trustd: Trust Daemon Experimental Testbed for Network Emulation

Kevin S. Chan*, Jin-Hee Cho*, Theron Trout†, Jason Wampler‡, Andrew Toth*, Brian Rivera*
*US Army Research Laboratory, Adelphi, MD
†Independent Consultant, Silver Spring, MD
‡INCA Engineering, Silver Spring, MD
Email: {kevin.s.chan, jin-hee.cho,andrew.j.toth, brian.m.rivera}.civ@mail.mil,
theron.t.trout.ctr@mail.mil, jason.wampler@incaeng.com

Abstract—Various trust models have recently been proposed in order to develop trust-based security and/or network services. Although the trust models have been tested through extensive simulation to verify and validate their performance gains, there has been little effort to validate the models using network emulation to provide more realistic network characteristics. In this work, we present trustd, a trust daemon created to provide a capability to evaluate trust in distributed network environments. trustd has been deployed in CORE (Common Open Research Emulator), which enables the system to conduct peer-to-peer trust estimation in a fully distributed emulated network. In this work, we present the architectural framework and implementation details of trustd. In particular, we discuss the process of taking the trust metric theory and implementing it as a UNIX-system daemon providing trust-evaluation services to the system. In addition, we compare our experimental and simulation results and analyze their trends in terms of the accuracy of trust estimation and trust evolution over time in the presence of malicious nodes in the network.

I. INTRODUCTION

Trust-based approaches have been developed for a variety of network security and/or management services in dynamic, hostile, and distributed network environments. Particularly, in military tactical networks, establishing and maintaining trust is critical to providing the basis for decision making, which directly impacts mission performance.

Trust management, including trust establishment, update, and revocation, in MANETs is also much more challenging than in traditional environments as it is assumed to be distributed and decentralized. First of all, collecting trust evidence to evaluate trustworthiness is difficult due to network dynamics caused by node mobility/failure. Further, resource constraints often confine trust assessment process only to local information. Due to these reasons, trust management is challenging for a fully distributed network environment where each node needs to perform peer-to-peer trust estimation towards each other based on local evidence [1].

In our previous work, we developed various trust-based approaches for network security and management services, focusing on theoretical approaches through modeling and simulation. In this work, we describe our implementation of our proposed trust metric [2] using CORE (Common Open Research Emulator) [3]. The development of trustd aims to validate the claims of our proposed trust model by comparing the simulation and emulation-based estimation of trust in MANET environments.

The contributions of this paper are as follows:

- We implement our trust model [2] as a UNIX daemon called Trust Daemon, trustd, in order to validate the performance claims of our prior theoretical trust model. We ran trustd on emulated wireless network using CORE. Although theory and modeling and simulation studies suggest the benefits of various trust models for MANETs, there has been little work that validates the proposed trust model in emulation environment.
- We describe the specific design components of trustd’s architectural framework which is developed with the goal of scalability, reconfigurability, and efficiency of experimental testing.
- Through the comparison of performance results from both simulation and emulation, we discuss general trends where we clearly explain why some discrepancies between these two results are produced.
- We discuss how this CORE-based emulation testing environment can be utilized in other part of military networks and organizations.

The rest of this paper is organized in the following. Section II discusses the concept of trust in distributed networks and its applicability in network and management services. Section III explains the theoretical framework of the proposed trust metric. Section IV describes the details of the architectural framework of the trustd and the developed interfaces for emulation experiments. Section V shows the simulation and emulation results and compares their performance in terms of the accuracy of trust estimation. Section VI discusses the applicability of the developed emulation platform of trustd and potential extensibility to other network domains related to tactical operations. Lastly, we conclude this work and suggest future directions.

II. RELATED WORK

The concept of “trust” originally is derived from the social sciences and is defined as the degree of a subjective belief about the behaviors of a particular entity [4]. Blaze et al. [5] first introduced the term “trust management” and identified it as a separate component of security services in networks. They
explained that trust management provides a unified approach for specifying and interpreting security policies, credentials, and relationships.” Trust management in MANETs is needed when participating nodes, without any previous interactions, desire to establish a network of nodes with an acceptable level of trust among themselves. Trust management has diverse applicability in many decision making situations including secure routing [6], [7], intrusion detection [8], key management [9], task assignment [10], service binding and composition [11], information sharing [12], and information fusion [13]. However, the above works [6]-[13] were developed based on modeling and simulation platforms and has not been validated using higher-fidelity network environments such as CORE.

The nature and characteristics of MANETs result in uncertain, incomplete trust evidence, which is dynamically changing over time [14], [15]. In our work, each node conducts dynamic peer-to-peer trust estimation based on the aggregation of direct and indirect evidence where past experience is fully leveraged to minimize the uncertainty due to the incomplete, unavailable evidence.

