Using Trust in Distributed Consensus with Adversaries in Sensor and Other Networks

Xiangyang Liu
Institute for Systems Research
University of Maryland College Park
Email: xyliu@umd.edu

John S. Baras
Institute for Systems Research
University of Maryland College Park
Email: baras@umd.edu

I. ABSTRACT

Extensive research efforts have been devoted to distributed consensus with adversaries. Many diverse applications drive this increased interest in this area including distributed collaborative sensor networks, sensor fusion and distributed collaborative control. We consider the problem of detecting Byzantine adversaries in a network of agents with the goal of reaching consensus. We propose a novel trust model that establishes both local trust based on local evidences and global trust based on local exchange of local trust values. We describe a trust-aware consensus algorithm that integrates the trust evaluation mechanism into the traditional consensus algorithm and propose various local decision rules based on local evidence. To further enhance the robustness of trust evaluation itself, we also provide a trust propagation scheme in order to take into account evidences of other nodes in the network. The algorithm is flexible and extensible to incorporate more complicated designs of decision rules and trust models. Then we show by simulation that the trust-aware consensus algorithm can effectively detect Byzantine adversaries and excluding them from consensus iterations even in sparse networks with connectivity less than $2f+1$, where f is the number of adversaries. These results can be applied for fusion of trust evidences as well as for sensor fusion when malicious sensors are present like for example in power grid sensing and monitoring.

Keywords—distributed consensus; trust; adversaries;

II. INTRODUCTION

In distributed systems, nodes in the network are programmed to calculate given functions of nodes' values. However, due to message transmission delays, crashes, value domain errors, or even Byzantine behavior of malicious nodes, different nodes would probably have different views of the input vector of these parameters. Consensus requires that every correct agent in the network reach an agreement on some value. We will study the problem of consensus in the face of Byzantine adversaries.

The issue of consensus has been investigated for decades in the computer science, communication and control communities. There are mainly two types of failures discussed. One is crash failure and the other is Byzantine failure. A crash failure refers to the case when a process stops working, while a Byzantine process may send arbitrary data to other processes. Byzantine failures are far more disruptive since they allow arbitrary behaviors.

In the computer science community, the consensus problem is to design a distributed protocol that allows processes to agree on a single decision value, which should be one of the initial values. Based on different failure types, oracle-based consensus protocols [1] and condition-based approaches [2] have been proposed to achieve the goal. For asynchronous distributed systems, Chandra and Toeug [3] introduced the concept of Failure Detector (FD) and showed that with FD, consensus can be achieved and possible failure processes can be found. In [4] a probabilistic solution was applied to solve the consensus problem under Byzantine failures with the condition $f_b < \frac{1}{5}n$, where n is the total number of processes and f_b is the number of Byzantine failures. The leader-based consensus developed by [1] Mostefaoui et al. required $f_c < \frac{1}{2}n$ to reach consensus with crash failures, where f_c is the number of crash failures, and the time and message costs of the protocol can be reduced when $f_c < \frac{1}{3}n$. Later in [2], the leader oracle approach, the random oracle approach and the condition-based approach are combined to provided a hybrid consensus protocol. [5] connected Error-Correcting Codes (ECC) to the condition-based approach and showed that consensus can be solved despite f_c crash failures if and only if the condition can be mapped to a code whose Hamming distance is $f_c + 1$, and Byzantine consensus can be solved despite f_b Byzantine faults when the Hamming distance of the code is $2f_b + 1$.

Different from the consensus problem discussed in the computer science community, consensus in a network of connected agents means reaching an agreement regarding the states (or values of certain agent variables) of all (benign) agents that are used in computing certain functions (or function). Without considering failures, for a certain node, the consensus problem can be as simple as a weighted average of its neighbors' states [6]. Ren [7], [8] proposed update schemes for consensus under dynamically changing directed interaction topologies, provided that the union of the directed interaction graphs have a spanning tree frequently enough as the system evolves. However the linear updating rules may not be resilient to misbehaving nodes; it was shown in [9] that a single misbehaving node can cause all agents to reach consensus on a wrong value, which potentially will result in a dangerous situation in physical systems. The framework for posing and solving consensus problems for multi-agent networked systems was analyzed in [9], [10], where key results on theory and applications of consensus problems in networked systems are presented. [11] showed that the resilience of a partially connected network to Byzantine adversaries is

characterized by its network topology, and a well-behaving agent can calculate any arbitrary function of all node values when the number of malicious nodes (f) is less than $1/2$ of the network connectivity, i.e the connectivity should be at least $2f+1$. [12] used system theory to reach a similar conclusion regarding network connectivity that the number of disjoint paths between any two nodes in the network should be greater than twice the number of adversaries. LeBlanc et al. developed the Adversarial Robust Consensus Protocol (ARC-P) [13], [14], which applied ideas from both the consensus algorithms resilient to Byzantine faults in distributed computing and the linear consensus protocols used for coordination of networked agents, and formulated the problem into a linear control problem where consensus could be reached among cooperative agents via agreement and validation conditions.

