Robust Average Consensus over Packet Dropping Links: Analysis via Coefficients of Ergodicity

Nitin H. Vaidya, Christoforos N. Hadjicostis, and Alejandro D. Domínguez-García

Abstract—We consider a networked system in which each component (node) iteratively exchanges information with its neighbors according to an arbitrary, possibly directed topology. Based on an iterative exchange of (local and possibly directed) information, we develop an average-consensus distributed algorithm that is robust to unreliable (packet-dropping) communication links. By introducing virtual nodes, we show that the execution of the proposed algorithm is mathematically equivalent to a finite inhomogenous Markov chain. Then, by using coefficients of ergodicity, we can prove convergence of the robust distributed algorithm to the exact average, in the presence of packet drops and under a very broad set of conditions.

I. INTRODUCTION

We consider a system in which each node exchanges information with its neighbors to compute, in a distributed fashion, the average of the nodes’ initial values. The communication links can be asymmetric (i.e., node j might be able to send information to node i, but not necessarily vice-versa). The links may also be lossy or unreliable. The behavior of unreliable links may be modeled using two different models:

1. Common knowledge: This models assumes that if a message sent over link (i, j) from node i to node j is lost due to link unreliability, then nodes i and j are both guaranteed to be immediately aware of the message loss. Such assumptions have been used in prior work (e.g., [1]).

2. Incomplete knowledge: This model assumes that if a message loss occurs over link (i, j), then node i does not immediately become aware of the message loss. We assume this model in our work. Such a scenario occurs in wireless networks when an unreliable “broadcast” is performed as a way of delivering an identical message to all neighbors. Some receivers will not receive the message due to transmission errors; however, the sender does not know which receivers lost the message. Acknowledgment mechanisms can be incorporated to improve reliability; however, this may not be feasible if communication links are asymmetric.

Consensus over unreliable links has received attention recently [2], [3], [4]. The work in [2] assumes that the graph describing the communication network is undirected and, when a communication link fails, it affects communication in both directions, but nodes can detect it and compensate for it. The work in [3] does not require the graph describing the communication network to be undirected and proposes two compensation methods to account for communication link failures; however, the value to which nodes reach consensus is not necessarily the average. The authors in [4] propose a strategy that corrects the errors in the state iteratively calculated by each node by incorporating corrective iterations; they also use retransmissions as a way to reduce the detrimental impact of link unreliability.

II. PRELIMINARIES

A. Network Model

The system is synchronous, and consists of a network of m nodes, \( V = \{1, 2, \ldots, m\} \), 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 on the average of these initial values, specifically, \( \sum_{i=1}^{m} V_i \). A directed link \((j, i)\) is said to “exist” if transmissions from node j may possibly (but not necessarily always) be received by node i. Let \( E \) denote the set of all directed links that exist in the network. Note that \((i, i) \in E, \forall i \in V\). Then, graph \( G = (V, E) \) represents the network connectivity. Define \( I_i = \{j \mid (j, i) \in E \} \) and \( O_i = \{j \mid (i, j) \in E \} \). The outdegree of node i, denoted as \( D_i \), is the size of set \( O_i \), thus, \( D_i = |O_i| \). Note that \( i \in I_i \) and \( i \in O_i, \forall i \in V \). We assume that the graph \( G = (V, E) \) is strongly connected. To make the discussion precise, we assume that a link \((i, j)\) exists (i.e., \((i, j) \in E\)) 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.

B. Ratio-Consensus Algorithm

Next, we summarize a consensus algorithm, which is similar to the push-sum algorithm in [5] and the weighted consensus algorithm in [6]. We rename the algorithm as “ratio consensus” since its output is obtained as the ratio of the state of two concurrent iterative computations. The ratio consensus algorithm is designed to perform correctly when all the links are perfectly reliable in each time slot.

