Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Reaching approximate Byzantine consensus with multi-hop communication ☆

Lili Su *, Nitin H. Vaidya

Department of Electrical and Computer Engineering, and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, United States

ARTICLE INFO

Article history:
Received 15 November 2015
Received in revised form 31 May 2016
Available online xxxx

Keywords:
Approximate agreement
Bounded length communication paths
Byzantine protocols
Synchronous systems

ABSTRACT

We are interested in approximate Byzantine consensus problem, wherein all the fault-free processes reach consensus asymptotically (approximately in finite time). In particular, we focus on the algorithms under which each process communicates with other processes that are up to l hops away and maintains minimal states across iterations. For a given l, we identify a necessary and sufficient condition on the network structure for the existence of iterative algorithms of interest. Our necessary and sufficient condition generalizes the tight condition identified in prior work for $l = 1$. For $l \geq l^*$, where l^* is the length of a longest cycle-free path in the given network, our condition is equivalent to the necessary and sufficient conditions for exact consensus in undirected and directed networks both.

1. Introduction

The problem addressed in this paper concerns a collection of processes that are connected by a network. Among the networked processes, some unknown processes may be compromised by an adversary, and be reprogrammed to behave arbitrarily, and adversarially try to degrade the behavior of the system. This fault model is referred as Byzantine fault [1]. In this paper, we are interested in approximate Byzantine consensus problem, wherein all the *faulty-free* processes reach consensus asymptotically (approximately in finite time). In particular, we focus on the algorithms under which each process communicates with other processes that are up to l hops away[1] and maintains *minimal* states across iterations – no messages received during previous iterations will be used in the state updates.

In practice, a faulty process does not necessarily behave entirely arbitrarily. However, since it is not always practical to precisely characterize the behaviors under all possible failures, the Byzantine fault model is often used in such cases as well due to its capability of dealing with all possible deviations from correct operations. The Byzantine fault-tolerance problem was first introduced in [2], and is one of the most fundamental problems in distributed computing. Ref. [3] showed

☆ This research is supported in part by NSF awards 1329681 and 1421918. 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.

* Corresponding author.
E-mail addresses: lilisu3@illinois.edu (L. Su), nhv@illinois.edu (N.H. Vaidya).
URL: https://sites.google.com/site/lilisuece/ (L. Su).

[1] Via synchronous FIFO (first-in-first-out) communication channels.

http://dx.doi.org/10.1016/j.ic.2016.12.003
0890-5401/

Please cite this article in press as: L. Su, N.H. Vaidya, Reaching approximate Byzantine consensus with multi-hop communication, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.12.003

that the fault-tolerant consensus problem cannot be solved in *asynchronous* system even in the presence of only one crash failure. A process suffering crash faults may unexpectedly stop participating the specified algorithms/protocols. As one way to circumvent this impossibility result, the notion of *approximate consensus* was introduced in [4] by requiring that the nodes agree with each other only asymptotically (approximately in finite time). The notion of approximate consensus is of interest in *synchronous* system as well [4–6]. The discussion in this paper applies to synchronous system.

Let n be the total number of processes and f be the upper bound on the number of faulty processes in the system. The actual number of compromised (faulty) processes may vary across executions, and may not known to the fault-free processes. However, each fault-free process knows that in each execution at most f processes may be faulty. In networks with bidirectional links, approximate consensus is achievable if and only if the network node-connectivity is at least $2f+1$ and less than one third of the processes can be faulty, i.e., $n \geq 3f+1$ [7]. Relaxing the bidirectional communication assumption, a tight condition for directed graphs was presented in [8]. There has been increasing interest in designing iterative variants of approximate Byzantine consensus where only local knowledge of the network topology (and local communication) is needed, and processes carry minimal state across iterations [9–11,5,6]. Ref. [11] studied the convergence rate of approximate consensus algorithms over complete networks. Refs. [5,6] considered arbitrary directed networks and derived tight (necessary and sufficient) topological conditions on the communication network. While [6] investigated the Byzantine fault model, [5] considered a restricted fault model in which the faulty nodes are restricted to sending identical messages to their neighbors. Independently, when $f=0$, such iterative approximate consensus algorithms have been well-studied in cooperative control community [12–14].

