8-11-2015

INDIGO: An In-Situ Distributed Gossip System Design and Evaluation

Paritosh Ramanan

INDIGO: AN IN-SITU DISTRIBUTED GOSSIP SYSTEM DESIGN AND EVALUATION

by

Paritosh P. Ramanan

Under the Direction of Wenzhan Song, PhD

ABSTRACT

Distributed Gossip in networks is a well studied and observed problem which can be accomplished using different gossiping styles. This work focusses on the development, analysis and evaluation of a novel in-situ distributed gossip protocol framework design called (INDIGO). A core aspect of INDIGO is its ability to execute on a simulation setup as well as a system testbed setup in a seamless manner allowing easy portability. The evaluations focus on application of INDIGO to solve problems such as distributed average consensus, distributed seismic event location and lastly distributed seismic tomography. The results obtained herein validate the efficacy and reliability of INDIGO.

INDEX WORDS: distributed computing, simulation, system design, gossip protocols

INDIGO: AN IN-SITU DISTRIBUTED GOSSIP SYSTEM DESIGN AND EVALUATION

by

Paritosh P. Ramanan

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Masters of Science

in the College of Arts and Sciences

Georgia State University

2015

INDIGO: AN IN-SITU DISTRIBUTED GOSSIP SYSTEM DESIGN AND EVALUATION

by

Paritosh P. Ramanan

Committee Chair: Wenzhan Song

Committee: Xiaolin Hu

Xiaojun Cao

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2015

DEDICATION

To Amma, for helping me stay strong,

To Papa, for egging me on when I was down,

To Prerana, for cheering me up when I was about to give up

Thanks!

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deep gratitude to Prof. Wenzhan Song for his continued trust in my abilities and his constant support in my research endeavors. I would also like to thank my colleagues at the Sensorweb Research Laboratory at Georgia State University for their support. I would like to also express my gratitude to my parents for their constant encouragement and moral support in helping me accomplish my goals.

Contents

List of Figures

1 Introduction

Sensor networks are becoming an important part of monitoring activities across various interdisciplinary domains. They have been successfully applied to solve problems like seismic activity monitoring and tomography[1], exploratory geophysics [2], wildfire and wildlife monitoring[3] among many things. Extracting optimal performance from sensors has always been a challenge[4] and it has led to a flurry of active research in recent times. Sensor networks come with their own set of constraints which cannot be overlooked. For instance, sensor networks often come with a very limited energy source, which makes it imperative to use system resources judiciously as well as keep communication costs at as minimum a level as possible. It is also quite likely that due to energy constraints the sensor network might be able to provide only limited amount of bandwidth for data transfer which makes communication a more precious affair.

Therefore, recent state-of-the art research in the area of sensor networks suggests that the latest trends appear to be focussing striking a balance between power consumption attributed to communication and system utilization. With sensor nodes becoming computationally more powerful and less resource hungry, the bottleneck of communication as a barrier for efficient utilization of system resources seems to persist. Due to the rise of increasingly power efficient sensor nodes it now makes more sense in some cases to delegate computation based tasks to the nodes themselves than to have them use up precious resources to depend on a central entity for computation. In recent times, the interleaving of the computational aspect of sensor networks with that of physical processes such as sensing has opened up new research avenues like *Cyber-Physical Systems* [5] and *in-network computing* [6].

One such research problem in which the centralized approach to problem solving is less efficient than an in-network approach is that of achieving consensus in a sensor network. Consensus problem in sensor network deals with each node arriving at a consensus of a measured parameter solely on the basis of exchange of information with its neighbor nodes. In order to achieve consensus in a network, an averaging problem of the following form must

be solved.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1}$$

where, $x_1, x_2 \ldots x_n$ are the individual observations recorded by each of the n nodes in the network. For instance, a bunch of nodes measuring the temperature of a room may do so by relaying their measured values to a central sink or exchange information amongst themselves and arrive at an average which would be the consensus. Mutual exchange of information is highly beneficial as it cuts down on otherwise expensive multi-hop communication to the sink. Although solving such a consensus problem is a trivial task in a centralized setup, there have been quite a number of research endeavors in the recent past which propose a distributed decentralized approach.

As an extension of the distributed consensus problem, the distributed consensus optimization problem has also been studied. Distributed consensus optimization is of the following form.

$$\begin{aligned} \text{minimize } F(x) &= \frac{1}{n} \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to } x_i &\in \chi_i \end{aligned} \tag{2}$$

where x_i,χ_i andf_i are the local estimate of the observed value, the set of constraints and the local objective function on the i^{th} node respectively. Distributed consensus optimization involves using consensus to propogate information to other nodes in the network and then solving a local optimization problem based on f_i known only to the i^{th} node.

It is in this regard that the problem of *asynchronous distributed gossip* has been proposed for consensus as well as consensus optimization in sensor networks. The idea is to be able to solve a computationally intensive problem by mutual exchange of information among nodes. The very basic case of distributed gossip is the distributed consensus problem. By attacking the distributed consensus problem, we can expect to solve much more computationally intensive problems.

