# Trustworthy Distributed Average Consensus

Christoforos N. Hadjicostis and Alejandro D. Domínguez-García

*Abstract*— **This paper proposes a distributed algorithm for average consensus in a multi-agent system under a fixed, possibly directed communication topology, in the presence of malicious agents (nodes) that may try to influence the average consensus value by manipulating their initial values and/or their updates in an arbitrary manner. The proposed algorithm is iterative and asymptotically converges to the average of the initial values of the non-malicious nodes (referred to as the *average of the trustworthy nodes*), as long as the underlying topology that describes the information exchange among the non-malicious nodes is strongly connected. The algorithm assumes that each node receives (at each iteration or periodically) side information about the trustworthiness of the other nodes, and it uses such trust assessments to determine whether or not to incorporate messages received from an in-neighbor or take into account, for its updates and transmissions, a particular out-neighbor. The algorithm allows the perceived trustworthiness of a node about another node to be asymmetric and to fluctuate during the iteration, and guarantees asymptotic convergence to the average of the trustworthy nodes, as long as the trust assessments for each non-malicious node eventually reflect correctly the status (malicious or non-malicious) of its neighboring nodes.**

**Keywords: Distributed averaging, multi-agent systems, fault-tolerant consensus, resilience, trustworthy computation, trust values.**

## I. Introduction and Motivation

A networked distributed system consists of a set of agents (nodes) that can share information with neighboring nodes via connection links (edges), forming a generally directed interconnection topology (digraph). In such systems, it is often necessary for all or some of the nodes to calculate a function of certain parameters that are held at individual nodes and are referred to as initial values. For example, when all nodes calculate the average of these initial values, they are said to reach average consensus. Average consensus and, more generally, consensus and distributed function calculation have received a tremendous amount of attention by many communities, including the control community (which has considered applications in multi-agent systems, formation control, and sensor networks), the communication community, and the computer science community [1]–[4]. In particular, average consensus has been studied extensively, primarily in settings where convergence is asymptotic and each node processes and transmits real-valued states with infinite precision [4], [5]; however, issues of finite time completion [6], [7], quantized transmissions [8]–[11] and event-triggered operation [12]–[14] have also been considered. Reference [15] discusses several applications of distributed average consensus.

In this paper, we propose and analyze a novel distributed algorithm, which enables the non-malicious nodes of a distributed system to calculate the *exact* average of their initial values, despite the actions of malicious nodes, which try to influence the outcome of the distributed computation by arbitrarily manipulating their initial values and/or their updates, possibly in a colluding manner. Non-malicious nodes are assumed to communicate over a strongly connected (possibly directed) communication topology and to, perhaps periodically, have access to (or be able to compute) a trust assessment for each of their in-neighbors and out-neighbors. It is assumed that these trust assessments allow non-malicious nodes to eventually identify correctly whether or not a neighboring node is malicious.

The proposed scheme essentially amounts to a modified version of the *running-sum ratio consensus* algorithm [15]. The basic version of this algorithm can operate over a static, directed communication topology under the assumption that each node is aware of the number of nodes that receive its transmissions (i.e., each node knows its out-degree), though extensions have been proposed that can handle network-induced errors (such as delays [16] and packet drops [17]) or even unknown out-degrees [18]. Running-sum ratio consensus is briefly reviewed later in the paper.

In the remainder of this section, we discuss some related work on trustworthy distributed average consensus. The work in [19] considered distributed function calculation using linear iterative strategies in the presence of malicious nodes. The main idea was to exploit the connectivity of the underlying topology in order to detect and isolate the effects of malicious nodes. More specifically, for any number of malicious nodes up to $f$, if the underlying graph is $(2f+1)$-connected (i.e., any two non-neighboring nodes have at least $2f+1$ node-disjoint paths that connect them), then one can systematically exploit information that arrives from these disjoint paths in order to withstand the actions of the malicious nodes (which are allowed to behave arbitrarily during the execution of the linear iteration). Another line of work relies on mean-subsequence-reduced (MSR) algorithms [20]–[24], which can be used to reduce the effects of misbehaving nodes and reach agreement to a value that is approximately correct. For example, the work in [24] considers a distributed protocol where, at each iteration, each node drops some of

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus, and also with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: chadjic@ucy.ac.cy.

Alejandro D. Domínguez-García is with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: aledan@illinois.edu.

the extremal values that it receives from its neighbors in order to be able to withstand at most $F$ misbehaving (faulty or malicious) nodes in its neighborhood. More recently, the work in [25] considers a distributed system where stochastic values of trust between the nodes are available, and nodes use these trust values to reach agreement to a common limit value. The authors show that, under certain conditions on the trust values, the deviation of the common limit value that is reached from the true consensus value is bounded.

