ARQ-based Average Consensus over Unreliable Directed Network Topologies

Evagoras Makridis, Themistoklis Charalambous, and Christoforos N. Hadjicostis

Abstract—In this paper, we address the problem of discrete-time average consensus, where agents (nodes) exchange information over unreliable communication links. We enhance the Robustified Ratio Consensus algorithm by exploiting features of the Automatic Repeat ReQuest (ARQ) protocol used for error control of data transmissions, in order to allow the agents to reach asymptotic average consensus even when operating within unreliable directed networks. This strategy, apart from handling time-varying delays induced by retransmissions of erroneous packets (which can be captured by the Robustified Ratio Consensus as well), can also handle packet drops that occur when exceeding a predefined packet retransmission limit imposed by the ARQ protocol. Invoking the ARQ protocol allows nodes to: (a) exploit the incoming error-free acknowledgement feedback signals to initially acquire or later update their out-degree, (b) know whether a packet has arrived or not, and (c) determine a local upper-bound on the delays which is imposed by the retransmission limit. By augmenting the network's corresponding weighted adjacency matrix, to handle time-varying (yet bounded) delays and possible packet drops, we show that nodes can make use of the proposed algorithm, herein called the ARQ-based Ratio Consensus algorithm, to reach asymptotic average consensus, while maintaining low running sum values, despite the fact that the communication links are unreliable.

Index Terms—Average consensus, digraphs, ratio consensus, ARQ feedback, heterogeneous time-varying delays, packet drops.

I. INTRODUCTION

Robust information exchange between computing nodes (usually referred to as agents or workers) is essential to apply various estimation and control algorithms in spatially distributed multi-agent systems [1]–[6]. Such distributed systems involve multiple nodes where each individual node has no knowledge of the entire system, but instead it only has some local information. Hence, to achieve certain network-wide (global) objectives, each node needs to either interact with a central master node (in a centralized master-worker architecture), or interact with its immediate neighbors (in a peer-to-peer decentralized/distributed architecture) via a communication network. Centralized topologies involve a master node collecting information from all other nodes to compute a global decision that achieves a network-wide objective. Decentralized topologies rely on nodes interacting with each other (without the need of a central coordinator) such that a global decision is obtained by combining local information (see [7] for a short review on network topology architectures for distributed computation). In practice, both communication network topologies pose some limitations and constraints on the information flow between nodes, which arise mainly due to network congestion and unreliable communication channels. Such impediments induce transmission delays and packet drops, affecting the performance and reliability of distributed computation algorithms. These issues are even more obvious in centralized network topologies, since the master node in large-scale centralized topologies may become a communication bottleneck, especially when the network resources are fixed and limited, because it needs to maintain communication with all worker nodes in the network [8]. Besides the communication bottleneck at the master node, another drawback of centralized networks is that they are vulnerable to single points of failure since a potential failure of the master node could lead to the failure of the entire network [7].

Distributed decision-making methods through peer-to-peer network topologies, often rely on nodes interacting with each other to compute a global value (*e.g.,* , the average) of a certain common quantity such that a global objective is achieved. The problem where a network of nodes maintain local information which is exchanged with neighboring nodes to compute their average value is called (distributed) average consensus (see [9] for an overview of consensus methods). Under specific conditions on the network topology and the interaction between the nodes, an accurate computation of the average value (*i.e.,* the network-wide average of the agents' initial values) can be obtained, either asymptotically [10]–[12], or in finite time [13]–[15]. Several practical examples limit the exchange of information from bidirectional to directional mainly due to diverse transmission power and interference levels of each individual node in the network. The information flow in such a paradigm is directed meaning that an agent v_j may be able to receive information from another agent v_i, but this does not necessarily imply that v_j is able to send information to v_i. In this configuration, the network topology can be modeled using directed graphs (*digraphs*), where the agents are represented by the graph's nodes, and the communication links that enable information exchange between agents are represented by the graph's edges.

For directed network topologies, the results in [12], [16], [17] have shown that agents can asymptotically reach the average. In the presence of computational and/or transmission delays on the information exchange between agents, the authors in [15], [18] proposed *ratio-consensus*-based algorithms, able

E. Makridis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus (e-mail: `makridis.evagoras@ucy.ac.cy`)

T. Charalambous is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus, and also with the Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland. (e-mail: `charalambous.themistoklis@ucy.ac.cy`)

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus, (e-mail: `chadjic@ucy.ac.cy`)

to reach (asymptotic in the former, finite-time in the latter) average consensus in directed communication networks in the presence of bounded time-invariant and time-varying delays. The ratio consensus algorithm proposed in [19], proved to be able to reach average consensus by computing in an iterative way the ratio of two concurrently running linear iterations with certain initial conditions that are properly specified. An extension of the ratio consensus algorithm is proposed in [20], where nodes exchange messages containing information of their running sums to handle potential loss of packets. They proved, by introducing extra virtual nodes and links describing the aforementioned communication scenarios and topology, that network nodes asymptotically converge to the exact average. For similar scenarios, the authors in [21] proposed a corrective consensus algorithm that enables the practical use of consensus by each node through extra variables, and extra corrective iterations, to prove almost sure convergence to the exact average in the presence of link losses.

In this paper, we cast the problem of discrete-time average consensus over possibly directed network topologies into the framework of the Automatic Repeat reQuest (ARQ) error correction protocol, which is the fundamental block for reliable communication in practical systems. Invoking the ARQ protocol allows nodes to exploit incoming error-free acknowledgement feedback signals to initially acquire or later update their out-degree, know whether a packet has arrived or not, and determine a local upper-bound on the delays, imposed by the ARQ retransmission limit. To the best of our knowledge, this is the first time that a realistic communication error correction protocol currently used in many real-life telecommunication systems, is integrated with distributed consensus iterations to compute the exact average consensus value over unreliable networks. Finally, we provide the proof of asymptotic convergence of the proposed method to the exact average and we evaluate its performance via simulations for different realistic scenarios. The proof is based on augmenting the weighted adjacency matrix that represents the network topology and the interactions between nodes, to model possible arbitrary and unknown time-varying (yet bounded) delays, as well as packet drops, on the communication links.

The remainder of this paper is organized as follows. Section II introduces the network model, the ARQ error correction protocol, the basic Ratio Consensus algorithm, and the Robustified Ratio Consensus algorithm for directed graphs. Section III describes the ARQ-based Ratio Consensus method for handling unreliable communication such as delays and packet-drops due to retransmissions and excess of a predefined retransmissions limit, respectively. In Section IV we present the augmented digraph representation needed for the proof of convergence of the proposed method, while in Section V we provide numerical simulations for different packet error probabilities. Finally, in Section VI we provide concluding remarks and present future directions.

II. BACKGROUND

A. Network Model

Consider a strongly connected network captured by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ is the set of nodes (representing the n agents) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (representing the communication links between agents). The total number of edges in the network is denoted by $m = |\mathcal{E}|$. A directed edge $\varepsilon_{ji} \triangleq (v_j, v_i) \in \mathcal{E}$, where $v_j, v_i \in \mathcal{V}$, indicates that node v_j can receive information from node v_i, *i.e.*, $v_i \rightarrow v_j$. The nodes that transmit information to node v_j directly are called in-neighbors of node v_j, and belong to the set $\mathcal{N}_j^- = \{v_i \in \mathcal{V} | \varepsilon_{ji} \in \mathcal{E}\}$. The number of nodes in the in-neighborhood is called in-degree and it is represented by the cardinality of the set of in-neighbors, $d_j^- = |\mathcal{N}_j^-|$. The nodes that receive information from node v_j directly are called out-neighbors of node v_j, and belong to the set $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} | \varepsilon_{lj} \in \mathcal{E}\}$. The number of nodes in the out-neighborhood is called out-degree and it is represented by the cardinality of the set of out-neighbors, $d_j^+ = |\mathcal{N}_j^+|$. Note that self-loops are included in digraph $\mathcal{G}$ and this implies that the number of in-going links of node v_j are $(d_j^- + 1)$ and similarly the number of its out-going links is $(d_j^+ + 1)$.

B. Ratio Consensus over Directed Graphs

The problem of average consensus involves a number of nodes in a network, represented by digraph (directed graph) $\mathcal{G}$, that cooperate by exchanging information to compute the network-wide average of their initial values, for a certain quantity of interest. At each time step k, each node v_j maintains a state variable $x_j[k] \in \mathbb{R}$ (initialized at $x_j[0] = V_j$, where V_j is the (arbitrary) initial value of node v_j), an auxiliary scalar variable, $y_j[k] \in \mathbb{R}^+$ (initialized at $y_j[0] = 1$), and $z_j[k] \in \mathbb{R}$ set to $z_j[k] = x_j[k]/y_j[k]$. The iterative scheme of the ratio consensus algorithm, involves each node updating its states for each iteration according to:

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}x_i[k], \tag{1a}$$

$$y_j[k+1] = p_{jj}y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} p_{ji}y_i[k], \tag{1b}$$

$$z_j[k+1] = \frac{x_j[k+1]}{y_j[k+1]}, \tag{1c}$$

where the collection of weights $P = \{p_{ji}\} \in \mathbb{R}_+^{n \times n}$ representing the interactions between nodes, forms a column-stochastic matrix. Often, each node v_j assigns the weight p_{lj} as:

$$p_{lj} = \begin{cases} \frac{1}{1+d_j^+}, & v_l \in \mathcal{N}_j^+ \cup \{v_j\}, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

which requires that the nodes have the knowledge of their out-degree. As soon as the weights are assigned by all nodes, the nonnegative weighted column-stochastic adjacency matrix P is formed, in which (possible) zero-valued entries denote the absence of communication links (edges) between the corresponding nodes in the digraph.