Applications of both kinds of consensus problems and formulations are appropriate for, and have been used in our earlier work, distributed filtering and estimation in heterogeneous sensor networks networks with applications to power grids [15]–[17]. A Model-Based Systems Engineering framework for distributed heterogeneous sensor networks was presented in [18].

Network connectivity conditions in most previous works are hard to satisfy in reality. In addition, the robustness condition in [13] is itself hard to verify. These motivate us to design a Byzantine adversary detection scheme based on trust evaluation. We introduce the notion of *trust* in the context of consensus algorithms with Byzantine adversaries. Works related to application of trust to distributed algorithms inlcude [15], [17] who embedded trust evaluation to distributed Kalman filtering (DKF) with applications to sensor networks in power grids, and [19] who proposed RoboTrust algorithm in consensus algorithms. Our work differs from [19] in the following ways: 1) The trust model is different; and 2) Trust in [19] is established only by local evidence while our trust model also depends on second-hand evidence. 3) Trust propagation in evaluating global trust is resistant to malicious voting. Our contributions are as follows:

- We propose a trust model with various decision rules based on local evidences in the setting of Byzantine adversaries.
- Our trust-aware algorithm is flexible and can be easily extended to incorporate more complicated designs of trust models and decision rules.
- Simulations show that our proposed trust-aware consensus algorithm can effectively detect various malicious strategies even in sparse networks where connectivity $< 2f+1$, where f is the number of adversaries.

III. Problem Formulation

Consider a communication network modeled by a directed graph $G(k) = (V, E(k))$, where V denotes the set of nodes in the network and $E(k)$ the set of edges at time k. If $e_{ij}(k) \in E(k)$, it means node i can hear node j's message at time k, i.e. node j is a neighboring node of i at time k. Whether node i is able to receive node j's message depends on their relative distance and j's transmission power. A node can reach a larger portion of nodes in the network if it transmits messages with higher power. Let $N_i(k) = \{j | e_{ij}(k) \in E(k), j \neq i\}$ denote the set of neighbor nodes that node i can hear from at time k, and $N_i^+(k) = N_i(k) \bigcup \{i\}$ denote the extended neighbor set of node i at time k. We assume all nodes' transmission power is fixed. Therefore we have $N_i(k) = N_i, \forall k \geq 0$.

Let $X(k)$ denote the N-dimensional vector of all nodes' states at time k. The nodes' beliefs evolve in time according to the dynamics:

$$x_i(k) = \sum_{j \in N_i} w_{ij}(k)\, x_j(k-1) + w_{ii}(k)\, x_i(k-1) \quad (1)$$

where $X(k) = [x_1(k), x_2(k), \cdots, x_N(k)]^T$, and $x_i(k)$ denotes node i's updated state at time k and $w_{ij}(k) \geqslant 0, j \neq i$ is the weight that node i puts on node j's belief at the previous time instant for the calculation of its state update. $w_{ii}(k) \geqslant 0$ is the weight that node i puts on its own previous state. Coefficients are normalized and satisfy $\sum_{j \in N_i(k)} w_{ij}(k) + w_{ii}(k) = 1$. We denote by $W(k)$ the $N \times N$ matrix with element $W_{ij}(k)$. Equation (1) is a standard module where all nodes are normal.

In a distributed environment, due to lack of central monitoring, nodes are subject to various attacks. In the worst case, some nodes might be hacked and do not function as they are originally programmed. We consider the following Byzantine adversary model:

Definition 3.1: **Byzantine adversary** may behave arbitrarily. It does not follow the prescribed distributed consensus update rule, i.e. at some time instant $k > 0$, a Byzantine adversary i sends incorrect message $v_i(k)$ other than $x_i(k)$ in equation (1) to its neighbors. We assume a broadcast model here, meaning adversaries send the same message to different neighbors. In addition, the adversary is assumed to have complete knowledge of the network topology, the states of all other nodes in the network, and the consensus update rule for all nodes.

Next, we define a normal node's behavior and the information it can get access to.

Definition 3.2: **A normal node** behaves according to the distributed consensus specification, i.e. it updates its states by combining the messages received from its neighbors using the specified coefficients. The normal node has access to just local information such as from its neighborhood, the messages sent by them, the coefficients it uses for updating states. Moreover, it does not know whether a neighbor is normal or malicious.

From the above definitions, we can see that a normal node i has no way of determining whether one of its neighbors j is malicious or not since it can not get access to all the messages sent by its 2-hop neighbors $l \in N_j \setminus \{i\}$. To detect malicious nodes locally, we introduce the trust model and establish local trust between nodes based on first-hand evidence and define several decision rules that map from local evidence to local decisions. Often local evidence is not sufficient to reach useful decisions. We therefore define global trust based on both first-hand evidence and evidence of other nodes in the network. Our proposed trust-aware consensus algorithm takes global trust values as input.

