Distributed Algorithms for Consensus and Coordination in the Presence of Packet-Dropping Communication Links

Part II: Coefficients of Ergodicity Analysis Approach

Nitin H. Vaidya, *Fellow, IEEE*
Christoforos N. Hadjicostis, *Senior Member, IEEE*
Alejandro D. Domínguez-García, *Member, IEEE*

September 2011

Abstract

In this two-part paper, we consider multicomponent systems in which each component can iteratively exchange information with other components in its neighborhood in order to compute, in a distributed fashion, the average of the components' initial values or some other quantity of interest (i.e., some function of these initial values). In particular, we study an iterative algorithm for computing the average of the initial values of the nodes. In this algorithm, each component maintains two sets of variables that are updated via two identical linear iterations. The average of the initial values of the nodes can be asymptotically computed by each node as the ratio of two of the variables it maintains. In the first part of this paper, we show how the update rules for the two sets of variables can be enhanced so that the algorithm becomes tolerant to communication links that may drop packets, independently among them and independently between different transmission times. In this second part, by rewriting the collective dynamics of both iterations, we show that the resulting system is mathematically equivalent to a finite inhomogenous Markov chain whose transition matrix takes one of finitely many values at each step. Then, by using e a coefficients of ergodicity approach, a method commonly used for convergence analysis of Markov chains, we prove convergence of the robustified consensus scheme. The analysis suggests that similar convergence should hold under more general conditions as well.

Note to readers: Section I discusses the relation between Part II (this report) and the companion Part I of the report, and discusses some related work. The readers may skip Section I without a loss of continuity.

University of Illinois at Urbana-Champaign. Coordinated Science Laboratory technical report UILU-ENG-11-2208 (CRHC-11-06)

N. H. Vaidya and A. D. Domínguez-García are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: {nhv, aledan}@ILLINOIS.EDU.

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

The work of A. D. Domínguez-García was supported in part by NSF under Career Award ECCS-CAR-0954420. The work of C. N. Hadjicostis was supported in part by the European Commission (EC) 7th Framework Programme (FP7/2007-2013) under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. The work of N. H. Vaidya was supported in part by Army Research Office grant W-911-NF-0710287 and NSF Award 1059540. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

I. INTRODUCTION

The focus of this paper is to analyze the convergence of the robustified double-iteration[1] algorithm for average consensus introduced in Part I, utilizing a different framework that allows us to move away from the probabilistic model describing the availability of communication links of Part I. More specifically, instead of focusing on the dynamics of the first and second moments of the two iterations to establish convergence as done in Part I, we consider a framework that builds upon the theory of finite inhomogenous Markov chains. In this regard, by augmenting the communication graph, we will show that the collective dynamics of each of the two iterations can be rewritten in such a way that the resulting system is mathematically equivalent to a finite inhomogenous Markov chain whose transition matrix takes values from a finite set of possible matrices. Once the problem is recasted in this fashion, tools, such as coefficients of ergodicity, commonly used in the analysis of inhomogenous Markov chains (see, e.g., [**?**]) are used to prove the convergence of the algorithm.

Recalling from Part I, when the communication network is perfectly reliable (i.e., in the absence of packet drops), the collective dynamics of the linear iterations can be described by a discrete-time transition system with no inputs in which the transition matrix is column stochastic and primitive. Then, each node runs two identical copies of a linear iteration, with each iteration initialized differently depending on the problem to be solved. This double-iteration algorithm is a particular instance of the algorithm in [**?**] (which is a generalization of the algorithm proposed in [**?**]), where the matrices describing each linear iteration are allowed to vary as time evolves, whereas in our setup (for the ideal case when there are no communication link failures) the transition matrix is fixed over time. In general, the algorithm described above is not robust against packet-dropping communication links. It might be possible to robustify it by introducing message delivery acknowledgment mechanisms and retransmission mechanisms, but this has certain overhead and drawbacks as discussed in Section II-C. Also, in a pure broadcast system, which is the communication model we assume in this work, it is easy to see that the double-iteration algorithm above will not work properly. The mechanism we proposed in Part I to robustify the double iteration algorithm was for each node i to keep track of three quantities of interest: i) its own internal state (as captured by the state variables maintained in the original double iteration scheme of [**?**], [**?**]; ii) an auxiliary variable that accounts for the total mass broadcasted so far by node i to (all of) its neighbors; and iii) another auxiliary variable that accounts for the total received mass from each node j that sends information to node i. The details of the algorithm are provided in Section III, but the key in analyzing convergence of the algorithm is to show that the collective system dynamics can be rewritten by introducing additional nodes—virtual buffers—that account for the difference between these two auxiliary variables. The resulting enhanced system is equivalent to an inhomogenous Markov chain whose transition matrix takes values from a finite set.

As discussed in Part I, even if relying on the ratio of two linear iterations, our work is different from the work in [**?**] in terms of both the communication model and also the nature of the protocol itself.

[1]In this second part we will also refer to this algorithm as "ratio consensus" algorithm and will use both denominations interchangeably.

In this regard, a key premise in [**?**] is that stochasticity of the transition matrix must be maintained over time, which requires sending nodes to know the number of nodes that are listening, suggesting that i) either the communication links are perfectly reliable, or ii) there is some acknowledgment and retransmission mechanism that ensures messages are delivered to the listening nodes at every round of information exchange. In our work, we remove both assumptions, and assume a pure broadcast model without acknowledgements and retransmissions. It is very easy to see that in the presence of lossy communication links, the algorithm in [**?**] does not solve the average consensus problems as stochasticity of the transition matrix is not preserved over time. Thus, as mentioned above, the key in the approach we follow to analyze convergence is to augment the communication graph by introducing additional nodes, and to establish the correctness of the algorithms and establish that the collective dynamics of the resulting system is equivalent to a finite inhomogenous Markov chain with transition matrix that values values from a finite set. Once the system is rewritten in this fashion, the robust algorithm for ratio consensus reduces to a similar setting to the one in [**?**], except for the fact that some of the the resulting transition matrices might not have positive diagonals, which is required for the proof in [**?**]. Thus, in this regard, our approach may be also viewed as a generalization of the main result in [**?**].

The idea of augmenting the communication graph has been used in consensus problems to study the impact of bounded (fixed and random) communication delays [**?**], [**?**], [**?**]. In our work, the augmented communication graph that results from rewriting the collective system dynamics has some similarities to the augmented communication graph in [**?**], where the link from node i to node j is replaced by several paths from node i to node j, in order to mimic the effect of communication delays. In particular, in [**?**], for a maximum delay of B steps, B paths are added in parallel with the single-edge path that captures the non-delayed message transmission. The added path corresponding to delay b ($1 \leq b \leq B$) has b nodes, for a total of $B(B+1)/2$ additional nodes capturing the effect of message transmission delays from node i to node j. At every time step, a message from node i to node j is randomly routed through one of these paths; the authors assume for simplicity that each of the paths is activated with probability $1/B$. For large communication graphs, one of the drawbacks of this model is the explosion in the number of nodes to be added to the communication graph to model the effect of delays. In our work, for analysis purposes, we also use the idea of augmenting the communication graph, but in our case, a single parallel path is sufficient to capture the effect of packet-dropping communication links. As briefly discussed later, it is easy to see that our modeling formalism can also be used to capture random delays, with the advantage over the formalism in [**?**] that in our model, it is only necessary to add a single parallel path with B nodes (instead of the $B(B+1)/2$ nodes added above) per link in the original communication path, which reduces the number of states added. Additionally, our modeling framework can handle any delay distribution, as long as the equivalent augmented network satisfies properties (M1)-(M5) discussed in Section IV-A.

In order to make Part II self-contained, we review several ideas already introduced in Part I, including

the double-iteration algorithm formulation over perfectly reliable networks and its robustified version. In Part II, we will embrace the common convention utilized in Markov chains of pre-multiplying the transition matrix of the Markov chain by the corresponding probability vector.

The remainder of this paper is organized as follows. Section II introduces the communication model, briefly describes the non-robust version of the double-iteration algorithm, and discusses some issues that arise when implementing the double-iteration algorithm in networks with unreliable links. Section III describes the strategy to robustify the double-iteration algorithm against communication link failures. Section IV reformulates each of the two iterations in the robust algorithm as an inhomogeneous Markov chain. We employ coefficients of ergodicity analysis to characterize the algorithm behavior in Section V. Convergence of the robustified double-iteration algorithm is established in Section VI. Concluding remarks and discussions on future work are presented in Section VII.

II. PRELIMINARIES

This section describes the communication model we adopt throughout the work, introduces notation, reviews the double-iteration algorithm that can be used to solve consensus problems when the communication network is perfectly reliable, and discusses issues that arise when implementing the double-iteration algorithm in networks with packet-dropping links.