1.1 Purpose of Study

The distributed gossip approach is a very promising one in the world of Cyber-Physical Systems [7]. In the recent past, a key implementation of CPS has been in the area of seismic monitoring [8][1]. As an extension of the above work, research is being conducted for performing seismic tomography [9]. Seismic tomography is the process of determining with good accuracy, a profile of the earth under the surface. It is extremely helpful in the area of geophysics for disaster planning and preparedness. A tomography problem can be modelled as a linear least squares problem of the following form.

$$x_{LS} = \arg\min_{x} \frac{1}{2}||Ax - b||_2^2 \tag{3}$$

where $x \in \mathbb{C}^n$, $A \in \mathbb{C}^{m \times n}$ and $b \in \mathbb{C}^m$

Currently, most tomography approaches use a centralized technique where information is relayed to a sink where the system of equations represented by Equation 3 is formulated and solved[9]. However, with distributed gossip, one can hope to minimize this cost, make the system and the network more efficient and expect it to be more reactive. In this regard distributed gossip techniques have an edge over existing algorithms.

Although asynchronous distributed gossip protocols have been well studied, there has been little development in terms of adapting these protocols to the realm of real world WSNs. There is also a dire need to have a flexible testing environment for distributed algorithms which could interface with real data already available and would enable one to observe the performance and behavior of the algorithm without the need for actual field deployment. This paper talks about the design, development and analysis of an In-Situ Distributed Gossip(INDIGO) framework for sensor networks. The INDIGO framework caters to the aforementioned needs and enables one to test the robustness and scalability of algorithms.

The rest of the paper is organized as follows. Section 2 talks about the existing state-of-the-art gossip algorithms which INDIGO implements. Section 3 presents an overview of

the random and broadcast gossip protocol as implemented under INDIGO and presents an emperical analysis of the performance of the framework on the basis of some newly introduced parameters like *sleep time* and *wait time*. Section 4 talks about how we have implemented the aforementioned algorithms both on system and testbed platforms. Section 5 demonstrates the various results we have obtained using INDIGO and Section 6 concludes the study by highlighting the various aspects of the study as well as pointing at the future direction of research in this area.

2 Related Work

Distributed Gossip in sensor networks is a well studied problem. The types of gossip can be broadly categorized into three types *i.e. broadcast, random and geographic* [10][11][12]. In this study we limit ourselves to the domain of only broadcast and random gossip and describe the various published works which have inspired this study. As already mentioned the main aim of this study is to implement established gossip algorithms on a system level and help in observing their behavior in different scenarios. Random Gossip was first proposed by Boyd *et al.* [10] based on the asynchronous time model. Random Gossip chooses nodes at random from its neighbors to exchange information and calculate the average. The paper proves that the algorithm converges to the consensus which is the average almost surely. The important thing about random gossip is that at any time instant there can be only one exchange taking place between two particular nodes. This implies that while the process of averaging or gossip is going on, no other third node can indulge either of the nodes in gossip. It is only after both the nodes have successfully performed gossip that they are free to choose other nodes to perform gossip with at random. The work done by Aysal *et al.* discusses the broadcast gossip algorithm [11]. This paper takes the above work by Boyd *et al.* a step further and proves that if the node were to broadcast its values to all its neighbors, then the algorithm converges in expectation. This paper also uses the asynchronous time model and uses a similar technique of a stochastic mixing matrix with a different set of constraints.

Although both random and broadcast gossip aim to achieve average consensus among nodes, their style of performing gossip is radically different. While random gossip chooses to perform gossip with its immediate neighbors, a node can only perform gossip with only one other particular node at any given time. Broadcast gossip on the other hand performs gossip by broadcasting its values to its neighbors. While random gossip is suited to any type of network with a static topology, broadcast gossip is more relevant in case of wireless sensor networks where the underlying communication pattern is broadcast driven.

The work done by Dimakis *et al.* [13] presents a broad overview of the recent developments in the area of gossip protocols. It describes the convergence rate of gossip protocols in relation to the number of transmitted messages as well as energy consumption and also discuss about gossip characteristics over wireless links. Further, the work done by Denantes *et al.* [14] presents an interesting evaluation on a mathematical basis of certain metrics which may be useful in choosing an apt algorithm for performing distributed gossip. Instead of focussing on a time-invariant scenarios, these metrics are evaluated on the basis of time-varying networks culminating in the provision of an upper bound on the convergence speed.

The work done by Braca *et al.* [15] investigate an important and crucial problem of when to begin averaging and when to end sensing. They propose an alternative novel approach of running consensus where the sensing and averaging happen in a simultaneous fashion. The paper [16] provides a very novel application of gossip protocols. By investigating the problem of consensus in a multi-agent system, it demonstrates a practical application of gossip protocols towards a Distributed Flight Array (DFA). DFA is a set of multiple agents, which co-ordinate amongst themselves to arrive at a consensus and fly in a variety of combinations While both the works [11] and [10] present an astute theoretical analysis of their respective gossip technique, they make a number of assumptions which may not hold good in case of a real implementation.