III. TRUST METRIC

In this section, we briefly describe how each node estimates another node’s trust by aggregating both direct and indirect trust evidence. Interested readers can refer to [2] for more details. First, we describe the trust components considered in this trust metric based on the multidimensional concept of trust.

A. Multidimensional Trust

The concept of trust has been defined numerously and differently in different disciplines including psychology, sociology, philosophy, organizational management, economics, and autonomic computing [16]. In this work, we define trust in a context of the computing and networking domain and use the concept of trust as the subjective probability that a node shows reliable behavior for given operations under a situation with potential risk [16].

Trust-based networking has been proposed as a method to enhance mission performance in various networked environments. The concept behind these approaches is the multi-genre nature of the networking environment in which the service will be deployed. Traditional network approaches employ single concepts and less flexible approaches to decision-making within protocols and algorithms.

In this work, we consider the multidimensional concept of trust based on the nature of a multi-genre network. Elements of the individual layers of these networks (e.g., tactical networks) can be characterized by different dimensions of trust. For example, we assume that the multi-genre tactical network is comprised of communication, information and social/cognitive network layers. To various extents, policies and protocols on each of these layers can be configured, and as a result, there are contributions of each of these layers to the performance of the multi-genre network.

Considering the multidimensional concept of trust, multiple trust components $T_{ij}^X(t)$ are considered and can be incorporated into the overall trust of node $i$ in node $j$, $T_{ij}(t)$ as follows

$$T_{ij}(t) = \sum_{X \in A} w_X T_{ij}^X(t)$$ (1)

where $\sum_{X \in A} w_X = 1$. $X$ indicates an element in a set $A$ and $X$ refers to a trust component. The weights of trust components, $w_X$ can be configured based on system application requirements. The trust metric above is a general-ized format that can consider various dimensions of trust. In this work, we consider two trust components, reliability and competence. Our approach to obtaining estimates of reliability and competence trust is explained in Section V-A.

B. Aggregation of Direct & Indirect Trust

The proposed trust metric is a distributed peer-to-peer protocol that enables trust estimation based on local information without the help of a centralized authority in a network. Each node evaluates another node based on both direct and indirect evidence. We discuss how direct and indirect evidence is calculated to estimate the overall trust value of a node’s trust in trust dimension $X$.

Direct evidence is the evidence that can be collected based on direct observation of another node. Thus, this evidence collection is only possible when a trustee node (i.e., a node to be evaluated) is a direct neighbor (i.e., 1-hop neighbor) of a trustor node (i.e., a node to evaluate another node). The trustor also considers indirect evidence forwarded by 1-hop neighbors of the trustee node via recommendations. In any cases where the trustor does not receive any recommendations from direct neighbors of the trustee or detects the trustee node as a remote node, no new evidence, either direct or indirect evidence, may be received by the trustor. In that case, the trustor relies on past experience to evaluate the trustee. Fig. 1 (a) illustrates the case that trustor node $i$ and trustee node $j$ are direct neighbors.

When trustee node $j$ is not a direct neighbor, but a distant node from the trustor node $i$, trustor node $i$ can refer to recommendations for the portion of indirect evidence from the direct neighbors of trustee node $j$ while it can rely on past experience for the portion of direct evidence. In any case when no evidence is available because no direct observation is possible or recommendation packets are lost on the way to the trustor, $i$ relies on past experience towards $j$ with a decay factor in order to estimate $j$’s trust. This is also illustrated in Fig. 1 (b).

The aggregated overall trust for trust component $X$ is calculated by

$$T_{ij}^X(t) = \beta T_{ij}^{X-D}(t) + (1-\beta) T_{ij}^{X-ID}(t)$$ (2)

$T_{ij}^{X-D}(t)$ refers to direct trust value of node $j$ for trust component $X$ evaluated by node $i$ at time $t$ and $T_{ij}^{X-ID}(t)$ is indirect trust value of node $j$ for trust component $X$ evaluated by node $i$ at time $t$. $\beta$ is a weight to consider direct trust evidence while $(1-\beta)$ is a weight for indirect trust.
The direct trust (based on direct observations) of node $j$ evaluated by node $i$ on trust component $X$ at time $t$, $T^{D-X}_{ij}(t)$, is computed as

$$T^{D-X}_{ij}(t) = \begin{cases} P^{D-X}_{ij}(t) & \text{if } HD(i,j) == 1 \\ \gamma T^{X}_{ij}(t - \Delta t) & \text{otherwise} \end{cases}$$

(3)

When nodes $i$ and $j$ encounter as 1-hop neighbors (i.e., $HD(i,j) == 1$) during the time period $(t - \Delta t)$, node $i$ can collect direct evidence based on its own observations or experiences $P^{D-X}_{ij}(t)$. When nodes $i$ and $j$ are distant with more than 1 hop distances, node $i$ relies on its past experience to assess the direct trust of node $j$. We assume that trust is updated every $\Delta t$. $P^{D-X}_{ij}(t)$ for trust component $X$ is computed based on the positive experience over the negative experience associated with $X$ as

$$P^{D-X}_{ij}(t) = \begin{cases} \frac{r}{r+s} & \text{if } r+s > 0 \\ 0 & \text{otherwise} \end{cases}$$

(4)

where $r$ is the number of positive experiences and $s$ is the number of negative experiences.