A. Network Communication Model

The system under consideration consists of a network of m nodes, $\mathcal{V} = \{1, 2, \ldots, m\}$, each of which has some initial value $v_i,\ i = 1, 2, \ldots, m$, (e.g., a temperature reading). The nodes need to reach consensus to the average of these initial values in an iterative fashion. In other words, the goal is for each node to obtain the value $\frac{\sum_{j=1}^{m} v_j}{m}$ in a distributed fashion. We assume a synchronous[2] system in which time is divided into *time steps* of fixed duration. The nodes in the network are connected by a certain directed network. More specifically, a directed link (j, i) is said to "exist" if transmissions from node j can be received by node i infinitely often over an infinite interval. Let $\mathcal{E}$ denote the set of all directed links that exist in the network. For notational convenience, we take that $(i, i) \in \mathcal{E},\ \forall i$, so that a self-loop exists at each node. Then, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents the network connectivity. Let us define $\mathcal{I}_i = \{j \mid (j, i) \in \mathcal{E}\}$ and $\mathcal{O}_i = \{j \mid (i, j) \in \mathcal{E}\}$. Thus, $\mathcal{I}_i$ consists of all nodes from whom node i has incoming links, and $\mathcal{O}_i$ consists of all nodes to whom node i has outgoing links. For a set S, we will denote the cardinality of set S by $|S|$. The outdegree of node i, denoted as D_i, is the size of set $\mathcal{O}_i$, thus, $D_i = |\mathcal{O}_i|$. Due to the assumption that all nodes have self-loops, $i \in \mathcal{I}_i$ and $i \in \mathcal{O}_i$, $\forall i \in \mathcal{V}$. We assume that graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is strongly connected. Thus, in $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, there exists a directed path from any node i to any node j, $\forall i, j \in \mathcal{V}$ (although it is possible that the links on such a path between a pair of nodes may not all be simultaneously reliable in a given time slot).

[2]We later discuss how the techniques we develop for reaching consensus using the double iteration algorithm in the presence of packet-dropping links naturally lead to an asynchronous computation setup.

The iterative consensus algorithms considered in this paper assume that, at each step of the iteration, each node transmits some information to all the nodes to whom it has a reliable directed link during that iteration (or "time step"). The iterative consensus algorithm summarized in Section II-B assumes the special case wherein all the links are *always reliable* (that is, all links are reliable in every time step). In Section III, and beyond, we consider a network with potentially unreliable links. Our work on iterative consensus over unreliable links is motivated by the presence of such links in wireless networks. Suppose that the nodes in our network communicate over wireless links, with the node locations being fixed. In such a wireless network, each node should generally be able to communicate with the other nodes in its vicinity. However, such transmissions may not always be reliable, due to channel fading and interference from other sources. To make our subsequent discussion precise, we will assume that a link (i, j) exists (i.e., $(i, j) \in \mathcal{E}$) only if each transmission from i is successfully received by node j with probability q_{ij} ($0 < q_{ij} \leq 1$). We assume that successes of transmissions on different links are independent of each other; also, successes of different transmissions on any given link are independent of each other. As we will see, these independence assumptions can be partially relaxed but we adopt them at this point for simplicity.

We assume that all transmissions from any node i are *broadcasts*,[3] in the sense that, every node j, such that $(i, j) \in \mathcal{E}$, may receive i's transmission with probability q_{ij} independently between nodes and transmission steps. As seen later, this broadcast property can potentially be exploited to make communication more efficient, particularly when a given node i wants to send identical information to all the nodes in $\mathcal{O}_i$. When node i broadcasts a message to its neighbors, the reliabilities of receptions at different nodes in $\mathcal{O}_i$ are mutually independent. Each node i is assumed to be aware of the value of D_i (i.e., the number of nodes in $\mathcal{O}_i$), and the identity of each node in set $\mathcal{I}_i$. This information can be learned using *neighbor discovery* mechanisms used in wireless ad hoc or mesh networks. Note that node i does not necessarily know whether transmissions to nodes in $\mathcal{O}_i$ are successful.

B. Ratio Consensus Algorithm in Perfectly Reliable Communication Networks

In this section, we summarize a consensus algorithm for a special case of the above system, wherein all the links in the network are *always reliable* (that is, reliable in every time step). The "ratio consensus" algorithm presented here performs two iterative computations in parallel, with the solution of the consensus algorithm being asymptotically obtained as the *ratio* of the outcome of the two parallel iterations. We will refer to this approach as *ratio consensus*. In prior literature, similar approaches have also been called *weighted consensus* [**?**], [**?**].

Each node i maintains at iteration k state variables $y_k[i]$ and $z_k[i]$. At each time step k, each node i

[3]As elaborated later, the results in this paper can also be applied in networks wherein the transmissions are unicast (not broadcast).

updates its state variable as follows:

$$y_k[i] = \sum_{j \in \mathcal{I}_i} y_{k-1}[j] \,/\, D_j \ , \qquad k \geq 1, \tag{1}$$

$$z_k[i] = \sum_{j \in \mathcal{I}_i} z_{k-1}[j] \,/\, D_j \ , \qquad k \geq 1, \tag{2}$$

where $y_0[j] = v_j, \ \forall j = 1, \ldots, m$, and $z_0[j] = 1, \ \forall j = 1, \ldots, m$.

To facilitate implementation of the above iterations, at time step k, each node i broadcasts a message containing values $y_{k-1}[i]/D_i$ and $z_{k-1}[i]/D_i$ to each node in $\mathcal{O}_i$, and awaits reception of a similar message from each node in $\mathcal{I}_i$. When node i has received, from each node $j \in \mathcal{I}_i$, a value (namely, $y_{k-1}[j]/D_j$ and $z_{k-1}[i]/D_j$) at step k, node i performs the above update of its state variables (by simply summing the corresponding values). Hereafter, we will use the phrase "message v" to mean "message containing value v".

The above two iterations are represented in a matrix notation in (3) and (4), where y_k and z_k are row vectors of size m, and M is an $m \times m$ primitive matrix[4], such that $M[i,j] = 1/D_i$ if $j \in \mathcal{O}_i$ and 0 otherwise. Compactly, we show

$$y_k = y_{k-1} \ M, \qquad k \geq 1 \ , \tag{3}$$

$$z_k = z_{k-1} \ M, \qquad k \geq 1. \tag{4}$$

It is assumed that $z_0[j] = 1$ and $y_0[j] = v_j$ are the initial values at each node $j \in \mathcal{V}$. Each node i calculates, at each time step k, the ratio

$$v_k[i] = \frac{y_k[i]}{z_k[i]} \ .$$

For the transition matrix M, (a) $M[i,j] \geq 0$, and (b) for all i, $\sum_j M[i,j] = 1$. Any matrix that satisfies these two conditions is said to be a *row stochastic* matrix. It has been shown in [**?**] that $v_k[i]$ asymptotically converges to the average of the elements of y_0, provided that M is *primitive* and *row stochastic*. That is, if M is a primitive row stochastic matrix, then

$$\lim_{k \to \infty} \ v_k[i] = \frac{\sum_j \ y_0[j]}{m}, \ \ \forall i \in \mathcal{V}, \tag{5}$$

where m is the number of elements in vector y_0.

C. Implementation Aspects of Ratio Consensus Algorithm in the Presence of Unreliable Links

Let us consider how we might implement iterations (3) and (4) in a wireless network. Since the treatment for the y_k and z_k iterations is similar, let us focus on the y_k iteration for now. Implementing (3) requires that, at iteration k (to compute y_k), node i should transmit message $y_{k-1}[i] \, M[i,j]$ to each

[4]A finite square matrix A is said to be *primitive* if for some positive integer p, $A^p > 0$, that is, $A^p[i,j] > 0, \ \forall i,j$.

node $j \in \mathcal{O}_i$. Conveniently, for all $j \in \mathcal{O}_i$, the values $M[i,j]$ are identical, and equal to $1/D_i$. Thus, node i needs to send message $y_{k-1}[i]/D_i$ to each node in $\mathcal{O}_i$. Let us define

$$\mu_k[i] \equiv y_{k-1}[i]\,/D_i\ , \qquad k \geq 1.$$

In a wireless network, the two approaches described next may be used by node i to transmit message $\mu_k[i]$ to all the nodes in $\mathcal{O}_i$.

