Performance Comparison of Randomized Gossip, Broadcast Gossip and Collection Tree Protocol for Distributed Averaging

Jun Ye Yu
Electrical and Computer Engineering
McGill University
Montreal, Quebec
Email: jun.y.yu@mail.mcgill.ca

Michael Rabbat
Electrical and Computer Engineering
McGill University
Montreal, Quebec
Email: michael.rabbat@mcgill.ca

***Abstract*—Gossip and tree-based aggregation algorithms are two popular solutions for distributed averaging in wireless networks. The former uses only local message exchanges and requires no routing structures whereas the latter requires building a spanning tree. In this paper we provide a detailed comparison of their performance in terms of communication overhead, accuracy, latency and energy consumption using the network simulator Castalia. We use randomized gossip, broadcast gossip and the collection tree protocol as practical representatives in each category. Through simulations, we show that broadcast gossip requires, in general, the least communication overhead and lowest latency and energy at the expense of lower accuracy. Randomized gossip requires more transmissions than broadcast gossip, but has higher accuracy. The collection tree protocol requires, in general, the most communication overhead.**

I. Introduction

There is currently a large amount of literature dedicated to distributed applications for *wireless sensor networks* (WSN). The problem of distributed averaging in particular has received much attention because it can be used as a basic building block for more complex applications in distributed signal processing. The premise of distributed averaging is simple. All nodes in a network have an initial data value. They must communicate with only their neighboring nodes to determine the network-wide average.

Gossip algorithms and tree-based aggregation algorithms have emerged as two popular paradigms for distributed averaging. Randomized gossip does not require any network-wide coordination. Rather, in its simplest form [1], at the tick of a Poisson clock, two randomly selected neighbors exchange messages and average their values. If the network is strongly connected, all nodes' values can be made arbitrarily close to the network average after sufficiently many exchanges. There is no need to set up and maintain specific routing structures and there is no single point of failure or bottleneck. Alternatively, in tree-based algorithms, a spanning tree is used for aggregating data from all nodes up to the root node, and then the resulting aggregate is sent back down the tree so that all nodes receive the result. Although fewer messages are required, in principle, for aggregation, these algorithms require additional overhead for constructing and maintaining the spanning tree.

The existing literature has mainly focused on reducing the communication overhead and convergence time of gossip algorithms (e.g., [2], [3]) and constructing the minimum-cost spanning trees in static networks (e.g., [4], [5]). Yet very little work has compared the two approaches directly. Which algorithm has lower overall communication overhead, especially in a dynamic network? One previous work [6] provides partial answers to these questions and shows that broadcast gossip has the lowest communication overhead, followed by the collection tree protocol, and then randomized gossip. However, the simulations in [6] use a simplified model that omits key aspects such as packet collisions and asymmetric links, which may well lead to detrimental effects on the final results.

To avoid the same pitfalls, we use Castalia, a popular WSN simulator that provides features such as path loss calculation and SINR-based packet reception [7], so we may conduct complex simulations while focusing only on the algorithms. In this paper we adopt *broadcast gossip* (BG) and *randomized gossip* (RG) as representatives of gossip algorithms. BG exploits the broadcast nature of wireless networks and does not require nodes to have any prior knowledge of the network topology, making it attractive for dynamic networks. RG preserves the network sum so nodes converge to the true average almost surely rather than in expectation. We adopt the *collection tree protocol* (CTP) as a representative of tree-based algorithms because it is included in the TinyOS [8] distribution and is thus widely used in WSNs.

The main contributions of our work are twofold. First, we present a performance comparison of RG, BG and CTP in terms of communication overhead, accuracy, latency and energy consumption using Castalia. Second, we address some subtle issues in algorithm implementations often overlooked in the existing literature. For instance, existing gossip algorithms have no self-termination. Nodes continually exchange messages until the per-node variance is below a threshold. We present implementations for all three algorithms that address these issues. The remainder of the paper is organized as follows. Section II provides the background on the topics of interest. Section III outlines the three algorithms' implementations in Castalia. Section IV presents our main results, and Section V concludes the paper.