The work done by Tsianos *et al.* [17] presents a practical approach for asynchronous

gossip protocols but they do not use a bi-directional mechanism and opt for a one-directional variant instead and their evaluations are performed on an MPI cluster which has different constraints from an actual WSN.

2.1 Motivation and Key Contribution

Therefore, there have been very few attempts at providing a real implementation framework for a WSN which could provide unmatched flexibility and ease for evaluation and testing of various distributed algorithms. It is these aspects which serve as a motivation for the development and design of the INDIGO framework which have been expounded in a systematic way. Firstly, the INDIGO protocol design overview is given which provides a practical and a near-accurate way of realizing both random and broadcast gossip protocols on an actual system setup. We also present an analysis of the effect certain newly introduced parameters like sleep time and wait time have on the framework performance and attempt to find a *sweet spot* with respect these. Further, the implementation details are elaborated upon which describe the framework on a system platform utilizing the standard TCP/IP stack and on a testbed platform comprising of the BeagleBone Black coupled with an XBee radio. Lastly, the protocol framework is evaluated on the basis of various applications and the performance of the algorithms analyzed.

3 INDIGO Protocol Overview and Design

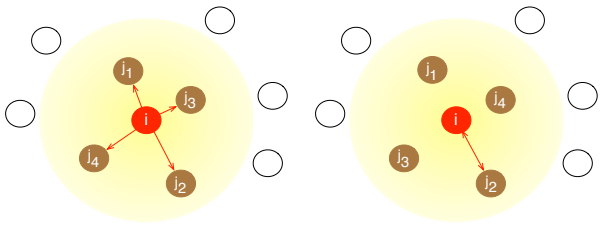
As described in the previous section, gossip protocols can be broadly categorized into *random* and *broadcast* gossip protocols. In this section is presented a novel and practical framework design that aims to bring forth the true spirit of the aforementioned protocols. The idea is to create a flexible framework design which can be extended into a platform on the basis of which various novel algorithms can be evaluated upon.

Let us consider a graph $G(V, E)$, with V, E being the vertex set and edge set respectively. Since distributed gossip occurs among neighbors, we denote the neighborhood of any node

$i \in V$ as follows,

$$\mathcal{N}_i = \{j | j \in V, \mathcal{W}_{ij} = 1\}, \tag{4}$$

where $\mathcal{W}$ is the *adjacency matrix* of graph G. In the design of a gossip protocol framework,



(a) Broadcast Gossip (b) Random Gossip

Figure 1: Illustration of Random and Broadcast gossip with respect to Node i and its neighborhood $j_k \in \mathcal{N}_i, \forall k \in \{1, |\mathcal{N}_i|\}$

an important feature of *exclusivity* needs to be preserved to reflect the true nature of the process. Exclusivity implies that a node when in the process of performing gossip cannot entertain gossip requests from a third party node, thereby discarding any other packets until the ongoing gossip exchange succeeds. An important outcome of exclusivity is that the node which is soliciting has no way of knowing whether its destination has received its request or not. In a real setting it is important to take into account the fact that, packets may get lost and moreover, even if the packet is received, the destination might be involved in gossip with some other of its neighbor and may simply discard this request. If these situations are not handled properly, the gossip protocol may never terminate or worse it may lead to contradictory results. In order to solve this problem a concept of *wait time*, denoted by σ is introduced. It denotes the duration of time any node waits before it deems the gossip exchange to have failed. Wait time insulates nodes from the phenomenon of waiting forever to hear from their solicited neighbors and also handles the aspect of packet loss. With the wait time concept in place, if the packet has not been received or has been discarded by the receiver, the sender can resume gossip afresh.

Another important feature that needs to be preserved is the *stochastic nature* of the gossip

process. There has to be a degree of randomness associated when a particular node begins gossip. Failure to maintain this feature would lead to a deterministic output. Absence of this feature may also cause deadlock among nodes or cause a heavy rate of failure of gossip exchanges. To maintain stochastic behavior a parameter known as *maximum sleep time*, denoted by ρ has been introduced which is nothing but an upper bound on the random interval of time a node sleeps before attempting a gossip exchange.

We now describe the various terminologies related to both *random* and *broadcast* gossip and proceed to give a detailed description of the sequence of events in each.