Approach 1: In this approach, each node i ensures that its message $\mu_k[i]$ is delivered reliably to all the nodes in $\mathcal{O}_i$. One way to achieve this goal is as follows. Node i can broadcast the message $\mu_k[i]$ on the wireless channel, and then wait for acknowlegdements (ack) from all the nodes in $\mathcal{O}_i$. If such acks are not received from all nodes in $\mathcal{O}_i$ within some timeout interval, then i can retransmit the message. This procedure will be repeated until acks are received from all the intended recipients of $\mu_k[i]$. This procedure ensures that the message is received by each node in $\mathcal{O}_i$ reliably in each step k of the iteration. However, as an undesirable side-effect, the time required to guarantee the reliable delivery to all the neighboring nodes is not fixed. In fact, this time can be arbitrarily large with a non-zero probability, if each transmission on a link $(i,j) \in \mathcal{E}$ is reliable with probability $q_{ij} < 1$. Different nodes may require different amounts of time to reliably deliver their message to their intended recipients. Thus, if a fixed finite interval of time is allocated for each step k, then it becomes difficult to guarantee that the iterations will be always performed *correctly* (because some messages may not be delivered within the fixed time interval).

Approach 2: Alternatively, each node i may just broadcast its message $\mu_k[i]$ once in time step k, and hope that all the nodes in $\mathcal{O}_i$ receive it reliably. This approach has the advantage that each step of the iteration can be performed in a short (and predictable) time interval. However, it also has the undesirable property that all the nodes in $\mathcal{O}_i$ may not receive the message (due to link unreliability), and such nodes will not be able to update their state correctly. It is important to note that, since there are no acknowlegements being sent, a node i cannot immediately know whether a node $j \in \mathcal{O}_i$ has received i's message or not.

Considering the shortcomings of the above two approaches, it appears that an alternative solution is required. Our solution to the problem (to be introduced in Section III) is to maintain *additional* state at each node, and utilize this state to mitigate the detrimental impact of link unreliability. To put it differently, the additional state can be used to design an iterative consensus algorithm *robust* to link unreliability. In particular, the amount of state maintained by each node i is proportional to $|\mathcal{I}_i|$. In a large scale wireless network (i.e, with large m) with nodes spread over large space, we would expect that for any node i, $|\mathcal{I}_i| << m$. In such cases, the small increase in the amount of state is a justifiable cost to achieve robustness in presence of link unreliability.

Although $M[i,j]$ is identical (and equal to $1/D_i$) for all $j \in O_i$ in our example above, this is not necessary. So long as M is a primitive row stochastic matrix, the above iteration will converge to the correct consensus value (provided that the transmissions are always reliable). Thus, it is possible that in

a given iteration, node i may want to send different messages to different nodes in O_i. This goal can be achieved by performing unicast operation to each node in O_i. In this situation as well, two approaches analogous to Approaches 1 and 2 may be used. The first approach would be to reliably deliver the unicast messages, using as many retransmissions as necessarys. The second approach may be to transmit each message just once. In both cases, it is possible that the iterations may not be performed correctly. To simplify the discussion in this paper, we assume that each node i needs to transmit identical message to the nodes in $\mathcal{O}_i$. However, it is easy to extend the proposed scheme so that it is applicable to the more general scenario as well.

III. Robustification of Ratio Consensus Algorithm

In this section, we present the proposed ratio consensus algorithm that is robust in presence of link unreliability. The correctness of the proposed algorithm is established in Section VI. As before, each node maintains state variables $y_k[i]$ and $z_k[i]$. Additional state maintained at each node will be defined soon. Iterative computation is performed to maintain y_k and z_k. For brevity, we will focus on presenting the iterations for y_k, but iterations for z_k are analogous, with the difference being in the initial state. The initial values of y and z are assumed[5] to satisfy the following conditions:

1) $y_0[i] \geq 0, \ \forall i$,
2) $z_0[i] \geq 0, \ \forall i$,
3) $\sum_i z_0[i] > 0$.

Our goal for the robust iterative consensus algorithm is to allow each node i to compute (asymptotically) the ratio

$$\frac{\sum_i y_0[i]}{\sum_i z_0[i]}.$$

With a suitable choice of $y_0[i]$ and $z_0[i]$, different functions may be calculated [**?**]. In particular, if the initial input of node i is denoted as v_i, then by setting $y_0[i] = w_i v_i$ and $z_0[i] = w_i$, where $w_i \geq 0, \ \forall i$, the nodes can compute the weighted average $\frac{\sum_i w_i v_i}{\sum_i w_i}$; with $w_i = 1, \ \forall i \in \mathcal{V}$, the nodes calculate average consensus.

A. Intuition Behind the Robust Algorithm

To aid our presentation, let us introduce the notion of "mass." The initial value $y_0[i]$ at node i is to be viewed as its initial mass. If node i sends a message v to another node j, that can be viewed as a "transfer" of an amount of mass equal to v to node j. With this viewpoint, it helps to think of each step k as being performed over a non-zero interval of time. Then, $y_k[i]$ should be viewed as the mass at node i at the *end* of time step k (which is the same as the *start* of step $k+1$). Thus, during step k, each node i transfers (perhaps unsuccessfully, due to unreliable links) some mass to nodes in $\mathcal{O}_i$, the

[5]The assumption that $y_0[i] \geq 0, \ \forall i$, can be relaxed, allowing for arbitrary values for $y_0[i]$.

amount being a function of $y_{k-1}[i]$. The mass $y_k[i]$ is the accumulation of the mass that i receives in messages from nodes in $\mathcal{I}_i$ during step k.

Now, $\sum_i y_0[i]$ is the total mass in the system initially. If we implement iteration (3) in the absence of packet drops, then for all iterations k

$$\sum_i y_k[i] = \sum_i y_0[i].$$

That is, the total mass in the system remains constant. This invariant is maintained because M is a row stochastic matrix. However, if a message v sent by node i is not received by some node $j \in \mathcal{O}_i$, then the mass in that message is "lost," resulting in reduction of the total mass in the system.

Our robust algorithm is motivated by the desire to avoid the loss of mass in the system, even in the presence of unreliable links. The proposed algorithm uses Approach 2 for transmission of messages. In particular, in our algorithm (and as in the original ratio consensus), at each step k, each node i wants to transfer $\mu_k[i] = y_{k-1}[i]/D_i$ amount of mass to each node in $\mathcal{O}_i$. For this purpose, node i broadcasts[6] message $\mu_k[i]$. To make the algorithm robust, let us assume that, for each link $(i,j) \in \mathcal{E}$, a "virtual buffer" is available to store the mass that is "undelivered" on the link. For each node $j \in \mathcal{O}_i$, there are two possibilities:

(P1) Link (i,j) is not reliable in slot k: In this case, message $\mu_k[i]$ is *not* received by node j. Node i believes that it has transferred the mass to j (and thus, i does not include that mass in its own state $y_k[i]$), and at the same time, that mass is not received at node j, and therefore, not included in $y_k[j]$. Therefore, let us view this missing mass as being "buffered on" link (i,j) in a *virtual buffer*. The virtual buffer for each directed link (i,j) will be viewed as a *virtual node* in the network. Thus, when link (i,j) is unreliable, the mass is transferred from node i to "node" (i,j), instead of being transferred to node j. Note that when link (i,j) is unreliable, node j neither receives mass directly from node i, nor from the virtual buffer (i,j).

(P2) Link (i,j) is reliable in slot k: In this case, message $\mu_k[i]$ is received by node j. Thus, $\mu_k[i]$ contributes to $y_k[j]$. In addition, all the mass buffered in the virtual buffer (i,j) will also be received by node j, and this mass will also contribute to $y_k[j]$. We will say that buffer (i,j) "releases" its mass to node j.

We capture the above intuition by building an "augmented" network that contains all the nodes in $\mathcal{V}$, and also contains additional virtual nodes, each virtual node corresponding to the virtual buffer for a link in $\mathcal{E}$. Let us denote the augmented networks by $\mathcal{G}^a = (\mathcal{V}^a, \mathcal{E}^a)$ where $\mathcal{V}^a = \mathcal{V} \cup \mathcal{E}$ and

$$\mathcal{E}^a = \mathcal{E} \cup \{((i,j),j) \mid (i,j) \in \mathcal{E}\} \cup \{i,(i,j) \mid (i,j) \in \mathcal{E}\}.$$

In case (P2) above, the mass sent by node i, and the mass released from the virtual buffer (i,j), both

[6]In the more general case, node i may want to transfer different amounts of mass to different nodes in $\mathcal{O}_i$. In this case, node i may send (unreliable) unicast messages to these neighbors. The treatment in this case will be quite similar to the restricted case assumed in our discussion, except that node i will need to separately track mass transfers to each of its out-neighbors.

contribute to the new state $y_k[j]$ at node j. In particular, it will suffice for node j to only know the *sum* of the mass being sent by node i at step k and the mass being released (if any) from buffer (i,j) at step k. In reality, of course, there is no virtual buffer to hold the mass that has not been delivered yet. However, an equivalent mechanism can be implemented by introducing additional state at each node in $\mathcal{V}$, which exploits the above observation. This is what we explain in the next section.

