p \approx p_{nx}^{\log_2(n)-3} \approx 1 - (\log_2(n) - 3) * (1 - p_{nx}) \qquad (7)$$

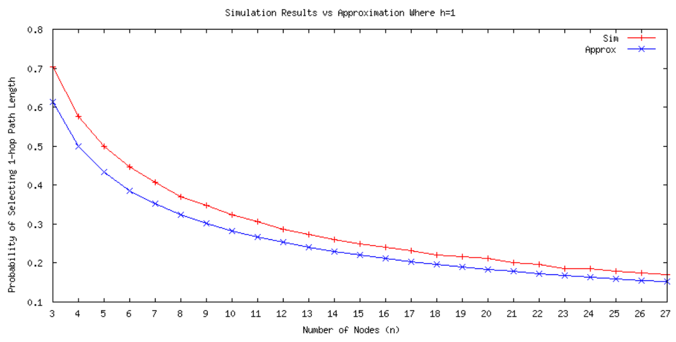
The final approximation holds only for $p_{nx} > 0.8$ and moderate $9 < n < 256$. In summary, the average path length grows and the the probability $p$ decays slower than $log_2(n)$.

**Simulation Results:** We simulated the behavior of the simple routing protocol by creating a complete graph with $n$ nodes. Each edge has a random weight selected uniformly from [0.0,1.0] range. We then determined all possible paths and the probability each would be used to ultimately determine the probability of using a path of $h$ hops in order to confirm our approximations from equations 3, 6 and 7. Each simulation was run 50,000 times for each value from $n = 3$ to $n = 27$ and the results were averaged. Figure 1 shows how the probability of using a path with $h$ hops changes as the number of nodes in the network increases. Once $n > 9$, there is a higher probability of using a path with two hops than with one. Three hops paths occur more often than one hop ones when $n > 17$.

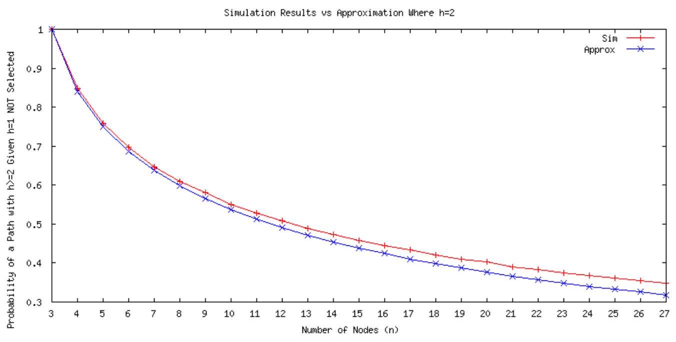
Equation 4 gives our expected value for $p_1^{(3)} = 0.7046$ the simulation results are $p_1^{(3)} = 0.7045$. Equation 5 gives our expected value for $p_1^{(4)} = 0.5763$ the simulation results are $p_1^{(4)} = 0.5755$. This confirms our intuition about how the protocol reacts, but as stated above it is not a practical approach for larger $n$. Figure 2a shows the approximation for $h = 1$ and figure 2b shows the approximation for $h = 2$ using equation 6. While the approximation is not exact, it gives a tight lower bound as $n$ increases.

The previous section outlined a process for determining the probability across all paths given a trust level in the network. Below we outline what happens when $k$ segments arrive and one or more segments are corrupt. It is the added benefit of waiting for $(k + m)^{th}$ segment.