IV. TRUST MODEL

There are two possible connections from node i to node j. One is communication connection. If $j \in N_i(k)$, node j is said to be in the communication neighborhood of node i at time instant k. The other is trust connection. Denote the set of trust relations at time instant k as $E_c(k)$. A directed edge from node i to node j, denoted as $e_{ij}^c(k) \in E_c(k)$, represents the trust relation node i has towards node j. We assume that if there is communication connection from node i to node j at time k, there must exist trust relation $e_{i,j}^c(l), \forall l \geq k$ since node i receives message from node j at time k which forms the direct evidence for i to evaluate j's trustworthiness at the current iteration k and future iterations. However, if there exists trust relation $e_{ij}^c(k)$, communication connection e_{ij} does not necessarily exist because the trust relation is possibly derived from indirect observations of other nodes in the network.

We associate a local trust value $c_{ij}(k) \in [0,1)$ with the trust connection $e_{ij}^c(k) \in E_c(k)$. It represents the belief node i holds about j at time instant k based on its local interactions with node j. The value $c_{ij}(k)$ can be seen, in node i's perspective, as the probability of node j behaving normally at time instant k. It depends on both the interaction between i and j at time k and history records i has on node j. We utilize the Beta reputation system [20] to model local trust values. Denote the probability that node j behaves normally at time instant k in node i's perspective as $p_{ij}(k)$. $p_{ij}(k)$ is assumed to have beta distribution with parameter $\alpha_{ij}(k)$ and $\beta_{ij}(k)$:

$$\begin{aligned} &f\left(p_{ij}(k)|\alpha_{ij}(k),\beta_{ij}(k)\right) \\ &= \frac{\Gamma\left(\alpha_{ij}(k)+\beta_{ij}(k)\right)}{\Gamma\left(\alpha_{ij}(k)\right)\Gamma\left(\beta_{ij}(k)\right)} p_{ij}(k)^{\alpha_{ij}(k)-1}\left(1-p_{ij}(k)\right)^{\beta_{ij}(k)-1} \end{aligned} \tag{2}$$

where Γ is Gamma distribution. Let $r_{ij}(k) = \alpha_{ij}(k) - 1$ and $s_{ij}(k) = \beta_{ij}(k) - 1$, to represent the number of events that node j is normal and the number of events j is malicious up to time k in the perspective of node i respectively. The local trust value $c_{ij}(k)$ is defined to be the mean value of $p_{ij}(k)$, i.e.

$$c_{ij}(k) = \mathbb{E}\left[p_{ij}(k)\right] = \frac{r_{ij}(k)+1}{r_{ij}(k)+s_{ij}(k)+2} \tag{3}$$

Considering the time-varying behavior of node j, old records are less important than more recent observations. We introduce positive forgetting factors ρ_1 and ρ_2 such that

$$\begin{aligned} r_{ij}(k+1) &= \rho_1 r_{ij}(k) + I_{ij}(k+1) \\ s_{ij}(k+1) &= \rho_2 s_{ij}(k) + 1 - I_{ij}(k+1), \end{aligned} \tag{4}$$

where $I_{ij}(k+1)$ equals 1 if node i believes that node j behaves normally at time $k+1$ and equals 0 otherwise. In practice, we may choose $\rho_1 < \rho_2$ so that bad behaviors are remembered for longer period of time relative to good behaviors. Note that $I_{ij}(k+1)$ can also take fractional values from $[0,1]$ which allows us to encode the uncertainty of local decisions.

A. Local Trust Evaluation

We now discuss the question of evaluating the indicator function $I_{ij}(k)$ based on node i's local observation about node j at time instant k, which involves the interplay of the consensus algorithm and the trust computation. To answer this question, we have to examine what can be used as local evidence for node i to determine whether a neighbor j behaves normally or maliciously at time instant k. Essentially, we want to find a mapping from trust evidence to a binary decision. There are many choices of evidences available and the mapping can also be arbitrary. Denote the values $v_j(k)$ received from $j \in N_i(k)$ as the vector $r_i(k)$. We discuss three decision rules (mappings) below. One is based on clustering technques, the second is based on distance of the messages, and the third on the consistency of 2-hop messages.

1) Clustering-based Decision Rule: The motivation behind clustering-based decision rules is the observation that the malicious node's message tends to deviate from normal nodes' messages. Therefore the node whose message is far away from the rest is likely to be malicious. Formally, we define the deviation of a message sent by node j from the all other messages received by node i as

$$\text{dev}_{ij}(k) = \sum_{l \in N_i^+} \frac{|v_l(k) - v_j(k)|^2}{|N_i^+|} \tag{5}$$

The ranking of the deviation in equation (5) itself can not indicate whether node j is malicious or not because a normal node might be the one deviating the most when convergence is almost reached and all nodes' messages are close to each other. Therefore we propose a decision rule based on relative ranking:

$$I_{ij}(k) = \begin{cases} 1 & \text{if dev}_{ij}(k) \leq \text{Th}_i * \text{median}\left(\{dev_{ij}(k)\}\right) \\ 0 & \text{o.w} \end{cases} \tag{6}$$

where Th_i is the threshold used by node i and median$(\cdot)$ denotes the median of values within the bracket. The above decision rule reflects the heuristics that the node whose message is too far away from the median is deemed to be malicious.

