Random Consensus Protocol in Large-Scale Networks

Zhipu Jin and Richard M. Murray

Abstract—One of the main performance issues for consensus protocol is the convergence speed. In this paper, we focus on the convergence behavior of discrete-time consensus protocol over large-scale sensor networks with uniformly random deployment, which are modeled as Poisson random graphs. Instead of using “random rewiring” procedure, we introduce a deterministic principle to locate certain “chosen nodes” in the network and introduce “virtual” shortcuts among them so that the number of iterations to achieve average consensus drops dramatically. Extensive simulation results are presented to verify the efficiency of this approach. Moreover, a random consensus protocol is proposed, in which virtual shortcuts are implemented by random routes.

Index Terms—Random consensus protocol, small-world effect, distributed algorithms, convergence speed, random deployment, sensor network.

I. INTRODUCTION

Recently, consensus seeking in networked multi-agent systems has been extensively studied by many researchers from different disciplines. Starting from the Vicsek’s model for self-driven particles [1], Jadabaie et al. give a theoretical explanation based on stochastic matrix theory in [2]. Olfati-Saber and Murray [3] propose a continuous-time consensus protocol and show that this protocol achieves the average consensus for a balanced directed graph. Other cases for consensus seeking are discussed in [4], [5], [6]. Consensus protocol is quickly employed in many applications, such as coordination control [7], peer-to-peer networks [8], distributed Kalman filters [9], swarming and flocking [10], and oscillators synchronization [11].

Convergence speed of consensus protocol has been identified as an important performance issue, which is determined by the topology of the network and the local weights. According to [3], the convergence speed of the continuous-time consensus protocol is bounded by algebraic connectivity, which is the second smallest eigenvalue of Laplacian matrix. An explicit formula is given in [12] to show that the algebraic connectivity of a regular lattice converges to zero as the size of the lattice goes to infinity, which means that the consensus protocol needs infinite time to converge. Another work is reported in [13] where the concept of “effective resistance” for lattice graphs is used to bound the convergence rate. A multi-hop consensus protocol is proposed in [14] so that, without physically changing the topology, the convergence speed is improved by systematically using multi-hop paths in the network. However, this method only moderates the problem. Inspired by the idea of “small-world networks” and a “random rewiring” procedure developed in [15], the author of [16] claims that, with the same large number of nodes, a small-world network has a much bigger algebraic connectivity than a regular lattice. Another type of networks, Ramanujan graph [17], is also discussed in a recent work [12] since it expresses very quick convergence behavior due to its special topology. On the choice of local weights, [18] treats a discrete-time consensus process as an optimal linear iteration problem and shows that the convergence speed can be increased by finding the optimal local weights when the global structure of the network is known beforehand.

One potential application for consensus protocol is data fusion in sensor networks. In order to monitor an interesting area, large number of small but “smart” sensors may be deployed randomly and collect data such as sound, motion, temperature, etc. Due to energy constraints, each sensor may only set up a wireless communication link with another sensor when the distance between them is shorter than a certain range. Also, the number of links that one sensor can have is limited. Issues on deployment method, data collection, optimal coverage, and energy consumption has been extensively studied [19], [20], [21], [22], [23] during the last several years. Certain communication links will inevitably become the bottleneck for data fusion if a centralized approach is used. On the other hand, decentralized approach, such as a consensus protocol, is criticized due to its long processing time. In this paper, we focus on consensus convergence behavior for a large-scale sensor network with random deployment. Assuming the topology is an undirected graph. Instead of using the random rewiring procedure, we add small amount of “virtual” links among certain “chosen nodes” as “shortcuts” to join geographically remote parts to one another. More importantly, we give out a principle to locate those chosen nodes only based on local information and pre-defined parameters, such as sensor density. A random consensus protocol is proposed to implement those shortcuts by random routes in the network. We claim that the iteration number to achieve a certain accuracy for average consensus becomes incredibly small for large scale networks if we choose nodes and shortcuts based on this principle.

