IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Utilizing Feedback Channel Mechanisms for Reaching Average Consensus over 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 embedding the Automatic Repeat ReQuest (ARQ) protocol used for error control of data transmissions, in order to allow the agents to reach asymptotic average consensus while handling time-varying delays induced by retransmissions of erroneous packets, and possible packet drops that occur due to excess of 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. The analysis of our proposed algorithm, herein called the ARQ-based Ratio Consensus algorithm, relies on augmenting the network's corresponding weighted adjacency matrix, to handle time-varying (yet bounded) delays and possible packet drops. To the best of the authors' knowledge, this is the first consensus algorithm that incorporates a communication protocol for error control used in real communication systems with feedback.**

Index Terms— **average consensus, digraphs, unreliable communication, retransmissions, delays, packet drops, ARQ protocol.**

I. Introduction

Robust information exchange between computing nodes (usually referred to as agents) is essential to apply various estimation and control algorithms in spatially distributed multi-agent systems [1]–[3]. 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. In decentralized (peer-to-peer) network topologies, to achieve certain network-wide (global) objectives, each node needs to interact with its immediate neighbors via a communication network (see [4] for a short review on network topology architectures for distributed computation). In practice, communication networks 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.

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 [5] 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 [6]–[8], or in finite time [9]–[11]. Several practical examples limit the exchange of information from bidirectional to directional mainly due to diverse transmission power and interference levels at 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. Such network topologies 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 [8], [12], [13] 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 [11], [14] proposed *ratio-consensus*-based algorithms, able 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 [15], is 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 [16], where nodes exchange messages containing information of their running sums to handle potential loss of packets. The aforementioned references proved, by introducing extra virtual nodes and links describing the underlying communication scenarios and topology, that network nodes asymptotically converge to the exact average.

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 (i) exploit

The authors are with the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus. E-mails: {makridis.evagoras, charalambous.themistoklis, chadjic}@ucy.ac.cy.

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

incoming error-free acknowledgement feedback signals to initially acquire or later update their out-degree, (ii) know whether a packet has arrived or not, and (iii) 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 prove asymptotic convergence of the proposed method to the exact average and we evaluate its performance via simulations for different scenarios. The proof is based on augmenting the weighted adjacency matrix that represents the network topology, to model possible arbitrarily time-varying (yet bounded) delays and packet drops on the communication links.

II. Background

A. Network Model

Consider n agents (represented by graph's nodes) communicating over a strongly connected network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ is the set of nodes 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 $v_j, v_i \in \mathcal{V}$, represents that node v_j can receive information from node v_i, *i.e.,* $v_i \to v_j$. 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. Robustified Ratio Consensus over Directed Graphs

The problem of average consensus involves a number of nodes in a network represented by the digraph $\mathcal{G}$, that cooperate by exchanging information to compute the network-wide average of their initial values. 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 an arbitrary value of node v_j), an auxiliary scalar variable, $y_j[k] \in \mathbb{R}^+$ (initialized at $y_j[0] = 1$), and $z_j[0] \in \mathbb{R}$ set to $z_j[k] = x_j[k]/y_j[k]$. In practice, packet transmissions are usually delayed due to poor channel conditions and network congestion. To overcome this issue, the authors in [14] proposed the *Robustified Ratio Consensus* 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 their proposed protocol, 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 on link ε_{ji} is denoted by $\bar{\tau}_{ji}$, while 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 [14] involves each node updating its states for each iteration according to

$$\begin{aligned} x_j[k+1] &= p_{jj} x_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji} x_i[k-r] \iota_{ji}[k-r], \\ y_j[k+1] &= p_{jj} y_j[k] + \sum_{v_i \in \mathcal{N}_j^-} \sum_{r=0}^{\bar{\tau}} p_{ji} y_i[k-r] \iota_{ji}[k-r], \\ z_j[k+1] &= \frac{x_j[k+1]}{y_j[k+1]}, \end{aligned} \tag{1}$$

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

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

Since each node v_j assigns the weights according to (2), it is required 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 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

$$\iota_{ji}[k-r] = \begin{cases} 1, & \text{if } \tau_{ji}[k-r] = r, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