2) Distance-based Decision Rule: Denote the distance between node i's state at time instant k, $x_i(k)$ and the value $v_j(k)$ as

$$d_{ij}\left(x_i(k-1), v_j(k-1)\right) = \|x_i(k-1) - v_j(k-1)\|_2 \tag{7}$$

$d_{ij}\left(x_i(k-1), v_j(k-1)\right)$ measures the degree of disagreement of node j from node i at time $k-1$. We measure the degree of cooperation by $\Delta_{ij}(k)$ defined as:

$$\Delta_{ij}(k) = d_{ij}\left(x_i(k-1), v_j(k-1)\right) - d_{ij}\left(x_i(k-1), v_j(k)\right) \tag{8}$$

$\Delta_{ij}(k)$ measures the degree of cooperation in the sense that if node j cooperates (normal), its state is expected to be closer to node i as the iteration goes on and that if node j does not cooperate (Byzantine), it may sends value $v_j(k)$ that are far away from $x_i(k-1)$. The decision rule therefore can be specified as:

$$I_{ij}(k) = \begin{cases} 1 & \text{if } \Delta_{ij}(k) \geq 0 \\ 0 & \text{o.w} \end{cases} \tag{9}$$

3) Consistency-based Decision Rule: For node i, to verify the correctness of message $v_j(k)$, it needs more information than $v_j(k)$. We propose to augment the message sent from node j as follows. Several new notations will be introduced. The **inner message** of node j at time instant k is defined as the messages that node j collects from its neighbors and we denote it as $X_j^*(k) = \{x_{jl}^*(k-1), l \in N_j^+\}$ with $x_{jj}^*(k-$

$1) = x_j^c(k-1)$, the calculated state of node j at super-step $k-1$. The caculated state is defined in Table I. Similarly, we define the **outer message** of node j at time instant k to be the messages broadcast by node j and we denote it as $X_j(k) = \{x_{jl}(k-1), l \in N_j^+\}$, where $x_{jj}(k-1)$ might not equal to $x_j^c(k-1)$ for malicious nodes. For normal nodes, the inner message and the outer message equals, i.e. $X_j(k) = X_j^*(k)$. However, for malicious nodes, they can choose to broadcast messages different from inner message, i.e. $X_j(k) \neq X_j^*(k)$. The local decision node i makes about node j is not a scalar value. Instead, it consists of a set of values $I_{ij}^l(k), \forall l \in N_j^+$, where $I_{ij}^l(k)$ is node i's local decision about node j's message $x_{jl}(k-1)$. Therefore the decision rule becomes

$$I_{ij}^l(k) = \begin{cases} 1, & x_{il}^*(k-1) = x_{jl}(k-1), l \in N_i \\ 0, & x_{il}^*(k-1) \neq x_{jl}(k-1), l \in N_i \\ 0.5, & l \notin N_i \end{cases} \tag{10}$$

where $x_{il}^*(k-1)$ denotes the inner message element acquired by node i about l's state if node l can be heard by node i. When $l \notin N_i$, local evidence available to node i is not sufficient to reach any decision. Therefore $I_{ij}^l(k) = 0.5$. We need to specify a function that maps from these atomic decisions $I_{ij}^l(k)$ to a single scalar decision about j. We choose the following mapping as in [21]:

$$I_{ij}(k) = \prod_{l \in N_j} I_{ij}^l(k) \tag{11}$$

Since $I_{ij}^l(k) \in [0,1]$, it can be interpreted as the probability of node j behaving maliciously regarding x_{jl}. The mapping in equation (11) indicates that I_{ij} is closer to 1 only if all $I_{ij}^l(k)$'s are closer to 1 and that I_{ij} is closer to 0 if any $I_{ij}^l(k)$ is closer to 0. The aggregated decision toward node j at super step k, $I_{ij}(k)$, is then used to update node i's local trust value toward node j, c_{ij}, according to equation (3).

B. Global Trust Evaluation

Node i maintains its local trust opinion $c_{ij}(k)$ about node j for $j \in N_i$. However, when node i wants to get a more accurate estimation of the trustworthiness of node j, it needs to rely on the observations of other nodes in the network, i.e. node i needs to aggregate local trust values through trust propagation (we use global trust evaluation and trust propagation interchangably from now on). Denote the global trust of node j in the perspective of node i as t_{ij}. [22] suggests to aggregate local trust values by weighing node i's neighbors' local trust opinions about node j. Before the consensus algorithm advance to the next iteration, we want to obtain the equilibrium global trust opinion through the trust propagation subroutine, which is also an iterative process. Therefore, we assume $c_{ij}, \forall i,j$ remain constant during the iterations to obtain global trust and use τ to denote the iteration number of global trust values. It is a smaller time scale compared to consensus iteration number k. Omitting the time instant k for convenience, we have

$$t_{ij}^\tau = \begin{cases} 1 & \text{if } i = j \\ \frac{1}{z_i} \sum_{l \in N_i, l \neq j} c_{il} t_{lj}^{\tau-1} & \text{if } i \neq j \end{cases} \tag{12}$$

where the normalizing constant is $z_i = \sum_{l \in N_i, l \neq j} c_{il}$.