- x_{self}: The estimate of a node's measurement where $x_{self} \in \mathbb{C}^{n\times 1}$
- σ: The maximum duration of time after which a gossip exchange is deemed a failure.
- ρ: The upper bound on the random interval of time a node sleeps before initiating gossip.
- $\mathcal{M}$: The maximum number of gossip updates to be performed by all nodes.
- $\mathcal{N}_i$: The neighborhood of node i.
- $recv(k, x_k)$: An estimate x_k recieved from node k.
- $send(k, x_{self})$: A node's self estimate *unicasted* to node k.
- $\chi_{self} = [x_j \dots x_{j+m}]$, matrix of values recieved from m nodes to be averaged where $\chi \in \mathbb{C}^{n\times m}$
- $broadcast(x_{self})$: A node's self estimate broadcasted to all neighbors.

3.1 Random Gossip

Based on the above features and using aforementioned terminologies we have Algorithm 1 which describes the Random Gossip protocol encapsulated as a function. In the beginning of each batch of gossip each node goes to sleep for a random interval of time $t \leq \rho$. A

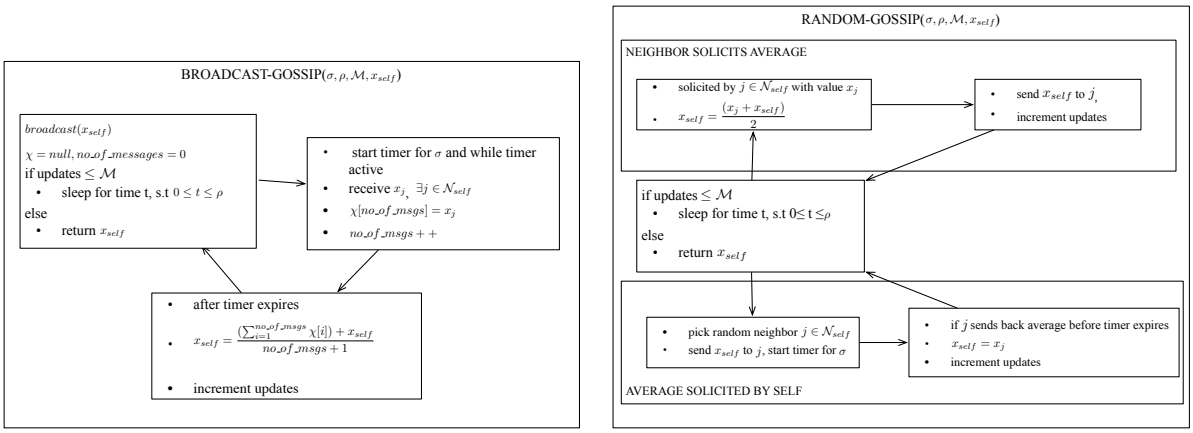
Algorithm 1 Random Gossip Algorithm

```
function RANDOM-GOSSIP (σ, ρ, M, x_self)
    while updates < M do
        sleep for time t, s.t. 0 ≤ t ≤ ρ
        if solicited by j ∈ N_self with value x_j then
            x_self = (x_j + x_self)/2
            send(j, x_self)
            updates ← updates + 1
        else
            pick random neighbor j ∈ N_self
            send(j, x_self) and start timer for σ
            if recv(j, x_j)&!timer.expire() then
                x_self = x_j
                updates ← updates + 1
            end if
        end if
    end while
return x_self
end function
```

node wakes up from sleep and chooses a random peer from its routing table and *solicits* an average. It starts a timer for $t \leq \sigma$ in order to wait for the solicited node to respond. If a node is in solicitation mode, it will discard any other solicitation request by a third party node. The σ timer expires with the solicited node failing to respond. In such a case the node again goes to sleep for a random interval of time $t \leq \rho$. The solicited node responds before timer expires. It updates its current value with the newly received value and goes to sleep for time $t \leq \rho$. A node wakes up from sleep and finds that there is already a request for average by one of its peer. In such a case the node performs the average and sends back the result to the solicitor node. This process is summarized by Figure 2(b) which summarizes the sequence of events disscussed in Algorithm 1.

3.2 Broadcast Gossip

Broadcast gossip varies from random gossip in its demand for exclusivity. Since broadcast gossip exploits the underlying broadcast nature of the network, there is no explicit requirement for exclusivity. However, in broadcast gossip, a node still needs to maintain the



(a) Broadcast Gossip (b) Random Gossip

Figure 2: Flow Diagram depicting Broadcast and Random Gossip algorithm

stochastic nature and for this purpose the concept of *maximum sleep time* is maintained. Also, in broadcast gossip, a node is expected to wait for receiving values from its neighbors. During this process, there should be a way to determine when to stop accepting the values and perform the average. This can be done in two ways, either wait for a fixed number of neighbors to respond and then do the average or wait for a fixed amount of time and do the average with whatever values have been received until then. Logically, the latter is a better way due to many reasons. Firstly, this technique does not depend on the node degree. Secondly, it does not go into an indefinite wait on not receiving anything from a fixed set of neighbors. Lastly, it preserves the stochastic and asynchronous nature of the algorithm. Therefore, we incorporate the concept of *wait time* to mark the cut-off time for performing the average. While the average is being computed any received requests will be dropped. Based on the above features Algorithm 2 presents the algorithm for the broadcast gossip protocol encapsulated as a function. In Broadcast Gossip too each node goes to sleep for a random interval of time $t \leq \rho$. A node that has just woken up from sleep and broadcasts its value to neighbors. It then waits for interval of time $t \leq \sigma$. It performs the average with whatever values have been received in the interim period and again goes to sleep for random interval of time $t \leq \sigma$. Figure 2(a) summarizes the sequence of events disscussed