The remainder of this paper is organized as follows: In Section II, consensus behavior for large size sensor networks with uniformly random deployment is formulated. We then propose a deterministic principle to locate chosen nodes in the network so that adding virtual links among them can significantly improve the convergence speed. Section
IV is devoted for a random consensus protocol, in which those virtual links is implemented in the existing network. Examples and simulation results are also provided. Finally, conclusions and future work are summarized in Section V.

II. AVERAGE CONSENSUS OVER SENSOR NETWORKS WITH RANDOM DEPLOYMENT

Most practical deployment methods in large scale sensor networks is “random, or at best, can be controlled with coarse granularity” [23]. Suppose there is a sufficient large 2-D square area \( \Omega \in \mathbb{R}^2 \) where \( \mathbb{R}^2 \) denotes a two-dimension Euclidian space. We randomly place \( N \) (a large number) sensors in \( \Omega \) and the distribution is uniform. If the distance between any two sensors is shorter than a certain communication range \( R \), a wireless link is set up between them. Figure 1 shows the topologies of three sensor networks.

Suppose the size of the square area is \( L^2 \). Since the deployment is uniformly random, the probability that there exists a link between any two sensors is

\[
p = \frac{\pi R^2}{L^2}.
\]

(1)

except those nodes who are close to the boundary. We assume that the average sensor density is constant and the communication range \( R \) is preset. Then the average number of links that each sensor has is

\[
E[d] = NP = N \frac{\pi R^2}{L^2}.
\]

(2)

It is obvious that, when \( N \to \infty \), the degree distribution for the network can be approximated by Poisson distribution. In other words, the probability of any node to have degree \( k \) is

\[
p(k) = \frac{e^{-\lambda} \lambda^k}{k!},
\]

(3)

where \( \lambda = E[d] \), and the degree distribution is \( NP(k) \).

Figure 2 shows the degree distribution of a network with 1000 nodes. Please note, since nodes near the boundary may have less nodes, the degree distribution is skew to left a little comparing with the one generated by Poisson distribution.

Passion random graph has been studied by mathematicians and physicists since long time ago [24]. Many interesting properties of random graphs are identified in the limit of large graph size. For a good review, please refer to [25]. It has been noticed that, when the average degree is bigger than 1, all nodes are joined together in a single “giant component” with high probability. In other words, if \( E[d] > 1 \), the topology will most likely be connected. Networks in Figure 1 also verify this property. Thus, we may use consensus protocol to calculate the average value of the data collected by the network.

There are at least two types of discrete-time consensus protocol reported in current literature [2], [8]. One is directly derived from the continuous-time consensus protocol. Let \( x_i(k) \) denote the state of node \( i \) at time \( k \) and \( \mathcal{N}(i) \) denote the set of neighbors. The consensus protocol is represented by

\[
x_i(k) = x_i(k-1) - \gamma \sum_{j \in \mathcal{N}(i)} (x_i(k-1) - x_j(k-1))
\]

(4)

where \( \gamma \) is the step size. The consensus process is presented by

\[
X(k) = X(k-1) - \gamma LX(k-1)
\]

(5)

where \( X = [x_1, \cdots, x_n]' \) and \( L \) is the Laplacian matrix of the network. This protocol solves the average consensus problem for a connected graph as long as \( \gamma \) is strictly less than the inverse of twice the largest eigenvalue of \( L \). An sufficient condition for that is given in [8] as

\[
0 < \gamma < \frac{1}{2d_{\text{max}}}
\]