Each node \( i \) maintains at iteration \( k \) state variables \( y_k[i] \) and \( z_k[i] \). At each time step \( k \), each node \( i \) updates its state variable as follows:

\[
\begin{align*}
\hat{y}_k[i] &= \sum_{j \in I_i} y_{k-1}[j] / D_j, \quad k \geq 1, \\
\hat{z}_k[i] &= \sum_{j \in I_i} z_{k-1}[j] / D_j, \quad k \geq 1,
\end{align*}
\]
where $y_0[j] = V_j$ and $z_0[j] = 1$, for $j \in V$. (It is also possible to use this structure to compute an arbitrary weighted average of the inputs.)

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 the nodes in $O_i$, and awaits reception of a similar message from each node in $I_i$. Provided that all the links are always reliable, each transmission by each node $j$ will be received by all the nodes in $O_j$. When node $i$ has received, from each node $j \in I_i$, a message (consisting of $y_{k-1}[i]/D_i$ and $z_{k-1}[i]/D_i$) at step $k$, node $i$ performs the above update of its state variables (by simply summing the corresponding values).

The above two iterations are represented in matrix notation below, where $y_k$ and $z_k$ are row vectors of size $m$, and $M$ is an $m \times m$ matrix, such that $M[i, j] = 1/D_i$ if $j \in O_i$ and 0 otherwise.

\[
y_k = y_{k-1} M, \quad k \geq 1,
\]

\[
z_k = z_{k-1} M, \quad k \geq 1.
\]

Each node $i$ calculates, at each time step $k$, the ratio $v_k[i] = y_k[i]/z_k[i]$. For the transition matrix $M$, $M[i, j] \geq 0$, and, for all $i$, $\sum_j M[i, j] = 1$. Any matrix that satisfies these conditions is said to be row stochastic. In this regard, it has been shown in [7] 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 $\lim_{k \to \infty} v_k[i] = \frac{\sum_j y_0[j]}{m}, \quad \forall i \in V$, where $m$ is the number of elements in vector $y_0$.

### III. ROBUST RATIO-CONSENSUS ALGORITHM

In this section, we present a ratio-consensus algorithm that is robust in the presence of link unreliability. The proposed algorithm is obtained by augmenting the two iterations presented above with additional state. We first introduced the robust algorithm in [8]; the main contribution of this paper is to show that, by representing the behavior of the unreliable links using “virtual nodes”, and suitably rewriting the iteration dynamics, the resulting system is mathematically equivalent to a finite inhomogenous Markov chain. Then, by using a coefficients of ergodicity approach, a commonly used method, we can prove convergence of robust consensus. In [8], we focused on analyzing the dynamics of the first and second moments of the two iterations under the assumption that all communication links drop packets with equal probabilities; we recently extended those results to the case when packet-drop probabilities are heterogeneous [9].

As before, each node maintains state variables $y_k[i]$ and $z_k[i]$. Additional state maintained at each node will be defined soon. Also as before, $y_0[j] = V_j$ and $z_0[j] = 1$, for $1 \leq j \leq m$. For convenience of the presentation, we assume that $V_j \geq 0$ for all $j$. (The assumption of non-negative $V_j$ can be relaxed easily.)

Iterative computation is performed in parallel to maintain state variables $y_k$ and $z_k$ both. 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.

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 nonzero 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 $O_i$, the 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 $I_i$ during step $k$.

Now, $\sum_i y_0[i]$ is the total mass in the system initially. When iteration (3) is performed on perfectly reliable links, then for all iterations $k$, $\sum_i y_k[i] = \sum_i y_0[i]$. That is, provided that the links are always reliable, the total mass in the system remains constant. However, if a message $v$ sent by node $i$ is not received by some node $j \in O_i$, 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. 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 $O_i$. For this purpose, node $i$ broadcasts message $\mu_k[i]$ to its neighbors in $O_i$. However, this message may not be received by all the nodes in $O_i$. Additional state is maintained at each node to allow the algorithm to perform correctly despite the link unreliability. The additional state is used to emulate a “virtual buffer” for each directed link. As will become clearer later, the “virtual buffer” can be seen as holding mass that may have otherwise been lost due to link unreliability; the mass in the virtual buffer is “released” to the intended receiver whenever the corresponding link operates reliably. These virtual buffers are then represented by adding “virtual nodes” to the original network, to obtain an “augmented network”. The idea of augmenting the communication graph has also been used to study the impact of communication delays (e.g., [10], [11], [12]).