It is worth pointing out that the idea of adding and removing nodes from a distributed average consensus computation appears in other works that rely on the running-sum ratio consensus algorithm (e.g., [18], [26]) or in works that deal with dynamic average consensus (e.g., [27], [28]). Unlike the work in this paper, however, the aforementioned works deal with nodes that willingly remove themselves from the computation of the average.

## II. Mathematical Background and Notation

A directed graph (digraph) of order $N$ ($N \geq 2$), is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of vertices (nodes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of links (edges). A directed edge from node $v_i$ to node $v_j$ is denoted by $(v_j, v_i) \in \mathcal{E}$, and indicates that node $v_i$ can send information to node $v_j$.

A digraph is called *strongly connected* if for each pair of nodes $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path* from $v_i$ to $v_j$ i.e., we can find a sequence of nodes $v_i =: v_{l_0}, v_{l_1}, \ldots, v_{l_t} := v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$. All nodes that can send information to node $v_j$ directly are said to be its in-neighbors and belong to the set $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$, the cardinality of which is referred to as the *in-degree* of $v_j$ and is denoted by $D_j^-$. The nodes that can receive information from node $v_j$ are said to be its out-neighbors and belong to the set $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$, the cardinality of which is referred to as the *out-degree* of $v_j$ and is denoted by $D_j^+$.

Consider a distributed system, captured by a nominal directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has an initial value $x_j$. Ratio consensus and push sum are iterative distributed algorithms that allow the nodes to asymptotically calculate the average $\overline{X} = \frac{1}{N}\sum_{l=1}^{N} x_l$ by performing two linear iterations. Here, we describe the push sum algorithm for the case when the communication topology is time-varying, i.e., at iteration $k$, the topology is captured by a digraph $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ where $\mathcal{E}[k] \subseteq \mathcal{E}$. Below, we use $\mathcal{N}_j^+[k]$ ($D_j^+[k]$) to denote the set (number) of out-neighbors of node $v_j$ at iteration $k$, and $\mathcal{N}_j^-[k]$ ($D_j^-[k]$) to denote the set (number) of in-neighbors of node $v_j$ at iteration $k$.

In the push sum algorithm, each node $v_j$ maintains two state variables, $y_j[k]$ and $z_j[k]$, $k \geq 0$, and updates them as follows:

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} y_i[k]/(1 + D_i^+[k]), \quad (1)$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} z_i[k]/(1 + D_i^+[k]), \quad (2)$$

where $y_j[0] = x_j$, and $z_j[0] = 1$, for all $v_j \in \mathcal{V}$. The protocol assumes that each node $v_j$ is aware of its (instantaneous) out-degree $D_j^+[k]$ at iteration $k$, and transmits the values $\overline{y}_j[k] := y_j[k]/(1 + D_j^+[k])$, $\overline{z}_j[k] := z_j[k]/(1 + D_j^+[k])$ to all of its out-neighbors; each receiving node simply adds the values it receives from all of its in-neighbors.

Under some joint connectivity assumptions on the digraphs $\mathcal{G}[k]$, $k = 0, 1, 2, ...$, it can be shown (see, for example, [15]) that the ratio $r_j[k] := y_j[k]/z_j[k]$ asymptotically converges to the average of the initial values, i.e.,

$$\lim_{k\to\infty} r_j[k] = \frac{\sum_l y_l[0]}{\sum_l z_l[0]} = \overline{X}\,, \quad \forall v_j \in \mathcal{V}\,. \quad (3)$$

A sufficient condition for reaching asymptotic average consensus (as in (3)) is to be able to find a finite $K$ such that each union graph $(\mathcal{V}, \mathcal{E}[\tau K] \cup \mathcal{E}[\tau K+1] \cup \ldots \cup \mathcal{E}[\tau K+K-1])$ for $\tau = 0, 1, 2, \ldots$ is strongly connected.

The *running-sum ratio consensus* algorithm [15] is a variation of the ratio consensus algorithm used to overcome packet drops or other network-induced uncertainties such as unknown out degrees [18]. The ratio consensus algorithm is a time-invariant version of the push-sum algorithm described above, where the communication topology is fixed and the weights used in the iterations (1)–(2) are constant. This means that the out-degrees $D_j^+[k]$, $v_j \in \mathcal{V}$, are also fixed but in the discussion below we allow them to vary to maintain consistency with (1)–(2).

The main idea in running-sum ratio consensus is that, at time step $k$, instead of broadcasting $\overline{y}_j[k] := y_j[k]/(1 + D_j^+[k])$ and $\overline{z}_j[k] := z_j[k]/(1 + D_j^+[k])$, node $v_j$ broadcasts the so-called $y$- and $z$-running sums, denoted by $\sigma$ and $\eta$ respectively, and defined as follows:

$$\sigma_j[k+1] := \sum_{t=0}^{k} y_j[t]/(1 + D_j^+[t])\,,$$

$$\eta_j[k+1] := \sum_{t=0}^{k} z_j[t]/(1 + D_j^+[t])\,.$$

Clearly, each receiving node can recover $\overline{y}_j[k] := y_j[k]/(1 + D_j^+[k])$, $\overline{z}_j[k] := z_j[k]/(1 + D_j^+[k])$, by taking the differences $\sigma_j[k+1] - \sigma_j[k]$ and $\eta_j[k+1] - \eta_j[k]$ (i.e., the difference between two consecutive messages received from node $v_j$). Thus, apart from broadcasting/receiving $y$- and $z$-running sums, the ratio consensus algorithm could be run in exactly the same way (by first recovering the needed values at each iteration), modulo some additional bookkeeping.