(6)
becomes bigger. From now on, we use Equation (8) as the iterations. It also indicates that the converge speed, in terms of iteration steps, becomes larger as the network expands. Figure 3 shows how two consensus processes closely converge to the average value since $W$ is the adjacency matrix. There is no parameter design issue for this protocol. But it cannot guarantee to converge to the average value since $W$ may not be symmetric. However, for large scale Poisson random graphs, most of the nodes have close degrees and this protocol still converges to a very good approximation of the average value. Figure 3 shows how two consensus processes closely converge to the average value after 150 iterations. For the network with 100 nodes, the average value is 9.65 and the consensus process reaches to $9.63 \pm 0.01$ after 150 iterations. For the network with 1000 nodes, the average value is 10.08 and the consensus process reaches to $10.11 \pm 0.08$ after 500 iterations. It also indicates that the converge speed, in terms of the number of iterations, becomes larger as the network becomes bigger. From now on, we use Equation (8) as the local updating rule for discrete-time consensus seeking.

III. DETERMINISTIC APPROACH FOR SMALL-WORLD EFFECT

According to Equation (8) and [18], if we want $X(k)$ converges to the average vector

$$\bar{X} = \frac{1}{N} \mathbf{1}^T \cdot X(0),$$

it must be true that

$$\lim_{k \to \infty} W^k = \frac{\mathbf{1}^T}{n}$$

(9)

where $\mathbf{1}$ denotes the vector with all ones and $X(0)$ is the initial condition. Thus, the asymptotic convergence factor is defined as

$$r(W) = \sup_{X(0) \neq \bar{X}} \lim_{k \to \infty} \left( \frac{\|X(k) - \bar{X}\|_2}{\|X(0) - \bar{X}\|_2} \right)^{1/k}$$

(10)

and the convergence time $t(W)$, in terms of iteration steps, is given by

$$t(W) = -\frac{1}{\log(r(W))}.$$ 

(11)

Moreover, [18] shows that

$$r(W) = \rho(W - \frac{\mathbf{1}^T}{n})$$

(12)

if Equation (9) holds, where $\rho(\cdot)$ denotes the spectral radius of a matrix. Based on the observation on Figure 3, we use $t(W)$ to indicate the convergence rate in the rest of this paper.

When we keep the sensor density constant, the shape of degree distribution $p(k)$ is determined by $E[d]$ and $N$. However, the convergence time $t(W)$ increases quickly when the network expands. Figure 4 shows some simulation results about $t(W)$ where we increase the number of sensors and keep $E[d]$ constant.

One challenge for consensus protocols in large scale networks is how to keep the convergence time scalable. Manipulating the network topology is a possible approach. In the literature, there exists at least two methods to change a regular lattice into a small-world graph. One is called random rewiring procedure, which randomly takes a small
fraction of the existing links and moves one end of each link to a new location chosen uniformly at random from other nodes. Another method is adding a small amount of shortcuts randomly into the network [26], [27]. Those two methods have been proved to provide similar “small-world effect” [15], which dramatically improve the speed of information propagation over the network. We choose the second method here because we believe that it is more suitable for real sensor networks with random deployment.

The first question for adding shortcuts into the network is how to locate certain nodes, we call them chosen nodes, so that links are added among them as shortcuts. We are looking for a complete decentralized principle so that, as long as the sensor network is deployed, each sensor can automatically determine if it is a chosen node or not only based on local information. The local information we use here is the degree, i.e., how many links a node has after the deployment.

Figure 2 shows the degree distribution of a sensor network with 1000 nodes, which is a Poisson distribution scaled by \( N \). Figure 5 shows the locations of sensors with certain degree. It is surprising to see that nodes with high degree are highly clustered. Their locations are close and most of them are already connected with each other. Thus, adding links among them may not improve the convergence time. For those nodes with low degree, they are more likely located along the boundary and are not good choices either. For those nodes with medium degree, i.e., expected degree \( E[d] \), they happen to be good candidates since their locations evenly cover the whole interesting area.