The auxiliary scalar variable $y[k]$ is used to asymptotically compute the right Perron eigenvector of P which is not equal to $\mathbf{1}_n$ since P is not row-stochastic [22]. Forcing the auxiliary variable $y[k]$ to be initialized at value 1 for each node, we can verify that $\lim_{k \to \infty} y_j[k] = n\,[\boldsymbol{\pi}_c]^j$ and that $\lim_{k \to \infty} x_j[k] = (\sum_{i=1}^{n} x_i[0])\,[\boldsymbol{\pi}_c]^j$, where $\boldsymbol{\pi}_c$ is the right eigenvector of P ($\boldsymbol{\pi}_c$ will be strictly positive if P is primitive column-stochastic). Hence, the limit of the ratio $x_j[k]$ over $y_j[k]$, is the average of the initial values and is given by [19], [23]:

$$\lim_{k \to \infty} z_j[k] = \lim_{k \to \infty} \frac{x_j[k]}{y_j[k]} = \frac{\left(\sum_{i=1}^{n} x_i[k]\right)[\boldsymbol{\pi}_c]^j}{n\,[\boldsymbol{\pi}_c]^j} = \frac{1}{n}\sum_{i=1}^{n} x_i[0].$$
(3)

Remark 1. The Ratio Consensus algorithm assumes that communication links within the network are perfectly reliable, and that each node v_j is aware of its out-degree.

C. Robustified Ratio Consensus over Directed Graphs

In practice, packet transmissions are often delayed due to bad channel conditions and network congestion. To overcome this issue, the authors in [18] proposed a protocol that handles time-varying delays to ensure asymptotic consensus to the exact average, despite the presence of time-varying (yet bounded) delays in the communication links.

To get intuition about the protocol proposed in [18], consider that node v_j at time step k undergoes an *a priori* unknown delay denoted by a bounded positive integer $\tau_{ji}[k] \leq \bar{\tau}_{ji} < \infty$. The maximum delay in the network is denoted by $\bar{\tau} = \max\{\bar{\tau}_{ji}\}$. Moreover, the own value of node v_j is always instantly available without delay, *i.e.*, $\tau_{jj}[k] = 0, \forall\, k$. Based on this notation, the strategy proposed in [18] involves each node updating its states for each iteration according to

$$x_j[k+1] = p_{jj}x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji}x_i[k-r]\iota_{ji}[k-r],$$

$$y_j[k+1] = p_{jj}y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji}y_i[k-r]\iota_{ji}[k-r],$$

$$z_j[k+1] = \frac{x_j[k+1]}{y_j[k+1]}.$$
(4)

The indicator function, $\iota_{ji}[k-r]$, indicates whether the bounded delay $\tau_{ji}[k-r] \leq \bar{\tau}_{ji}$ on link ε_{ji} at iteration $k-r$, equals r (*i.e.*, the transmission on link ε_{ji} at $k-r$ arrives at node v_j at iteration k) and is defined as

$$\iota_{ji}[k-r] = \begin{cases} 1, & \text{if } \tau_{ji}[k-r] = r, \\ 0, & \text{otherwise.} \end{cases}$$
(5)

Here it is important to note that, a transmission on link ε_{ji} undergoes an *a priori* unknown delay, for which node v_j is unaware of, but instead, it processes (delayed) packets as soon as they arrive successfully. Clearly, in the absence of delays, this strategy reduces to the Ratio Consensus algorithm in [19].

Remark 2. The Robustified Ratio Consensus algorithm handles (possibly) delayed information exchange between nodes, neglecting any packet drops due to erroneous packets, while assuming that each node v_j is aware of its out-degree.

D. Error Correction Protocol - Automatic Repeat reQuest (ARQ)

Several message transmission protocols such as the Transmission Control Protocol (TCP), the High-Level Data Link protocol, and others, utilize Automatic Repeat reQuest (ARQ), an error control protocol for data transmission used to maintain reliable packet transmissions over unreliable communication [24], [25, §6], [26, §5]. ARQ uses error-detection codes, acknowledgment (ACK) or negative acknowledgment (NACK) messages, and timeouts to maintain the reliability of data transmissions. An acknowledgment is a feedback signal sent by the data receiver to notify the data transmitter whether a packet has been successfully received or not. A graphical representation of the ARQ error control feedback mechanism is shown in Fig. 1.
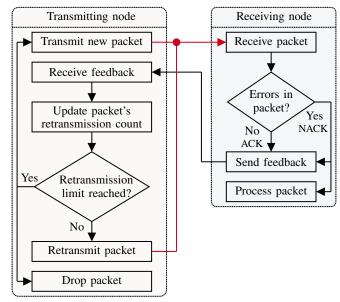

Fig. 1: ARQ error control feedback mechanism.

Here it is important to note that the ARQ feedback is sent over a narrowband error-free (with negligible probability of error in the reception of this small packet, which usually comprises 1 bit) feedback channel that is not used for data transmission, hence the network topology could still be assumed directed. The ARQ feedback procedure is repeated until a packet is successfully received (without errors) or dropped due to excess of a predefined limit of retransmissions. In other words, the receiver has up to a predefined number of retransmission trials to receive the data packet correctly, otherwise the packet is dropped. A retransmission requested by data receivers introduces a delay of one time slot, since the receiver will get the intended packet as soon as the transmission is successfull.

III. ARQ-BASED RATIO CONSENSUS

In this section we propose a distributed ratio consensus method, herein called the ARQ-based Ratio Consensus algorithm, to ensure asymptotic convergence to the network-wide average value, based on the ARQ protocol for reliable information exchange (described in §II-D). Under the consensus framework, each node acts as a data transmitter (receiver) when transmitting (receiving) data to (from) its out-neighbors (in-neighbors) utilizing the ARQ protocol described. In particular, at each time slot, all nodes send their values to their adjacent neighbors, and they receive an ACK as soon as the data packets containing the nodes' values are correctly decoded. In contrast, a NACK is fed back to the transmitting node, if the receiving node failed to decode the data packet. Unsuccessful decoding induces one time slot delay since the same packet needs to be retransmitted. Excess of the ARQ retransmission limit could be modeled as a packet loss if the packet arrives in error during its last retransmission trial.

Each node assigns its self-weight and the weights of its out-neighbors using (2). Here, it is important to mention that each node can acquire its out-degree by summing the number of its incoming ARQ feedback signals (ACK/NACK) which equals the number of its out-neighbors. This can be established at the initialization, by having the nodes broadcast a dummy packet in order to identify their out-neighbors. As aforementioned, although one could argue that the use of ARQ feedback contradicts the assumed directed network topology, this is not the case because it does not imply bidirectional (undirected) communication, since the ARQ feedback signal is transmitted over a narrowband feedback channel, which cannot support data transmission due to the high data rate required and, hence, larger bandwidth and higher-order modulation. As soon as the weights are assigned, each node updates its own consensus variables using received information from its in-neighbors.

A. Handling Erroneous Transmissions (delays) and Packet Drops

Recall that a NACK sent by the receiving node v_j, implies that the transmitting node v_i should retransmit the same packet in the next time slot, unless the retransmission limit, $\bar{\tau}_{ji}$, has been exceeded. For simplicity of exposition, it is assumed that the communication link from v_i to v_j is stationary. Hence, the probability that a transmitted packet contains an error (detected by the receiver) is constant and it is denoted by q_{ji}. For a packet that was initially transmitted at time k on link ε_{ji}, the delay of information is denoted by $\tau_{ji}[k]$, as shown in Fig 2. During the last trial of a packet retransmission (when the maximum retransmission limit has been reached, i.e., $\tau_{ji}[k] = \bar{\tau}_{ji}$), the transmitting node receives the last ARQ feedback for this particular packet from the receiving node. If the ARQ feedback is NACK, then the packet is dropped. Otherwise, the packet is received successfully at the receiving node. Since all nodes in the network utilize internally an ARQ protocol, it is natural to imply that the delays are bounded by

the predefined maximum allowable number of retransmissions, for all $k \geq 0$, i.e., $0 \leq \tau_{ji}[k] \leq \bar{\tau}_{ji} \leq \bar{\tau}$.