Fig. 2: Approximation Fit for One and Two Hops



(a) Simulation and Approximation Results $p_1^{(n)} = f_1^{(n)}$



(b) Simulation and Approximation Results $p_2^{(n)}$

*2) Cost and Benefit:* The total probability of being able to assemble the message from $k+m$ segments is as follows:

$$P_{k+m} = \sum_{i=0}^{m} p_{k+i} \tag{8}$$

For $m > 0$, the change in probability from $k+m-1$ to $k+m$ can be expresses as:

$$\Delta P_{k+m} = p_{k+m} = \frac{(k+m-1)!p^k(1-p)^m}{(k-1)!m!} \tag{9}$$

The justification is simple. Since $k+m-1$ segments were not enough to recreate the message but $k+m$ are, the $(k+m)^{th}$ segment has to be correct and the $k+m-1$ segments received previously must contain exactly $k-1$ correct segments. Thus, $p^k$ defines probability of having $k$ correct segments, while $(1-p)^m$ is the probability of having $m$ corrupt segments and the rest of the expression defines the number of ways $k-1$ correct segments can be chosen from $k+m-1$ segments previously received.

The cost of being able to assemble the message from $k+m$ segments is as follows:

$$C_{k+m} = \alpha \sum_{i=0}^{m} c_{k+i} \tag{10}$$

where $\alpha$ defines the ratio of the cost of computation to the value of increased probability of being able to recreate the message from additional segments. When the $(k+m)^{th}$ segment arrives it is compared to all $k-1$ subsets of $k+m-1$ segments already received but not sufficient to recreate the message, hence, for all $m > 0$ the change in cost from $k+m-1$ to $k+m$ can be expressed as follows

$$\Delta C_{k+m} = \alpha \frac{(k+m-1)!}{(k-1)!m!} \tag{11}$$

*3) Utility Functions:* There are two options after receiving $k+m-1$ segments. The first is to wait for the $k+m$ segment. The second is to request sending the message again. To make this decision, we can use one of the two criteria listed below.

1) The first criterion is to compare the expected gain in probability as expressed in equation 9 with the cost of equation 11.
2) In the second criterion, the price of receiving the next segment versus the price of resending the message that is the expected gain of receiving the message is $P_k = p_k = p^k$ at the cost of $n*r+\alpha$ which is the cost of resending $nr$ plus the cost of unpacking the segments $\alpha$.

Using the first criterion, we wait for the next segment if the gain in probability to recreate the message is greater than the cost. The second is to receive the best price per increase in probability. The former is a simplified function while the later is more comprehensive since it takes into account the cost of resending the message.

The first utility function is $G_{k+m} = p_{k+m} - \alpha c_{k+m}$. It is beneficial to wait for the $k+m$ segment if $G_{k+m} > 0$. Substituting the values from equation 9 and equation 11, the following holds:

$$p^k(1-p)^m > \alpha \tag{12}$$

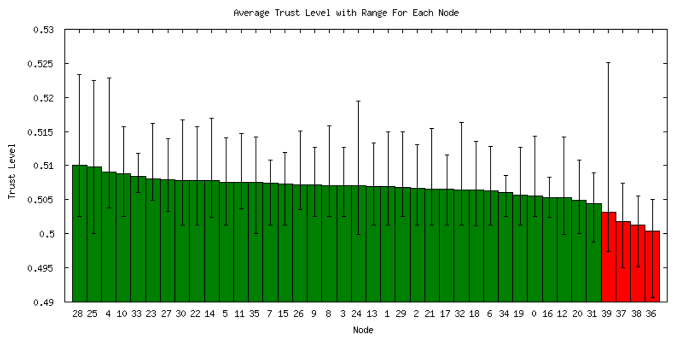
Equation 12 has its merits; however, it it based solely on the increase in probability of being able to recreate a message with an additional segment being greater than the cost of this segment processing. The increase in probability shrinks very quickly limiting the number of segments for which the destination waits before requesting a message be resent, which intuitively makes sense.

The second more complicated utility function takes into account the cost of having to resend the message a second time. Ultimately it compares the price increase between waiting for another segment versus that of resending the message. This gives the inequality of $\alpha\frac{c_{k+m}}{p_{k+m}} < \frac{(\frac{k}{r}+\alpha)}{p^k}$. Since $\frac{c_{k+m}}{p_{k+m}} = \frac{1}{p^k(1-p)^m}$, the inequality can be reduced to $\frac{\alpha}{(1-p)^m} < \frac{k}{r}+\alpha$. The cost of $\frac{k}{r} \gg \alpha$ so the inequality can be rewritten as $\frac{\alpha r}{k} < (1-p)^m$. Taking the natural log of both sides gives the final inequality as follows.