Algorithm 2 Broadcast Gossip Algorithm

function BROADCAST-GOSSIP $(\sigma, \rho, \mathcal{M}, x_{self})$
  while $updates < \mathcal{M}$ **do**
    $broadcast(x_{self})$
    sleep for time t, s.t. $0 \leq t \leq \rho$
    $\chi \leftarrow null$
    $no\_of\_msgs \leftarrow 0$
    start timer for σ
    while !$timer.expire()$ **do**
      $recv(j, x_j), \exists j \in \mathcal{N}_{self}$
      $\chi[no\_of\_msgs] = x_j$
      $no\_of\_msgs \leftarrow no\_of\_msgs + 1$
    end while
    $x_{self} = \frac{(\sum_{i=1}^{no\_of\_msgs} \chi[i]) + x_{self}}{no\_of\_msgs+1}$
    $updates \leftarrow updates + 1$
  end while
return x_{self}
end function

in Algorithm 2. We will now turn our attention to the effects ρ and σ have on the gossip performance.

3.3 Sweet Spot Analysis

It is of primary interest to determine whether these parameters have any bearing on the success of a gossip exchange. Moreover, it is also of importance to find out whether there exists a *Sweet Spot*, *i.e* a range of values of ρ and σ value which could yield a near optimal probability of success. To accomplish this, numerous experiments were conducted with $0 \leq \rho \leq 10$ on a 3×3 simulation setup configured for random gossip. We varied the value of σ with respect to ρ and plotted the average probability of success of each gossip exchange. The result is presented in Figure 3 Figure 3 depicts the p_s, the probability of success on the y-axis and the ρ values on the x-axis respectively. The probability of success p_s is determined by the relation,

$$p_s = \sum_{i=1}^{n} \frac{N_{s_i}}{N_{t_i}} \tag{5}$$

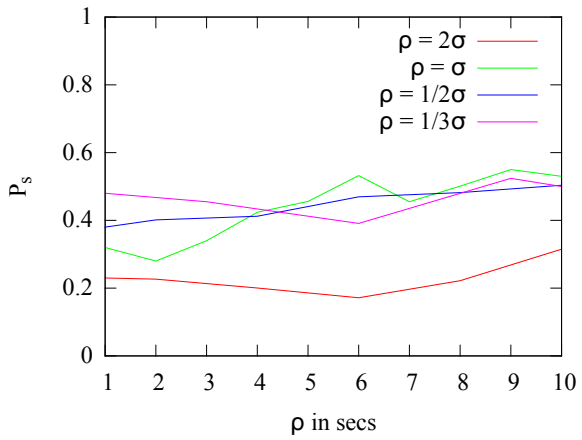


Figure 3: Sweet Spot Analysis

where N_{s_i} is the total number of successful gossip attempts and N_{t_i} is the total number of attempts obtained on the i^{th} node. Each curve in Figure 3 represents a particular relation between ρ and σ. With σ being the dependent variable and ρ being the independent variable, we collect values for a variety of combinations of ρ and σ. From the figure, it can be observed that there indeed exists a sweet spot for the set of relations $\rho = k\sigma$ where $0 \leq k \leq 1$ while for the relation $\rho = 2\sigma$, the value of p_s turns out to be sub optimal.Although this experiment is in no way exhaustive and further trends may emerge on detailed analysis with other values of ρ, σ, we can draw a number of inferences from this figure. Firstly, the trends follow the intuitive notion that if the maximum time a node can sleep is less than the maximum time it is ready to wait then the probability of success increases and vice versa. Secondly, with further reduction in the ratio $\rho : \sigma$, there appears to be a saturation point and further decrease will not yield greater improvement. Lastly, for this network setup, the region around $\rho \geq 6$ seems to be a favorable position because in all relations, there is a noticeable improvement of performance. From this analysis it becomes quite clear that ρ, σ do have an effect on the probability of success of gossip exchanges and there does exist a *sweet spot* for these values.

In the following sections, we discuss the implementation details and a testbed setup description of INDIGO before proceeding forward to analyze the results in the form of various case studies.