Note that in a distributed environment, only $c_{il}, l \in N_i$ are stored locally at node i while c_{lj} is sent by node i's neighbor l. However, node l might be malicious and lying about c_{lj}. Specifically, if node j is normal node, node l intentionally report to node i that $c_{lj} = 0$. Similarly, if node j is Byzantine, node l protects its peer by reporting $c_{lj} = 1$. To cope with this concern, we introduce a set of pre-trusted nodes in the network, namely *headers*.

Definition 4.1: **Headers** are a set of pretrusted nodes besides V. They are equipped with more security measures and therefore more reliable. The header's identity remains anonymous, i.e. neither a normal node in V nor an adversary knows if a given node is header or not. Therefore, a normal node can choose to trust or distrust a header since it does not know which node is header.

The introduction of anonymous headers induce costs since they come with higher level of reliability. Therefore we might prefer to deploy headers in denser areas instead of sparse areas in order to make the most use of them. The problem of how to deploy headers optimally with fixed number of headers is beyond the scope of this paper.

Denote the trust that node i places on a header h as p_{ih}. Node i aggregates local trust from both its neighbors in N_i and headers that it can receive message from. The global trust vector $\vec{t_i}$ evolves according to:

$$t_{ij}^\tau = \begin{cases} 1 & \text{if } i = j \\ \frac{1}{z_i + b_H} \left(\sum_{l \in N_i, l \neq j} c_{il} t_{lj}^{\tau-1} + \sum_{h \in H} p_{ih} c_{hj} \right) & \text{if } i \neq j \end{cases} \tag{13}$$

where H is the set of headers, $b_H = \sum_{h \in H} p_{ih}$, $\sum_{m \in N_i, m \neq j} c_{il}$, and $z_i + b_H$ as a whole serves as normalizing constant. Since node i does not know which node is header, p_{ih} might be smaller than 1.

The introduction of trust propagation (global trust evaluation) to the trust model can incur high computational cost since it is an embedded iterative process. Therefore, we can choose to either activate this part or mute it wisely in practice. In this paper, we give simulation results for both cases and leave the discussion of when to activate trust propagation and when to mute it as future work.

V. Trust Aware Consensus Algorithm

Given the trust model above, we are now in the position to propose our trust-aware consensus algorithm. The state dynamics goes as follows:

$$x_i(k) = \frac{1}{A_i(k)} \sum_{j \in N_i} t_{ij}(k) v_j(k-1) \tag{14}$$

where $A_i(k) = \sum_{j \in N_i} t_{ij}(k)$, $t_{ij}(k)$ is the equilibrium global trust value of node j in equation (13), and $v_j(k-1)$ is the message sent by node j. If j is malicious, $v_j(k-1)$ is not necessarily $x_j(k-1)$. Using $t_{ij}(k)$ as the linear coefficient to combine message $v_j(k-1)$ is just one way to do it. There are other more complex choices.

Note that the trust model as well as the decision rules defined in the previous section become a pluggable component to the trust-aware consensus algorithm because it only needs as input the trust dynamics given states, ignoring the details

of trust model design and decision rules. This makes the algorithm highly extensible to future developments of more complicated trust models and decision rules.

The full algorithm consisting of the trust-aware consensus algorithm and the trust model based on cluster-based decision rule or distance-based decision rule is shown in Algorithm 1.

Algorithm 1: Trust-Aware Consensus Algorithm

Input: initial states $x_i(0), \forall i \in V$ and initial local trust $c_{ij}(0), \forall i \in V \setminus \mathcal{F}, \forall j \in V \cup H$, ϵ
Output:
for $\forall i \in V \setminus \mathcal{F}$ **do**
  repeat
    Receive messages from neighbors $v_j(k-1), \forall j \in N_i$
    `// Update local trust values`
    for $\forall j \in N_i$ **do**
      $r_{ij}(k+1) = \rho_1 r_{ij}(k) + I_{ij}(k+1)$
      $s_{ij}(k+1) = \rho_2 s_{ij}(k) + 1 - I_{ij}(k+1)$
      $c_{ij}(k) = \mathbb{E}[p_{ij}(k)] = \frac{r_{ij}(k)+1}{r_{ij}(k)+s_{ij}(k)+2}$
    end
    `// Compute global trust values` $t_{ij}(k)$
    for $\forall j \in N_i$ **do**
      `// The local trust values` $c_{ij}(k)$ `remain constant within trust iterations.`
      repeat
        update $t_{ij}^{\tau}(k)$ according to equation (13)
      until $|t_{ij}^{\tau}(k) - t_{ij}^{\tau-1}(k)| < \epsilon$;
    end
    Update state x_i based on global trust values according to equation (14)
  until $|x_i(k) - x_i(k-1)| < \epsilon$;
end

The trust-aware consensus algorithm with consistency-based decision rule based on augmented messages is shown in Algorithm 2. In practice, an ensemble of the various choices of decision rules will be used.