#### A. Proposed Robust Ratio-Consensus Algorithm

Presently we only discuss the iteration for state $y_k$ (similar steps are performed for state $z_k$ as well). 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 $\sum_j \mu_j[i]$, denoted as $\sigma_k[i]$. Thus, $\sigma_k[i]$ is the total mass that node $i$ wants to transfer to each node in $O_i$ through the first $k$ steps.
- Each node $i$ maintains, in addition to state variable $y_k[i]$, also a state variable $\rho_k[j, i]$ for each node $j \in I_i$. As
seen below, $\rho_k[j,i]$ keeps track of the mass received by node $i$ from node $j$.

We define $\sigma_0[i] = 0, \forall i \in V$ and $\rho_0[i,j] = 0, \forall (i,j) \in E$. The operations performed at node $i$ in step $k \geq 1$ are:

- Compute

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

and broadcast $\sigma_k[i]$ to nodes in $O_i$.

- For each $(j,i) \in E$: If message $\sigma_k[j]$ is received by $i$ on link $(j,i)$ in step $k$, then

$$\rho_k[j,i] = \sigma_k[j]$$

else

$$\rho_k[j,i] = \rho_{k-1}[j,i].$$

- Compute

$$y_k[i] = \sum_{j \in I_i} (\rho_k[j,i] - \rho_{k-1}[j,i]).$$

**Intuition:** When link $(j,i) \in E$ is reliable, $\rho_k[j,i]$ becomes equal to $\sigma_k[j]$. This can be viewed equivalent to node $i$ receiving any new mass node $j$ wants to send at step $k$, as well as any mass previously stored in the “virtual buffer” for link $(j,i)$. 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 $j$. 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 (7).

As presented above, $\sigma$ and $\rho$ variables increase monotonically with time. This is undesirable in practice; however, node $i$ can communicate intermittently to node $j$ (over a multi-hop path if needed) the amount of mass it has received from $j$ through a certain number of iterations. This information can then be used to “reset” the state maintained for the corresponding virtual buffer, avoiding the monotonic increase in the state value. We omit the details here for brevity.

**IV. INHOMOGENEOUS MARKOV CHAIN FORMULATION**

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

Let us call the augmented network $G^a$. With a slight abuse of notation, let us denote by $y_k$ the state of the nodes in the augmented network $G^a$. The vector $y_k$ for $G^a$ is an augmented version of $y_k$ for $G$. In addition to $y_k[i]$ for each $i \in V$, the augmented $y_k$ vector also includes elements $y_k[(i,j)]$ for each $(i,j) \in E$, with $y_0[(i,j)] = 0$. Due to the manner in which the $y_k[i]$’s are updated, $y_k[i], i \in V$, are identical in the original network and the augmented network; therefore, we do not distinguish between them. We next rewrite the algorithm in (5)–(7) into matrix form:

$$y_k = y_{k-1}M_k,$$

for appropriate 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 $(j,i) \in E$ at each time step $k$ as follows:

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

We will now reformulate the iteration (5)–(7) and show how it can be described in matrix form as in (8), where the matrix transition matrix $M_k$ is a function of the indicator variables defined in (9). First, by using the indicator variables at time step $k$, as defined in (9), it follows from (6) that

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

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 (5) and (10) that

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

Also, from (5) and (10), it follows that (7) can be written as

$$y_k[i] = \sum_{j \in I_i} X_k[j,i] \left( \frac{y_k[j]}{D_j} + \nu_{k-1}[j,i] \right), \quad k \geq 1.$$