Hence, the probabilities for different number of consecutive NACKs for a packet initially transmitted at time instant k over link ε_{ji} are defined as follows:

$$\mathbb{P}(\tau_{ji}[k] = 0) = 1 - q_{ji}, \qquad \text{(no error)}$$
$$\mathbb{P}(\tau_{ji}[k] = 1) = q_{ji}(1 - q_{ji}), \qquad \text{(1 NACK)}$$
$$\vdots$$
$$\mathbb{P}(\tau_{ji}[k] = \bar{\tau}_{ji}) = (q_{ji})^{\bar{\tau}_{ji}}(1 - q_{ji}), \qquad (\bar{\tau}_{ji} \text{ NACKs})$$
$$\mathbb{P}(\tau_{ji}[k] = \bar{\tau}_{ji} + 1) = (q_{ji})^{\bar{\tau}_{ji}+1}, \qquad \text{(packet drop)}$$

where $\mathbb{P}(\cdot)$ refers to the probability that the event in the argument "$\cdot$" will occur. A packet initially transmitted during time slot k may be dropped if the maximum allowable retransmission number ($\bar{\tau}_{ji}$) is exceeded. A successful reception of a packet sent over link ε_{ji}, is denoted by the indicator variable $\mathbb{I}_{ji}[k + \bar{\tau}_{ji}] = 1$, while a failed reception implies that the packet is dropped, which is denoted by $\mathbb{I}_{ji}[k + \bar{\tau}_{ji}] = 0$.
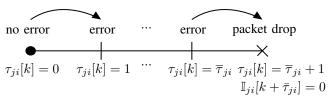
Fig. 2: Erroneous packet propagation.

B. Feedback Schemes for Ratio Consensus

In this work, we consider two different schemes for utilizing the ACK/NACK feedback for the ARQ-based Ratio Consensus algorithm. The mechanisms for these two schemes arise naturally since multiple packets (new or delayed) may be transmitted over the same link ε_{ji} within a time slot. Both schemes involve receivers that feed back ACK/NACK signals for each transmission that arrives. However, the former involves multiple individual transmissions and their corresponding ARQ feedback signals, while the latter involves single transmissions that contain the aggregated information and their corresponding ARQ feedback signal (that signifies whether the aggregated packet has been successfully decoded).

1) Multiple Transmissions Multiple Feedback (MTMF): In this scheme, each new packet is transmitted individually along with possibly retransmitted (delayed) packets. To get some intuition, consider the simple example with two nodes v_1 and v_2, shown in Fig. 3. During the first time slot (starting at $k = 1$), a new packet (containing a weighted version of the state according to (2)) is sent over link ε_{ji}. Node v_2 detects an error in the received packet, and sends back a NACK. By the second time slot, node v_1 has the NACK feedback sent from v_2, and retransmits the previous packet $w_{21}[1]$, along with the new packet $w_{21}[2]$. Within this time slot, node v_2 successfully receives the delayed packet $w_{21}[1]$ for which it sends an ACK, but it detects an error in the new packet $w_{21}[2]$ for which it

sends a NACK. The same procedure is followed for the latter transmissions. Note that, this scheme requires transmissions and ARQ feedback signals to be identified (matched to packets).
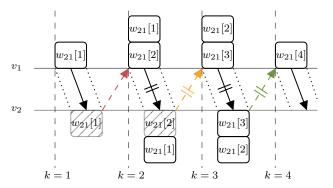

Fig. 3: An individual ARQ feedback (ACK/NACK) is sent for each packet, within each time slot. Red single arrows denote a NACK; orange bus arrows denote multiple ARQ feedback of ACKs and NACKs; green bus arrows denote multiple ACKs; shaded rectangles represent erroneous packets; transmissions involving multiple packets, ACKs or NACKs are indicated by arrows with multiple vertical lines.

2) Single Transmissions Single Feedback (STSF): In this scheme, transmitting nodes aggregate the information (from new and delayed packets) in a single packet to be transmitted within a single time slot. To get some intuition, consider the simple example with two nodes v_1 and v_2, shown in Fig. 4. Node v_1 transmits a packet $w_{21}[1]$, to node v_2 within the first time slot. Within the same time slot, node v_2 detects an error and requests a retransmission by sending back a NACK. By the next time slot, node v_1 has the NACK and calculates the summation of the weighted values of the delayed packet and the new packet, in order to transmit a single packet $w_{21}[2]$. Under this scheme, node v_2 sends only a single ARQ feedback to node v_1.
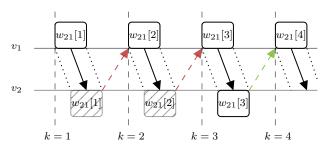

Fig. 4: Receiving node sends an ARQ-feedback (ACK or NACK) for the aggregated information of all received packets, at each time step k. Red arrows denote NACKs; green arrows denote ACKs; shaded rectangles represent erroneous packets.

C. ARQ-based Ratio Consensus Algorithm

Herein, we describe the ARQ-based Ratio Consensus algorithm for the feedback schemes considered in §III-B. In the following steps, we describe MTMF in detail, but one can adjust the algorithm for the STSF scheme. Specifically,

each node v_j executes the steps in Algorithm 1, which can be summarized as follows:

- First, the initial state $x_j[0] = V_j$ is determined by the information (often local sensor reading) on which the network of nodes must reach average consensus. As soon as the input has been processed, the auxiliary variable $y_j[k]$ is initialized at $y_j[0] = 1$. To keep track of possibly lost information due to packet drops, each node maintains the accumulated x_j and y_j masses that node v_j wants to transmit to each of its out-neighbors, which are denoted by $\eta_j[k]$ and $\sigma_j[k]$, and initialized at $\eta_j[0] = 0$ and $\sigma_j[0] = 0$, respectively. Similarly, each node maintains the variables $\psi_{ji}[k]$ and $\chi_{ji}[k]$ that keep track of the accumulated x_j and y_j masses received from each node $v_i \in \mathcal{N}_j^-$; these variables are initialized at $\psi_{ji}[0] = 0$ and $\chi_{ji}[0] = 0$.
- Second, the out-degree of node v_j is acquired by broadcasting a dummy package and summing the number of received ARQ feedback signals from its out-neighbors. Note that, one could improve the out-degree acquisition accuracy by executing this process several times.
- Node v_j updates its states $x_j[k + 1]$ and $y_j[k + 1]$ using the equations in lines 11-12, whenever the actual packet retransmissions over link ε_{ji} have not reached the maximum retransmission limit ($\tau_{ji}[k] < \bar{\tau}_{ji}$) imposed by the ARQ protocol.
- In case the retransmission limit of a packet has been reached, then the running sum mechanism, in lines 14-28, is activated. At this stage, the running sum variables $\sigma_j[k+1]$ and $\eta_j[k+1]$ are updated according to lines 14-15. The accumulated information in the running sum variables of each node v_j is transmitted to its out-neighbors. If the packet arrives at the receiving node v_i without errors, then variables $\chi_{ji}[k + 1]$ and $\psi_{ji}[k + 1]$ are updated with the accumulated masses $\sigma_j[k+1]$ and $\eta_j[k+1]$, respectively. In contrast, if the packet is erroneous, then the auxiliary state variables $\chi_{ji}[k+1]$ and $\psi_{ji}[k+1]$ remain unchanged. With the next (possibly delayed) transmission, the information held in the auxiliary variables is released to its destined node accordingly.

D. Simple Example

To get intuition about the ARQ-based Ratio Consensus algorithm, we provide the following example with the aid of a graph representation. Here, it is important to note that, the augmented graph representation is only used to exploit the modeling and analysis of the algorithm without affecting its actual implementation. More details regarding the augmentation of the digraph are presented in the subsequent section. Consider the digraph of two nodes, shown in Fig. 5, where nodes v_1 and v_2 utilize an ARQ protocol with maximum retransmission limit $\bar{\tau}_{12} = 1$, and $\bar{\tau}_{21} = 2$, respectively. In this graph representation, we augment the original digraph by adding extra virtual nodes and extra virtual buffer nodes to capture the effect of delays due to retransmissions, and possible packet drops. Initially, transmitted information will reach its recipient through virtual nodes (shown in orange squares) that model the induced delays