Note that the actions within the for loops in Algorithm 1 and Algorithm 2 are executed in parallel instead of in a sequential way. Our trust-aware consensus algorithm is not restricted to the three decision rules in Section IV. We can readily place more delicate decision rules into the algorithm. Moreover, the model can incorporate more complex update schemes for trust compared to equation (4). Therefore, the trust-aware consensus algorithm is highly extensible. Table I shows the common notations used in this paper.

VI. Simulations

We consider a sensor network with 7 nodes (sensors) shown in Fig. 2. The shaded node is a Byzantine adversary and all the other nodes are normal nodes. In the simulation, each normal node uses both cluster-based decision rule and distance-based decision rule to generate local decisions. The consistency-based decision rule is evaluated theorectically in [21]. We assume the probability of choosing either one is set to 0.5.

Algorithm 2: Trust-Aware Consensus Algorithm Based on Consistency

Input: initial states $x_i(0), \forall i \in V$ and initial local trust $c_{ij}(0), \forall i \in V \setminus \mathcal{F}, \forall j \in V \cup H$, ϵ
Output:
for $\forall i \in V \setminus \mathcal{F}$ **do**
  repeat
    Receive augmented message vectors from neighbors $X_j(k-1), \forall j \in N_i$
    `// Update local trust values`
    for $\forall j \in N_i$ **do**
      compute $I_{ij}^{l}(k+1)$ according to equation (10)
      compute $I_{ij}(k+1)$ according to equation (11)
      $r_{ij}(k+1) = \rho_1 r_{ij}(k) + I_{ij}(k+1)$
      $s_{ij}(k+1) = \rho_2 s_{ij}(k) + 1 - I_{ij}(k+1)$
      $c_{ij}(k) = \mathbb{E}[p_{ij}(k)] = \frac{r_{ij}(k)+1}{r_{ij}(k)+s_{ij}(k)+2}$
    end
    `// Compute global trust values` $t_{ij}(k)$
    for $\forall j \in N_i$ **do**
      `// The local trust values` $c_{ij}(k)$ `remain constant within trust iterations.`
      repeat
        update $t_{ij}^{\tau}(k)$ according to equation (13)
      until $|t_{ij}^{\tau}(k) - t_{ij}^{\tau-1}(k)| < \epsilon$;
    end
    Update state x_i based on global trust values according to equation (14)
  until $|x_i(k) - x_i(k-1)| < \epsilon$;
end

TABLE I. Commonly Used Notations

Notation	Definition
k	time step at the top layer (communication graph)
τ	time step at the bottom layer (trust graph) which is a smaller time scale compared to k
V	set of nodes in the network (nodes are the same in both layers)
$x_j^c(k)$	calculated state of node j according to equation (14)
$x_j(0)$	initial state of node j
$X_j^*(k)$	messages that node j hears from its neighbors $N_j^+(k-1)$
$X_j(k)$	messages that node j broadcast about what it hears and what it calculates
$c_{ij}(k)$	local trust value node i has about node j at iteration k for consensus algorithm
$t_{ij}^{\tau}(k)$	global trust opinion held by node i toward j at iteration τ for kth round of consensus algorithm
$t_{ij}(k)$	equilibrium global trust opinion held by node i toward j at iteration k for consensus algorithm

In practice, each node (sensor) can have its own parameters for the probabilities of randomly choosing decision rules. For simulation purposes, we consider the following 4 malicious strategies adopted by the Byzantine adversary:

1) *Remain constant*: the adversary, disregarding the update rule in equation (14), holds a constant value.
2) *Random vibration*: the adversary switches between several values randomly at each iteration.
3) *Random noise*: the adversary adds a random noise to the state calculted if it is a normal node.
4) *Fixed noise*: the adversary adds a fixed input to the state

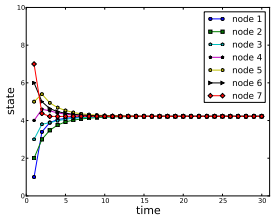
calculated if it is a normal node.

The simulation results are shown in Fig. 1. First look at Fig. 1(a) and Fig. 1(b). When no adversaries exist, the use of global trust (trust propagation) can speed up convergence. When node 7 is set to be adversary and it adopts remain constant strategy, all nodes can still reach convergence as shown in Fig. 1(c). However, all nodes are dragged to closer to the constant input of node 7 because local evidence of nodes are not sufficient to reach good decision at the beginning of the consensus iterations and it takes a period of time before all other nodes can detect the adversary and exclude it from consensus updates. This problem is mitigated when we invoke global trust and take advantage of trust decisions made by other nodes in the network as shown in Fig. 1(d). Nodes can detect adversary much faster than in Fig. 1(c) and all normal nodes remain relatively unaffected by node 7. The effects of earlier detection also happpens for use of trust propagation when adversary adopts random vibration strategy as in Fig. 1(e) and Fig. 1(f). Detection of adversary with random noise and fixed noise is more subtle. Local evidences are not sufficient to detect this malicious behavior as shown in Fig. 1(g) and Fig. 1(i). With global trust, adversaries can still be detected Fig. 1(h) and Fig. 1(j).