978-1-4673-3146-3/13/$31.00 ©2013IEEE

II. Background

A. Gossip Algorithms

Boyd et al. [1] propose randomized pairwise gossip, one of first gossip algorithms for distributed averaging in arbitrarily connected networks. Each node maintains a Poisson clock and has knowledge of its one-hop neighbors. When node i's clock ticks, it selects a random neighbor j and the two exchange messages to average their values. If the network is strongly connected, after sufficiently many iterations, all nodes' values can be arbitrarily close to the true average with high probability. The algorithm is asynchronous, fully distributed, fault-tolerant and has no single point failure; however, the algorithm has slow convergence speed due to slow diffusion of information across the network.

Broadcast gossip is first proposed by Aysal et al. [2] as the first gossip algorithm to fully exploit the broadcast nature of wireless transmissions. Unlike gossip algorithms requiring two-way message exchanges, BG is an asynchronous push-only protocol. Each node has a Poisson clock and an initial data value $x_i(0)$. At the t^{th} clock tick, node i broadcasts its value, $x_i(t)$. All nodes j receiving the broadcast update their value as

$$x_j(t+1) = \delta x_j(t) + (1-\delta)x_i(t) \quad \forall (i,j) \in E \qquad (1)$$

where $\delta > 0$ is a mixing parameter controlling the trade-off between convergence speed and mean-square error with an optimal value of 0.5 [2]. All other nodes in the network maintain their current value. Broadcasting removes the need to keep track of neighboring nodes since messages have no particular destination, a process often ignored or taken for granted in the existing literature. On the downside, broadcast gossip updates are not guaranteed to preserve the network sum, and the algorithm only converges to initial average $x_{ave}(0) = \sum_{i=1}^{n} \frac{x_i(0)}{n}$ in expectation.

B. Collection Tree Protocol

CTP [9] is designed for data aggregation. We present here only a brief overview of its core mechanisms. Interested readers are referred to [9], [10] for more technical and implementation details.

Two types of messages are used in CTP. Beacons are broadcast messages used to build and maintain the spanning tree. They contains key information such as distance to the root and the parent node ID. Data packets contain the actual application values being aggregated.

The cost metric CTP uses for spanning tree formation is the *expected number of transmissions* (ETX) to the root. The ETX from a node i to its one-hop neighbor j is calculated as a function of both incoming and outgoing link qualities. The outgoing link quality is the ratio of the transmitted messages to the received acknowledgments. The incoming link quality depends on the number of received beacons from j and total number of beacons from j. The multi-hop ETX of a node is the sum of the one-hop ETX to its parent and the multi-hop ETX of its parent to the root. To form a minimum-cost spanning tree, non-root nodes always choose the neighboring node with lowest multi-hop ETX as parent. Note that the root node is identified with an ETX of 0.

Because link qualities can be highly dynamic, beacons must be broadcast periodically to inform neighboring nodes of the current ETX. However, it is difficult to determine the proper beacon interval due to the trade-off between fast response and energy conservation. CTP bypasses this problem using adaptive beaconing. If there is no change in the network, after each beacon is sent, the beaconing interval is doubled until an upper limit is reached. Specific events such as detection of loops will reset the beaconing interval.

Finally, loops are detected through datapath validation. Whenever a node transmits a data packet to its parent, it includes its own ETX in the packet header. If a parent receives a packet from its child with an ETX lower than its own, there is a routing loop and the parent node requests a routing update through its own beacons.

C. Castalia

Castalia is an event-driven simulator for wireless sensor networks and body area networks based on OMNeT++ [7]. It has a modular design mimicking the IP protocol stack, enabling users to easily change components based on their simulation needs. For our purpose, we focus on implementing the three algorithms in the application module.

Castalia bases its channel model on the work of Zuniga et al. [11] and uses the lognormal-shadowing model to calculate path loss. The path loss between nodes at distance d is

$$PL(d) = PL(d_0) + 10\eta \log(\frac{d}{d_0}) + X_\sigma \qquad (2)$$

where $PL(d_0)$ is the path loss at a reference distance d_0, η is the path-loss exponent and X_σ is a Gaussian zero-mean random noise with standard deviation σ. The path loss and transmission power are used to calculate *received signal strength indicator*, RSSI, which in turn is used to calculate *signal to interference and noise ratio*, SINR, and to determine packet reception.