To execute the running-sum ratio consensus algorithm, each node $v_j$ maintains (i) the broadcast running sum values $\sigma_j[k]$ and $\eta_j[k]$ (described above); and (ii) for each in-neighbor $v_i \in \mathcal{N}_j^-$, node $v_j$ maintains two incoming running sums, namely $\rho_{ji}[k]$, which keeps track of the $y$-running sum broadcast by node $v_i$, with $\rho_{ji}[0] = 0$, and $\nu_{ji}[k]$, which keeps track of the $z$-running sum broadcast by node $v_i$, with $\nu_{ji}[0] = 0$. Indirectly, this assumes that each node is able to associate an incoming message with the node that sends it (e.g., via a unique identifier).

## III. Trustworthy Distributed Averaging Algorithm

### A. Problem Formulation and Trust Assessments

We are given a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which describes the (fixed) communication topology among a set of nodes in a distributed system. We assume a wireless broadcast model as described in the following assumption.

**Assumption 0.** Each transmission by node $v_j \in \mathcal{V}$ is received by *all* out-neighbors of node $v_j$ (i.e., all nodes in the set $\mathcal{N}_j^+$). Furthermore, we assume that each transmission is associated with a unique node ID that allows receiving nodes to identify the sending node.

Each node $v_j \in \mathcal{V}$ has an initial value $x_j$. A certain subset $\mathcal{V}_T$, $\mathcal{V}_T \subseteq \mathcal{V}$, of the nodes is trustworthy (non-malicious), whereas the remaining nodes in $\mathcal{V}_M = \mathcal{V} \setminus \mathcal{V}_T$ are malicious. Malicious nodes can choose their initial values arbitrarily and may also collude to behave unpredictably during the execution of the algorithm. The goal of the trustworthy nodes is to compute the *average of the trustworthy nodes*, defined by

$$\overline{X}_T = \frac{\sum_{v_l \in \mathcal{V}_T} x_l}{|\mathcal{V}_T|} , \tag{4}$$

despite the initial values or any actions (such as incorrect updates) by the malicious nodes.

**Assumption 1.** The digraph induced from $\mathcal{G}$ by restricting attention to the trustworthy nodes, denoted by $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$, where $\mathcal{E}_T = \{(v_j, v_i) \in \mathcal{E} \mid v_j, v_i \in \mathcal{V}_T\}$, is strongly connected.

At each iteration $k$, each node $v_j \in \mathcal{V}_T$ has access to an assessment about the trustworthiness of each other node.[1] This assessment could be derived based on measurements of some sort, such as the *stochastic values of trust*, $a_{lj} \in (0, 1)$ in [25], [29], which approach 1 when node $v_l$ should be trusted by node $v_j$ and approach 0 when node $v_l$ should not be trusted by node $v_j$; or it could be based on checks like the ones in [30], [31], where, using two-hop communication in undirected graphs, neighbors of node $v_l$ assess the computations performed by node $v_l$ in order to determine their correctness. Such measurements could be used so that, at any given iteration $k$, each node $v_j$ reaches an assessment about the trustworthiness of another node $v_l$ (for example, using a likelihood test based on the stochastic values of trust over time, as done in [25], [29]).

In this paper, we abstract away the specific mechanism of measuring and assessing trust, and assume that, at each iteration $k$, each node $v_j$ has access to an assessment about the trustworthiness of another node $v_l$. In particular, $t_{lj}[k] \in \{0, 1\}$ is a binary indicator that captures the trustworthiness of node $v_l$ as perceived by node $v_j$ at iteration $k$: $t_{lj}[k] = 1$ ($t_{lj}[k] = 0$) indicates that node $v_j$ considers node $v_l$ to be trustworthy (malicious) at iteration $k$. We use $\mathcal{T}_j[k] = \{v_l \mid t_{lj}[k] = 1\}$ to denote the set of nodes that are considered trustworthy by node $v_j$ at iteration $k$; we assume that $t_{jj}[k] = 1$ and thus $v_j \in \mathcal{T}_j[k]$ for all $k$. Without loss of generality, we assume that initially $\mathcal{T}_j[0] = \mathcal{V}$ (i.e., at the start of the algorithm execution, all nodes are considered trustworthy by each node $v_j$). We also require that asymptotically $\lim_{k\to\infty} \mathcal{T}_j[k] = \mathcal{V}_T$, at least for nodes $v_j$ that are trustworthy. Note that convergence of $\mathcal{T}_j[k]$ to $\mathcal{V}_T$ as $k$ goes to infinity does not have to be monotonic as $t_{lj}[k]$ may fluctuate between 1 and 0.