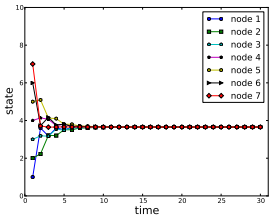
Next we present the performance of trust-aware consensus algorithm in an even sparser sensor network shown in Fig. 3. The communication graph contains seven nodes numbering from 1 to 7 plus a triangle node. The triangle node is a header node and the links connecting the header node and others are not part of the communication graph. However, the dotted links are in the trust graph. Therefore network connectivity of the communication graph in this example network is 2, rendering connectivity-based approaches in most of previous works invalid because connectivity $< 2f+1$. The header node does not participate in consensus iterations but is involved in the global trust evaluation process in equation (13). The dotted lines indicate that node 5 and node 6 can obtain trust decisions from header. We assume header node can provide trust decisions about node 1 and 4 and since header is anonymous in the eye of normal nodes, normal nodes do not necessarily trust headers. Therefore we set the local trust value from node 5 and 6 toward header to be 0.5. The results are shown in Fig. 4. We observe that even in this sparse network, normal nodes can still detect node 7 and reach consensus eventually.

VII. Conclusions

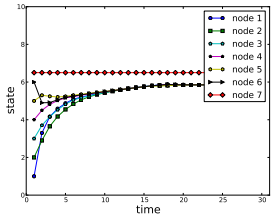
In this paper, we proposed a trust model with various decision rules based on local evidence in the setting of distributed consensus with adversaries. The global trust evaluation (trust propagation) can be used to obtain more accurate trust evaluation results if local evidences alone are not sufficient. The design of the trust-aware consensus algorithm is flexible in that it can incorporate more delicate decision rules and trust models. To evaluate the performance of the trust-aware consensus algorithm, we ran simulations and we showed that our proposed trust-aware consensus algorithm could effectively detect various malicious strategies even in network with very low connectivity. The results can be applied to distributed collaborative sensor networks, sensor fusion and collaborative control.



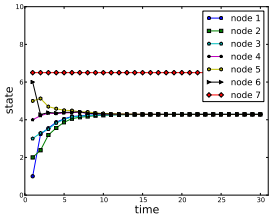
(a) No trust propagation and no adversaries.



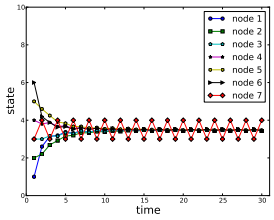
(b) No adversary. Use trust propagation.



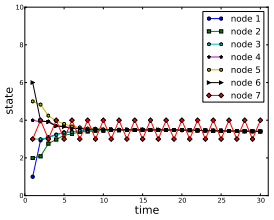
(c) Node 7 is adversary. No trust propagation and adversary adopts remain constant strategy.



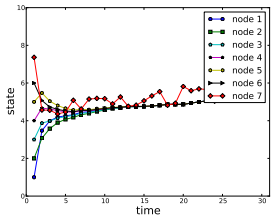
(d) Node 7 is adversary. Use trust propagation. Adversary adopts remain constant strategy.



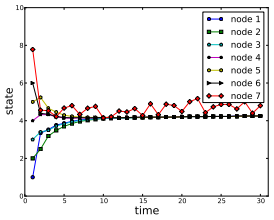
(e) Node7 is adversary. No trust propagation and adversary adopts random vibration strategy.



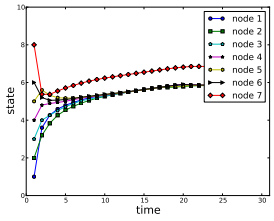
(f) Node 7 is adversary. Use trust propagation. Adversary adopts random vibration strategy.



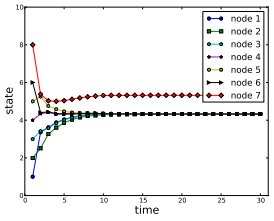
(g) Node 7 is adversary. No trust propagation. Adversary adopts random noise strategy.



(h) Node 7 is adversary. Use trust propagation. Adversary adopts random noise strategy.



(i) Node 7 is adversary. No trust propagation. Adversary adopts fixed noise strategy.



(j) Node 7 is adversary. Use trust propagation. Adversary adopts fixed noise strategy.

Fig. 1. Trust-aware consensus algorithm in situations with or without trust propagation, with or without adversary, and under 4 simulated adversary strategies.

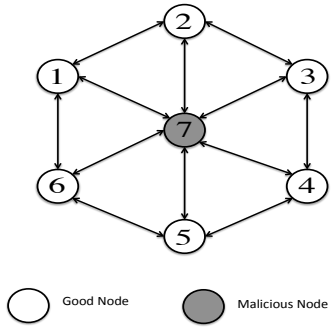


Fig. 2. Sensor network of 7 nodes (sensors) with the centering shaded node as Byzantine adversary.