At every step $k$ 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 zero and its value at slot $k - 1$ is received by node $i$ as can be seen in (12). Thus, from (11) and (12), 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 $V$ computes $\nu_k$) that just result from combining, as explained above, variables that the nodes in $V$ compute. The reason for doing this is so that the resulting model is equivalent to an inhomogeneous Markov chain. This can be easily seen by stacking up (12) for all nodes indexed in $V$, i.e., the computing nodes, and (11) for all virtual buffers $(j,i)$, with $(j,i) \in E$, and rewriting the resulting expressions in matrix form, from where the expression in (8) results, as elaborated in the next section.

Thus, the state vector $y_k$ used in the analysis below is obtained by stacking the original $y$ states for the nodes in $V$, and $\nu$ states for the virtual buffers. Next, we discuss the structure of the $M_k$’s and obtain their entries by inspection of (11) and (12).
1) Structure of $M_k$: Let us first define the entries in row $i$ of matrix $M_k$ that corresponds to $i \in V$. For $(i,j) \in E$, there are two possibilities: $X_k[i,j] = 0$ or $X_k[i,j] = 1$. If $X_k[i,j] = 0$, then the mass $\mu_k[i] = \frac{y_k[i]}{D_i}$ 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,

$$M_k[i,(i,j)] = \frac{1 - X_k[i,j]}{D_i}.$$  \hspace{1cm} (13)

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

$$M_k[i,j] = X_k[i,j]/D_i,$$  \hspace{1cm} (14)

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

$$M_k[i,j] + M_k[i,(i,j)] = 1/D_i,$$  \hspace{1cm} (15)

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

Now define row $(i,j)$ of matrix $M_k$, which describes how the mass of the virtual buffer $(i,j)$, for $(i,j) \in E$, 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 transferred from buffer $(i,j)$ to node $j$. These conditions are captured by defining $M_k$ entries as follows:

$$M_k[(i,j),j] = X_k[i,j],$$  \hspace{1cm} (16)

$$M_k[(i,j),(i,j)] = 1 - X_k[i,j].$$  \hspace{1cm} (17)

Also, for obvious reasons,

$$M_k[(i,j),p] = 0, \forall p \neq j, p \in V,$$  \hspace{1cm} (18)

$$M_k[(i,j),(s,r)] = 0, \forall (i,j) \neq (s,r), (s,r) \in E. \hspace{1cm} (19)$$

Clearly, all the entries of the row labeled $(i,j)$ add up to 1, which results in $M_k$ being row stochastic 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 $X_k[i,j]$, $(i,j) \in E$, $k \geq 1$) 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. (M2): Each matrix in $\mathcal{M}$ is a finite-dimensional square row stochastic matrix. (M3): Each positive element of any matrix in $\mathcal{M}$ is lower bounded by a positive constant. Let us denote this lower bound as $c$; due to the manner in which matrices in $\mathcal{M}$ are constructed, we can define $c$ to be the positive constant obtained as $c = \min_{i,j,M \in \mathcal{M}, M[i,j] > 0} M[i,j] = \min_{i \in V} 1/D_i$. (M4): The matrix $M_k$, $k \geq 0$, may be chosen to be any matrix $M \in \mathcal{M}$ with a nonzero 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). (M5): There exists a finite positive integer $l$ such that, for any $i \in V$, it is possible to find $l$ 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; $l$ is independent of node $i$.

Property (M5) states that, for each $i \in V$, there exists a matrix $T_i^*$, obtained as the product of $l$ matrices in $\mathcal{M}$ that has the following properties:

$$T_i^*[i,i] > 0, \quad \forall j \in V,$$  \hspace{1cm} (20)

$$T_i^*[(j_1,j_2),i] > 0, \quad \forall (j_1,j_2) \in E. \hspace{1cm} (21)$$

Property M5 follows from the fact that graph $G$, and therefore augmented graph $G^a$, is strongly connected.