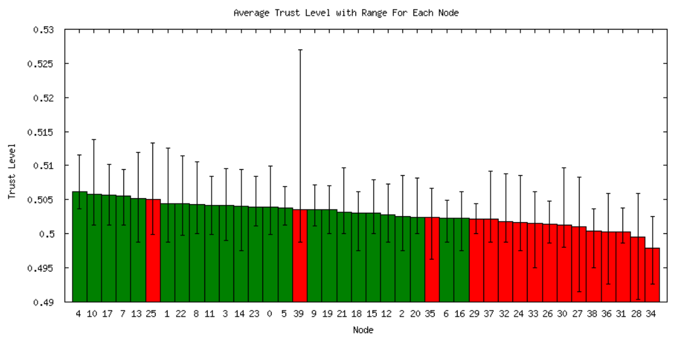
$$m < \frac{ln\left(\frac{\alpha r}{k}\right)}{ln(1-p)} \tag{13}$$

In this inequality, $\alpha$ represent technology factors, as the node processing becomes faster and faster at the same price due to the chip technology, $\alpha$ becomes smaller and smaller.

Fig. 3: Ranking of All Nodes According to Their Trustworthiness



(a) 90% Path Trustworthiness



(b) 60% Path Trustworthiness

Yet, the value of $m$ grows only logarithmically with this gain. The erasure coding algorithm parameters $r$ and $k$ change relatively little for different applications, so their impact on $m$ can be ignored. Finally, $1 - p$, measuring how "pollute" and to lesser degree how large (see equation 7) the network is has a large impact; the higher the pollution level the larger $m$ the destination should wait before requesting resend. This again is intuitively clear, as the resent message will face the same treacherous journey to destination as the original message did.

## III. EC Trust Management Simulations

In order to test the proposed trust management scheme, a number of simulations were executed using NS3. This paper uses the work done by *Lakkokorpi et al.* in [8]. The available DTN module for NS3 implements the DTN bundle layer first proposed by the Delay Tolerant Networking Research Group (DTNRG) and codified in a number of DTNRG request for comments [9], [10]. The bundle layer protocol manages application to application transportation with each bundle usually being larger than a normal IP packet. Each bundle has a timer and if it is not delivered within a specified time it is deleted from a nodes buffer. The point to point connections between nodes can be managed by UDP or TCP. There are numerous routing protocols proposed for use in DTNs. While most were not written to use the bundle layer, almost all can be modified to do so.

The latest available DTN module for NS3 is compatible with version 3.18 and includes the implementation of two DTN routing protocols: endemic [11] and stray and wait [12]. For each protocol a number of duplicate bundles are sent from source to destination to increase the probability that one will arrive. In the former protocol, a bundle is send to any node that does not yet have a copy in its buffer. This has been shown to significantly clog the network and use unnecessary resources. The later is a more refined approach and sends a smaller number of copies and then waits to see if the bundle arrives. It only increases the number of nodes it forwards to if it does not receive an acknowledgement packet from the destination in a specified time.

The proposed trust management scheme is built using the concept of erasure coding to determine which nodes, if any, modify a segment of the data transmission between the source and destination. The authors in [8] give a good breakdown of the code they created. This includes the main DTN module *dtn.cc* and the bundle layer encapsulation headers *mypacket.cc* and *mypacket.h*. The necessary modification to simulate EC and the proposed trust management scheme are broken down into two main categories: simulate erasure coding and simulate trust management at the node level. Section III-A gives an overview of the modifications to allow for erasure coding. Section III-B give an overview of the changes to replicate node trust management. Section III-C explains the simulations run and Section III-D presents results.