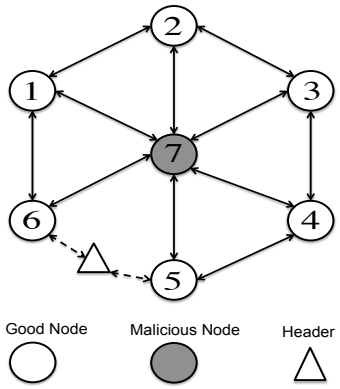


Fig. 3. Network of 8 nodes (sensors) with the centering shaded node as Byzantine adversary and triangle node as header. The dotted links incident on the header node are not in the communication graph. Instead, they belong to trust graph. Nodes 1 to 7 form a communication graph of connectivity 2.

ACKNOWLEDGMENT

Research partially supported by grants AFOSR MURI FA-9550-10-1-0573, NSF CNS-1035655, NIST 70NANB11H148, NSF CNS-1018346.

REFERENCES

[1] A. MOSTEFAOUI and M. RAYNAL, "Leader-based consensus," *Parallel Processing Letters*, vol. 11, no. 01, pp. 95–107, 2001.

[2] A. Mostéfaoui, S. Rajsbaum, and M. Raynal, "A versatile and modular consensus protoco," in *Proceedings of the 2002 International Conference on Dependable Systems and Networks*, ser. DSN '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 364–373.

[3] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, pp. 225–267, Mar. 1996.

[4] M. Ben-Or, "Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols," in *Proceedings of the second annual ACM symposium on Principles of distributed computing*, ser. PODC '83. New York, NY, USA: ACM, 1983, pp. 27–30.

[5] R. Friedman, A. Mostefaoui, S. Rajsbaum, and M. Raynal, "Asynchronous agreement and its relation with error-correcting codes," *IEEE Trans. Comput.*, vol. 56, no. 7, pp. 865–875, Jul. 2007.

[6] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, 2003.

[7] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 1859–1864 vol. 3.

[8] W. Ren and R. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *Automatic Control, IEEE Transactions on*, vol. 50, no. 5, pp. 655–661, 2005.

[9] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, 2004.

[10] R. Olfati-saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, 2007, p. 2007.

[11] S. Sundaram and C. Hadjicostis, "Distributed function calculation via linear iterative strategies in the presence of malicious agents," *Automatic Control, IEEE Transactions on*, vol. 56, no. 7, pp. 1495–1508, 2011.

[12] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus computation in unreliable networks: A system theoretic approach," *Automatic Control, IEEE Transactions on*, vol. 57, no. 1, pp. 90–104, 2012.

[13] H. J. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos, "Consensus of multi-agent networks in the presence of adversaries using only local information," in *Proceedings of the 1st international conference on High Confidence Networked Systems*, ser. HiCoNS '12. New York, NY, USA: ACM, 2012, pp. 1–10.

[14] H. J. LeBlanc and X. D. Koutsoukos, "Low complexity resilient consensus in networked multi-agent systems with adversaries," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: ACM, 2012, pp. 5–14.

[15] K. K. Somasundaram and J. S. Baras, "Performance improvements in distributed estimation and fusion induced by a trusted core," in *Information Fusion, 2009. FUSION'09. 12th International Conference on*. IEEE, 2009, pp. 1942–1949.

[16] J. S. Baras, T. Jiang, and P. Purkayastha, "Constrained coalitional games and networks of autonomous agents," in *Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on*. IEEE, 2008, pp. 972–979.

[17] I. Matei, J. S. Baras, and T. Jiang, "A composite trust model and its application to collaborative distributed information fusion," in *Information Fusion, 2009. FUSION'09. 12th International Conference on*. IEEE, 2009, pp. 1950–1957.

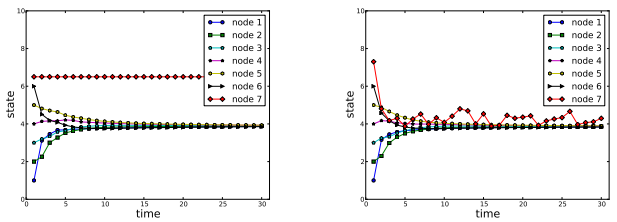
[18] B. Wang and J. S. Baras, "Integrated modeling and simulation framework for wireless sensor networks," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on*. IEEE, 2012, pp. 268–273.

[19] D. G. Mikulski, F. L. Lewis, E. Y. Gu, and G. R. Hudas, "Trust method for multi-agent consensus," in *Proc. of SPIE Vol*, vol. 8387, 2012, pp. 83 870E–1.

[20] A. Jsang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th bled electronic commerce conference*, 2002, pp. 41–55.

[21] X. Liu, P. Gao, and J. S. Baras, "Trust aware consensus with adversaries," 2014, submitted for publication.

[22] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 640–651.



(a) Adversary takes constant strategy. (b) Adversary takes random noise strategy.

Fig. 4. Trust-aware consensus algorithm in sparse network Fig. 3 of connectivity 2.