[6] also allows for time-varying transition matrices, but requires the matrices to have nonzero diagonals. Our robust algorithm also results in time-varying transition matrices, but not all the diagonal entries in these matrices are necessarily nonzero.

V. Ergodicity Analysis of Products of $M_k$’s

We will next analyze the ergodic behavior of the forward product $T_k = M_1M_2 \cdots M_k = \Pi_{p=1}^k M_p$, where $M_p \in \mathcal{M}, \forall j = 1, 2, \ldots, k$. 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 [13], [14], [15].

A. Results Pertaining Coefficients of Ergodicity [14]

A coefficient of ergodicity of a matrix $A$ characterizes how different two rows of $A$ are. For a row stochastic matrix $A$, coefficients of ergodicity $\delta(A)$ and $\lambda(A)$ are defined as [14]:

$$\delta(A) := \max_j \max_{i_1,i_2} |A[i_1,j] - A[i_2,j]|.$$  \hspace{1cm} (22)

$$\lambda(A) := 1 - \min_{i_1,i_2} \sum_j \min(A[i_1,j],A[i_2,j]).$$  \hspace{1cm} (23)

**Proposition 1:** For any $p$ square row stochastic matrices

$$A_1, A_2, \ldots, A_{p-1}, A_p,$$  \hspace{1cm} [14]

$$\delta(A_1A_2 \cdots A_{p-1}A_p) \leq \left( \prod_{i=1}^{p-1} \lambda(A_i) \right) \delta(A_p) \leq \prod_{i=1}^{p} \lambda(A_i).$$  \hspace{1cm} (24)

**Definition 1:** A matrix $A$ is said to be a scrambling matrix, if $\lambda(A) < 1$ [13].

Note that, if any one column of a row stochastic matrix $A$ contains only nonzero 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

$$y_k = y_{k-1}M_k, \quad k \geq 1,$$  \hspace{1cm} (25)
where $M_k \in M$, $\forall k$. As already discussed, the sequence of $M_k$’s that will govern the progress of $y_k$ is determined by communication link reliability. Defining $T_k = \Pi_{j=1}^{k} M_j$, we obtain:

$$y_k = y_0 M_1 M_2 \cdots M_k = y_0 \Pi_{j=1}^{k} M_j = y_0 T_k, \quad k \geq 1. \quad (26)$$

$T_k$, a product of row stochastic matrices, is row stochastic. By convention, $\Pi_{j=1}^{k} M_i = I$ 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,

$$W_k = \Pi_{j=(k-1)l+1}^{kl} M_j, \quad k \geq 1, \quad M_j \in M, \quad (27)$$

**Lemma 1:** There exist constants $\alpha$, $\beta$, and $w$ where $0 < \alpha < 1$, $0 \leq \beta < 1$ and $w > 0$, such that, with probability greater than $(1 - \alpha^k)$, $\delta(T_k) \leq \beta^k$ for $k \geq 8l/w$. The proof, which uses properties (M4) and (M5), is omitted for lack of space (see [16]).

**Lemma 2:** $\delta(T_k)$ converges almost surely to 0.

**Proof:** For $k \geq 8l/w$, from Lemma 1, we have that $\Pr\{\delta(T_k) \geq \beta^k\} \leq \alpha^k$, $0 < \alpha < 1$, $0 \leq \beta < 1$. 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 OF PROPOSED ROBUST ALGORITHM

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

$$y_k = y_{k-1} M_k, \quad k \geq 1, \quad (28)$$

$$z_k = z_{k-1} M_k, \quad k \geq 1, \quad (29)$$

where $M_k \in M$, $k \geq 1$, $y_0[i] \geq 0$, $\forall i$, $z_0[i] \geq 0$, $\forall i$, and $\sum_j z_0[j] > 0$, and $y_0[i, j] = z_0[i, j] = 0$, $\forall (i, j) \in E$. The same matrix $M_k$ is used at step $k$ of the iterations in (28) and (29), however, $M_k$ may vary over $k$. Recall that $y_k$ and $z_k$ in (28) and (29) have $n$ elements, but only the first $m$ elements correspond to computing nodes in $G^n$; the remaining entries in $y_k$ and $z_k$ correspond to virtual buffers.