The indirect trust of node $j$ evaluated by node $i$ on trust component $X$ at time $t$, $T^{I-D-X}_{ij}(t)$, is obtained by:

$$T^{I-D-X}_{ij}(t) = \begin{cases} \sum_{k \in R_j} T^{X}_{lk}(t) & \text{if } |R_j| > 0 \\ \gamma T^{X}_{ij}(t - \Delta t) & \text{otherwise} \end{cases}$$

(5)

where $R_j$ is a set of recommendations about $j$ received by $i$. When node $i$ receives correct recommendations with $|R_j| > 0$, node $i$ uses the average of the recommendations to derive the overall indirect trust. If $R_j$ is an empty set, node $i$ will use its past experience $T_{ij}^{X}(t - \Delta t)$, with decay weighted by $\gamma$, due to no correct recommendations received.

IV. TRUST DAEMON trustd UNIX SERVICE

We now provide an overview of the implementation of trust and experimentation with CORE. A trust daemon trustd has been created to perform the trust evaluation described in the previous sections. In this section, we describe the implementation and the modules in the daemon.

A. Architectural Framework

The architecture of trustd is arranged into main subsystems: evidence collection and trust instrumentation, trust management, and command interpretation. An instance of trustd is run on each of the nodes. A high-level architecture is shown in Fig. 2. The network is comprised of trustd nodes, who communicate with other nodes through CORE links. To receive trust information, the Trust Data Manager receives trust information from each node’s trust_daemon. This information is sent to the Telemetry database, which in turn forwards the content-specific information regarding trust and other experiment decisions.

B. Trust Evaluation Process

1) Direct Evidence Collection and Evaluation: The trust system creates a set of “judges” for each one-hop neighbor detected on the network. Each judge is dedicated to evaluating one or more characteristics of the neighboring nodes. Fig. 3 only depicts a pair of judges, one integrity and the other for competence, per neighbor node. There could be multiple judges working in parallel to produce a multitude of different trust dimensions.

Measurement Acquisition: When a measurement is triggered, through either the timeout of the measurement interval or a detected event which indicates that performing a new measurements is appropriate, the judges are activated. The judge will collect the measurement in whatever form is appropriate. This is described as “direct measurements.” These measurements are acquired by what are dubbed “trust instruments” in the trustd system.

Measurement Scoring: The raw evidence resulting from the measurement acquisition step is passed to the judges who use “trust calculators” to convert each raw measurement into a discrete trust score. Trust calculators maintain associated trust metrics used to calculate trust scores.

Each trust calculator is given an opportunity to review each and every piece of raw trust evidence collected by the trust instruments running on the node. Each trust calculator can
then choose whether to produce a trust score for the piece of evidence or to ignore it. Trust is evaluated for each trust component: Integrity, $T^{int}$ and Competence $T^{comp}$. For the trust management scheme, we define the trust components as follows. Competence trust is a representation of the node’s participation in the trust evaluation process. Integrity trust is a measure of the behavior of the node in terms of its conformance to expected beneficial behaviors such as forwarding data to others without modification and in a timely manner, and a lack of malicious actions (scanning other nodes in the network, repeatedly attempting and failing to authenticate when requesting services from other nodes on the network). In addition to the trust estimates, each judge can provide a confidence value for each trust score it is providing.

**Trust Value Determination:** After a trust calculator produces a trust score, these trust scores are sent to the Trust Manager. The trust manager uses the formulation found in Section III-B to incorporate these individual scores into a final trust score for the each node.

2) **Indirect Trust Handling:** Reception of Indirect Trust Broadcasts: When indirect trust broadcasts are received by the trustd system, these scores are sent to the trust manager for incorporating into the final trust value through two paths. The first path is via the judges which are each given an opportunity to provide a score they think is appropriate. Second, after all judges have commented on the original trust value, it is sent to the trust manager who can then determine how to aggregate the original indirect evidence data with other scores provided by other judges.

**Judge Comments on Trust Broadcasts:** If a judge opts to comment on the indirect evidence data provided, it does so by preparing a trust score and attaching the indirect evidence object to it. This affords the opportunity for each judge to advise the trust manager on how the indirect trust score would impact its opinion of subject of the indirect evidence. The judge sends the following information to the trust manager: the judge’s estimated trust value, a confidence value of this estimate, the trust component (Integrity or Competence), and the original indirect evidence.