To the best of our knowledge, no attempts have been made on investigating the impact of each process' communication range on the network condition for a correct iterative approximate consensus algorithm to exist. In this paper, we model the network as a directed graph, and we focus on the family of algorithms in which a process communicates with processes that are up to l hops away by forwarding messages through intermediate processes. The directed graph model is motivated by the presence of directed links in wireless networks. Our goal is to identify a necessary and sufficient condition on the network structure for the algorithms of interest to exist. Note that, in this paper, the algorithms considered are "non-terminating" in the sense that consensus is only achieved asymptotically. In fact, it can be seen from our proofs that for any given ϵ, we are able to identify a termination condition such that the disagreement among the fault-free processes is at most ϵ.

Contributions Our main contribution is to identify a necessary and sufficient condition on the network structure for a given l, named Condition NC for a given l. Informally speaking, our Condition NC states that for any four set process partition L, R, C, and F such that both L and R are non-empty and $|F| \leq f$, with up to l-hop communication, at least one process in L is influenced by processes in $R \cup C$ or at least one process in R is influenced by processes in $L \cup C$. Condition NC will be formally stated in Section 3. Our sufficiency proof is shown by constructing a simple iterative algorithm, whose trim function is defined based on a *minimal messages cover* property that we introduce in this paper.

The tight condition we found is consistent with the tight condition identified in [6] when only local communication is allowed, i.e., $l=1$. For $l \geq l^*$, where l^* is the length of a longest cycle-free path in the given network, our condition is equivalent to the necessary and sufficient condition for consensus in undirected networks [7] as well as exact consensus in directed networks [8].

Organization The rest of the paper is organized as follows. Section 2 presents our models and the structure of the iterative algorithms considered in our work. Our necessary condition is presented in Section 3, and its sufficiency is proved constructively in Section 4. The correspondence between our condition and the results in [4,7,8] is discussed in Section 5. Section 6 discusses possible relaxations of our fault model and concludes the paper.

2. Problem setup and structure of iterative algorithms

Communication model The system is assumed to be *synchronous*. The communication network is modeled as a simple *directed* graph G with self-loop at each process. Denote $\mathcal{V}(G)=\{1,\ldots,n\}$ as the set of n processes, where $n \geq 2$, and $\mathcal{E}(G)$ as the set of directed links between processes in $\mathcal{V}(G)$. In general, $\mathcal{V}(\cdot)$ and $\mathcal{E}(\cdot)$ are two functions defined over graphs that return the vertex set and the edge set, respectively, for a given graph. For instance, let H be a graph, then $\mathcal{V}(H)$ and $\mathcal{E}(H)$ are the vertex set and edge set of H. In this paper, we use "process" and "node" interchangeably, and use "link" and "edge" interchangeably.

Let l be a positive integer. For each node i, let N_i^{l-} be the set of nodes that can reach node i via at most l hops. Similarly, denote the set of nodes that are reachable from node i via at most l hops by N_i^{l+}. Note that $i \in N_i^{l-}$ and $i \in N_i^{l+}$. When $l=1$, we write N_i^{1-} and N_i^{1+} as N_i^- and N_i^+, respectively, for simplicity. We also assume each node i knows the entire network topology.

Node i may send messages to node j via different i,j-paths with intermediate nodes on an i,j-path forwarding messages accordingly. To capture this distinction in transmission routes, we represent a message as a tuple $m=(w,P)$, where $w \in \mathbb{R}$ is the message content, and P indicates the path via which message m should be transmitted. It is assumed that the network layer in the system delivers the messages along the specified paths. The intermediate nodes on the paths do not view the message values (i.e., the message values are not used by intermediate nodes in performing consensus). Four functions are defined over message m, corresponding to message content, transmission route, message source, and message destination,

Please cite this article in press as: L. Su, N.H. Vaidya, Reaching approximate Byzantine consensus with multi-hop communication, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.12.003