**Assumption 2.** The trust assessments $t_{lj}[k]$, $v_l, v_j \in \mathcal{V}$, are such that for each $v_j \in \mathcal{V}_T$, there exists a finite $k_j$ such that $\mathcal{T}_j[k] = \mathcal{V}_T$ , for $k \geq k_j$ (where $\mathcal{T}_j[k] = \{v_l \mid t_{lj}[k] = 1\}$).

### B. Out-Degree Update based on Trust Assessments

The trustworthy distributed calculation of the average $\overline{X}_T$ in (4) is based on a variation of the running-sum ratio algorithm where each node $v_j$ carefully tracks its trustworthy out-neighbors at iteration $k$, given by $\mathcal{N}_j^+[k] = \mathcal{N}_j^+ \cap \mathcal{T}_j[k]$, and its trustworthy in-neighbors at iteration $k$, given by $\mathcal{N}_j^-[k] = \mathcal{N}_j^- \cap \mathcal{T}_j[k]$. More specifically, node $v_j$ uses the number of its trustworthy out-neighbors at iteration $t$, given by $D_j^+[t] = |\mathcal{N}_j^+[t]| = |\mathcal{N}_j^+ \cap \mathcal{T}_j[t]|$, to update its $y$- and $z$-running sums as

$$\begin{aligned} \sigma_j[k+1] &:= \sum_{t=0}^{k} y_j[t]/(1 + D_j^+[t]) , \\ \eta_j[k+1] &:= \sum_{t=0}^{k} z_j[t]/(1 + D_j^+[t]) . \end{aligned}$$

Note that these sums can be easily updated recursively since $\sigma_j[k+1] = \sigma_j[k] + y_j[k]/(1 + D_j^+[k])$ and $\eta_j[k+1] = \eta_j[k] + z_j[k]/(1 + D_j^+[k])$.

Moreover, node $v_j$ updates its $y$ and $z$ values using only the running sums it receives from its trustworthy in-neighbors, i.e.,

$$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k]) , \tag{5}$$

$$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k]) , \tag{6}$$

where (apart from some exceptional conditions discussed in the next section) $\rho_{jj}[k] = \sigma_j[k]$ and $\nu_{jj}[k] = \eta_j[k]$.

With the above modifications, it should be clear that the malicious nodes are excluded from the distributed computation. In fact, if $\mathcal{T}_j[k] = \mathcal{V}_T$ for all $k$ (starting from $k = 0, 1, 2, ...$), then the above modified version of the running-sum ratio consensus algorithm effectively operates on the digraph $\mathcal{G}_T$ induced from $\mathcal{G}$ by restricting attention to the trustworthy nodes, which is assumed to be strongly connected (Assumption 1). Therefore, the algorithm will converge to the average of the initial values of the trustworthy nodes comprising the digraph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$, namely $\overline{X}_T$ given in (4). The main challenge is that trust assessments are not available at the initialization of the algorithm and may also fluctuate during the operation of the algorithm.

[1] It will become obvious in our development that node $v_j$ only needs information about the trustworthiness of its in-neighbors and out-neighbors, and not necessarily all other nodes; however, for ease of notation, we assume that such information is made available at node $v_j$ for all other nodes.

### C. Adding and Removing Malicious Nodes

In order to perform the calculation of the trustworthy average, $\overline{X}_T$, trustworthy nodes need a way to add or remove the effect of the malicious nodes on the distributed computation, using the available trust assessments at each iteration. This is mainly done by adjusting their out-degree and ignoring in-neighbors that are considered malicious (as described in the previous section). In addition, they need to take specific actions when a previously considered malicious node becomes trustworthy and vice-versa. The actions taken, however, are different for out-neighbors and in-neighbors. Below, we discuss the actions from the perspective of a trustworthy node $v_j$, which has out-neighbors $v_l \in \mathcal{N}_j^+$ and in-neighbors $v_i \in \mathcal{N}_j^-$.

Case 1: Previously thought trustworthy out-neighbor $v_l$ becomes untrustworthy ($t_{lj}[k] = 0$ and $t_{lj}[k-1] = 1$). In this case, apart from decreasing its out-degree by one ($D_j^+[k] = D_j^+[k-1] - 1$), node $v_j$ needs to compensate for all transmissions that it sent to node $v_l$ while it was considered trustworthy. It does so by adding to its $y$ and $z$ values its corresponding $y$-running sum and $z$-running sum:

$$\begin{aligned} y_j[k] &:= y_j[k] + \sigma_j[k] \,, \\ z_j[k] &:= z_j[k] + \eta_j[k] \,. \end{aligned}$$

The reason is that $\sigma_j[k]$ ($\eta_j[k]$) represents the cummulative $y$ ($z$) values that were sent to the now untrustworthy out-neighbor $v_l$ (or any out-neighbor for that matter). This adjustment essentially "diverts" all messages sent to node $v_l$ up to time instant $k-1$, back to node $v_j$.