Algorithm 1 ARQ-based Ratio Consensus

1: **Input:** $x_j[0] = V_j$
2: **Initialization:** $\sigma_j[0] = 0$, $y_j[0] = 1$, $\eta_j[0] = 0$,
3: $\quad\quad\quad\quad \chi_{ji}[0] = 0$, $\psi_{ji}[0] = 0$, $\forall i \in \mathcal{N}_j^-$
4: **Out-degree acquisition:** $d_j^+ = |\mathcal{N}_j^+|$
5: **for** $k \geq 0$: **do**
6: $\quad$ **Transmit to all** $v_i \in \mathcal{N}_j^+$ **:**
7: $\quad\quad x_j[s]$ and $y_j[s]$, $\forall s = k - \bar{\tau}_{ji}, \ldots, k$, for $k \geq \bar{\tau}_{ji}$
8: $\quad$ **Receive from all** $v_i \in \mathcal{N}_j^-$ **:**
9: $\quad\quad x_i[h]$ and $y_i[h]$, $\forall h = k - \tau_{ji}[h]$, for $0 \leq h \leq k$
10: $\quad$ **if** $\tau_{ji}[k] < \bar{\tau}_{ji}$
11: $\quad\quad x_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \sum_{r=0}^{\bar{\tau}_{ji}} \iota_{ji}[k-r] p_{ji} x_i[k-r]$
12: $\quad\quad y_j[k+1] = \sum_{v_i \in \mathcal{N}_j^- \cup \{v_j\}} \sum_{r=0}^{\bar{\tau}_{ji}} \iota_{ji}[k-r] p_{ji} y_j[k-r]$
13: $\quad$ **else**
14: $\quad\quad \sigma_j[k+1] = \sigma_j[k] + p_{lj} x_j[k]$
15: $\quad\quad \eta_j[k+1] = \eta_j[k] + p_{lj} y_j[k]$
16: $\quad\quad$ **Transmit to all** $v_l \in \mathcal{N}_j^+$ **:** $\sigma_j[k+1]$ and $\eta_j[k+1]$
17: $\quad\quad$ **Receive from all** $v_i \in \mathcal{N}_j^-$ **:** $\sigma_i[k+1]$ and $\eta_i[k+1]$
18: $\quad\quad$ **for** $v_i \in \mathcal{N}_j^-$ **: do**
19: $\quad\quad\quad$ **if** $\mathbb{I}_{ji}[k + \bar{\tau}_{ji}] = 1$
20: $\quad\quad\quad\quad \chi_{ji}[k+1] = \sigma_i[k+1]$
21: $\quad\quad\quad\quad \psi_{ji}[k+1] = \eta_i[k+1]$
22: $\quad\quad\quad$ **else**
23: $\quad\quad\quad\quad \chi_{ji}[k+1] = \chi_{ji}[k]$
24: $\quad\quad\quad\quad \psi_{ji}[k+1] = \psi_{ji}[k]$
25: $\quad\quad\quad$ **end**
26: $\quad\quad$ **end**
27: $\quad\quad x_j[k+1] = x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} (\chi_{ji}[k+1] - \chi_{ji}[k])$
28: $\quad\quad y_j[k+1] = y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} (\psi_{ji}[k+1] - \psi_{ji}[k])$
29: $\quad$ **end**
30: $\quad$ **Output:** $z_j[k+1] = \frac{x_j[k+1]}{y_j[k+1]}$
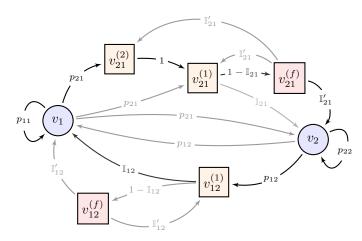31: **end for**

Fig. 5: Two node digraph denoting ARQ-based transmissions. A packet transmitted from node v_1 to v_2 is delayed by two time slots, and is dropped in the subsequent time slot. A packet transmitted from node v_2 to v_1 is delayed by one time slot.

due to retransmissions. In this example, we set the probability that a node receives an erroneous packet to be $q_{ji} = 0.4$. Hence, packets sent from v_1 to v_2 are dropped with probability $(q_{21})^{\bar{\tau}_{21}+1} = 0.4^3 = 0.064$, while packets sent from node v_2 to v_1, are dropped with probability $(q_{12})^{\bar{\tau}_{12}+1} = 0.4^2 = 0.16$. As soon as the packets are dropped, then the corresponding edges are activated to propagate the information held in the running sum variables to the virtual buffer nodes as described in Algorithm 1. Consider the case where two consecutive NACKs have been sent back from node v_2 to node v_1. Then, we can model the delays on the initial information by introducing two virtual nodes, $v_{21}^{(1)}$ and $v_{21}^{(2)}$. In the last retransmission trial, the running sum mechanism will be activated to handle possible packet drops. During this time slot, the packet drop handling mechanism updates the running sum variables based on the indicator variable $\mathbb{I}_{21}[k + \bar{\tau}_{21}] = \mathbb{I}_{21}[k + 2]$. If $\mathbb{I}_{21}[k + 2] = 1$, then the initially transmitted information will be received by node v_2 successfully. Otherwise, if node v_2 detects another error in the initially transmitted packet, $\mathbb{I}_{21}[k + 2] = 0$, then the running sum information will propagate to the virtual

buffer $v_{21}^{(f)}$ (shown in red square). The information held in $v_{21}^{(f)}$, is released to the actual node v_2 along with the (possibly delayed) packet that is intended to be transmitted in the next time slot. The information release is modeled by activating one of the virtual buffer node's $v_{21}^{(f)}$ outgoing links, $\mathbb{I}_{21}'$. Packet transmission from node v_2 to node v_1, is delayed by one time slot which equals the maximum retransmission limit. However, the packet arrives successfully at node v_1 since no further error was detected in the last retransmission trial.

To emphasize the difference between the ARQ-based Ratio Consensus algorithm and the Robustified Ratio Consensus algorithm (described in §II-C), we introduce the digraph shown in Fig. 6. This digraph models the same communication topology between nodes, as in Fig. 5. However, it is easy to see that the digraph that models the Robustified Ratio Consensus algorithm transmissions, cannot handle packet drops, and hence, the digraph only consists of the actual nodes and the virtual nodes that model the transmission delays. Consequently, nodes utilizing the Robustified Ratio Consensus algorithm, cannot converge to the average of the network's wide initial values if packet drops occur during information exchange.

Remark 3. The ARQ-based Ratio Consensus algorithm handles unreliable packet transmissions, by utilizing 1-bit feedback signals that can be also used to determine the nodes' out-degrees, and local upper-bounds on the delays imposed by the ARQ retransmission limit.

IV. AUGMENTED DIGRAPH REPRESENTATION

In this section, we analyse the convergence of Algorithm 1, by first introducing the random weighted adjacency matrix that corresponds to the delayed and possibly packet dropping communication topology. To simplify the analysis, we consider identical ARQ protocols for each node. This implies that the maximum retransmission limit is the same for all nodes, and therefore the maximum delay on each link $\bar{\tau}_{ji}$, is
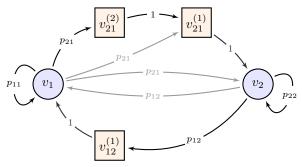
Fig. 6: Two node digraph denoting transmissions using the Robustified Ratio Consensus algorithm. A packet transmitted from node v_1 to v_2 is delayed by two time slots, while from v_2 to v_1 is delayed by one time slot.

bounded by the maximum delay in the network, $\bar{\tau}$. Hence, for each link (edge) $\varepsilon_{ji} \in \mathcal{E}$, we add $\bar{\tau}$ extra virtual nodes, $v_{ji}^{(1)}, v_{ji}^{(2)}, \ldots, v_{ji}^{(\bar{\tau})}$, where the virtual node $v_{ji}^{(r)}$ holds the information that is destined to arrive at actual (original) node v_j after r time steps. In case of a packet drop, $\mathbb{I}_{ji}[k] = 0$, the delayed information (held at the corresponding virtual node of maximum delay) will propagate to the virtual buffer node of the original node v_j instead of being received at the original node. During the next time slot, the information will be released either to the original node if $\tau_{ji}[k+1] = 0$, or to the corresponding virtual node if $\tau_{ji}[k+1] > 0$. Note that, the links that correspond to self-loops do not induce delays or packet losses, and hence no virtual nodes for self-loops are considered in the augmented graph representation. Finally, the augmented digraph, $\mathcal{G}^a = (\mathcal{V}^a, \mathcal{E}^a)$, that models the delayed network with packet drops consists of at most $\tilde{n} = |\mathcal{E}|(\bar{\tau}+1) + n \leq n(n-1)(\bar{\tau}+1) + n$ nodes in $\mathcal{V}^a$, where n nodes are original nodes, $|\mathcal{E}|\bar{\tau}$ nodes are virtual nodes due to delays, and $|\mathcal{E}|$ are virtual nodes due to packet losses.

Hence, the consensus iterations performed by Algorithm 1, can be rewritten in matrix form as

$$\tilde{x}[k+1] = \Xi[k]\tilde{x}[k], \tag{6a}$$
$$\tilde{y}[k+1] = \Xi[k]\tilde{y}[k], \tag{6b}$$

where $\tilde{x}[k]$ and $\tilde{y}[k]$ are vectors containing the variables of both the actual and virtual nodes at time step k. Note that $\tilde{x}[k]$, and $\tilde{y}[k]$ have $\tilde{n} \geq n$ elements, where the first n elements correspond to the actual nodes, while the remaining elements correspond to the virtual nodes of the augmented digraph. Matrix $\Xi[k] \in \mathbb{R}_+^{\tilde{n} \times \tilde{n}}$ is a nonnegative random matrix associated with the augmented digraph:

$$\Xi[k] \triangleq \begin{pmatrix} P^{(0)}[k] & D^{\text{succ}}[k] & 0 & \cdots & 0 & D^{(0)}[k+1] \\ P^{(1)}[k] & D^{\text{exc}}[k] & I & \cdots & 0 & D^{(1)}[k+1] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P^{(\bar{\tau}-1)}[k] & 0 & 0 & \cdots & I & D^{(\bar{\tau}-1)}[k+1] \\ P^{(\bar{\tau})}[k] & 0 & 0 & \cdots & 0 & D^{(\bar{\tau})}[k+1] \\ 0 & D^{\text{pd}}[k] & 0 & \cdots & 0 & D^{\text{sl}}[k+1] \end{pmatrix} \tag{7}$$

where each element of $P^{(r)}[k]$ is determined by:

$$P^{(r)}[k](j,i) = \begin{cases} P(j,i), & \text{if } \tau_{ji}[k] = r, \ (j,i) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

In other words, if the index r that corresponds to the block matrix $P^{(r)}$ is equal to the packet retransmission number $\tau_{ji}[k]$, then the link ε_{ji} will be weighted by the actual (original) weight p_{ji}, otherwise its weight will be zero.