Due to the manner in which matrices $M_k$ are defined, the $M_k$ matrices do not necessarily have all the diagonal elements positive (in particular, see (17)). The analysis below shows that the ratio algorithm achieves asymptotic consensus correctly even if diagonals of the transition matrices ($M_k$’s) are not always strictly positive. Aside from this important difference, our proof technique for the augmented network is similar to the one in [6].

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

$$\pi^* = \frac{\sum_{j \in V} y_0[j]}{\sum_{j \in V} z_0[j]} = \frac{\sum_{j \in V} V_i}{m}. \quad (30)$$

Note that $y_0$ and $z_0$ state for each virtual buffer is 0, and, therefore, it does not affect $\pi^*$. To achieve the above goal, each node $i \in V$ calculates

$$\pi_k[i] = \frac{y_k[i]}{z_k[i]} \quad (31)$$

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

$$z_k[i] \geq \mu, \quad (32)$$

for some constant $\mu > 0$ to be defined later. We will show that, for each $i = 1, 2, \ldots, n$, the sequence $\pi_k[i]$ thus calculated asymptotically converges to the desired consensus value $\pi^*$. To show this, we first establish that (32) occurs infinitely often, thus computing nodes can calculate the ratio in (31) infinitely often. Then, we will show that as $k$ goes to infinity, the sequence of ratio computations in (31) will converge to the value in (30).

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:

$$\tilde{y}_k[i] = \frac{y_k[i]}{\sum_j y_0[j]}, \quad \forall i, \quad (33)$$

$$\tilde{z}_k[i] = \frac{z_k[i]}{\sum_j z_0[j]}, \quad \forall i. \quad (34)$$

Thus, $\tilde{y}_0$ and $\tilde{z}_0$ are defined by normalizing $y_k$ and $z_k$. It follows that $\tilde{y}_k$ and $\tilde{z}_k$ 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 $\mu_z$. In particular, since $z_0[j] = 1$ for $j \in V$, define $\mu_z = 1$. (When the algorithm is used to compute weighted average, $z_0[j]$ will be typically positive for all $j$, and this value can be used by node $j$ as the lower bound.) We also assume that there exists an upper bound, say $\mu_\alpha$, on $\sum_j y_0[j]$. Define

$$\mu = \frac{\mu_z + \mu_\alpha}{n}. \quad (35)$$

where $c$ was defined in property (M3). As time evolves, each node $i \in V$ will calculate a new estimate of the consensus value whenever $z_k[i] \geq \mu$. The next lemma establishes that nodes can perform this calculation infinitely often.

**Lemma 3:** Let $T_i = \{\tau_1, \tau_2, \cdots\}$ denote the sequence of time steps when node $i$ updates its estimate of the consensus using (31), and obeying (32), where $\tau_i^j < \tau_i^{j+1}, \quad j \geq 1$. The sequence $T_i$ contains infinitely many elements with probability 1.

**Proof:** To prove the lemma, it will suffice to prove that for infinitely many values of $k$, $z_k[i] \geq \mu$, with probability 1. Assumptions (M1)-(M5) imply that each matrix $W_j$ is row stochastic. In (27) contains a strictly positive column corresponding to index $i \in V$ with a nonzero 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 nonzero elements of each matrix in $M$ are all greater than or equal to $c$, $c > 0$ (by property M3), and
since \( W_j \) is a product of \( l \) matrices in \( M \), it follows that all the nonzero elements of each \( W_j \) are lower bounded by \( c^j \).