The second question is how many shortcuts we should add. The number of nodes whose degree equals to \( E[d] \) can be approximated by

\[
M \approx \text{round}(N \cdot p(E[d])) = \text{round}\left(N \cdot \frac{e^{-\lambda} \lambda^d}{d!}\right)
\]

where the function around(·) rounds the input to the nearest integer and \( \lambda = N \pi R^2 / L^2 \). Thus, \( M \propto N \). We randomly choose a fraction, \( 0 < \beta \leq 1 \), of all possible \( 1/2 \cdot M(M-1) \) links among those chosen nodes as shortcuts and test the convergence time. Figure 6 shows the topology of a network with 1000 nodes and 250 shortcuts, which is about \( \beta = 8\% \) of all possible shortcuts. Figure 7 shows how the convergence time \( t(W) \) changes with different amount of shortcuts when \( N \) increases. It is true that, by properly choosing \( \beta \) and shortcuts, the trend of increase for \( t(W) \) may be stopped and even reversed. According to Figure 7, we can make \( t(W) \) less than 50 iterations for a network up to 5000 nodes. Also, when the network is large, adding more shortcuts is not necessary better than adding less. For example, for a network with 5000 nodes, \( \beta = 5\% \) shortcuts do a better job than \( \beta = 60\% \) shortcuts since the small-world effect is attenuated by too many nearby chosen nodes. This indicates that the best number of shortcuts is not proportional to the network size.

Thus, given a sensor network with random deployment and the expected degree \( E[d] = \lambda \), the principle for chosen nodes is simple and can be implement using Algorithm 1. Each sensor sets up links with its nearby neighbors, compares its degree with \( E[d] \), and identifies itself as a chosen node if they are equivalent. Then they should try to connect to and exchange information with other chosen nodes over those shortcuts.

**IV. RANDOM CONSENSUS PROTOCOL FOR SENSOR NETWORKS**

In this section, we present a random consensus protocol, which includes state updating and random routing for shortcuts. Besides the packets each sensor directly sends to its
Algorithm 1 Locating chosen nodes

Require: $d_i, E[d]$

Ensure: $\text{Chosen\_flag} = 1$ if this node is a chosen node
1: $\text{Chosen\_flag} \leftarrow 0$
2: if $d_i = E[d]$ then
3: $\text{Chosen\_flag} \leftarrow 1$
4: end if

Algorithm 2 Random consensus protocol

Require: $d_i, E[d], \text{Chosen\_flag}, m, n$

Ensure: $x_i$ is updated based on neighbors (and shortcuts).
Suppose there are $L$ shortcut packets are received from neighbors. Each shortcut packet has three parts: last node id, hop_counter, and value $p$.

1: $\text{sum} \leftarrow 0$
2: for $j = 0$ to $d_i$ do
3: $\text{sum} \leftarrow \text{sum} + x_j$
4: end for
5: $c \leftarrow 0$

6: If $\text{Chosen\_flag} \neq 1$ then
7: for $l = 0$ to $L$ do
8: if Packet is too old then
9: Discard it;
10: else
11: if hop_counter $\neq 0$ then
12: hop_counter $\leftarrow$ hop_counter $- 1$
13: end if
14: $\text{sum} \leftarrow \text{sum} + p$
15: $c \leftarrow c + 1$
16: Random pick one neighbor except the last node id;
17: Send $p$ to it with new hop_counter and id $\leftarrow i$
18: end if
19: end for
20: $x_i \leftarrow (\text{sum} + x_i)/(d_i + c + 1)$
21: send $x_i$ to its neighbors;
22: else {Is a chosen node}
23: for $l = 0$ to $L$ do
24: if Packet is too old or created by itself then
25: Discard it;
26: else if hop_counter $\neq 0$ then
27: hop_counter $\leftarrow$ hop_counter $- 1$
28: Random pick one neighbor except $j$
29: Send $p$ to it with new hop_counter and id $\leftarrow i$
30: else
31: $\text{sum} \leftarrow \text{sum} + p$
32: $c \leftarrow c + 1$
33: end if
34: end for
35: $x_i \leftarrow (\text{sum} + x_i)/(d_i + c + 1)$
36: send $x_i$ to its neighbors;
37: {Randomly generate the same shortcut packet $m$ times}
38: for $i = 0$ to $m$ do
39: Random pick one neighbor;
40: Send $x_i$ to it as a shortcut packet with id $\leftarrow i$, hop_counter $\leftarrow n$, and value $p = x_i$
41: end for
42: end if

neighbors, we define a new packet type as “shortcut packet”. Shortcut packets are generated only by chosen nodes. If a node is not a chosen one, it passes any shortcut packet it receives to its neighbors randomly. Using this mechanism, shortcuts among chosen nodes are implemented.