B. Robust Ratio Consensus Algorithm

We will mitigate the shortcomings of Approach 2 described in Section II-C by changing our iterations to be tolerant to missing messages. The modified scheme has the following features:

- Instead of transmitting message $\mu_k[i] = y_{k-1}[i]/D_i$ at step k, each node i broadcasts at step k a message with value $\sum_{j=1}^{k} \mu_k[i]$, denoted as $\sigma_k[i]$. Thus, $\sigma_k[i]$ is the total mass that node i wants to transfer to each node in $\mathcal{O}_i$ through the first k steps.
- Each node i maintains, in addition to state variables $y_k[i]$ and $z_k[i]$, also a state variable $\rho_k[j,i]$ for each node $j \in \mathcal{I}_i$; $\rho_k[j,i]$ is the total mass that node i has received either directly from node j, or via virtual buffer (j,i), through step k.

The computation performed at node i at step $k \geq 1$ is as follows. Note that $\sigma_0[i] = 0$, $\forall i \in \mathcal{V}$ and $\rho_0[i,j] = 0,\ \forall (i,j) \in \mathcal{E}$.

$$\sigma_k[i] = \sigma_{k-1}[i] + y_{k-1}[i]/D_i, \tag{6}$$

$$\rho_k[j,i] = \begin{cases} \sigma_k[j], & \text{if } (j,i) \in \mathcal{E} \text{ and message } \sigma_k[j] \text{ is received by } i \text{ from } j \text{ at step } k, \\ \rho_{k-1}[j,i], & \text{if } (j,i) \in \mathcal{E} \text{ and no message is received by } i \text{ from } j \text{ at step } k, \end{cases} \tag{7}$$

$$y_k[i] = \sum_{j \in \mathcal{I}_i} (\rho_k[j,i] - \rho_{k-1}[j,i]). \tag{8}$$

When link $(j,i) \in \mathcal{E}$ is reliable, $\rho_k[j,i]$ becomes equal to $\sigma_k[j]$: this is reasonable, because i receives any new mass sent by j at step k, as well as any mass released by buffer (j,i) at step k. On the other hand, when link (j,i) is unreliable, then $\rho_k[j,i]$ remains unchanged from the previous iteration, since no mass is received from j (either directly or via virtual buffer (j,i)). It follows that, the total new mass received by node i at step k, either from node j directly or via buffer (j,i), is given by $\rho_k[j,i] - \rho_{k-1}[j,i]$, which explains (8).[7]

IV. Robust Algorithm Formulation as an Inhomogeneous Markov Chain

In this section, we reformulate each iteration performed by the robust algorithm as an inhomogeneous Markov chain whose transition matrix takes values from a finite set of matrices. We will also discuss some properties of these matrices, and analyze the behavior of their products, which helps in establishing the convergence of the robustified ratio consensus algorithm.

[7]As per the algorithm specified above, observe that the values of σ and ρ increase monotonically with time. This can be a concern for a large number of steps in practical implementations. However, this concern can be mitigated by "resetting" these values, e.g., via the exchange of additional information between neighbors (for instance, by piggybacking cumulative acknowledgements, which will be delivered whenever the links operate reliably).

A. Matrix Representation of Each Individual Iteration

The matrix representation is obtained by observing an equivalence between the iteration in (6)–(8), and an iterative algorithm (to be introduced soon) defined on the augmented network described in Section III-A. The vector state of the augmented network consists of $n = m+|\mathcal{E}|$ elements, corresponding to the mass held by each of the m nodes, and the mass held by each of the $|\mathcal{E}|$ virtual buffers: these n entities are represented by as many nodes in the augmented network.

With a slight abuse of notation, let us denote by y_k the state of the nodes in the augmented network $\mathcal{G}^a$. The vector y_k for $\mathcal{G}^a$ is an augmented version of y_k for $\mathcal{G}$. In addition to $y_k[i]$ for each $i \in \mathcal{V}$, the augmented y_k vector also includes elements $y_k[(i,j)]$ for each $(i,j) \in \mathcal{E}$, with $y_0[(i,j)] = 0$.[8] Due to the manner in which the $y_k[i]$'s are updated, $y_k[i],\ i \in \mathcal{V}$, are identical in the original network and the augmented network; therefore, we do not distinguish between them. We next translate the iterative algorithm in (6)–(8) into the matrix form

$$y_k = y_{k-1}M_k, \tag{9}$$

for appropriately row-stochastic matrices M_k (to be defined soon) that might vary as the algorithm progresses (but nevertheless take values from a finite set of possible matrices).

Let us define an indicator variable $X_k[j,i]$ for each link $(j,i) \in \mathcal{E}$ at each time step k as follows:

$$X_k[j,i] = \begin{cases} 1, & \text{if link } (j,i) \text{ is reliable at time step } k, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

We will now reformulate the iteration (6)–(8) and show how, in fact, it can be described in matrix form as shown in (9), where the matrix transition matrix M_k is a function of the indicator variables defined in (10). First, by using the indicator variables at time step k, as defined in (10), it follows from (6) that

$$\rho_k[j,i] = X_k[j,i]\sigma_k[j] + (1 - X_k[j,i])\rho_{k-1}[j,i]. \tag{11}$$

Now, for $k \geq 0$, define $\nu_k[j,i] = \sigma_k[j] - \rho_k[j,i]$ (thus $\nu_0[j,i] = 0$). Then, it follows from (6) and (11) that

$$\nu_k[j,i] = (1 - X_k[j,i]) \left(\frac{y_{k-1}[j]}{D_j} + \nu_{k-1}[j,i] \right), \qquad k \geq 1. \tag{12}$$

Also, from (6) and (11), it follows that (8) can be rewritten as

$$y_k[i] = \sum_{j \in \mathcal{I}_i} X_k[j,i] \left(\frac{y_{k-1}[j]}{D_j} + \nu_{k-1}[j,i] \right), \qquad k \geq 1. \tag{13}$$

At every instant k that the link (j,i) is not reliable, it is easy to see that the variable $\nu_k[j,i]$ increases by an amount equal to the amount that node j wished to send to node i, but i never received due to the link failure. Similarly, at every instant k that the link (j,i) is reliable, the variable $\nu_k[j,i]$ becomes

[8] Similarly, $z_0[(i,j)] = 0$.

zero and its value at $k-1$ is received by node i as can be seen in (13). Thus, from (12) and (13), we can think of the variable $\nu_k[j,i]$ as the state of a virtual node that buffers the mass that node i does not receive from node j every time the link (j,i) fails. It is important to note that the $\nu_k[j,i]$'s are virtual variables (no node in $\mathcal{V}$ computes ν_k) that just result from combining, as explained above, variables that the nodes in $\mathcal{V}$ compute. The reason for doing this is that the resulting model is equivalent to an inhomogeneous Markov chain. This can be easily seen by stacking up (13) for all nodes indexed in $\mathcal{V}$, i.e., the computing nodes, and (12) for all virtual buffers (j,i), with $(j,i) \in \mathcal{E}$, and rewriting the resulting expressions in matrix form, from where the expression in (9) results.

B. Structure and Properties of the Matrices M_k

Next, we discuss the sparsity structure of the M_k's and obtain their entries by inspection of (12) and (13). Additionally, we will explore some properties of the M_k's that will be helpful in the analysis conducted in Section V for characterizing the behavior of each of the individual iterations.

1) Structure of M_k: Let us first define the entries in row i of matrix M_k that corresponds to $i \in \mathcal{V}$. For $(i,j) \in \mathcal{E}$, there are two possibilities: $X_k[i,j]=0$ or $X_k[i,j]=1$. If $X_k[i,j]=0$, then the mass $\mu_k[i] = y_k[i]/D_i$ that node i wants to send to node j is added to the virtual buffer (i,j). Otherwise, no new mass from node i is added to buffer (i,j). Therefore,

$$M_k[i,(i,j)] = (1 - X_k[i,j])/D_i. \tag{14}$$

The above value is zero if link (i,j) is reliable at step k, and $1/D_i$ otherwise. Similarly, it follows that

$$M_k[i,j] = X_k[i,j]/D_i, \tag{15}$$

which is zero whenever link (i,j) is unreliable at step k, and $1/D_i$ otherwise. Observe that for each $j \in \mathcal{O}_i$,

$$M_k[i,j] + M_k[i,(i,j)] = 1/D_i, \tag{16}$$

with, in fact, one of the two quantities zero and the other equal to $1/D_i$. For $(i,j) \notin \mathcal{E}$, it naturally follows that $M_k[i,j]=0$. Similarly,

$$M_k[i,(s,r)] = 0, \qquad \text{whenever} \;\; i \neq s \;\; \text{and} \;\; (s,r) \in \mathcal{E}. \tag{17}$$

Since $|\mathcal{O}_i| = D_i$, all the elements in row i of matrix M_k add up to 1.