Note that if multiple out-neighbors become untrustworthy this adjustment has to be performed for each such out-neighbor. In other words, if we let $\Delta U_j^+[k] = |\mathcal{N}_j^+ \cap (\mathcal{T}_j[k-1] \setminus \mathcal{T}_j[k])|$ be the number of previously trustworthy out-neighbors of node $v_j$ that become untrustworthy at iteration $k$, then

$$\begin{aligned} y_j[k] &:= y_j[k] + \Delta U_j^+[k]\sigma_j[k] \,, \\ z_j[k] &:= z_j[k] + \Delta U_j^+[k]\eta_j[k] \,. \end{aligned}$$

Case 2: Previously thought untrustworthy out-neighbor $v_l$ becomes trustworthy ($t_{lj}[k] = 1$ and $t_{lj}[k-1] = 0$). In this case, apart from increasing its out-degree by one ($D_j^+[k] = D_j^+[k-1] + 1$), node $v_j$ needs to adjust for all transmissions that node $v_l$ received from $v_j$ while it was considered untrustworthy by node $v_j$ (note that, under Assumption 0, node $v_l$ was receiving transmissions from node $v_j$ even though node $v_j$ did not consider node $v_l$ in its computations). Node $v_j$ can achieve this adjustment by subtracting from its own $y$ and $z$ values, its corresponding $y$-running sum and $z$-running sum:

$$\begin{aligned} y_j[k] &:= y_j[k] - \sigma_j[k] \,, \\ z_j[k] &:= z_j[k] - \eta_j[k] \,. \end{aligned}$$

As in Case 1, the above adjustment has to be performed for each out-neighbor that becomes trustworthy, i.e., if we let $\Delta T_j^+[k] = |\mathcal{N}_j^+ \cap (\mathcal{T}_j[k] \setminus \mathcal{T}_j[k-1])|$ be the number of previously untrustworthy out-neighbors of node $v_j$ that become trustworthy at iteration $k$, then

$$\begin{aligned} y_j[k] &:= y_j[k] - \Delta T_j^+[k]\sigma_j[k] \,, \\ z_j[k] &:= z_j[k] - \Delta T_j^+[k]\eta_j[k] \,. \end{aligned}$$

Cases 1 and 2 can be easily merged together: at iteration $k$, node $v_j$ can adjust its $y_j[k]$ and $z_j[k]$ values as

$$\begin{aligned} y_j[k] &:= y_j[k] + (\Delta U_j^+[k] - \Delta T_j^+[k])\sigma_j[k] \,, \\ z_j[k] &:= z_j[k] + (\Delta U_j^+[k] - \Delta T_j^+[k])\eta_j[k] \,. \end{aligned}$$

Later on, when updating its $\sigma_j[k+1]$ and $\eta_j[k+1]$ running sums, node $v_j$ uses the number of trustworthy out-neighbors at iteration $k$, given by $D_j^+[k] = |\mathcal{N}_j^+[k]| = |\mathcal{N}_j^+ \cap \mathcal{T}_j[k]|$.

Case 3: Previously thought trustworthy in-neighbor $v_i$ becomes untrustworthy ($t_{ij}[k] = 0$ and $t_{ij}[k-1] = 1$). In this case, apart from ignoring node $v_i$'s transmissions, node $v_j$ has to also remove the effect of all previous transmissions that it received from node $v_i$. This can be done by subtracting from its $y$ and $z$ values the corresponding $y$-running sum and $z$-running sum:

$$\begin{aligned} y_j[k] &:= y_j[k] - \rho_{ji}[k] \,, \\ z_j[k] &:= z_j[k] - \nu_{ji}[k] \,. \end{aligned}$$

Node $v_j$ also has to set $\rho_{ji}$ and $\nu_{ji}$ to zero in order to ensure that, if node $v_i$ becomes trustworthy in the future, node $v_i$'s effect on the computation will be properly accounted for. The above has to be done for every node in the set $\Delta\mathcal{T}_j^-[k] = \mathcal{N}_j^- \cap (\mathcal{T}_j[k-1] \setminus \mathcal{T}_j[k])$, which captures the subset of in-neighbors of node $v_j$ that have become untrustworthy at iteration $k$. In other words, node $v_j$ sets

$$\begin{aligned} y_j[k] &:= y_j[k] - \sum_{v_i \in \Delta\mathcal{T}_j^-[k]} \rho_{ji}[k] \,, \\ z_j[k] &:= z_j[k] - \sum_{v_i \in \Delta\mathcal{T}_j^-[k]} \nu_{ji}[k] \,. \end{aligned}$$

and then sets $\rho_{ji}[k] = \nu_{ji}[k] = 0$ for all $v_i \in \Delta\mathcal{T}_j^-[k]$.