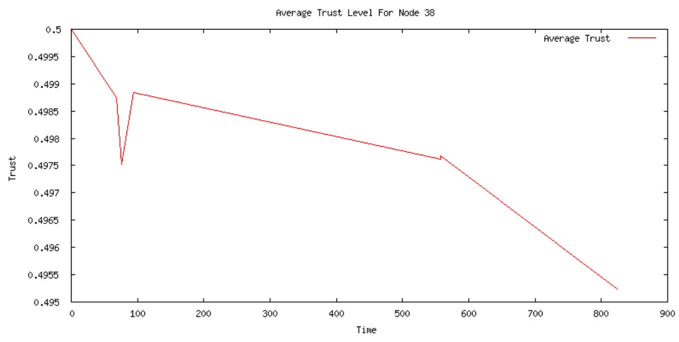
### A. Erasure Coding Simulation for DTN Module in NS3

In order to simulate Erasure Coding some modifications to the existing routing protocols in the DTN module were required. Erasure coding as described above multiplies a message payload by a replication factor $r$, this modified message is divided and each segment is sent from source to destination. Each message segment is sent as a bundle with its own individual bundle ID. For a given message, there is a single message ID designated the PID. When $k$ unique unmodified bundles arrive at the destination with the same PID the message is recreated.

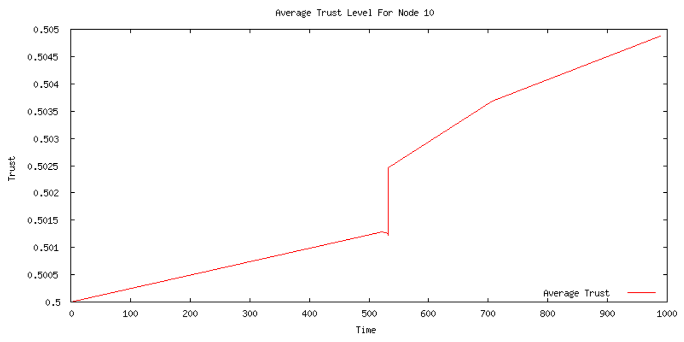
The user provides $k$, $\epsilon$ (that for simplicity we assumed to be 0 in the current simulation) and $s$ (which defines $r$) as initial input to the simulation setup to allow for the segmentation of messages into $kr/(1+\epsilon)$ bundles. Based on the values for $k$ and $r$, each bundle is scheduled for transmission. For the first hop from the source a node can only accept one bundle with a particular PID. This forces multiple paths from source to destination.

While currently erasure coding is implemented in an austere manner and utilizes the already implemented endemic routing protocol to move the bundles, it is useful for showing preliminary result using EC for trust management. Although we are using epidemic routing, the approximate metrics found in section II apply to our simulation results as explained in that section.

Fig. 4: Trace of a Given Untrustworthy Node over Time



(a) Trace of Node 38, 90% Path Trustworthiness



(b) Trace of Node 10, 60% Path Trustworthiness

## *B. Trust Management Object in NS3 for use in the DTN Module*

Like in many distributed trust management schemes, each node maintains a structure that includes the trust level for each node it interacts with. Additionally, many schemes aggregate trust levels with neighbors through periodic comparison of stored trust values. To simulate such a scheme in NS3, a trust management object was added. This was created for use with each NS3 DtnApp outlined in [8]. The DtnTrust object initializes all nodes in the network to a trust level $t_{node} = 0.5$, $0.0 \leq t_{node} \leq 1.0$. This DtnTrust object tracks the bundles that arrive and once $k$ unmodified unique bundles with the same PID arrive recreates the original message.

After the successful message recreation, each node that sent an unmodified segment has its trust level increased to $t_{node} = t_{node} + .05$. For each node that sent a malicious segments the nodes trust is decremented to $t_{node} = t_{node} - .05$. Generally this only occurs at the destination; however, certain paths might contain a single node that forwards many bundles that are part of a given message and such nodes may also compute trust for their neighbors. The node, where the successful message recreation occurred, continues to receive and check validity of addition received segments for time $T$ updating trust accordingly.