Now define row (i,j) of matrix M_k, which describes how the mass of the virtual buffer (i,j), for $(i,j) \in \mathcal{E}$, gets distributed. When link (i,j) works reliably at time step k (i.e., $X_k[i,j]=1$), all the mass buffered on link (i,j) is transferred to node j; otherwise, no mass is trasferred from buffer (i,j) to node j and the buffer retains all its previous mass and increases it by a quantity equal to the mass

that node i fail to send to node j. These conditions are captured by defining M_k entries as follows:

$$M_k[(i,j),j] = X_k[i,j], \tag{18}$$

$$M_k[(i,j),(i,j)] = 1 - X_k[i,j]. \tag{19}$$

Also, for obvious reasons,

$$M_k[(i,j),p] = 0, \quad \forall p \neq j, \quad p \in \mathcal{V}, \tag{20}$$

$$M_k[(i,j),(s,r)] = 0, \quad \forall (i,j) \neq (s,r), \quad (s,r) \in \mathcal{E}. \tag{21}$$

Clearly, all the entries of the row labeled (i,j) add up to 1, which results in M_k being a row stochastic matrix for all $k \geq 1$.

2) Properties of M_k: Let us denote the set of all possible instances (depending on the values of the indicator variables $X_k[i,j],\ (i,j) \in \mathcal{E},\ k \geq 1$) of matrix M_k as $\mathcal{M}$. The matrices in the set $\mathcal{M}$ have the following properties:

(M1) *The set $\mathcal{M}$ is finite.*
Each distinct matrix in $\mathcal{M}$ corresponds to different instantiations of the indicator variables defined in (10), resulting in exactly $2^{|\mathcal{E}|}$ distinct matrices in $\mathcal{M}$.

(M2) *Each matrix in $\mathcal{M}$ is a finite-dimensional square row stochastic matrix.*
The number of rows of each matrix $M_k \in \mathcal{M}$, as defined above, is $n = m + |\mathcal{E}|$, which is finite. Also, from (14)–(21), theses matrices are square row-stochastic matrices.

(M3) *Each positive element of any matrix in $\mathcal{M}$ is lower bounded by a positive constant.*
Let us denote this lower bound as c. Then, due to the manner in which matrices in $\mathcal{M}$ are constructed, we can define c to be the positive constant obtained as

$$c = \min_{i,j,M \ \mid M \in \mathcal{M}, M[i,j]>0} M[i,j].$$

(M4) The matrix $M_k,\ k \geq 0$, may be chosen to be any matrix $M \in \mathcal{M}$ with a non-zero probability. The choice of the transition matrix at each time step is independent and identically distributed (i.i.d.) due to the assumption that link failures are independent (between nodes and time steps).
Explanation: The probability distribution on $\mathcal{M}$ is a function of the probability distribution on the link reliability. In particular, if a certain $M \in \mathcal{M}$ is obtained when the links in $\mathcal{E}' \subseteq \mathcal{E}$ are reliable, and the remaining links are unreliable, then the probability that $M_k = M$ is equal to

$$\Pi_{(i,j)\in\mathcal{E}'}\ q_{ij}\ \Pi_{(i,j)\in\mathcal{E}-\mathcal{E}'}\ (1-q_{ij}). \tag{22}$$

(M5) *For each $i \in \mathcal{V}$, there exists a finite positive integer l_i such that it is possible to find l_i matrices in $\mathcal{M}$ (possibly with repetition) such that their product (in a chosen order) is a row stochastic matrix with the column that corresponds to node i containing strictly positive entries.*
This property states that, for each $i \in \mathcal{V}$, there exists a matrix T_i^*, obtained as the product of l_i

matrices in $\mathcal{M}$ that has the following properties:

$$T_i^*[j,i] > 0, \qquad \forall\ j \in \mathcal{V}, \tag{23}$$

$$T_i^*[(j_1,j_2),i] > 0, \qquad \forall\ (j_1,j_2) \in \mathcal{E}. \tag{24}$$

This follows from the fact that the underlying graph $\mathcal{G}^a$ is strongly connected (in fact, it can be easily shown that $l_i \leq m$). To simplify the presentation below, and due to the self-loops, we can take l_i to be equal to a constant l, for all $i \in \mathcal{V}$. However, it should be easy to see that the arguments below can be generalized to the case when the l_i's may be different.

We can also show that under our assumption for link failures, there exists a single matrix, say T^*, which simultaneously satisfies the conditions in (23)–(24) for all $i \in \mathcal{V}$. When all the links in the network operate reliably, network $\mathcal{G}(\mathcal{V},\mathcal{E})$ is strongly connected (by assumption). Since $\mathcal{G}$ is strongly connected, there is a directed path between every pair of nodes i and j, i.e., $i,j \in \mathcal{V}$. In the augmented network $\mathcal{G}^a$, for each $(i,j) \in \mathcal{E}$, there is a link from node i to node (i,j), and a link from node (i,j) to node j. Thus, it should be clear that the augmented network $\mathcal{G}^a$ is strongly connected as well. Consider a spanning tree rooted at node 1, such that all the nodes in $V = \mathcal{V} \cup \mathcal{E}$ have a directed path towards node 1, and also a spanning tree in which all the nodes have directed paths *from* node 1. Choose that matrix, say $M^* \in \mathcal{M}$, which corresponds to all the links on these two spanning trees, as well as self-loops at all $i \in \mathcal{V}$, being reliable. If the total number of links that are thus reliable is e, it should be obvious that $(M^*)^e$ will contain only non-zero entries in columns corresponding to $i \in \mathcal{V}$. Thus, l defined above may be chosen as e. There are several other ways of constructing T^*, some of which may result in a smaller value of l.

V. Ergodicity Analysis of Products of Matrices M_k

We will next analyze the ergodic behavior of the *forward product* $T_k = M_1 M_2 \ldots M_k = \Pi_{j=1}^k M_j$, where $M_j \in \mathcal{M},\ \forall j = 1,2,\ldots,k$. Informally defined, weak ergodicity of T_k obtains if the rows of T_k tend to equalize as $k \to \infty$. In this work, we focus on the weak ergodicity notion, and establish probabilistic statements pertaining the ergodic behavior of T_k. The analysis builds upon a large body of literature on products of nonnegative matrices (see, e.g., [**?**] for a comprehensive account). First, we introduce the basic toolkit adopted from [**?**], [**?**], [**?**], and then use it to analyze the ergodicity of T_k.

A. Some Results Pertaining Coefficients of Ergodicity

Informally speaking, a coefficient of ergodicity of a matrix A characterizes how different two rows of A are. For a row stochastic matrix A, proper[9] coefficients of ergodicity $\delta(A)$ and $\lambda(A)$ are defined

[9]Any scalar function $\tau(\cdot)$ continuous on the set of $n \times n$ row stochastic matrices, which satisfies $0 \leq \tau(A) \leq 1$, is said to be a proper coefficient of ergodicity if $\tau(A) = 0$ if and only if $A = e^T v$, where e is the all-ones row vector, and $v \geq 0$ is such that $ve^T = 1$ [**?**].

as:

$$\delta(A) := \max_{j} \ \max_{i_1,i_2} \ |A[i_1,j] - A[i_2,j]|, \tag{25}$$

$$\lambda(A) := 1 - \min_{i_1,i_2} \sum_{j} \min(A[i_1,j], A[i_2,j]). \tag{26}$$

It is easy to see that $0 \leq \delta(A) \leq 1$ and $0 \leq \lambda(A) \leq 1$, and that the rows are identical if and only if $\delta(A) = 0$. Additionally, $\lambda(A) = 0$ if and only if $\delta(A) = 0$.

The next result establishes a relation between the coefficient of ergodicity $\delta(\cdot)$ of a product of row stochastic matrices, and the coefficients of ergodicity $\lambda(\cdot)$ of the individual matrices defining the product. This result will be used in the proof of Lemma 2. It was established in [**?**] and also follows from the more general statement of Theorem 4.8 in [**?**].

Proposition 1: For any p square row stochastic matrices $A_1, A_2, \ldots A_{p-1}, A_p$,

$$\delta(A_1 A_2 \cdots A_{p-1} A_p) \ \leq \ \left(\Pi_{i=1}^{p-1} \lambda(A_i)\right) \delta(A_p) \ \leq \ \Pi_{i=1}^{p} \ \lambda(A_i). \tag{27}$$

The result in (27) is particularly useful to infer ergodicity of a product of matrices from the ergodic properties of the individual matrices in the product. For example, if $\lambda(A_i)$ is less than 1 for all i, then $\delta(A_1 A_2 \cdots A_{p-1} A_p)$ will tend to zero as $p \to \infty$. We will next introduce an important class of matrices for which $\lambda(\cdot) < 1$.

Definition 1: A matrix A is said to be a *scrambling* matrix, if $\lambda(A) < 1$ [**?**].