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 [15]. Forcing the auxiliary variable y_k to be initialized at value 1 for each node, we can verify that $\lim_{k\to\infty} y_j[k] = n\boldsymbol{\pi}_j$ and that $\lim_{k\to\infty} x_j[k] = \boldsymbol{\pi}_j \sum_{i=1}^n x_i[0]$, where $\boldsymbol{\pi}$ is the right eigenvector of P ($\boldsymbol{\pi}$ 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 [15], [17]:

$$\lim_{k\to\infty} z_j[k] = \lim_{k\to\infty} \frac{x_j[k]}{y_j[k]} = \frac{\boldsymbol{\pi}_j \sum_{i=1}^n x_i[0]}{n\boldsymbol{\pi}_j} = \frac{1}{n} \sum_{i=1}^n x_i[0].$$

Remark 1. The Robustified Ratio Consensus algorithm handles (possibly) delayed information exchange between nodes, neglecting any packet drops due to erroneous

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

packets, while assuming that each node v_j is aware of its out-degree.

C. 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 [18, §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. 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 successful.

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-C). 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 or reception of a data packet 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 broadcasting a dummy packet in order to identify their out-neighbors. 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 error-free feedback channel (with negligible probability of error in the reception of this packet, which usually consists of a single bit), and 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 the 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 $\tau_{ji}[k]$. 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 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 $\mathbb{I}_{ji}[k + \bar{\tau}_{ji}] = 0$.

B. Feedback Scheme for Ratio Consensus

In this work, we consider a feedback scheme (hereinafter called multiple-transmissions-multiple-feedback MTMF) where each new packet is transmitted individually along with possibly retransmitted (delayed) packets over the same link ε_{ji} within a time slot. Note that, one could utilize a different feedback scheme, where each transmitting node aggregates the information (from new and delayed packets) in a single packet to be transmitted within a single time slot, and receive back its corresponding ARQ feedback signal (see [19] for more information). To get some intuition about the MTMF feedback scheme consider a simple example with two nodes v_1 and v_2, shown in Fig. 1. During the first time slot (starting at $k = 1$), a new packet $w_{21}[1]$ 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

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

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).

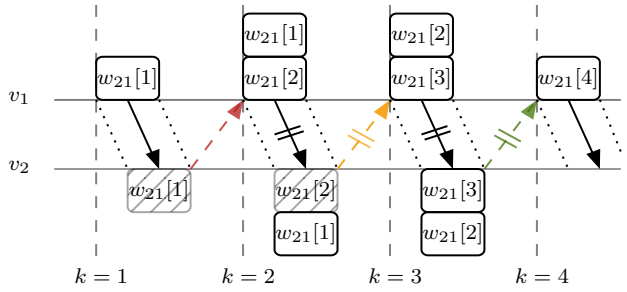

Fig. 1: MTMF ARQ feedback scheme. Red single arrows denote a NACK; orange bus arrows denote multiple ARQ feedback of ACKs and NACKs; green bus arrows denote multiple ACKs; shaded rectangles represent erroneous packets; transmissions involving multiple packets, ACKs or NACKs are indicated by arrows with multiple vertical lines.

C. ARQ-based Ratio Consensus Algorithm

Herein, we describe the ARQ-based Ratio Consensus algorithm for the MTMF feedback schemes considered in §III-B, although one can adjust the algorithm for other feedback schemes [19]. Specifically, each node v_j executes the steps in Algorithm 1, summarized as follows:

- First, the initial state $x_j[0] = V_j$ is determined by the information 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 $v_i \in \mathcal{N}_j^-$, which 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 (or more for improved out-degree acquisition accuracy) and summing the number of received ARQ feedback signals from its out-neighbors.
- 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 arrived 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; otherwise, 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.