Currently the trust level is not used to deter sending a bundle to a node; however, there are functions to allow a node to check the current trust of all other nodes. An extension is to phase out nodes that do no behave properly. Those nodes would no longer be allowed to send or receive bundles.

## *C. Simulation Overview*

The NS3 simulator modified with DTN extensions, is used to conduct the simulations presented. The threat model against which this scheme is attempting to protect includes an adversary who either hijacks, reprograms, or adds malicious nodes to the DTN that modify all packets that arrive prior to sending them on as per the routing protocol. There are more sophisticated models where the attacker intermittently modifies packets or where nodes collude. While it is likely this scheme with full use of the Bundle layers security module [10] would provide useful results, such extensions are left for future endeavors. This adversarial model is simulated in NS3 by making x% of the nodes always act maliciously by modifying all bundles they receive prior to forwarding. The malicious node does not modify bundles when it is the source or destination.

The NS3 simulation tool provides the user with of a number of traffic and mobility models. Similar to [8], a simple traffic model is used in our simulation. Each node sends a number of variable sized messages, which are broken into a set of bundles defined by $k$ and $r$, at a random time within each 200 second interval of simulation run time to another random node. For node to node transmission, the segments are fragmented into 1500-byte data-grams using UDP, with retransmission, to provide reliability.

The mobility model used is the random way point (RWP) model, which is conveniently built into the NS3 simulator. For each simulation, a total of 40 nodes move in a 2500m x 2500m grid. These nodes are evenly distributed and select direction and speed at random (uniformly distributed) times. The maximum speed is 20m/s with a 2 second pause. All nodes pause between each movement. The simulation is run 10 times each with a different seed and all simulations are run for 1000 seconds.

## *D. Simulation Results*

The results show that over time the average probability across all nodes increase for good nodes and decrease for bad ones. There is a large contrast in result when the percentage of good nodes drops from 90% to 60%. This make sense because with a higher number of untrustworthy nodes, the likelihood that they would corrupt a message from source to destination and negatively impact trust for a good node increases.

*1) 90% Path Trustworthiness:* Figure 3a shows the average trust level for each of the 40 nodes sorted in the decreasing order of their trustworthiness. While the decrease is not steep, the four untrustworthy nodes that modified all forwarded bundles have the lowest average trustworthiness among all nodes (those are nodes 35-39 and colored red). The error bars show the largest and smallest individual run. This gives an idea of the range of values.

Figure 4a illustrates one malicious node trust evolution over a complete simulation. This figure shows the change in average trust of node 38 from time 0.0 seconds to time 1000 seconds. The trend is down as expected. At approximately 90 seconds there is an increase in trust. It appears when node 38 is the source of a message and sends it directly to the destination. In that case, the trust is increased by the receiving node because in the threat model all messages start without modification, which make sense because even if the message payload contains malicious content, the checksum is being generated at the source and it would be correct.

*2) 60% Path Trustworthiness:* Figure 3b shows the average trust level for each of the 40 nodes sorted by their trustworthiness. Due to the number of malicious nodes, 16 in this case, there is no steep slope, however, the majority of the malicious nodes, colored red, have the lowest average trust and only three have trust higher than a good node colored green.

Figure 4b illustrates the change of a trustworthy node over time. Its average trust goes generally up, but lowers and then spikes around 550 seconds due to passing on a tainted message segment and then sending a number of trustworthy segments.

## IV. Future Works and Conclusions

Based on the results of the simulation, discussed in the previous section, a number of additional improvements are merited as part of future work. They include the expansion of the threat model to include a larger array of adversaries, sliding trust windows, implementation and simulation of node exclusion based on changing trust, integration of a trust sharing extension, implementation and experimentation with the bundle security module, ability to add and delete nodes from the network, and analysis of the effects of false positives and false negatives on the network. Additionally, a more thorough comparison to other trust management schemes [13]–[15] is merited upon completion of further research.