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, \( z_{jk} = \tilde{z}_{(j-1)k} W_j \), \( \tilde{z}_k \) is a stochastic vector. Thus, for any \( k \geq 0 \), \( \sum_k \tilde{z}_k[i] = 1 \) and, at least one of the elements of \( \tilde{z}_{(j-1)k} \) must be greater than or equal to \( 1/n \). Also, all the elements in columns of \( \tilde{z}_k \) indexed by \( i \in V \) are lower bounded by \( c^j \) (recall that we are now only considering those \( j \) for which the \( i \)-th column of \( W_j \) is positive). This implies that, \( z_{jk}[i] \geq c^j/n \), thus \( z_j[l] \geq \left( \sum_j z[j] \right) c^j/n \), from where it follows that \( z_{jl}[i] \geq \mu_z c^j/n \). Then, since infinitely many \( W_j \)'s contain a positive \( i \)-th column (with probability 1), by (35), we finally obtain that \( z_{jl}[i] \geq \mu_z \) holds for infinitely many \( j \) with probability 1. Thus, with probability 1, the set \( T_i = \{ \tau_i^1, \tau_i^2, \cdots \} \) contains infinitely many elements, for any \( i \in V \).

**Theorem 1:** Let \( \pi_i[t] \) denote node \( i \)'s estimate of the consensus value calculated at time \( t_i \). For each node \( i \in V \), with probability 1, \( \pi_i[t] \) converges to \( \pi^* = \sum_j y_j \pi[j] \).

**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 2 implies that \( \Pr \{ \lim_{t \to \infty} \delta(T_k) = 0 \} = 1 \). Together, Lemmas 2 and 3 imply that, with probability 1, for a chosen execution, (i) for any \( \psi > 0 \), there exists a finite \( k_\psi \) 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 \( z_k[i] > \mu_z \).

Consider any \( k \geq k_\psi \) such that \( z_k[i] > \mu_z \). Since \( \delta(T_k) \leq \psi \), the rows of matrix \( T_k \) are “within \( \psi \)” of each other. Observe that \( \tilde{y}_k \) is obtained as the product of stochastic row vector \( y_0 \) 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 \( \psi \) of each other, for all \( j \). Therefore,

\[
\left| \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} - 1 \right| \leq \frac{\psi}{\tilde{z}_k[i]} \tag{36}
\]

\[
\Rightarrow \left| \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} - \sum_j y[j][i] \right| \leq \psi \sum_j y[j][i] \frac{1}{\tilde{z}_k[i]} \tag{37}
\]

\[
\Rightarrow \left| \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} - \sum_j y[j][i] \right| \leq \frac{\psi}{\tilde{z}_k[i]} \mu \tag{38}
\]

Given any \( \epsilon > 0 \), let us choose \( \psi = \epsilon \mu / \mu_z \). Then (37) implies that \( \left| \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} - \sum_j y[j][i] \right| \leq \epsilon \) whenever \( k \geq k_\psi \) and \( k \in T_i \). Therefore, \( \frac{\tilde{y}_k[i]}{\tilde{z}_k[i]} \) for \( k \in T_i \) converges to \( \epsilon \) in the limit. This result holds with probability 1, since conditions (i) and (ii) above hold with probability 1.

**VII. CONCLUDING REMARKS**

The paper considers an average consensus algorithm that performs correctly despite link unreliability. We model the behavior of the lossy links using virtual nodes, which allows us to rewrite the algorithm dynamics as an inhomogeneous Markov chain. We then use results involving coefficients of ergodicity to prove the convergence of the algorithm.

**ACKNOWLEDGMENTS**

Vaidya was supported by Army Research Office grant W-911-NF-0710287 and NSF grant CNS-1059540. Domínguez-García was supported by NSF grants ECCS-CAR-0954420 and ECCS-CPS-1135598. Hadjicostis was supported in part by the European Commission (EC) Seventh Framework Programme (FP7/2007-2013) under grants INFSO-ICT-223844 and PIRG02-GA-2007-224877. 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 or the EC.

**REFERENCES**