To capture the variation in path loss between the two directions of a link, Castalia calculates the average path loss for both direction and adds a Gaussian zero-mean random variable with standard deviation equal to the parameter `bidirectionSigma`. A value of 0 will create symmetric links while a large value will create directed links.

Other features of Castalia include the modeling of temporal variation in network connectivity, calculation of energy consumption and inclusion of mobile sensor nodes. Interested readers are referred to [7] for more details.

III. Algorithm Implementations in Castalia

A. Randomized Gossip

Three issues need to be addressed in order to implement randomized gossip in an actual network: neighbor discovery, communication delay during gossiping and self-termination. We consider each in turn.

We adopt and modify the algorithm presented in [12] for neighbor discovery. Each node needs to broadcast a fixed number of "Hello" beacons. Nodes receiving these beacons add the sender as a potential neighbor, but do not reply. As

network connectivity can vary over time, nodes also rely on overheard data packets to keep track of their neighbors.

After a node i sends a data request to neighboring node j, it starts a timer, `Discovery Delay`. All incoming data requests during this time interval are ignored. If no response is received when the timer expires, node i initiates another data request with another randomly selected neighbor. Nodes with repeated lack of responses are removed from neighbor list.

Finally, we adopt the silencing rule, based on [13], so nodes can determine locally when to stop gossiping. Whenever a node updates its value, it increments a counter if its value does not change by more than a threshold, τ, and resets it to 0 otherwise. When the counter exceeds an upper limit, C, the node stops requesting data, but still replies when it receives a request. For optimal values of τ and C, the randomized gossip is guaranteed to converge with bounded error [13]. For simplicity, we fix C to 1 and let τ be an algorithm parameter in our implementation. The algorithm terminates when all nodes stop gossiping.

B. Broadcast Gossip

Broadcast gossip does not require neighbor discovery nor two-way message exchanges. Therefore, we only include the silencing rule. The algorithm terminates when all nodes stop broadcasting.

C. Collection Tree Protocol

CTP is included as part of standard Castalia release, but only provides data gathering features. We modify this code to perform in-network aggregation and broadcasting. Distributed averaging using CTP follows three basic steps. First, a spanning tree is constructed. Then, nodes send their data to their parent. Once the root receives the data from all its children, it calculates the average, and then broadcasts it down the tree.

The simple algorithm raises several questions. How do nodes know when the tree is constructed, so they can start aggregation? How do nodes know when they have all the data from their children? Finally, how can we guarantee that all nodes receive the calculated average from their parent?

To determine when the tree is constructed, nodes start a timer, `Startup Delay`, after acquiring the initial data value. To determine when they receive data packets from all their children, nodes maintain a second wait timer after which they combine whatever data packets they already have received and send the aggregated data to their parent. We define a parameter, `Max ETX`, and calculate the wait time of node i as the ratio of `Max ETX` to the multi-hop ETX of node i. For root node with an ETX of 0, the timer is initialized to `Max ETX`. As a starting point, we set `Max ETX` to $5n$ so it is much larger than any node's multi-hop ETX. This reduces the probability of discarding delayed packets. Finally, nodes send out an acknowledgment when they receive the calculated average from either their parent or a neighbor with higher ETX (in case the update from the parent is lost). We also introduce a third timer, `Acknowledge Delay`, which specifies how long nodes wait for acknowledgments before rebroadcasting the average, up to a maximum of three attempts. The algorithm terminates when all nodes receive the update or all non-leaf nodes broadcast the average three times, whichever comes first.