Case 4: Previously thought untrustworthy in-neighbor $v_i$ becomes trustworthy ($t_{ij}[k] = 1$ and $t_{ij}[k-1] = 0$). In this case, node $v_j$ needs to simply incorporate the effect of node $v_i$'s transmissions: since $\rho_{ji}[k]$ and $\nu_{ji}[k]$ have been set to zero in previous iterations, all node $v_j$ needs to do is to simply incorporate the values of the $y$-running sum and the $z$-running sum it last received from node $v_i$ into its update in (5) and (6) with

$$\begin{aligned} \rho_{ji}[k+1] &:= \sigma_i[k+1] \,, \\ \nu_{ji}[k+1] &:= \eta_i[k+1] \,. \end{aligned}$$

Note that Cases 3 and 4 can be easily handled by having node $v_j$ simply set $\rho_{ji}[k+1] = 0$ and $\nu_{ji}[k+1] = 0$ for all in-neighbors $v_i$ that are untrustworthy at iteration $k$, and then update $y_j[k+1]$ and $z_j[k+1]$ as normal. In other words, if we let $\mathcal{N}_j^-[k] = \mathcal{N}_j^- \cap \mathcal{T}_j[k]$ be the set of trustworthy in-neighbors of node $v_j$, then node $v_j$ can handle Cases 3 and 4 by taking the following steps once it receives $\sigma_i[k+1]$

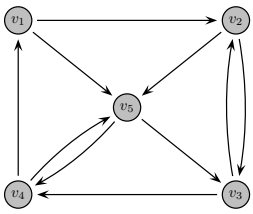


Fig. 1. Digraph considered in the example.

and $\eta_i[k+1]$ from its in-neighbors: first it sets

$$\rho_{ji}[k+1] = \begin{cases} \sigma_i[k+1]\,, & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}\,, \\ 0\,, & \text{otherwise.} \end{cases}$$

$$\nu_{ji}[k+1] = \begin{cases} \eta_i[k+1]\,, & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\}\,, \\ 0\,, & \text{otherwise.} \end{cases}$$

and then using the updates in (5)–(6) with all in-neighbors $\mathcal{N}_j^-$ used in the summations.

The pseudocode for the algorithm described above is presented as Algorithm 1 from the perspective of node $v_j$. The main convergence result for Algorithm 1 is stated next. Due to space considerations, the proof is not included; it will be provided in an extended version of the paper.

*Theorem 1:* Consider a distributed system, captured by a nominal directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in which each node $v_j \in \mathcal{V}$ has an initial value $x_j$. A certain subset $\mathcal{V}_T$, $\mathcal{V}_T \subseteq \mathcal{V}$, of the nodes are trustworthy (non-malicious), whereas the remaining nodes in $\mathcal{V}_M = \mathcal{V} \setminus \mathcal{V}_T$ are malicious. At various points in time, each node $v_j$ receives information that allows it to compute a binary indicator $t_{lj}$ regarding the trust it places to node $v_l$. Under Assumptions 0–2, if the non-malicious nodes execute Algorithm 1, they asymptotically converge to the average $\overline{X}_T$.

## IV. Example

Consider the digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in Fig. 1. We assume that the set of trustworthy nodes is $\mathcal{V}_T = \{v_1, v_2, v_3, v_4\}$ and the set of malicious nodes is $\mathcal{V}_M = \{v_5\}$. Notice that the induced digraph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$ (where $\mathcal{E}_T = \{(v_2, v_1), (v_3, v_2), (v_2, v_3), (v_4, v_3), (v_1, v_4)\}$) is strongly connected. The initial values of the nodes are $x_i = i$ for $i = 1, 2, ..., 5$, so that $\overline{X} = 3$ and $\overline{X}_T = 2.5$.

We next execute different scenarios to illustrate the operation of the proposed trustworthy distributed averaging algorithm. More specifically, we illustrate three runs of the proposed algorithm, which differ in terms of when trust assessments converge to the correct values and/or the behavior of the malicious node. In all three cases, the non-malicious nodes in $\mathcal{V}_T$ converge to the average of the trustworthy nodes $\overline{X}_T = 2.5$, whereas the malicious node may or may not converge depending on its own behavior.