In a scrambling matrix A, since $\lambda(A) < 1$, for each pair of rows i_1 and i_2, there exists a column j (which may depend on i_1 and i_2) such that $A[i_1,j] > 0$ and $A[i_2,j] > 0$, and vice-versa. As a special case, if any one column of a row stochastic matrix A contains only non-zero entries, then A must be scrambling.

B. Ergodicity Analysis of Iterations of the Robust Algorithm

We next analyze the ergodic properties of the products of matrices that result from each of the iterations comprising our robust algorithm. Let us focus on just one of the iterations, say y_k, as the treatment of the z_k iteration is identical. As described in Section IV, the progress of the y_k iteration can be recast as an inhomogeneous Markov chain

$$y_k = y_{k-1} M_k, \ k \geq 1, \tag{28}$$

where $M_k \in \mathcal{M}, \ \forall k$. As already discussed, the sequence of M_k's that will govern the progress of y_k is determined by communication link availability. (28). Defining $T_k = \Pi_{j=1}^{k} M_j$, we obtain:

$$\begin{aligned} y_k &= y_0 M_1 M_2 \cdots M_k \\ &= y_0 \Pi_{j=1}^{k} M_j = y_0 T_k, \ k \geq 1. \end{aligned} \tag{29}$$

By convention, $\Pi_{i=k}^{0} M_i = I$ for any $k \geq 1$ (I denotes the $n \times n$ identity matrix).

Recalling the constant l defined in (M5), define W_k as follows,

$$W_k \;=\; \Pi_{j=(k-1)l+1}^{kl} M_j,\ k \geq 1,\ M_j \in \mathcal{M}, \tag{30}$$

from where it follows that

$$T_{lk} \;=\; \Pi_{j=1}^{k}\ W_k,\ k \geq 1. \tag{31}$$

Observe that the set of time steps "covered" by W_i and W_j, $i \neq j$, are non-overlapping. It is also important to note for subsequent analysis that, since the M_k's are row stochastic matrices and the product of any number of row stochastic matrices is row stochastic, all the W_k's and T_k's are also row stochastic matrices.

Lemma 2 will establish that as the number of iteration steps goes to infinity, the rows of the matrix T_k tend to equalize. For proving Lemma 2, we need the result in Lemma 1 stated below, which establishes that there exists a nonzero probability of choosing matrices in $\mathcal{M}$ such that the W_k's as defined in (30) are scrambling.

Lemma 1: There exist constants $w > 0$ and $d < 1$ such that, with probability equal to w, $\lambda(W_k) \leq d$ for $k \geq 1$, independently for different k.

Proof: Each W_k matrix is a product of l matrices from the set $\mathcal{M}$. The choice of the M_k's that form W_i and W_j is independent for $i \neq j$, since W_i and W_j "cover" non-overlapping intervals of time. Thus, under the $i.i.d.$ assumption for selection of matrices from $\mathcal{M}$ $\big($property (M4)$\big)$, and property (M5), it follows that, with a non-zero probability (independently for W_k and $W_{k'}$ for $k \neq k'$), matrix W_k for each k is scrambling. Let us denote by w the probability that W_k is scrambling.

Let us define $\mathcal{W}$ as the set of all possible instances of W_k that are scrambling. The set $\mathcal{W}$ is finite because the set $\mathcal{M}$ is finite, and $\mathcal{W}$ is also non-empty (this follows from the discussion of (M5)). Let us define d as the tight upper bound on $\lambda(W)$, for $W \in \mathcal{W}$, i.e.,

$$d \;\equiv\; \max_{W \in \mathcal{W}}\ \lambda(W). \tag{32}$$

Recall that $\lambda(A)$ for any scrambling matrix A is strictly less than 1. Since $\mathcal{W}$ is non-empty and finite, and contains only scrambling matrices, it follows that

$$d < 1. \tag{33}$$

■

Lemma 2: There exist constants α and β ($0 < \alpha < 1$, $0 \leq \beta < 1$) such that, with probability greater than $(1 - \alpha^k)$, $\delta(T_k) \leq \beta^k$ for $k \geq 8l/w$.

Proof: Let $k^* = \left\lfloor \frac{k}{l} \right\rfloor$ and $\Delta = k - lk^*$. Thus, $0 \leq \Delta < l$. From (29) through (31), observe that

$$T_k = T_{lk^*+\Delta} = T_{lk^*}\ \Pi_{j=1}^{\Delta}\ M_{lk^*+j},$$

where T_{lk^*} is the product of k^* of W_j matrices, where $1 \leq j \leq k^*$. As per Lemma 1, for each W_j,

the probability that $\lambda(W_j) \leq d < 1$ is equal to w. Thus, the expected number of scrambling matrices among the k^* matrices is wk^*. Denote by S the actual number of scrambling W_j matrices among the k^* matrices. Then the Chernoff lower tail bound tells us that, for any $\phi > 0$,

$$\Pr\{S < (1-\phi)E(S)\} < e^{-E(S)\phi^2/2} \tag{34}$$

$$\Rightarrow \qquad \Pr\{S < (1-\phi)(wk^*)\} < e^{-(wk^*)\phi^2/2}. \tag{35}$$

Let us choose $\phi = \frac{1}{2}$. Then,

$$\Pr\{S < (wk^*)/2\} < e^{-wk^*/8} \tag{36}$$

$$\Rightarrow \qquad \Pr\{S \geq wk^*/2\} > 1 - e^{-wk^*/8}. \tag{37}$$

Thus, at least $\lfloor wk^*/2 \rfloor$ of the W matrices from the k^* matrices forming T_{lk^*} are scrambling (each with λ value $\leq d$, by Lemma 1) with probability greater than $1 - e^{-wk^*/8}$. Proposition 1 then implies that

$$\delta(T_k) = \delta(T_{lk^*+\Delta}) = \delta\left(\left(\Pi_{i=1}^{k^*} W_i\right)\left(\Pi_{i=1}^{\Delta} M_{lk^*+i}\right)\right) \leq \left(\Pi_{i=1}^{k^*}\lambda(W_i)\right)\left(\Pi_{i=1}^{\Delta}\lambda(M_{lk^*+i})\right)$$

Since at least $\lfloor wk^*/2 \rfloor$ of the W_i's have $\lambda(W_i) \leq d$ with probability greater than $1 - e^{-wk^*/8}$, and $\lambda(M_j) \leq 1, \ \forall j$, it follows that

$$\delta(T_k) \leq d^{\lfloor wk^*/2 \rfloor} \tag{38}$$

with probability exceeding

$$1 - e^{-wk^*/8}. \tag{39}$$

Let us define $\alpha = e^{-\frac{w}{16l}}$ and $\beta = d^{\frac{w}{8l}}$. Now, if $k \geq 8l/w$, then if follows that $k \geq 2l$, and

$$k^* \ = \ \left\lfloor \frac{k}{l} \right\rfloor \geq \frac{k}{2l} \tag{40}$$

$$\Rightarrow \qquad \left\lfloor \frac{wk^*}{2} \right\rfloor \geq \left\lfloor \frac{wk}{4l} \right\rfloor \tag{41}$$

$$\Rightarrow \qquad \left\lfloor \frac{wk^*}{2} \right\rfloor \geq \frac{wk}{8l} \tag{42}$$

$$\Rightarrow \qquad d^{\left\lfloor \frac{wk^*}{2} \right\rfloor} \leq d^{\frac{wk}{8l}} \qquad \text{(because } \ 0 \leq d < 1) \tag{43}$$

$$\Rightarrow \qquad d^{\left\lfloor \frac{wk^*}{2} \right\rfloor} \leq \beta^k. \tag{44}$$

Similarly, if $k \geq 8l/w$, it follows that

$$k^* = \left\lfloor \frac{k}{l} \right\rfloor \geq \frac{k}{2l} \tag{45}$$

$$\Rightarrow \quad e^{-wk^*/8} \leq e^{-\frac{wk}{16l}} = \alpha^k \tag{46}$$

$$\Rightarrow \quad 1 - e^{-wk^*/8} \geq 1 - \alpha^k. \tag{47}$$

By substituting (44) and (47) into (38) and (39), respectively, the result follows. ■

Note that α and β in Lemma 2 are independent of time. The threshold on k for which Lemma 2 holds, namely $k \geq 8l/w$, can be improved by using better bounds in (40) and (45). Knowing a smaller threshold on k for which Lemma 2 holds can be beneficial in a practical implementation. In the above derivation for Lemma 2, we chose a somewhat loose threshold in order to maintain a simpler form for the probability expression (namely, $1 - \alpha^k$) and also a simpler expression for the bound on $\delta(T_k)$ (namely, β^k).

Lemma 3: $\delta(T_k)$ converges almost surely to 0.