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

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, which is only used to exploit the modeling and analysis of the algorithm without affecting its actual implementation. The network represented by the augmented digraph in Fig. 2 involves two nodes, *i.e.*, v_1 and v_2, that utilize an ARQ protocol with maximum retransmission limit $\bar{\tau}_{12} = 1$, and $\bar{\tau}_{21} = 2$, respectively. To capture the effect of delays due to retransmissions, and possible packet drops, we augment the original digraph

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

by adding extra virtual nodes (shown in yellow squares) and extra virtual buffer nodes (shown in red squares), respectively. 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. 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 $\mathbb{I}_{21}[k+\bar{\tau}_{21}]$. If $\mathbb{I}_{21}[k+\bar{\tau}_{21}]=1$, 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, $\mathbb{I}_{21}[k+\bar{\tau}_{21}]=0$, then the running sum information will propagate to the virtual buffer $v_{21}^{(f)}$. The information held in $v_{21}^{(f)}$, is released (by activating one of the virtual links $\mathbb{I}'_{21}$) to the actual node v_2 along with the (possibly delayed) packet that is intended to be transmitted in the next time slot. Packet transmission from node v_2 to node v_1, is delayed by one time slot which equals the maximum retransmission limit. If no further error was detected in the last retransmission trial, then the packet will arrive successfully at node v_1.

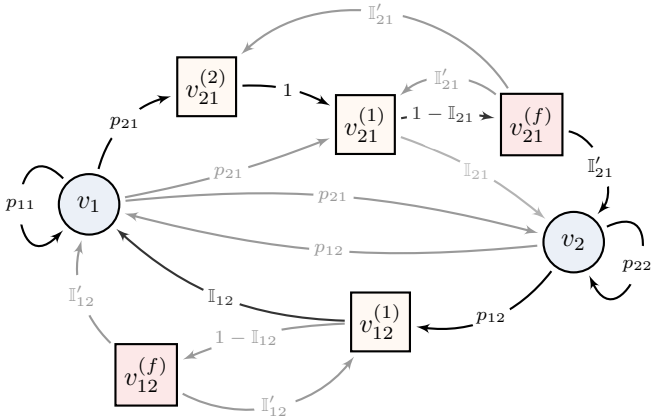

Fig. 2: Two node digraph denoting ARQ-based transmissions. A packet transmitted from node v_1 to v_2 is delayed by two time slots, and is dropped in the subsequent time slot. A packet transmitted from node v_2 to v_1 is delayed by one time slot.

Remark 2. The ARQ-based Ratio Consensus algorithm handles unreliable packet transmissions, by utilizing 1-bit feedback signals that can also be 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 $\bar{\tau}_{ji}=\bar{\tau}$. 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 node v_j after r time steps. In case of a packet drop, $\mathbb{I}_{ji}[k]=0$, the delayed information (held in the corresponding virtual node of maximum delay) will propagate to the virtual buffer node of the original node v_j instead of being received by 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 $\tau_{ji}[k+1]\geq 0$. Finally, the augmented digraph, $\mathcal{G}^a=(\mathcal{V}^a,\mathcal{E}^a)$, that models the delayed network with packet drops consists of exactly $\tilde{n}=|\mathcal{E}|(\bar{\tau}+1)+n\leq n(n-1)(\bar{\tau}+1)+n$ nodes $\in\mathcal{V}^a$, where n nodes are actual nodes, $|\mathcal{E}|\bar{\tau}$ 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

$$\tilde{x}[k+1]=\Xi[k]\tilde{x}[k], \tag{4a}$$

$$\tilde{y}[k+1]=\Xi[k]\tilde{y}[k], \tag{4b}$$

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 time-varying matrix associated with the augmented digraph:

$$\Xi[k]\triangleq\begin{pmatrix} P^{(0)}[k] & D^{\text{succ}}[k] & 0 & \cdots & 0 & D^{(0)}[k+1] \\ P^{(1)}[k] & D^{\text{exc}}[k] & I & \cdots & 0 & D^{(1)}[k+1] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ P^{(\bar{\tau}-1)}[k] & 0 & 0 & \cdots & I & D^{(\bar{\tau}-1)}[k+1] \\ P^{(\bar{\tau})}[k] & 0 & 0 & \cdots & 0 & D^{(\bar{\tau})}[k+1] \\ 0 & D^{\text{pd}}[k] & 0 & \cdots & 0 & D^{\text{sl}}[k+1] \end{pmatrix} \tag{5}$$

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

$$P^{(r)}[k](j,i)=\begin{cases} P(j,i), & \text{if } \tau_{ji}[k]=r,\ (j,i)\in\mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

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 value 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

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

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 (6). 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 these properties, it is clear that matrix $\Xi[k]$ maintains column-stochasticity, albeit the transmission links between nodes might be unreliable.