On the left of Fig. 2, we see the behavior of the network when nodes are aware that node $v_5$ is malicious from the very beginning. In this simulation, the malicious node $v_5$ behaves normally (apart from providing an incorrect initial value) and we see that it also converges to the average of the trustworthy nodes. In the middle of Fig. 2, we see the behavior of the network when, up to iteration 20, nodes have randomly generated binary values for their trust assessments about other nodes; however, after iteration 20, they acquire correct trust assessments. We see that the trustworthy nodes converge to the average $\overline{X}_T$ after about 30 iterations. In this simulation, we also assume that the malicious node $v_5$ behaves normally (apart from providing an incorrect initial value) and we see that it also converges to the average $\overline{X}_T$. On the right of Fig. 2, we see a simulation with the same characteristics as the one in the middle, except that the malicious node $v_5$ behaves arbitrarily (more specifically, at each iteration, node $v_5$ adds a random offset to its $y$ value, which then propagates to the values it transmits to its out-neighbors). In this case, the malicious node does not converge to a value; however, the trustworthy nodes are able to converge to the average $\overline{X}_T$ after about 30 iterations.

**Algorithm 1: Trustworthy Distributed Averaging**

**Input:** Node $v_j$ knows $x_j$, $\mathcal{N}_j^+$, $\mathcal{N}_j^-$, and has access to sets $\mathcal{T}_j[k]$ for $k = 0, 1, 2, ...$

**Initialization:**

Node $v_j$ initializes $\mathcal{T}_j[-1] = \mathcal{N}_j^+ \cup \mathcal{N}_j^- \cup \{v_j\}$, and $y_j[0] = x_j$, $\sigma_j[0] = 0$, and $\rho_{ji}[0] = 0$, $\forall v_i \in \mathcal{N}_j^-$
$z_j[0] = 1$, $\eta_j[0] = 0$, and $\nu_{ji}[0] = 0$, $\forall v_i \in \mathcal{N}_j^-$

**for** $k \geq 0$ :

Receive $\mathcal{T}_j[k]$

**Utilize Trust Assessments of Out-Neighbors:**

Set $\mathcal{N}_j^+[k] = \mathcal{N}_j^+ \cap \mathcal{T}_j[k]$

Set $D_j^+[k] = |\mathcal{N}_j^+[k]|$

Set $\Delta U_j^+[k] = |\mathcal{N}_j^+ \cap (\mathcal{T}_j[k-1] \setminus \mathcal{T}_j[k])|$

Set $\Delta T_j^+[k] = |\mathcal{N}_j^+ \cap (\mathcal{T}_j[k] \setminus \mathcal{T}_j[k-1])|$

Cases 1 and 2:

$y_j[k] := y_j[k] + (\Delta U_j^+[k] - \Delta T_j^+[k])\sigma_j[k]$

$z_j[k] := z_j[k] + (\Delta U_j^+[k] - \Delta T_j^+[k])\eta_j[k]$

**Compute:**

$\sigma_j[k+1] = \sigma_j[k] + y_j[k]/(1 + D_j^+[k])$

$\eta_j[k+1] = \eta_j[k] + z_j[k]/(1 + D_j^+[k])$

**Broadcast:** $\sigma_j[k+1]$ and $\eta_j[k+1]$ to all $v_l \in \mathcal{N}_j^+$

**Receive:** $\sigma_i[k+1]$ and $\eta_i[k+1]$ from each $v_i \in \mathcal{N}_j^-$

**Utilize Trust Assessments of In-Neighbors:**

Set $\mathcal{N}_j^-[k] = \mathcal{N}_j^- \cap \mathcal{T}_j[k]$

Cases 3 and 4:

For each $v_i \in \mathcal{N}_j^-$ set

$$\rho_{ji}[k+1] = \begin{cases} \sigma_i[k+1]\,, & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \\ 0\,, & \text{otherwise} \end{cases}$$

$$\nu_{ji}[k+1] = \begin{cases} \eta_i[k+1]\,, & \forall v_i \in \mathcal{N}_j^-[k] \cup \{v_j\} \\ 0\,, & \text{otherwise} \end{cases}$$

**Compute:**

$y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\rho_{ji}[k+1] - \rho_{ji}[k])$

$z_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} (\nu_{ji}[k+1] - \nu_{ji}[k])$

**Output:** $r_j[k+1] = y_j[k+1]/z_j[k+1]$

## V. Conclusions and Future Work

In this paper, we have considered the problem of trustworthy distributed average consensus in multi-agent sys-