Clearly the structure of matrix $\Xi[k]$ depends on the realized number of retransmissions and packet drops. Block matrix $D[k]^{\text{succ}} \in \mathbb{R}_+^{n \times m}$ activates the virtual links that propagate the delayed information to actual nodes. This occurs whenever a packet is successfully received by its destined (actual) node without error, during its last retransmission trial, $\tau_{ji}[k] = \bar{\tau}_{ji}$. Block matrix $D^{\text{exc}}[k] \in \mathbb{R}_+^{m \times m}$, is a diagonal matrix with its diagonal elements having 1 if $\tau_{ji}[k] < \bar{\tau}_{ji}$, and 0 otherwise, for $j, i \in \{1, \ldots, n\}$ and $j \neq i$. This matrix activates the running sum mechanism whenever the retransmission limit of a packet has been reached. Block matrix $D^{\text{pd}}[k] \in \mathbb{R}_+^{m \times m}$ is a diagonal matrix where the diagonal elements take values $1 - \mathbb{I}_{ji}[k + \bar{\tau}_{ji}]$. This matrix propagates the running sum of each node to its virtual buffer node whenever a packet drop occurs. Block matrices $D^{(r)}[k+1]$ are of appropriate dimensions and handle the release of information from virtual buffer nodes to the corresponding actual or virtual node. The elements of these matrices are placed in the corresponding row of matrix $\Xi[k]$ similarly to (8). In other words, if a packet is dropped, the information will be held in the virtual buffer, until the next successful (possibly delayed) transmission. Block matrix $D^{\text{sl}}[k+1]$ is of appropriate dimensions, and models the self-loops of the virtual buffer nodes, whenever a packet arrives successfully to its intended actual node, without being dropped. Based on the aforementioned properties, it is clear that matrix $\Xi[k]$ maintains column-stochasticity, although the links that establish the transmissions between nodes might be unreliable.

Prior to establishing the convergence of each node to the network-wide average of the initial values, we first establish some properties of matrices $\Xi[k]$, similar to the ones established in [20] for packet dropping links. Matrix $\Xi[k]$ denotes a particular instance from the set of all possible instances of realized delays induced from packet retransmissions, and packet drops. All these possible instances are in the set $\mathcal{X}$, which consists of a finite number of matrices of identical dimensions $\tilde{n} \times \tilde{n}$, where each matrix in $\mathcal{X}$ corresponds to a distinct instantiation of the packet drop indicator variable $\mathbb{I}_{ji}[k]$ and the delay on the links $\tau_{ji}[k]$. Thus, each positive element of any matrix $\Xi[k]$ is lower bounded by a positive constant $c := \min_{j \in \mathcal{V}} 1/(1 + d_j^+)$, since the (i, j) element of any $\Xi[k] \in \mathcal{X}$ can take values of either 0, 1, or $1/(1 + d_j^+)$. Let λ be a finite positive integer. Then, a sequence of λ matrices in $\mathcal{X}$, possibly with repetitions and in a certain order, materializes with a strictly positive probability. The product of a particular choice of these λ matrices (in the given order) forms a column stochastic matrix, where the entries that correspond to the

actual network nodes, contain strictly positive values. This can be written as:

$$\mathbb{P}(\Xi[k+\lambda]=\Xi_\lambda, \Xi[k+\lambda-1]=\Xi_{\lambda-1},\dots,\Xi[k+1]=\Xi_1)$$
$$\geq p_{\min} > 0, \qquad (9)$$

where $p_{\min}$ is the strictly positive probability with which this sequence of λ matrices materializes. Note that $L = \Xi_\lambda \Xi_{\lambda-1},\dots,\Xi_1$ have been chosen so that the top n rows satisfy: $L(j,i) > 0$ for all $j,i \in \mathcal{V}$, (*e.g.*, $c = n$ and $\tau_{ji}[k] = 0$ for all links and all k). These properties can be exploited for the proof of convergence of the ARQ-based Ratio Consensus algorithm, stated in Theorem 1 below.

Recall that each node in the network aims at obtaining consensus to the network-wide average of the initial values:

$$z^* = \frac{\sum_{l\in\mathcal{V}^a}\tilde{x}_l[0]}{\sum_{l\in\mathcal{V}^a}\tilde{y}_l[0]} = \frac{\sum_{l\in\mathcal{V}}x_l[0]}{\sum_{l\in\mathcal{V}}y_l[0]} = \frac{1}{n}\sum_{l\in\mathcal{V}}x_l[0], \qquad (10)$$

by having network actual nodes calculate the ratio $z_j[k] = \tilde{x}_j[k]/\tilde{y}_j[k]$, whenever $\tilde{y}_j[k] \geq c^\lambda$. To establish the convergence of the proposed algorithm to the exact average of the network-wide initial values, we need to show that the augmented auxiliary variable $\tilde{y}_j[k] \geq c^\lambda$ occurs infinitely often, at actual nodes, and hence as k goes to infinity, the sequence of ratio computations $z_j[k]$ converges to the value in (10) almost surely.

Theorem 1. *Consider a strongly connected digraph $\mathcal{G}(\mathcal{V},\mathcal{E})$, where each node $v_j \in \mathcal{V}$ has some initial value $x_j[0]$, and $y_j[0] = 1$. Let $z_j[k]$ (for all $v_j \in \mathcal{V}$ and $k = 0,1,2,\dots$) be the output of the iterations in Algorithm 1. Then, the solution to the average consensus can be obtained by each node, by computing:*

$$\lim_{k\to\infty} z_j[k] = \frac{x_j[k]}{y_j[k]} = \frac{\sum_{v_i\in\mathcal{V}}x_i[0]}{n}, \ \forall v_j \in \mathcal{V}. \qquad (11)$$

Proof. See Appendix B. ∎

V. NUMERICAL EVALUATION

Consider the directed network shown in Fig. 7 consisting of five nodes ($n = 5$), with each node v_j choosing the weights on its out-going links (including its self-loop link) based on (2).

This digraph corresponds to the following column-stochastic weighted adjacency matrix:

$$P = \begin{pmatrix} 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/2 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/3 \\ 0 & 1/3 & 1/2 & 0 & 1/3 \end{pmatrix}. \qquad (12)$$

The variables of each node x_j, y_j are updated within each time slot using Algorithm 1 with initial values $\mathbf{x}[0] = (x_1[0]\dots x_n[0])^\top = (4\ 5\ 6\ 3\ 2)^\top$, and $\mathbf{y}[0] = (y_1[0]\dots y_n[0])^\top = (1\ 1\ 1\ 1\ 1)^\top$, respectively. In the examples that follow we consider that a transmitted packet may not arrive at the receiving node with probability q_{ji}.
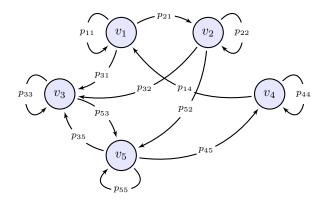

Fig. 7: Five node strongly-connected digraph.

Delayed packets negatively affect the performance of consensus algorithms, executed at each node, in terms of delaying convergence to the average consensus value. These effects can be quantified by capturing the absolute consensus error for different packet error probabilities q_{ji}. The absolute consensus error $|\hat{z} - z^*|$ with respect to the ARQ retransmission limit $\bar{\tau}$ for different packet error probabilities q_{ji}, is shown in Fig. 8. In this example, the absolute consensus error $|\hat{z} - z^*|$, where $\hat{z}$ is the average ratio among all actual nodes in the network over a total of 20 simulations of 200 consensus iterations using the ARQ-based Ratio Consensus algorithm, is captured for three different packet error probabilities, *i.e.*, $q_{ji} = 0.2$, $q_{ji} = 0.6$, and $q_{ji} = 0.8$, and z^* is the actual average consensus value of the network defined in (10). For each packet error probability, we choose a different retransmission limit for the ARQ protocol and we record the absolute consensus error. For low packet error probability, the absolute consensus error remains low, while for higher probability of packet error q_{ji}, the absolute consensus error grows at higher values.
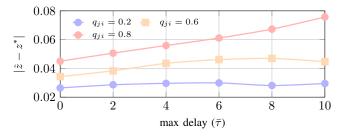

Fig. 8: Absolute consensus error, $|\hat{z} - z^*|$, for different upper bounds $\bar{\tau}$ on delays, and probabilities of erroneous packet reception.