4 System Implementation and Testbed Design

In this section, we describe in greater detail, the technical aspects of two evaluation platforms, *i.e.* a system platform and a testbed platform. System platform is intended to provide a generic evaluation platform using the standard TCP/IP stack based wireless mesh network. Although for evaluation purposes, such a robust system platform should be sufficient, we also require a testbed platform to emulate on-field environments using the very same hardware which would be used for deployment. Hence we propose and eventually describe a testbed platform as well comprising of BeagleBone Black coupled with an XBee radios. Since the testbed platform is an indoor setup, the nodes form a network which resembles a complete graph due to close radio proximity. A unique feature of INDIGO is its platform agnostic way of functioning which provides a flexible, rich and diverse testing environment. We draw a comparison between the two before proceeding towards evaluation with the help of case studies.

4.1 System Design

We utilize a mesh network model for implementing INDIGO. Mesh networks are those in which each node not only communicates with its peers but also serves as a relay point by facilitating the transfer of messages between two different nodes. Since maintaining proper end-to-end connectivity in a mesh network is a costly affair due to low link reliability, we employ a mechanism known as the Bundle Layer which is a *delay tolerant* technique of transmission. The key objective behind the Bundle Layer is to improve reliable transmission over wireless media over the TCP/IP stack. To accomplish this the Bundle Layer breaks down the notion of end-to-end among the various hops in between which would significantly reduce retransmission of packets. Under the Bundle Layer lies the actual transport layer which uses normal TCP and beneath which runs a distance vector routing protocol known as BATMAN (Better Approach to Mobile Ad-hoc Networking)[18]. The advantage of BATMAN lies in the fact that routing overhead is minimized by maintaining only the next hop neighbor

entry to forward messages to instead of maintaining the full route to the destination. The Bundle Layer along with BATMAN ensure reliable transmission of messages between source and destination.

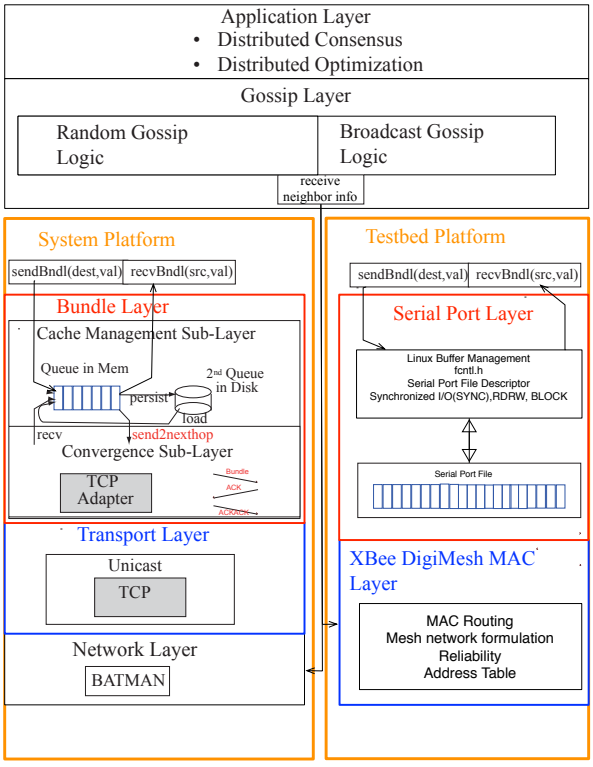
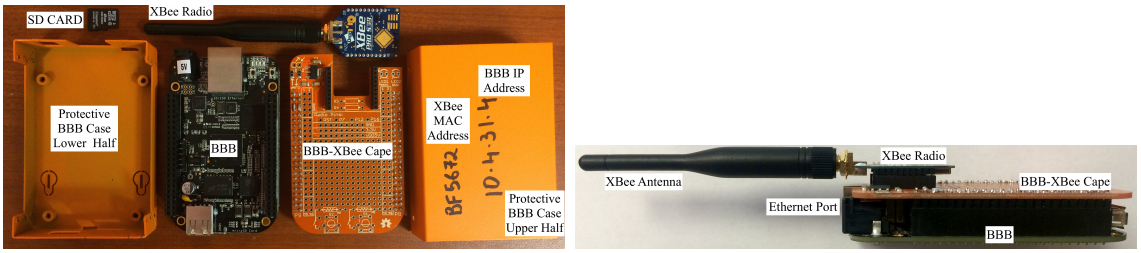


Figure 4: System design and Testbed Design : A comparison

4.2 Testbed Design

Our testbed setup comprises of the BeagleBone Black(BBB) interfaced with the XBee radio. The BBB is an inexpensive small palm sized computer which runs the Angstrom operating system which is a flavor of embedded linux. The BBB has a memory of 512 MB and has a single core CPU with clock rate of 1GHz. For radio communication we use the XBee PRO S3B 900 MHz version which is mesh network capable. The module comes with an onboard flash memory of 512 bytes and has a Freescale MC9S08QE32 microcontroller which allows for programmable control. Various network functionality have been abstracted by XBee including routing and mesh network capability. The programmable control allows us to operate the