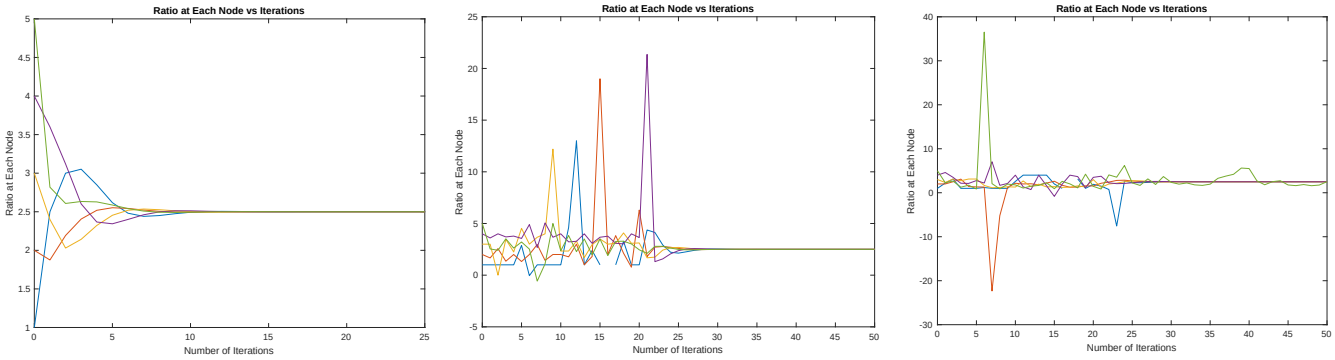


Fig. 2. Ratios of nodes in the example converge to $\overline{X}_T = 2.5$ under different scenarios regarding trust assessments and malicious node behavior.

tems, in the presence of malicious nodes that may try to influence the outcome of the computation via their (possibly collusive) choices of initial values and/or updates. The proposed algorithm allows the nodes to asymptotically converge to the average of the initial values of the trustworthy nodes, assuming that (i) the underlying directed topology that describes the information exchange among the non-malicious nodes is strongly connected, and (ii) the non-malicious nodes eventually receive correct information about the trustworthiness of other nodes. In our future work we plan to research into mechanisms for distributively obtaining the trust assessments that are required by this algorithm.

## REFERENCES


[1] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[2] N. A. Lynch, *Distributed Algorithms*. San Mateo: CA: Morgan Kaufmann Publishers, 1996.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

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

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

[6] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *Proc. of 2007 American Control Conference*, 2007, pp. 711–716.

[7] L. Wang and F. Xiao, "Finite-time consensus problems for networks of dynamic agents," *IEEE Trans. on Automatic Control*, vol. 55, no. 4, pp. 950–955, April 2010.

[8] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[9] T. C. Aysal, M. J. Coates, and M. G. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, October 2008.

[10] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, November 2009.

[11] K. Cai and H. Ishii, "Quantized consensus and averaging on gossip digraphs," *IEEE Trans. on Automatic Control*, vol. 56, no. 9, pp. 2087–2100, September 2011.

[12] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[13] A. I. Rikos and C. N. Hadjicostis, "Event-triggered quantized average consensus via ratios of accumulated values," *IEEE Trans. on Automatic Control*, vol. 66, no. 3, pp. 1293–1300, March 2020.

[14] ——, "Event-triggered quantized average consensus via mass summation," *arXiv preprint arXiv:2003.14183*, 2020.

[15] C. N. Hadjicostis, A. D. Domínguez-García, and T. Charalambous, "Distributed averaging and balancing in network systems, with applications to coordination and control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 3–4, 2018.

[16] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Trans. on Automatic Control*, vol. 59, no. 3, pp. 763–768, March 2014.

[17] A. D. Domínguez-García, C. N. Hadjicostis, and N. H. Vaidya, "Resilient networked control of distributed energy resources," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1137–1148, 2012.

[18] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Trans. on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, June 2016.

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

[20] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 766–781, 2013.

[21] S. M. Dibaji, H. Ishii, and R. Tempo, "Resilient randomized quantized consensus," *IEEE Trans. on Automatic Control*, vol. 63, no. 8, pp. 2508–2522, August 2017.

[22] Y. Wang and H. Ishii, "Resilient consensus through event-based communication," *IEEE Trans. on Control of Network Systems*, vol. 7, no. 1, pp. 471–482, 2019.

[23] D. Sakavalas, L. Tseng, and N. H. Vaidya, "Asynchronous Byzantine approximate consensus in directed networks," in *Proc. of the 39th Symposium on Principles of Distributed Computing*, 2020, pp. 149–158.

[24] 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 *Proc. of the 1st International Conference on High Confidence Networked Systems*, 2012, pp. 1–10.

[25] M. Yemini, A. Nedic, A. J. Goldsmith, and S. Gil, "Characterizing trust and resilience in distributed consensus for cyberphysical systems," *IEEE Trans. on Robotics*, vol. 38, no. 1, pp. 71–91, 2021.

[26] I. Eyal, I. Keidar, and R. Rom, "LiMoSense: live monitoring in dynamic sensor networks," *Distributed Computing*, vol. 27, no. 5, pp. 313–328, 2014.

[27] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[28] E. Montijano, J. I. Montijano, C. Sagüés, and S. Martínez, "Robust discrete time dynamic average consensus," *Automatica*, vol. 50, no. 12, pp. 3131–3138, 2014.

[29] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus, "Guaranteeing spoof-resilient multi-robot networks," *Autonomous Robots*, vol. 41, no. 6, pp. 1383–1400, 2017.

[30] L. Yuan and H. Ishii, "Resilient consensus with distributed fault detection," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 285–290, 2019.

[31] ——, "Secure consensus with distributed detection via two-hop communication," *Automatica*, vol. 131, p. 109775, 2021.