To get intuition regarding the convergence of the proposed algorithm, we consider the same ARQ mechanism for each node, with maximum allowable retransmission number set to $\bar{\tau}_{ji} = \bar{\tau} = 2$, which corresponds to the upper bound of the delays on the links. Fig. 9 shows the evolution of the ratio at node v_1 during the first time steps of the execution of the ARQ-based Ratio Consensus algorithm with the STSF feedback scheme.
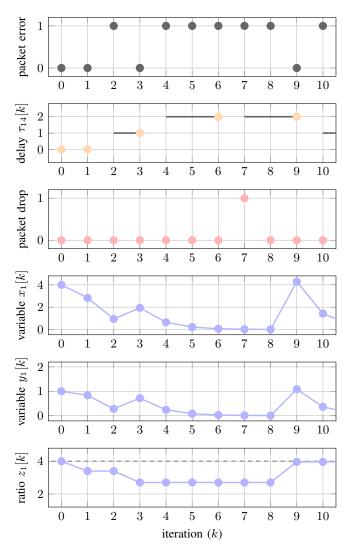
Fig. 9: Packet transmissions from node v_4 to node v_1 using the ARQ-based Ratio Consensus STSF algorithm and $q_{ji} = 0.6$, $\bar{\tau} = 2$.

Node v_1 receives information from node v_4 (and itself), according to the network topology modeled by the weighted adjacency matrix defined in (12). The packet error probability is set at $q_{ji} = 0.6$. As shown in Fig. 9, the ratio $z_1[k]$ is updated as soon as information packets from node v_4 arrive successfully at node v_1. It is worth mentioning that, at iteration $k = 6$, the maximum retransmission limit is reached $\tau_{14}[6] = 2$ and due to another packet error at the last trial, the packet is dropped.

Next, we evaluate the ARQ-based Ratio Consensus algorithm with STSF and MTMF feedback scheme (ARQ-RC-STSF and ARQ-RC-MTMF, respectively) for three different packet error probabilities: (a) $q_{ji} = 0.2$, (b) $q_{ji} = 0.6$, and (c) $q_{ji} = 0.8$. We assess these cases for two different configurations on the maximum retransmission limit of the ARQ-based Ratio Consensus algorithm, $\bar{\tau} = 2$, and $\bar{\tau} = 5$, and then we compare both with the Ratio Consensus via the Running Sums (RC-RS) algorithm in [20]. Fig. 10 depicts the evolution of variables $x_j[k]$ and $y_j[k]$ of each (non-virtual) node in the

network with packet error probability $q_{ji} = 0.2$, and maximum retransmission limit $\bar{\tau} = 2$. Due to the time-varying delays on the links and possible packet drops, we can see that the variables do not converge to a certain value. However, the ratio of each node converges to the exact average of the nodes' initial values, despite the time-varying delays due to retransmissions, and possible packet drops. This is seen in Fig. 12, Fig. 14 for the ARQ-RC-STSF and Fig. 13, Fig. 15 for the ARQ-RC-MTMF algorithm. Fig. 11 depicts the ratio of each node in the network for different packet error probabilities (i.e., $q_{ji} = 0.2$, $q_{ji} = 0.6$, and $q_{ji} = 0.8$) using the RC-RS algorithm that does not utilize any feedback mechanism. For lower packet error probabilities (i.e., $q_{ji} = 0.2$), the convergence of the ratio $z_i[k]$ is fast for all the algorithms, although the maximum retransmission limit is set to $\bar{\tau} = 2$ and $\bar{\tau} = 5$. This occurs because most of the packets will eventually arrive at their destined nodes successfully, with higher probability.
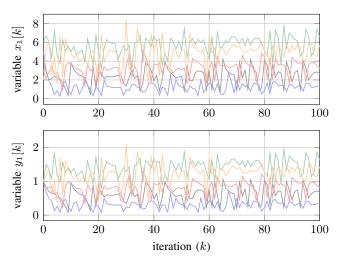

Fig. 10: Variables $x_j[k]$ (upper), and $y_j[k]$ (lower) of each node v_j, with maximum delay $\bar{\tau} = 2$, and probability of failure $q_{ji} = 0.2$.

In contrast, when the communication channel is highly unreliable (e.g., when $q_{ji} = 0.6$, and $q_{ji} = 0.8$), we can see that the convergence to the average consensus value is slower compared to lower packet error probabilities. It is important to mention that one can tune the ARQ-based Ratio Consensus algorithms based on the packet error probability, by adjusting the maximum retransmission limit such that a faster convergence is achieved. Nevertheless, for all the aforementioned scenarios, the ARQ-based Ratio Consensus algorithm guarantees that each node will converge to the average consensus value despite the maximum retransmission limit, and the packet error probability that captures the quality of the communication channels.

A. Discussion on the Convergence Rate

To get intuition regarding the convergence rate of the proposed algorithm, we consider the network in Fig. 7 with $\mathbf{x}[0] = (4\ 5\ 6\ 3\ 2)^\top$ and $\mathbf{y}[0] = (1\ 1\ 1\ 1\ 1)^\top$. For each packet transmitted over this network, we set the packet error probability at $q_{ji} = 0.6$, and we run 500 simulations of 300 consensus iterations to obtain the mean value $\frac{\|\mathbf{z}[k] - \mathbb{1}_n z^*\|}{\|\mathbb{1}_n z^*\|}$

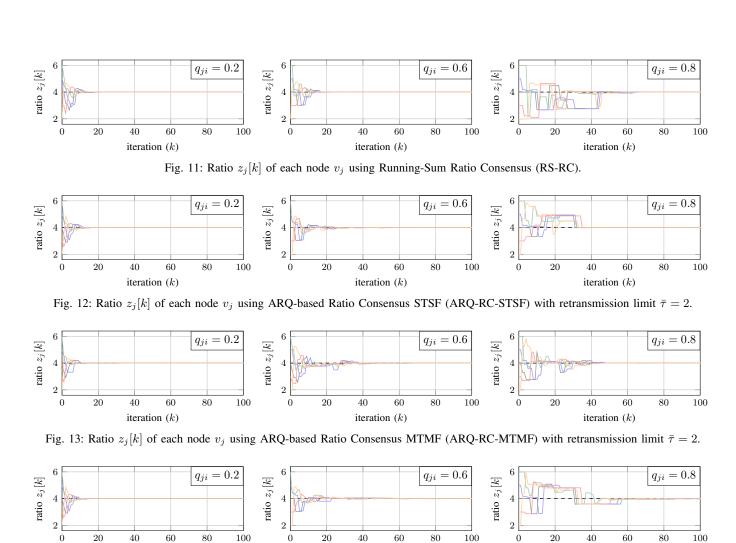Fig. 11: Ratio $z_j[k]$ of each node v_j using Running-Sum Ratio Consensus (RS-RC).

Fig. 12: Ratio $z_j[k]$ of each node v_j using ARQ-based Ratio Consensus STSF (ARQ-RC-STSF) with retransmission limit $\bar{\tau} = 2$.

Fig. 13: Ratio $z_j[k]$ of each node v_j using ARQ-based Ratio Consensus MTMF (ARQ-RC-MTMF) with retransmission limit $\bar{\tau} = 2$.

Fig. 14: Ratio $z_j[k]$ of each node v_j using ARQ-based Ratio Consensus STSF (ARQ-RC-STSF) with retransmission limit $\bar{\tau} = 5$.
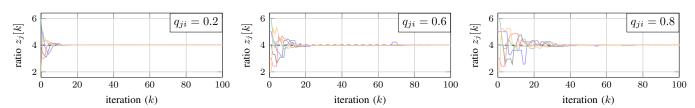

Fig. 15: Ratio $z_j[k]$ of each node v_j using ARQ-based Ratio Consensus MTMF (ARQ-RC-MTMF) with retransmission limit $\bar{\tau} = 5$.

at each consensus iteration k, over all simulations. Fig. 16 depicts the convergence of each algorithm to the average consensus value, in terms of the mean error. Clearly the RC-RS algorithm converges faster to the average consensus value relatively to the ARQ-RC-STSF and ARQ-RC-MTMF. However, as it can be seen from Fig. 16, the convergence of all algorithms follows a linear trend pattern. Moreover, setting the maximum retransmission limit at $\bar{\tau} = 5$ for ARQ-RC-STSF and ARQ-RC-MTMF (dashed blue and green lines respectively), results in a slower convergence relatively to $\bar{\tau} = 2$, while the ARQ-RC-MTMF outperforms the ARQ-RC-STSF in terms of convergence speed.

B. Discussion on the Accumulation of Running Sums

Implementing ratio consensus algorithms that handle packet drops would require nodes to maintain extra variables to store the information that is lost due to packet drops. This information is accumulated at each iteration in the running sum variables of each node. As discussed in [20], the values of running sums in the Ratio Consensus via Exchange of Running Sums algorithm, grow linearly in an unbounded manner with the number of iterations, as shown in Fig. 18. Although this would not be an issue for nodes' memory capabilities (assuming that there exist a distributed termination mechanism), it would require nodes to transmit packets of increasing length,
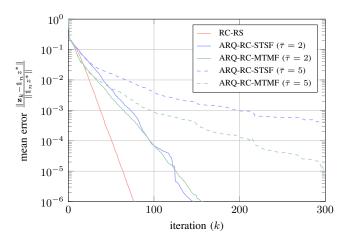
Fig. 16: Convergence rate comparison between RC-RS, ARQ-RC-STSF, and ARQ-RC-MTMF, with packet error probability $q_{ji} = 0.6$.

which would induce additional delays and packet drops due to the increased probability of packet errors. However, utilizing feedback mechanisms as in the ARQ-based Ratio Consensus algorithm, effectively allows the values of running sums to reset when the receiving nodes successfully receive packets.