Proof: For $k \geq 8l/w$, from Lemma 2, we have that $\Pr\{\delta(T_k) > \beta^k\} \leq \alpha^k$, $0 < \alpha < 1$, $0 \leq \beta < 1$. Then, it is easy to see that $\sum_k \Pr\{\delta(T_k) > \beta^k\} \leq 8l/w + \sum_k \alpha^k < \infty$. Then, by the first Borel-Cantelli lemma, $\Pr\{\text{the event that } \delta(T_k) > \beta^k \text{ occurs infinitely often}\} = 0$. Therefore, $\delta(T_k)$ converges to 0 almost surely. ■

VI. Convergence Analysis of Robustified Ratio Consensus Algorithm

The analysis below shows that the ratio algorithm achieves asymptotic consensus correctly in the presence of the virtual nodes, even if diagonals of the transition matrices (M_k's) are not always strictly positive. A key consequence is that the value of $z_k[i]$ is not necessarily greater from zero (at least not for all k), which creates some difficulty when calculating the ratio $y_k[i]/z_k[i]$. As noted earlier, aside from these differences, our algorithm is similar to that analyzed in [**?**]. Our proof has some similarities to the proof in [**?**], with the differences accounting for our relaxed assumptions.

By defining z_k in an analogous way as we defined state y_k in Section IV, the robustified version of the ratio consensus algorithm in (3)–(4) can be described in matrix form as

$$y_k = y_{k-1}\, M_k, \qquad k \geq 1, \tag{48}$$

$$z_k = z_{k-1}\, M_k, \qquad k \geq 1, \tag{49}$$

where $M_k \in \mathcal{M},\ k \geq 1,\ y_0[i] \geq 0,\ \forall i,\ z_0[i] \geq 0,\ \forall i$, and $\sum_j z_0[j] > 0$, and $y_0[(i,j)] = z_0[(i,j)] = 0,\ \forall (i,j) \in \mathcal{E}$. The same matrix M_k is used at step k of the iterations in (48) and (49), however, M_k may vary over k. Recall that y_k and z_k in (48) and (49) have n elements, but only the first m elements correspond to computing nodes in the augmented network $\mathcal{G}^a$; the remaining entries in y_k and z_k correspond to virtual buffers.

The goal of the algorithm is for each computing node to obtain a consensus value defined as

$$\pi^* = \frac{\sum_j \ y_0[j]}{\sum_j \ z_0[j]}. \tag{50}$$

To achieve this goal, each node $i \in \mathcal{V}$ calculates

$$\pi_k[i] \ = \ \frac{y_k[i]}{z_k[i]}, \tag{51}$$

whenever the denominator is large enough, i.e., whenever

$$z_k[i] \ \geq \ \mu, \tag{52}$$

for some constant $\mu > 0$ *to be defined later*. We will show that, for each $i = 1, 2, \ldots, m$, the sequence $\pi_k[i]$ thus calculated asymptotically converges to the desired consensus value π^*. To show this, we first establish that (52) occurs infinitely often, thus computing nodes can calculate the ratio in (51) infinitely often. Then, we will show that as k goes to infinity, the sequence of ratio computations in (51) will converge to the value in (50).

The convergence when $\sum_j y_0[j] = 0$ can be shown trivially. So let us now consider the case when $\sum_j y_0[j] > 0$, and define new state variables $\tilde{y}_k$ and $\tilde{z}_k$ for $k \geq 0$ as follows:

$$\tilde{y}_k[i] \ = \ \frac{y_k[i]}{\sum_j y_0[j]}, \ \forall i, \tag{53}$$

$$\tilde{z}_k[i] \ = \ \frac{z_k[i]}{\sum_j z_0[j]}, \ \forall i. \tag{54}$$

Thus, $\tilde{y}_0$ and $\tilde{z}_0$ are defined by normalizing y_k and z_k. It follows that $\tilde{y}_0$ and $\tilde{z}_0$ are stochastic row vectors. Also, since our transition matrices are row stochastic, it follows that $\tilde{y}_k$ and $\tilde{z}_k$ are also stochastic vectors for all $k \geq 0$.

We assume that each node knows a lower bound on $\sum_j z_0[j]$, denoted by μ_z. In typical scenarios, for all $i \in \mathcal{V}$, $z_0[i]$ will be positive, and, node $i \in \mathcal{V}$ can use $z_0[i]$ as a non-zero lower bound on $\sum_j z_0[j]$ (thus, in general, the lower bound used by different nodes may not be identical). We also assume an upper bound, say μ_y, on $\sum_j y_0[j]$.

Let us define

$$\mu = \frac{\mu_z \ c^l}{n}. \tag{55}$$

As time progresses, each node $i \in \mathcal{V}$ will calculate a new estimate of the consensus value whenever $z_k[i] \geq \mu$. The next lemma establishes that nodes will can carry out this calculation infinitely often.

Lemma 4: Let $\mathcal{T}_i = \{\tau_i^1, \tau_i^2, \cdots\}$ denote the sequence of time instances when node i updates its estimate of the consensus using (51), and obeying (52), where $\tau_i^j < \tau_i^{j+1}$, $j \geq 1$. The sequence $\mathcal{T}_i$ contains infinitely many elements with probability 1.

Proof: To prove the lemma, it will suffice to prove that for infinitely many values of k, $z_k[i] > \mu$, with probability 1. Assumptions (M1)-(M5) imply that each matrix $W_j,\ j \geq 1$ (defined in (30)) contains a strictly positive column corresponding to index $i \in \mathcal{V}$ with a non-zero probability, say $\gamma_i > 0$. Also, the choice of W_{k_1} and W_{k_2} is independent of each other for $k_1 \neq k_2$. Therefore, the second Borel-Cantelli lemma implies that, with probability 1, for infinitely many values of j, W_j will have the i-th column strictly positive. Since the non-zero elements of each matrix in $\mathcal{M}$ are all greater or equal to c, $c > 0$ (by property M3), and since W_j is a product of l matrices in $\mathcal{M}$, it follows that all the non-zero elements of each W_j must be lower bounded by c^l.

Consider only those $j \geq 1$ for which W_j contains positive i-th column. As noted above, there are infinitely many such j values. Now,

$$\tilde{z}_{jl} \ = \ \tilde{z}_{(j-1)l}\ W_j.$$

As noted above, $\tilde{z}_k$ is a stochastic vector. Thus, for any $k \geq 0$,

$$\sum_i \tilde{z}_k[i] \ = \ 1 \tag{56}$$

and, at least one of the elements of $\tilde{z}_{(j-1)l}$ must be greater or equal to $1/n$. Also, all the elements in columns of W_j indexed by $i \in \mathcal{V}$ are lower bounded by c^l (recall that we are now only considering those j for which the i-th column of W_j is positive). This implies that,

$$\tilde{z}_{jl}[i] \ \geq \ c^l/n \tag{57}$$

$$\Rightarrow \qquad z_{jl}[i] \ \geq \ \left(\sum_j z_0[j]\right)\ c^l/n \tag{58}$$

$$\Rightarrow \qquad z_{jl}[i] \ \geq \ \mu_z\ c^l/n \tag{59}$$

$$\Rightarrow \qquad z_{jl}[i] \ \geq \ \mu, \quad \forall i \in \mathcal{V} \qquad \text{(by (55))} \tag{60}$$

Since infinitely many W_j's will contain a positive i-th column (with probability 1), (60) holds for infinitely many j with probability 1. Therefore, with probability 1, the set $\mathcal{T}_i = \{\tau_i^1, \tau_i^2, \cdots\}$ contains infinitely many elements, for all $i \in \mathcal{V}$. ■

Finally, the next theorem shows that the ratio consensus algorithm will converge to the consensus value defined in (50).

Theorem 1: Let $\pi_i[t]$ denote node i's estimate of the consensus value calculated at time τ_i^t. For each node $i \in \mathcal{V}$, with probability 1, the estimate $\pi_i[t]$ converges to

$$\pi^* = \frac{\sum_j y_j[0]}{\sum_j z_j[0]}.$$

Proof: Note that the transition matrices $M_k,\ k \geq 1$, are randomly drawn from a certain distribution. By an "execution" of the algorithm, we will mean a particular instance of the M_k sequence. Thus, the

distribution on M_k's results in a distribution on the executions. Lemma 3 implies that,

$$\Pr\left\{\lim_{k\to\infty} \delta(T_k) = 0\right\} = 1.$$

Together, Lemmas 3 and 4 imply that, with probability 1, for a chosen execution, (i) for any $\psi > 0$, there exists a finite k_ψ such that, for all $k \geq k_\psi$, $\delta(T_k) < \psi$, and (ii) there exist infinitely many values of $k \geq k_\psi$ such that $z_k[i] \geq \mu$ (i.e., $k \in \mathcal{T}_i$ for the chosen execution).