IV. Simulation Results

A. Simulation Setup

We evaluate the algorithms' performance in terms of efficiency and accuracy. For efficiency, we consider the average number of transmissions per node, the latency and the energy consumption. For accuracy, we consider two metrics related to the average of nodes' values when the algorithm terminates at time T. First, we calculate the *normalized absolute error* (NAE) defined as $\frac{|x_{\text{ave}}(T)-x_{\text{ave}}(0)|}{x_{\text{ave}}(0)}$[1]. Second, we consider the standard deviation $\|x(T) - x_{ave}(T)\|_2$ of the nodes' final values. Note that CTP is a deterministic algorithm and all nodes are expected to receive the exact average. However, root may miss data packets and certain nodes may not receive the update. Therefore, for CTP, we only measure the efficiency when all nodes receive the exact average and we only measure the accuracy otherwise.

We compare the algorithms' performance in *random geometric graphs* (RGG). [14] shows that a connectivity radius scaling in $O(\sqrt{\log n/n})$ produces a connected network with high probability in an unit square. We extrapolate this result and set the ratio of the effective transmission radius (based on the transmission power) to $\sqrt{\log n/n}$ as the size of our network.

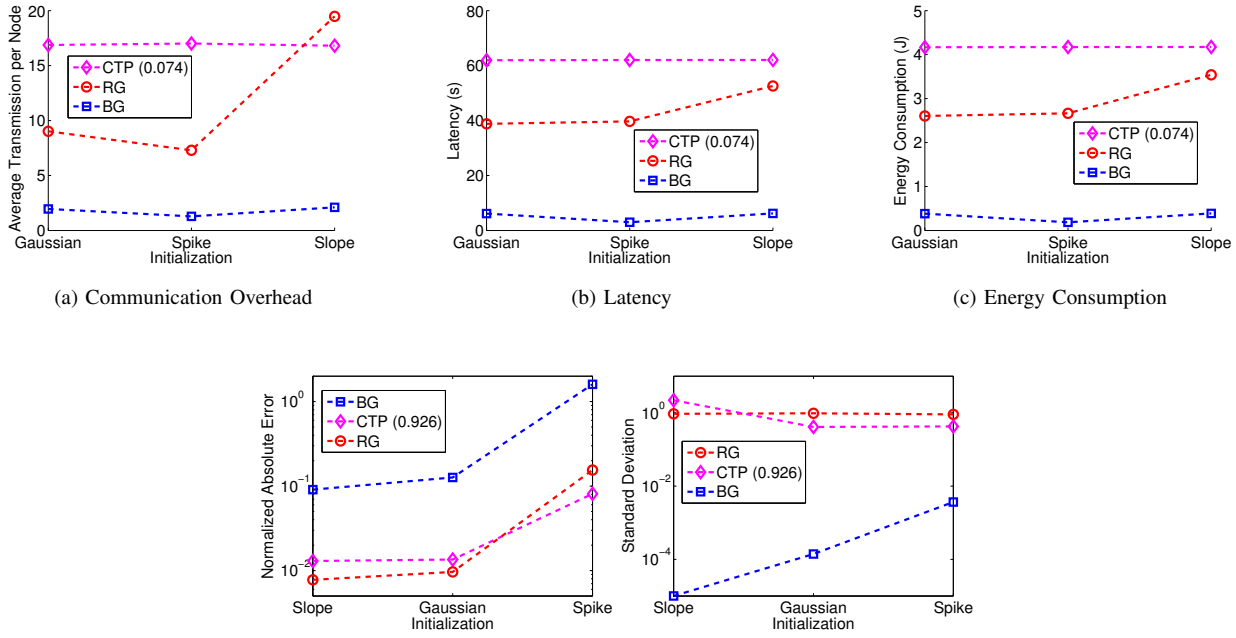
B. Performance Overview

Many algorithm parameters in our implementations are timers and have a direct impact on the latency and hence energy consumption. For a fair comparison, we select values that minimize latency without suffering degradation in accuracy by more than 20%. For all other parameters, we opt for optimal values, if these exist, or select values that minimize communication overhead if trade-off is required. We also consider different value initialization schemes due to their impact on the gossip algorithms [15]. In Gaussian initialization, the initial values are drawn from a Gaussian distribution. In slope initialization, the initial value is equal to the sum of node's X and Y coordinates. In spike initialization, one randomly selected node's value is equal to n and all other nodes' values are equal to 0.