**Trust Manager Evaluation:** After the trust manager has received all trust scores, the judges who opted to participate produced trustd will finally send the original indirect evidence object to the trust manager. This will act as a trigger to the trust manager that it now has all the information it is going to get about this indirect evidence item. Finally, based on the direct and indirect evidence collected, the trust manager determines if the final trust value of the subject node should be modified.

**C. Experimental Interface**

Several capabilities were developed to accompany trustd in the CORE environment to facilitate experimentation and testing.

**Trust Scenario Generation** For consistency similar to Section B, in order to provide a repeatable environment, we have
created a discrete event experiment driver that enables the control over the results produced by the various trust instrumentation and calculators. The format of an event is: \{time, trustor, trustee, trust integrity score, and trust competence score\}. This provides control over any direct observation nodes will send to the trust data manager for other nodes in the network. This will diffuse through the trust evaluation process in the indirect trust value distribution. Rather than directly varying the behavior of the nodes, this approach transforms each trust instrument (and judge) into a black box that produces individual trust scores that are consistent with how a real instrument would behave given the situation under evaluation.

Scenario Visualization: We show the screen shots of CORE network topology in Fig. 4. In addition to the CORE GUI, the Scripted Display Tools (SDT3D) [17] was used for playback and observation of emulation execution, whose screen shot is shown in Fig. 5. Visually, one can see the evolution of trust between nodes through the color coding of relative values.

![Fig. 4: trustd CORE topology screenshots](image)

![Fig. 5: trustd STD3D GUI](image)

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Experimental Setup

The emulation result obtained from the CORE-based trust implementation with each node deployed with trustd is compared with the simulation results implemented in SMPL (Simulation Modeling Programming Language) [18] which is an event-driven simulation language in C.

For comparison of simulation and emulation results, we set the default values of key parameters as summarized in Table I. We deploy \( N = 20 \) nodes in a mobile ad hoc network where each node moves around based on random way point model. We use \( \beta = 0.95 \) to consider direct evidence while \( 1 - \beta = 0.05 \) to consider indirect evidence to estimate overall trust per trust property. We equally weigh each trust property in integrity and competence when representing the overall trust values over time. We injected 20% of compromised nodes when the time is at 30s, performing bad-mouthing attacks. The compromised nodes are selected randomly with nodes 1, 6, 9, 15 with its ground truth integrity trust set to 0 at the specified compromised time. The CORE implementation is run for a minute to show the evolution of trust. For other default values of other key design parameters, refer to Table I.

B. Comparison of Simulation and Emulation Results

We consider the following scenario. At \( t = 30s \), nodes \{1, 6, 9, 15\} are compromised, and they inject fake recommendations which are used as indirect evidence for other nodes to estimate trust. If a node is compromised, it feeds same recommendation to all nodes by broadcasting fake recommendations. An attacker will feed a trust value recommendation of 0 for healthy nodes while feeding a trust value recommendation of 1 for compromised nodes.

![Fig. 6: Simulation overall trust estimate of trustor node 10 and trustee nodes 6 and 8.](image)

We show results from a simulations in Fig. 6 and emulations in Fig. 7. In this experiment, we show how a healthy node can estimate trust of a compromised node (i.e., node 6) and a healthy node (i.e., node 8). Note that a healthy node’s ground truth trust is set to 1 while a compromised node’s ground truth trust is 0. We show two curves where each curve indicates an estimated trust of node 6 (blue) and that of node 8 (gray).

We note several discrepancies between the emulation and simulation results. First, there were differences in the effective
topology from which the trust was being calculated. Although the position information was identical, there were differences in the actual links being established in the emulation and simulation. Thus, we see clear convergence to the ground truth trust in the simulation result from the beginning where node 6’s overall trust is 0.5 while node 8’s overall trust is 1 with approximately 0.1 inaccuracy, as shown in Fig. 6. On the other hand, the emulation result suffers from inaccurate trust values because the routes in the CORE emulation may not be completely established or maintained through the emulation. However, after an attack is injected at 30 s, 10 s later node 8’s trust is way above node 6’s trust based on the evaluation of node 8, which is reasonable although there are extra delay in the emulation. Second, as a result of the topology varying, the evolution of trust in the simulation occurs rather quickly, but there are sporadic updates in the emulated implementation.

VI. Conclusion

We described the implementation of trustd, a trust evaluation capability for mobile tactical networks. We have demonstrated its execution in an emulated environment and compared its performance to similar simulations. This provides a valuable platform on which to run future experimentation of trust-based protocols in networked environments.

In the current experiments, we showed the preliminary results based on one scenario. We plan to extend this work by (1) exploring further experiments in order to improve the performance of the CORE emulation with trustd in each node; (2) introducing trust threshold to filter trustworthy recommendations based on the trust of recommenders; and (3) trying other scenarios to increase the validity of this experiment platform.

REFERENCES