Prior to establishing the convergence of each node to the average consensus value, stated in Theorem 1, we first establish some properties of matrices $\Xi[k]$, similar to the ones established in [16] 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 $\tilde{n} \times \tilde{n}$. 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.

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

$$z^* = \frac{\sum_{l\in\mathcal{V}^a} \tilde{x}_l[0]}{\sum_{l\in\mathcal{V}^a} \tilde{y}_l[0]} = \frac{\sum_{l\in\mathcal{V}} x_l[0]}{\sum_{l\in\mathcal{V}} y_l[0]} = \frac{1}{n}\sum_{l\in\mathcal{V}} x_l[0], \qquad (7)$$

by calculating at each time step k, 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 consensus value, we need to show that the augmented auxiliary variable $\tilde{y}_j[k] \geq c^\lambda$ occurs infinitely often, and hence as k goes to infinity, the sequence of ratio computations $z_j[k]$ converges to the value in (7) 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 $k = 0, 1, 2, \ldots$) be the output of the iterations in Algorithm 1. Then, the solution to the average consensus can be obtained at each node, by computing:*

$$\lim_{k\to\infty} z_j[k] = \frac{x_j[k]}{y_j[k]} = \frac{\sum_{v_i\in\mathcal{V}} x_i[0]}{n}, \ \forall v_j \in \mathcal{V}. \qquad (8)$$

Proof. Theorem 1 is established in [19], following similar analysis as in [16] *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 protocol defined in (5). ■

V. Numerical Evaluation

Consider a directed network with each node v_j choosing its out-going links (including its self-loop link) based on (2), which results into the following column-stochastic matrix:

$$P = \begin{pmatrix} 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/2 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/3 \\ 0 & 1/3 & 1/2 & 0 & 1/3 \end{pmatrix}. \qquad (9)$$

The variables of each node x_j, y_j are updated within each time slot using Algorithm 1 with initial values $\mathbf{x}[0] = (x_1[0]\ldots x_n[0])^T = (4\ 5\ 6\ 3\ 2)^T$, and $\mathbf{y}[0] = (y_1[0]\ldots y_n[0])^T = (1\ 1\ 1\ 1\ 1)^T$, respectively. In the following example we consider the case where 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 algorithm is set to $\bar{\tau}_{ji} = \bar{\tau} = 2$. Fig. 3 shows the evolution of the ratio at node v_1 during the first time steps of the execution of the algorithm, where although the link ε_{14} is unreliable, the ratio $z_1[k]$ converges to the average consensus value.

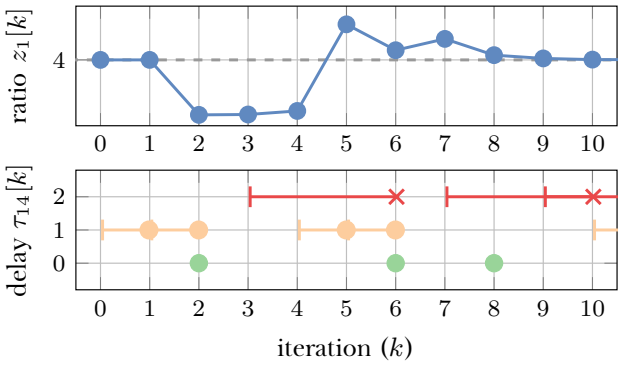

Fig. 3: Upper: Ratio $z_1[k]$ of node v_1 utilizing the ARQ-based Ratio Consensus algorithm with packet error probability $q_{ji} = 0.6$. Lower: Snapshot of packet transmissions from node v_4 to node v_1. Non-delayed transmissions are denoted with green circle marks, while $\tau_{14}[k] = 1$, and $\tau_{14}[k] = 2$ are denoted with orange and red, respectively. Packet drops are denoted with red cross marks.