To emphasize the superiority of ARQ-based Ratio Consensus algorithms, we consider the directed network shown in Fig. 17 consisting of 10 nodes, with each node v_j choosing the weights on its out-going links (including its self-loop link) based on (2). The variables of each node x_j, y_j are updated within each time slot with initial values $\mathbf{x}[0] = (x_1[0] \ldots x_n[0])^\top = (0\ 28\ 6\ 8\ 26\ -2\ 18\ 2\ 4\ 10)^\top$, and $\mathbf{y}[0] = (y_1[0] \ldots y_n[0])^\top = \mathbf{1}_n$, where $n = 10$. In the examples that follow we consider that a transmitted packet may arrive in error at the receiving node with probability $q_{ji} = 0.6$, while the maximum retransmission limit (for the ARQ-based Ratio Consensus algorithms) is set to $\bar{\tau} = 5$.
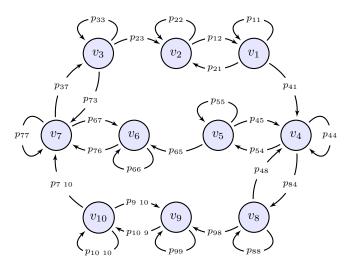

Fig. 17: Ten node strongly-connected digraph.

One can observe that utilizing ARQ-based Ratio Consensus algorithms (*i.e.,* ARQ-RC-STSF and ARQ-RC-MTMF), keeps the running sum values bounded and at low values as shown in Fig. 18 for the whole network, and in Fig. 19 for node v_2. In contrast, the RC-RS algorithm proposed in [20] does not reset the running sum values due to the absence of packet transmission feedback information. This fact results to growing values of the running sum in an unbounded manner with the number of iterations, and hence the transmission of such values may become costly in terms of communication since it would require a varying and possibly higher modulation and coding scheme (MCS) index.
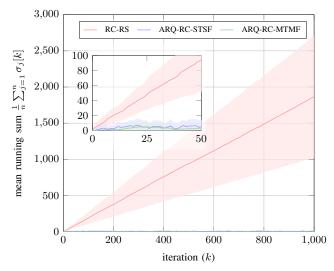

Fig. 18: Comparison of the running sum values of all the nodes in the network at each iteration k, using the RC-RS algorithm, and the ARQ-RC-STSF and ARQ-RC-MTMF schemes. Retransmission limit is set to $\bar{\tau} = 5$, and packet error probability is $q_{ji} = 0.6$.
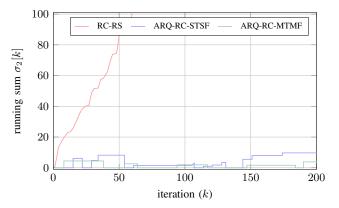

Fig. 19: Comparison of the running sum values at node v_2 at each iteration k, using the RC-RS algorithm, and the ARQ-RC-STSF and ARQ-RC-MTMF schemes. Retransmission limit is set to $\bar{\tau} = 5$, and packet error probability is $q_{ji} = 0.6$.

Remark 4. The ARQ-based Ratio Consensus algorithm resets the running sums values at successful receptions of packets by exploiting the ARQ feedback signals. Hence, the running

sums do not grow in an unbounded number with the number of iterations, and thus the running sums values can be transmitted albeit the communication bandwidth limitations.

VI. Conclusions and Future Directions

A. Conclusions

In this paper, we proposed a distributed algorithm to achieve discrete-time asymptotic average consensus, in the presence of time-varying delays induced by packet retransmissions, and packet-dropping communication links. By incorporating the ARQ protocol in a ratio-consensus-based algorithm, nodes can acquire their out-degree by utilizing the ARQ feedback signals sent by their out-neighbors, and determine a local upper-bound on the delays, imposed by the ARQ retransmission limit. We showed that the nodes of a strongly connected directed network can execute the ARQ-based Ratio Consensus algorithm to reach asymptotic average consensus over unreliable communication links.

B. Future Directions

A promising extension of this work to achieve faster convergence to the average consensus value, and maintain higher reliability of packet transmissions, is to devise a ratio consensus algorithm based on the Hybrid-ARQ (HARQ) protocol. This protocol incorporates Forward Error Correction (FEC) with the ARQ protocol, to reduce the frequency of retransmission by correcting the error patterns which occur most frequently, such that the system throughput increases, and hence the convergence of the average consensus is faster. This extension could exploit the potential of a wide variety of applications of distributed systems that involve wireless transmissions.

APPENDIX A
AUXILIARY LEMMAS

Lemma 1 implies that the event $\tilde{y}_j[k] \geq c^\lambda$, where c and λ were defined in §IV, occurs infinitely often and hence each node computes the ratio consensus value $z_j[k] = \tilde{x}_j[k]/\tilde{y}_j[k]$ at infinitely many time steps with probability 1. Lemma 2 implies that as k goes to infinity, the sequence of ratio computations $z_j[k]$ converges to the network-wide average of the initial values z^* in (10) almost surely (with probability 1).

Lemma 1. *Hadjicostis et. al. [20]. Let $\mathcal{K}_j = \{\kappa_j^1, \kappa_j^2, \ldots\}$ denote the sequence of time instants when $\tilde{y}_j[k] \geq c^\lambda$ and thus node $v_j \in \mathcal{V}$ updates its ratio consensus estimate using $z_j[k] = \tilde{x}_j[k]/\tilde{y}_j[k]$, where $\kappa_j^t < \kappa_j^{t+1}$, $t \geq 1$. The sequence $\mathcal{K}_j$ contains infinitely many elements with probability 1.*

Lemma 2. *Coefficient of Ergodicity, Hadjicostis et. al. [20]. Let $L_k = \prod_{\ell=0}^{k-1} \Xi[\ell]$ be the resulting column stochastic matrix from the product of column stochastic matrices $\Xi[\ell]$. Then the coefficient of ergodicity $\delta(L_k) := \max_j \max_{i_1, i_2} |L_k(j, i_1) - L_k(j, i_2)|$ converges almost surely to zero.*

APPENDIX B
PROOF OF THEOREM 1

Proof. Theorem 1 is established following similar analysis as in [20] *mutatis mutandis*, by replacing the matrix that corresponds to the network topology of the non-delayed case with the augmented matrix that incorporates the ARQ-based communication protocol defined in (7). First, it is clear to see that the evolution of the augmented variables in (6) can be written in the following form:

$$\tilde{x}[k] = \Xi[k-1]\Xi[k-2]\cdots\Xi[1]\Xi[0]\tilde{x}[0]$$
$$= \left(\Pi_{\ell=0}^{k-1}\Xi[\ell]\right)\tilde{x}[0] \equiv L_k\tilde{x}[0],$$
$$\tilde{y}[k] = \Xi[k-1]\Xi[k-2]\cdots\Xi[1]\Xi[0]\tilde{y}[0]$$
$$= \left(\Pi_{\ell=0}^{k-1}\Xi[\ell]\right)\tilde{y}[0] \equiv L_k\tilde{y}[0],$$

where $L_k = \Pi_{\ell=0}^{k-1}\Xi[\ell]$ is the forward product of column stochastic matrices $\Xi[\ell] \in \mathcal{X}$, $\forall \ell = 0, 1, \ldots, k-1$. From the above notation, it is clear that for each node $v_j \in \mathcal{V}$, we have $\tilde{x}_j[k] = L_k(j,:)\tilde{x}[0]$ and $\tilde{y}_j[k] = L_k(j,:)\tilde{y}[0]$, where $L_k(j,:)$ denotes the j-th row of the forward product matrix L_k. From Lemma 2 we can infer that $\delta(L_k) < \psi$ with probability $1 - \epsilon$ among chosen realizations for all $k \geq k_\psi$, where $0 < \psi \leq 1$ and $\epsilon > 0$. Considering any $k \geq k_\psi$ such that $y_j[k] \geq c^\lambda$, we can further infer that the maximum entry of $L_k(j,:)$ is at least c^λ/n since $y_j[k] = L_k(j,:)\tilde{y}[0] \geq c^\lambda$ and $\sum_l \tilde{y}_l[0] = n$. Moreover, since $\delta(L_k) < \psi$, the columns of matrix L_k are "within ψ" of each other. Hence, the j-th row of L_k can be written as $L_k(j,:) = c_j[k]\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)$, where $\mathbf{e}^\top[k] = [e_1[k], e_2[k], \ldots, e_m[k]]$ is an m-dimensional row vector that satisfies $e_{\max}[k] \equiv \max_l |e_l[k]| < \frac{\psi}{2c_j[k]}$, and $c_j[k] \geq \frac{c^\lambda}{n} - \frac{\psi}{2}$ (assume without loss of generality that $\psi < \frac{c^\lambda}{n}$), and $\mathbf{1}^\top$ is an all-ones row vector of appropriate dimension. Hence, at each node v_j, the ratio is obtained by:

$$z_j[k] = \frac{\tilde{x}_j[k]}{\tilde{y}_j[k]} = \frac{L_k(j,:)\tilde{x}[0]}{L_k(j,:)\tilde{y}[0]} = \frac{c_j[k]\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{x}[0]}{c_j[k]\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{y}[0]}$$
$$= \frac{\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{x}[0]}{\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{y}[0]}. \quad (13)$$