(a) Blowup of the BBB-XBee Node (b) Interfacing of BBB with XBee

Figure 5: Actual Testbed Node setup involving BBB and XBee

XBee in a variety of modes which makes it application flexible. Among the most important features, we could set the Power Level(PL) parameter which indicates the amount of power consumed during transmission. During run time, we can issue commands encapsulated in a pre-decided frame and pass it on to the device and expect to get encapsulated replies. Through programmable control one can even choose from a variety of sleep patterns already offered by the device. This greatly simplifies the process of deployment by having a robust network maintenance framework. Figure 5(a) presents a blow up of the different components which go into making one node on our testbed platform, while Figure 5(b) shows how the various hardware components fit together. For interfacing the BBB with the XBee it is configured as a peripheral UART (Universal Asynchronous Receiver Transmitter). Using the device tree overlay we are able to bring up a serial port for communication with the XBee. This serial port is memory mapped to the on board memory of the underlying XBee. Once this configuration is in place, we can communicate with the XBee and its peers through this serial port. For accomplishing this we have developed a host of XBee specific functions for sending and recieving information. The hallmark of these functions is that they allow for a flexible operation of the XBee with varying message types and message lengths. Figure 6 provides an overview of the XBee message structure for conducting distributed gossip. Another interesting point to note is that through a configuration of the serial port through the POSIX compliant serial port libraries in Linux, we can man this serial port with the effect of achieving simultaneous receiving and transmitting of data.

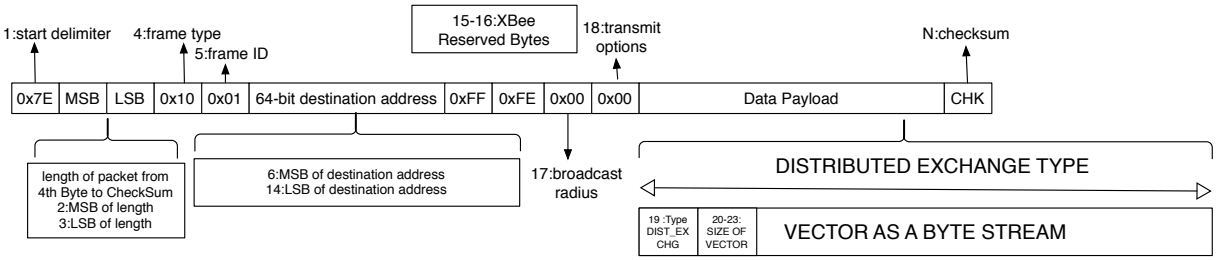


Figure 6: Distributed Gossip XBee Message Structure

5 Case Studies

This section focusses on the application based evaluation of INDIGO. We focus on two forms of evaluations.

- TYPE 1: *Distributed consensus* gathering of the form.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6}$$

- TYPE 2: *Distributed consensus optimization* of the form.

$$\begin{aligned} &\text{minimize } F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) \\ &\text{subject to } x_i \in \chi_i \end{aligned} \tag{7}$$

We start with the simple case of distributed consensus gathering which is of TYPE 1 in both the system as well as the testbed platform. Then we move to more complex cases like distributed event location on the testbed and finally to distributed tomography computation on a simulation setup which are problems of TYPE 2. For the system evaluation platform we employ a network emulator named CORE [19]. CORE creates virtual Network Interface Cards (NICs) for a specific network on a single host machine allowing emulation of actual network settings. The advantage of CORE is that traditional Unix like environment can be

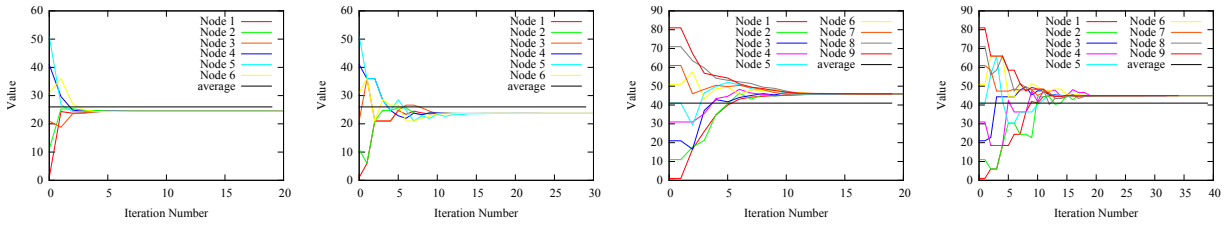
obtained on each of the nodes in the network which makes porting code to actual physical devices from the virtual nodes straightforward. For the testbed evaluation platform, we use the testbed consisting of 6 BBBs each connected to an XBee. The BBBs are connected to an Ethernet switch which is in turn connected to a host machine. While the distributed gossip occurs amongst the BBBs using the XBee radio, the Ethernet interface helps maintain control of the gossip process with a rich set of scripts via the host machine.

5.1 Simple Consensual Average

Distributed gossip protocols are evaluated [20] on the basis of their ability to converge to consensus based on two different types of initializations of data.