This paper proposes a novel trust management scheme that uses traits inherent with erasure coding and checksums to modify trust in a distributed manner. This scheme is ideally suited for use in delay tolerant networks. In erasure coding, a message is broken into a number of segments. Each segment is sent through the network from source to destination. In a DTN, segments will take multiple different paths. Once enough segments arrive at the destination, the message is recreated. By adding a checksum to the message prior to sending it to the destination faulty segments can be identified. If all segments are good then the message is recreated. If not, the destination continues to wait for an additional segment until the message is recreated or the cost of waiting for an additional segment is higher than resending the message. Once the message is recreated, all additional segments are used to modify the trust of the sender.

This paper approximates the probability that a good segment arrives, bounds the cost of waiting for an additional segment and proposes two utility functions to use in making the decision to request resending a message. The first is based on the change in probability that the next segment will recreate the message versus the cost of processing one more segment. The change in probability shrink exponentially and the cost increases linearly. This limits the usefulness of that particular utility function. The second utility function is used to minimize the price of increasing the probability of receiving a correct packet. This takes into account the cost of resending the message and is the utility function used to determine for how many additional segments to wait prior to requesting that a message be resent.

Based on the simulation results, the scheme shows promise and merits further research. A number of potential future research topics are listed above. With modifications this scheme can assist in filling the gap in information assurance in delay tolerant networks.

## References

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

[2] K. Fall, “A delay-tolerant network architecture for challenged internets,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM ’03. New York, NY, USA: ACM, 2003, pp. 27–34.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R.Durst, K. Scott, K. Fall, and H. Weiss, “Delay-tolerant networking architecture,” Internet Research Task Force (IRTF), RFC 4838, April 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt.pdf

[4] W. V. Maconachy, C. D. Schou, D. Ragsdale, and D. Welch, “A model for information assurance and integrated approach,” in *Proceeding of IEEE Workshop on Information Assurance and Security*, June 2001, pp. 306–310.

[5] J.-H. Cho, A. Swami, and I.-R. Chen, “A Survey on Trust Management for Mobile Ad Hoc Networks,” *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 4, pp. 562–583, 2011.

[6] E. Bulut, Z. Wang, and B. Szymanski, “Cost efficient erasure coding based routing in delay tolerant networks,” in *Communications (ICC), 2010 IEEE International Conference on*, 2010, pp. 1–5.

[7] Y. Wang, S. Jain, M. Martonosi, and K. Fall, “Erasure-coding based routing for opportunistic networks,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN ’05. New York, NY, USA: ACM, 2005, pp. 229–236.

[8] J. Lakkakorpi and P. Ginzboorg, “ns-3 module for routing and congestion control studies in mobile opportunistic dtns,” in *Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2013 International Symposium on*, July 2013, pp. 46–50.

[9] K. Scott and S. Burleigh, “Bundle protocol specification,” Internet Research Task Force (IRTF), RFC 5050, November 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt.pdf

[10] S. Symington and S. Farrell, “Bundle security protocol specification,” Internet Research Task Force (IRTF), RFC 6257, May 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6257.txt.pdf

[11] A. Vahdat and D. Becker, “Epidemic routing for partially-connected ad hoc networks,” Tech. Rep., 2000.

[12] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, “Spray and wait: an efficient routing scheme for intermittently connected mobile networks,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN ’05. New York, NY, USA: ACM, 2005, pp. 252–259.

[13] M. K. Denko, T. Sun, and I. Woungang, “Trust management in ubiquitous computing: A Bayesian approach,” *Computer Communications*, vol. 34, no. 3, pp. 398–406, Mar. 2011.

[14] E. Ayday and F. Fekri, “An iterative algorithm for trust management and adversary detection for delay-tolerant networks,” *Mobile Computing, IEEE Transactions on*, vol. 11, no. 9, pp. 1514–1531, Sept 2012.

[15] I.-R. Chen, F. Bao, M. Chang, and J.-H. Cho, “Dynamic trust management for delay tolerant networks and its application to secure routing,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 5, pp. 1200–1210, May 2014.