Algorithm 2 explains this protocol in detail and we assume that it runs on each sensor synchronously. There are a few input parameters: degree $d_i$, average degree for the network $E[d]$, chosen_flag generated by Algorithm 1, number of shortcut packets $m$ a chosen node should generate, and initial value of hop_counter $n$. Expected degree $E[d]$ is calculated by Equation (2). Degree $d_i$ and chosen_flag is determined right after the network is deployed. The value of $m$ determines how many shortcut packets a chosen node generates in each iteration. The value of $n$ denotes the number of hops a shortcut packet must be transmitted before it is discarded by other chosen nodes. How to choose the best $n$ is still under investigation, but it should be proportional to the average geodesic path length, which is $O(\log(N))$ in a Poisson random graph [25]. There are two possibilities that a shortcut packet is perished: it can be “eaten” when it reaches another chosen node after it has been passed longer than $n$ hops, or it can be discarded when it is too old. A delay threshold is used to judge if a shortcut packet is too old or not and a typical choice is $2n$.

Figure 8 shows simulation results on a network with 200 nodes that are randomly deployed. For the random consensus protocol, we have set parameters as $E[d] = 18, m = 30$, and $n = 6$. Comparing with the deterministic consensus protocol and the case where $80\%$ shortcuts are added, it is clear that using random consensus protocol effectively improves the convergence speed for average consensus seeking.

![Convergence time vs. network size with shortcuts.](image)
Also, as shown in Figure 7, the best number of shortcuts does not necessarily reach the real average protocol. Parameters $m$ and $n$ determine how much extra communication load is imposed to the network. Since routes are selected randomly, lots of shortcuts packets are discarded either because they reach other chosen nodes too late or they create self-loops. A better routing strategy is needed in order to reduce $m$. The parameter $n$ is directly associated with the delays along shortcuts. Delays on shortcuts may be able to un-stabilize the protocol and finding the optimal $n$ given the size of the network needs to consider the tradeoff between the performance and robustness.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we demonstrate that, for large scale sensor networks with random deployment, adding small amount of shortcuts can dramatically change the convergence behavior for consensus seeking. Based on local information and preset parameters, we claim that nodes with medium degree are good choices among whom shortcuts should be added. A random consensus protocol is proposed in which shortcut packets are transmitted along random routes in the network. Only the chosen nodes are “sources” of this shortcut packets, which either die out when they get old or are eaten by other chosen nodes. Simulation results verify the efficiency of this protocol.

Future work includes a quantitatively analysis on how close the consensus process can reach to the real average value in Passion random graphs with protocol Equation (7). Also, as shown in Figure 7, the best number of shortcuts does not necessary increase when the network size increases. It should be interesting to identify the optimal value of $\beta$ and its scalability properties. Furthermore, turning the parameters for random consensus protocol is important. Parameters $m$ and $n$ determine how much extra communication load is imposed to the network. Since routes are selected randomly, lots of shortcuts packets are discarded either because they reach other chosen nodes too late or they create self-loops. A better routing strategy is needed in order to reduce $m$. The parameter $n$ is directly associated with the delays along shortcuts. Delays on shortcuts may be able to un-stabilize the protocol and finding the optimal $n$ given the size of the network needs to consider the tradeoff between the performance and robustness.

VI. ACKNOWLEDGEMENTS

The authors would like to thank XXX, from XXX, for the fruitful discussions. This research is partly supported by grant XXX.

REFERENCES