Consider any $k \geq k_\psi$ such that $z_k[i] > \mu$. Since $\delta(T_k) \leq \psi$, the rows of matrix T_k are "within ψ" of each other. Observe that $\tilde{y}_k$ is obtained as the product of stochastic row vector $\tilde{y}_0$ and T_k. Thus, $\tilde{y}_k$ is in the convex hull of the rows of T_k. Similarly $\tilde{z}_k$ is in the convex hull of the rows of T_k. Therefore, the j-th elements of $\tilde{y}_k$ and $\tilde{z}_k$ are within ψ of each other, for all j. Therefore,

$$\begin{aligned}
& | \ \tilde{y}_k[i] - \tilde{z}_k[i] \ | \ \leq \ \psi && (61)\\
\Rightarrow & \left| \ \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} - 1 \ \right| \ \leq \ \frac{\psi}{\tilde{z}_k[i]} && (62)\\
\Rightarrow & \left| \ \frac{y_k[i]}{z_k[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \ \right| \ \leq \ \frac{\psi \ \sum_j y_0[j]}{z_k[i]} \quad \text{(by (53) and (54))} && (63)\\
\Rightarrow & \left| \ \frac{y_k[i]}{z_k[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \ \right| \ \leq \ \frac{\psi \ \mu_y}{z_k[i]} \quad \text{(because } \textstyle\sum_j y_0[j] \leq \mu_y\text{)} && (64)\\
\Rightarrow & \left| \ \frac{y_k[i]}{z_k[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \ \right| \ \leq \ \frac{\psi \ \mu_y}{\mu}. && (65)
\end{aligned}$$

Now, given any $\epsilon > 0$, let us choose $\psi = \epsilon\mu/\mu_y$. Then (65) implies that

$$\left| \ \frac{y_k[i]}{z_k[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \ \right| \leq \epsilon$$

whenever $k \geq k_\psi$ and $k \in \mathcal{T}_i$. Therefore, in the limit, $\frac{y_k[i]}{z_k[i]}$ for $k \in \mathcal{T}_i$ converges to $\frac{\sum_j y_0[j]}{\sum_j z_0[j]}$. This result holds with probability 1, since conditions (i) and (ii) stated above hold with probability 1. ■

The result above can be strengthened by proving convergence of the algorithm even if each node $i \in \mathcal{V}$ updates its estimate whenever $z_k[i] > 0$ (not necessarily $\geq \mu$). To prove the convergence in this case, the argument is similar to that in Theorem 1, with two modifications:

- Lemma 4 needs to be strengthened by observing that there exist infinitely many time instants at which $z_k[i] > \mu$ simultaneously for all $i \in \mathcal{V}$. This is true due to the existence of a matrix T^* (as seen in the discussion of (M5)) that contains positive columns corresponding to all $i \in \mathcal{V}$.
- Using the above observation, and the argument in the proof of Theorem 1, it then follows that, with probability 1, for any ψ, there exists a finite k_ψ such that $\delta(T_k) < \psi$ whenever $k \geq k_\psi$. As before, defining $\psi = \epsilon\mu/\mu_y$, it can be shown that for any ϵ, *there exists* a $k_\epsilon \geq k_\psi$ such that the

following inequality holds for all $i \in \mathcal{V}$ *simultaneously*.

$$\left| \frac{y_{k_\epsilon}[i]}{z_{k_\epsilon}[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \right| \leq \epsilon \quad \forall i \in \mathcal{V} \tag{66}$$

$$\Rightarrow \quad \frac{\sum_j y_0[j]}{\sum_j z_0[j]} - \epsilon \leq \frac{y_{k_\epsilon}[i]}{z_{k_\epsilon}[i]} \leq \frac{\sum_j y_0[j]}{\sum_j z_0[j]} + \epsilon \quad \forall i \in \mathcal{V} \tag{67}$$

Naturally, $z_{k_\epsilon}[i] \neq 0$, $\forall i \in \mathcal{V}$. It is now easy to argue that the above inequality will continue to hold for all $k > k_\epsilon$ and each $i \in \mathcal{V}$ whenever $z_i[k] > 0$. To see this, observe that, for $k > k_\epsilon$,

$$y_k = y_{k_\epsilon} \ \Pi_{k_\epsilon+1}^{k} \ M_k$$

Define $P = \Pi_{k_\epsilon+1}^{k} \ M_k$ and $\psi = \epsilon\mu/\mu_y$. Then, we have that, whenever $z_k[i] > 0$ for $k > k_\epsilon$,

$$\frac{y_k[i]}{z_k[i]} = \frac{\sum_{j=1}^n y_{k_\epsilon}[j] \ P[j,i]}{\sum_{j=1}^n z_{k_\epsilon}[j] \ P[j,i]} \tag{68}$$

$$= \frac{\sum_{j=1, P[j,i]\neq 0}^n y_{k_\epsilon}[j] \ P[j,i]}{\sum_{j=1, P[j,i]\neq 0}^n z_{k_\epsilon}[j] \ P[j,i]} \quad \text{(summation over non-zero } P[j,i] \text{ terms)}$$

$$\Rightarrow \min_{j, P[j,i]>0} \frac{y_{k_\epsilon}[j]}{z_{k_\epsilon}[j]} \leq \frac{y_k[i]}{z_k[i]} \leq \max_{j, P[j,i]>0} \frac{y_{k_\epsilon}[j]}{z_{k_\epsilon}[j]} \tag{69}$$

$$\Rightarrow \frac{\sum_j y_0[j]}{\sum_j z_0[j]} - \epsilon \leq \frac{y_k[i]}{z_k[i]} \leq \frac{\sum_j y_0[j]}{\sum_j z_0[j]} + \epsilon \quad \text{from (67)} \tag{70}$$

$$\Rightarrow \left| \frac{y_k[i]}{z_k[i]} - \frac{\sum_j y_0[j]}{\sum_j z_0[j]} \right| \leq \epsilon \quad \text{for all } i \in \mathcal{V} \text{ and } k \geq k_\epsilon \tag{71}$$

This proves the convergence of the algorithm in the limit. Recall that for this convergence it suffices if each node updates its estimate of the consensus whenever its z value is positive. (69) follows from the observation that $\frac{\sum_j a[j]u[j]}{\sum_j b[j]u[j]} = \sum_j \left[\frac{a[j]}{b[j]} \ \frac{b[j]u[j]}{\sum_k b[k]u[k]} \right]$ is a weighted average of $\frac{a[j]}{b[j]}$, and therefore, lower bounded by $\min_j \frac{a[j]}{b[j]}$ and upper bounded by $\max_j \frac{a[j]}{b[j]}$.

VII. Concluding Remarks and Future Work

Although our analysis above is motivated by wireless environments wherein transmissions may not succeed, the analysis is more general. In particular, it applies to other situations in which properties (M1)–(M5) are true. Indeed, property (M4) by itself is not as important as its consequence that a given W_k matrix has non-zero columns indexed by $i \in \mathcal{V}$.

A particular application of the above analysis is in the case when messages may be delayed. As discussed previously, mass is transfered by any node to its neighbors by means of messages. Since these messages may be delayed, a message sent on link (i, j) in slot k may be received by node j in a later slot. Let us denote by $V_k[i]$ the set of messages received by node i at step k. It is possible for $V_k[i]$ to contain multiple messages from the same node. Note that $V_k[i]$ may contain a message sent by

node i to itself as well. Let us define the iteration for y_k as follows:

$$y_k[i] \quad = \quad \sum_{v \in V_k[i]} v. \tag{72}$$

The iteration for z_k can be defined analogously. Our robust consensus algorithm essentially implements the above iteration, allowing for delays in delivery of mass on any link (i, j) (caused by link failures). However, in effect, the robust algorithm also ensures FIFO (first-in-first-out) delivery, as follows. In slot k, if node i receives mass sent by node $j \in \mathcal{I}_i$ in slot $s,\ s \leq k$, then mass sent by node j in slots strictly smaller than s is either received previously, or will be received in slot k.

The virtual buffer mechanism essentially models asynchronous communication, wherein the messages between any pair of nodes in the network may require arbitrary delay, governed by some distribution. It is not difficult to see that the iterative algorithm (72) should be able to achieve consensus correctly even under other distributions on message delays, with possible correlation between the delays. In fact, it is also possible to tolerate non-FIFO (or out-of-order) message delivery provided that the delay distribution satisfies some reasonable constraints. Delay of up to B slots on a certain link $(i, j) \in \mathcal{E}$ can be modeled using a single chain of B virtual nodes, with links from node i to every virtual nodes, and link from the last of the B nodes to node j—in this setting, depending on the delay incurred by a packet, appropriate link from node i to one of the virtual node on the delay chain (or to j, if delay is 0) is used.

Note that while we made certain assumptions regarding link failures, the analysis relies primarily on two implications of these assumptions, namely (i) the rows of the transition matrix T_k become close to identical as k increases, and (ii) $z_k[i]$ is bounded away from 0 for each i infinitely often. When these implications are true, similar convergence results may hold in other environments.