Fig. 1 presents the simulation results. Consider first the efficiency. BG requires the least amount of communication overhead regardless of initialization scheme. As expected, CTP is unaffected by the initialization scheme and generally has the highest communication overhead. RG is in-between but its slow diffusion speed causes very poor performance in slope initialization. BG also has the lowest latency because nodes can transmit data at any time without any restrictions. CTP on the other hand has several wait timers which establish a minimal latency and postpones data transmission whenever the spanning tree requires maintenance. The energy consumption depends directly on the latency as shown in Fig. 1c because the radio is always on. A higher latency translates directly into higher energy consumption.

Consider now the accuracy. BG only converges to the true average in expectation and, unsurprisingly, has a significantly higher NAE compared to RG and CTP. What is interesting is

[1]We slightly abuse the notations. $x_{\text{ave}}(T)$ is either a scalar or a vector.



(a) Communication Overhead (b) Latency (c) Energy Consumption

(d) Normalized Absoulte Error (e) Standard Deviation

Fig. 1. Overall Performance Comparison for Different Initializations, Network Size=100 nodes, Transmission Radius=11m, Network Area=43x43m^2. The number in brackets is the fraction of 500 CTP simulation runs applicable to calculate each performance metric

that, for Gaussian and slope initializations, RG has a lower NAE compared to CTP. As both algorithms preserve the network sum in theory, the error only occurs when data packets are lost. Our simulation suggests that lost data packets have a bigger impact on CTP than on RG as more nodes are affected. For standard deviation, we note the inverse trends. BG has the lowest standard deviation, followed by RG and CTP.

V. Conclusion

We conduct a series of simulations to evaluate and compare the performance of broadcast gossip, randomized gossip, and the collection tree protocol for distributed averaging. Overall, BG has the highest efficiency because it does not require any prior setup and can exchange data without any restriction, but it has the highest NAE because it does not preserve network sum. RG and CTP are hindered by various timers and unexpected delays which degrade their performance.

References

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508 – 2530, June 2006.

[2] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2748 –2761, July 2009.

[3] D. Ustebay, B. Oreshkin, M. Coates, and M. Rabbat, "Greedy gossip with eavesdropping," *Signal Processing, IEEE Transactions on*, vol. 58, no. 7, pp. 3765 –3776, July 2010.

[4] C. Yanez-Marquez, I. Lopez-Yanez, O. Camacho-Nieto, and A. J. Arguelles-Cruz, "Bdd-based algorithm for the minimum spanning tree in wireless ad-hoc network routing," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 11, no. 1, pp. 600–601, 2013.

[5] D. Montoya and J. Ramirez, "A minimal spanning tree algorithm for distribution networks configuration," in *Power and Energy Society General Meeting, 2012 IEEE*, 2012, pp. 1–7.

[6] B. Hakoura and M. Rabbat, "Data aggregation in wireless sensor networks: A comparison of collection tree protocols and gossip algorithms," in *Electrical Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, May 2012, pp. 1 –4.

[7] B. Athnassios, "Castalia a simulator for wireless sensor networks and body area networks," Mar 2011, available at http://castalia.npc.nicta.com.au/pdfs/Castalia%20-%20User%20Manual.pdf.

[8] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The collection tree protocol (ctp)," Feb 2007, available at http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09, 2009, pp. 1–14.

[10] U. Colesanti and S. Santini, *A performance evaluation of the collection tree protocol based on its implementation for the castalia wireless sensor networks simulator*. ETH, Department of Computer Science, 2010.

[11] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, 2004, pp. 517–526.

[12] "Implementation of average consensus protocols for commercial sensor networks platforms," in *Grid Enabled Remote Instrumentation*, ser. Signals and Communication Technology, F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, Eds., 2009.

[13] A. Daher, M. Rabbat, and V. Lau, "Local silencing rules for randomized gossip," in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, June 2011, pp. 1 –8.

[14] M. Penrose, *Random geometric graphs*. Oxford University Press Oxford, 2003, vol. 5.

[15] A. Dimakis, A. Sarwate, and M. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *Signal Processing, IEEE Transactions on*, vol. 56, no. 3, pp. 1205 –1216, March 2008.