First, we establish the bounds on the numerator of the above expression as

$$\Sigma_x - e_{\max}[k]\Sigma_{|x|} \leq \left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{x}[0] \leq \Sigma_x + e_{\max}[k]\Sigma_{|x|},$$

where $\Sigma_x = \sum_l x_l[0]$, and $\Sigma_{|x|} = \sum_l |x_l[0]|$, for $l = 1, \ldots, n$. Now, for the bounds on the denominator, we know that $\sum_l y_l[0] = n$, since $y_l[0] = 1$ for $l = 1, \ldots, n$. Hence, we can bound the denominator as

$$n\left(1 - e_{\max}[k]\right) \leq \left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{y}[0] \leq n\left(1 + e_{\max}[k]\right).$$

Then, the ratio of the bounded numerator and denominator gives:

$$\frac{\Sigma_x - e_{\max}[k]\Sigma_{|x|}}{n\left(1 + e_{\max}[k]\right)} \leq \frac{\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{x}[0]}{\left(\mathbf{1}^\top + \mathbf{e}^\top[k]\right)\tilde{y}[0]} \leq \frac{\Sigma_x + e_{\max}[k]\Sigma_{|x|}}{n\left(1 - e_{\max}[k]\right)}.$$

From (10) we have $z^* = \frac{1}{n} \sum_l x_l[0] = \frac{\Sigma_x}{n}$, and thus:

$$z^* - E[k] \le z_j[k] \le z^* + E[k], \qquad (14)$$

where $E[k] = z^* \left(\Sigma_x + \Sigma_{|x|} \right) e^{\max}[k]/\Sigma_x \left(1 - e_{\max}[k] \right)$. To ensure that $z^* - \epsilon \le z_j[k] \le z^* + \epsilon$ holds whenever $k \ge k_\psi$ and $k \in \mathcal{K}_j$, we can choose $\psi < (2c^\lambda \epsilon/(\Sigma_x + \Sigma_{|x|} + 2n\epsilon))$, such that for any desirable $\epsilon > 0$, $e^{\max}[k] < (n\epsilon/(\Sigma_x + \Sigma_{|x|} + n\epsilon)$, holds. Hence, the ratio at each node in (13) converges with probability 1 to z^* as k goes to infinity. ∎

REFERENCES

[1] L. Schenato, "Optimal Estimation in Networked Control Systems subject to Random Delay and Packet Drop," *IEEE Trans. Autom. Control*, vol. 53, no. 5, pp. 1311–1317, 2008.

[2] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam, "The Wireless Control Network: A New Approach for Control over Networks," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2305–2318, 2011.

[3] M. J. Hossain, M. A. Mahmud, F. Milano, S. Bacha, and A. Hably, "Design of Robust Distributed Control for Interconnected Microgrids," *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2724–2735, 2015.

[4] A. Varghese and D. Tandur, "Wireless Requirements and Challenges in Industry 4.0," in *Int. Conf. Cont. Computing and Inform.*, 2014, pp. 634–638.

[5] M. Weiner, M. Jorgovanovic, A. Sahai, and B. Nikolié, "Design of a Low-latency, High-reliability Wireless Communication System for Control Applications," in *IEEE Int. Conf. Commun.*, 2014, pp. 3829–3835.

[6] P. Millán, L. Orihuela, C. Vivas, F. Rubio, D. V. Dimarogonas, and K. H. Johansson, "Sensor-Network-based Robust Distributed Control and Estimation," *Control Eng. Pract.*, vol. 21, no. 9, pp. 1238–1249, 2013.

[7] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization," *Proc. IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[8] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent," *Adv. in Neur. Inf. Proc. Syst.*, vol. 30, 2017.

[9] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-agent Systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[10] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents using Nearest Neighbor Rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[11] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging," *Syst. and Control Lett.*, vol. 53, no. 1, pp. 65–78, 2004.

[12] K. Cai and H. Ishii, "Average Consensus on General Strongly Connected Digraphs," *Automatica*, vol. 48, no. 11, pp. 2750–2761, 2012.

[13] A. Y. Kibangou, "Graph Laplacian based Matrix Design for Finite-time Distributed Average Consensus," in *IEEE Amer. Control Conf.*, 2012, pp. 1901–1906.

[14] J. M. Hendrickx, R. M. Jungers, A. Olshevsky, and G. Vankeerberghen, "Graph Diameter, Eigenvalues, and Minimum-Time Consensus," *Automatica*, vol. 50, no. 2, pp. 635–640, 2014.

[15] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed Finite-Time Average Consensus in Digraphs in the Presence of Time Delays," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 370–381, 2015.

[16] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and Control of Distributed Energy Resources for Provision of Ancillary Services," in *IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 537–542.

[17] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed Averaging in Sensor Networks based on Broadcast Gossip Algorithms," *IEEE Sensors J.*, vol. 11, no. 3, pp. 808–817, 2010.

[18] C. N. Hadjicostis and T. Charalambous, "Average Consensus in the Presence of Delays in Directed Graph Topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 763–768, 2013.

[19] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed Strategies for Average Consensus in Directed Graphs," in *IEEE Conf. Decision Control*, 2011, pp. 2124–2129.

[20] Hadjicostis, Christoforos N and Vaidya, Nitin H and Domínguez-García, Alejandro D, "Robust Distributed Average Consensus via Exchange of Running Sums," *IEEE Trans. Autom. Control*, vol. 61, no. 6, pp. 1492–1507, 2015.

[21] Y. Chen, R. Tron, A. Terzis, and R. Vidal, "Corrective Consensus: Converging to the Exact Average," in *IEEE Conf. Decision Control*, 2010, pp. 1221–1228.

[22] E. Seneta, *Non-negative Matrices and Markov Chains*. Springer, 2006.

[23] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *IEEE Symp. Found. of Comput. Sci.*, 2003, pp. 482–491.

[24] S. Lin, D. J. Costello, and M. J. Miller, "Automatic-Repeat-ReQuest Error-Control Schemes," *IEEE Commun. Magaz.*, vol. 22, no. 12, pp. 5–17, 1984.

[25] E. Krouk and S. Semenov, *Modulation and Coding Techniques in Wireless Communications*. John Wiley & Sons, 2011.

[26] A. Leon-Garcia and I. Widjaja, *Communication Networks: Fundamental Concepts and Key Architectures*. McGraw-Hill New York, 2000, vol. 2.

Evagoras Makridis received his B.Sc. degree in Electrical Engineering from Cyprus University of Technology in June 2017, and his M.Sc. double degree in Autonomous Systems from KTH Royal Institute of Technology and Aalto University through the EIT Digital Master School. During his post-graduate (M.Sc.) studies, he worked as a Research Assistant at Aalto University and KTH Royal Institute of Technology, and subsequently he joined Ericsson AB to pursue his M.Sc. thesis. In 2021, he started his doctoral (Ph.D) studies, focusing on distributed optimization for decision and control of networked systems, at the Department of Electrical and Computer Engineering at the University of Cyprus, under the supervision of Prof. Themistoklis Charalambous.

Themistoklis Charalambous completed his PhD studies in the Control Laboratory of the Engineering Department, Cambridge University in 2009. Following his PhD, he joined the Human Robotics Group as a Research Associate at Imperial College London (2009-2010). He also worked as a Visiting Lecturer at the Department of Electrical and Computer Engineering, University of Cyprus (2010-2011), as a Postdoctoral Researcher at the Department of Automatic Control of the School of Electrical Engineering at the Royal Institute of Technology (KTH) (2012-2015), and as a Postdoctoral Researcher at the Department of Electrical Engineering at Chalmers University of Technology (2015- 2016). In 2017, he joined the Department of Electrical Engineering and Automation, Aalto University as a tenure-track Assistant Professor becoming a tenured Associate Professor in July 2020. Since September 2021, he has been a tenure-track Assistant Professor at the Department of Electrical and Computer Engineering at the University of Cyprus.

Christoforos N. Hadjicostis (M'99, SM'05, F'20) received the S.B. degrees in electrical engineering, in computer science and engineering, and in mathematics, the M.Eng. degree in electrical engineering and computer science in 1995, and the Ph.D. degree in electrical engineering and computer science in 1999, all from Massachusetts Institute of Technology, Cambridge. In 1999, he joined the Faculty at the University of Illinois at Urbana–Champaign, where he served as Assistant and then Associate Professor with the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Information Trust Institute. Since 2007, he has been with the Department of Electrical and Computer Engineering, University of Cyprus, where he is currently Professor.