Next, we consider two scenarios for different channel conditions, that are captured by: (a) packet error probability $q_{ji} = 0.2$, and (b) packet error probability $q_{ji} = 0.8$. We assess both cases for two different configurations on the maximum retransmission limit of the ARQ-based Ratio Consensus algorithm, $\bar{\tau} = 2$, and $\bar{\tau} = 10$. Variables $x_j[k]$ and $y_j[k]$, do not converge at a certain value due to the time-varying delays on the links and possible packet drops [14], [16]. However, the ratio of each node shown in Fig. 4 converges to the average consensus value, despite the time-varying delays due to retransmissions, and possible packet drops. The lower plot of Fig. 4 presents the convergence of the ratio $z_j[k]$ when the maximum retransmission limit is set to $\bar{\tau} = 10$. The convergence rate under this configuration is similar to the one with $\bar{\tau} = 2$ since the packet error probability is relatively small, and hence most of the packets will arrive at their destined nodes successfully.

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.

In contrast, when $q_{ji} = 0.8$, we can see in Fig. 5 that the convergence to the average consensus varies with the maximum retransmission limit. In particular, nodes converge faster with $\bar{\tau} = 2$ (upper plot in Fig. 5), while for $\bar{\tau} = 10$ (lower plot in Fig. 5), the convergence to the average consensus is significantly slower. However, for all the aforementioned scenarios, nodes utilizing the ARQ-based Ratio Consensus algorithm are guaranteed to converge to the average consensus value despite the experienced delays and packet drops induced by the unreliability of the channels.

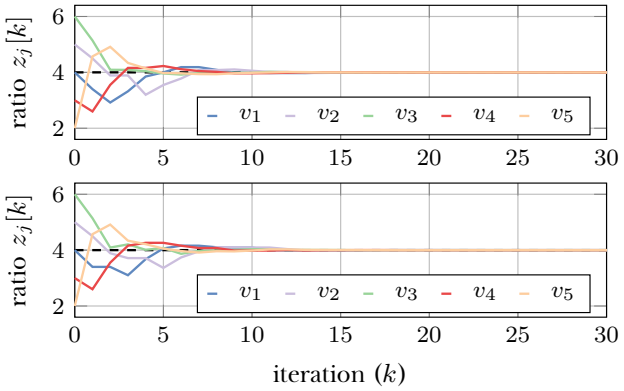

Fig. 4: Ratio $z_j[k]$ of each node v_j, with $\bar{\tau} = 2$ (upper), and $\bar{\tau} = 10$ (lower), and packet error probability $q_{ji} = 0.2$.

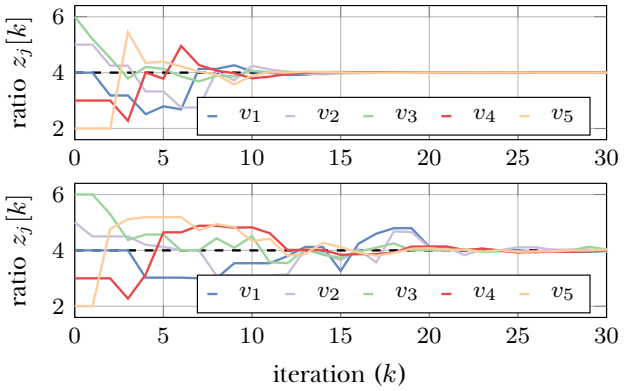

Fig. 5: Ratio $z_j[k]$ of each node v_j, with $\bar{\tau} = 2$ (upper), and $\bar{\tau} = 10$ (lower), and packet error probability $q_{ji} = 0.8$.

VI. Conclusions and Future Directions

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 in-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. A promising extension of this work to achieve faster convergence to the average consensus value, is to devise consensus algorithms based on a Hybrid-ARQ (HARQ) protocol which increases the system throughput, by correcting frequent error patterns.

Acknowledgement

This work was partly funded by the European Research Council (ERC) project MINERVA under the European Union's Horizon 2022 research and innovation programme (Grant agreement No. 101044629).

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] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] A. D. Dominguez-Garcia 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] E. Makridis, T. Charalambous, and C. N. Hadjicostis, "ARQ-based Average Consensus over Unreliable Directed Network Topologies," 2022. [Online]. Available: https://arxiv.org/abs/2209.14699

Authorized licensed use limited to: University of Cyprus. Downloaded on July 06,2023 at 16:25:54 UTC from IEEE Xplore. Restrictions apply.