- *Slope initialization*: All nodes in the network are initialized with a scalar value $x = k * nodeId$, where k is constant for all the nodes. The resultant set of values form a slope on a network of nodes. It is expected that on termination of the gossip protocol, the slope will have given way to a flat surface tending to average of the initial set.
- *Spike Initialization*: All but one of the nodes is initialized to a very high scalar value and the rest are set to 0. With this initialization it is expected that all the nodes will have the average of the spike value on termination.

5.1.1 Slope Initialization

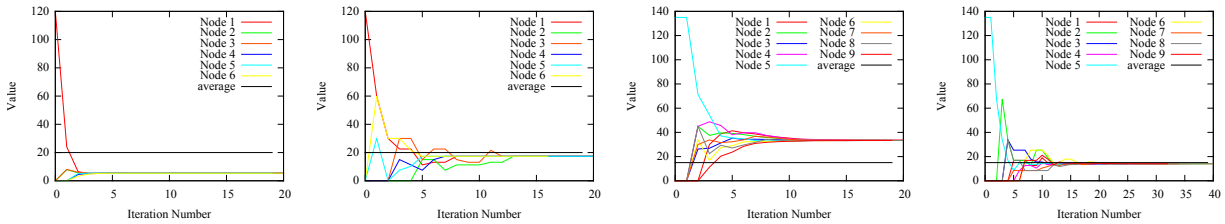


(a) Broadcast Gossip Slope Intialization on Testbed (b) Random Gossip Slope Intialization on Testbed (c) Broadcast Gossip Slope Intialization on System Emulation (d) Random Gossip Slope Intialization on System Emulation

Figure 7: Results of Slope Initialization

Figure 7 depicts the gossip trends arising out of slope initialization on the testbed platform and the system platform. This experiment was performed on the testbed platform mentioned in Section 4 using 6 Beaglebone Blacks and XBees with $\rho = 3$ and $\sigma = 3$ and on system emulation platform using the same values. As can be seen from the figure, the gossip yields very good results, with the protocol converging to a consensus which falls under a very close margin of the actual average.

5.1.2 Spike Initialization



(a) Broadcast Gossip Spike Intialization on Testbed (b) Random Gossip Spike Intialization on Testbed (c) Broadcast Gossip Spike Intialization on System Emulation (d) Random Gossip Spike Intialization on System Emulation

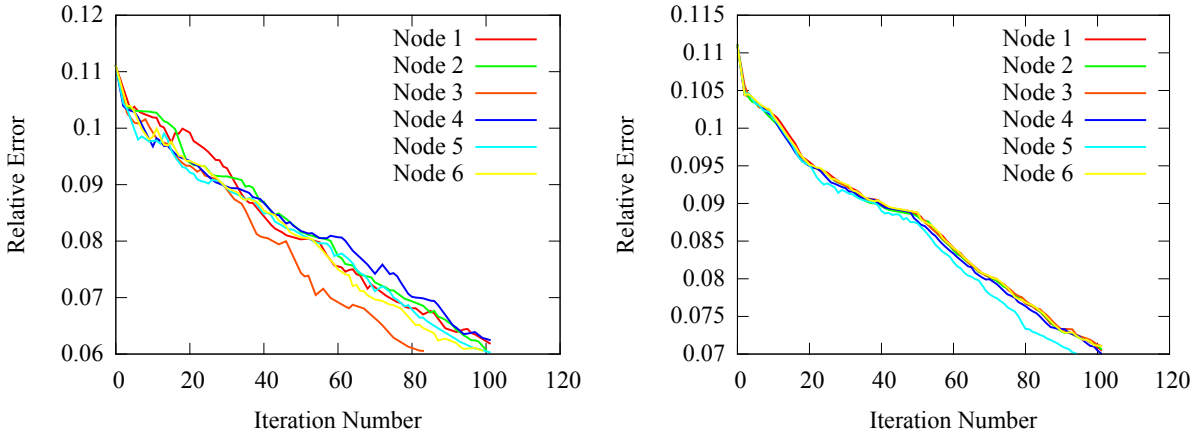
Figure 8: Results of Spike Initialization

Figure 8 depicts the gossip trends using a spike initialization. While the random gossip scheme performs well and converges to consensus within a close margin of average, the broadcast gossip converges to a consensus but isn't close to the actual average. This is expected behavior as it has been anticipated in [11] that broadcast gossip only converges to average consensus in expectation.

5.2 Distributed Event Location

Distributed Event Location is a process of localizing a seismic event. This is done through a process known as Geigers method [21] wherein a system of equations of the form represesented in Equation 7 is solved. Therefore, distributed event location falls under TYPE 2. We can solve these system of equations using any least squares technique like Bayesian ART [22]. The whole idea behind this experiment is to make the process of Event Location as

mentioned in [21] distributed. For performing this experiment we used the system testbed which comprised of 6 Beaglebone Blacks communicating with each other using the XBee radio. Figure 9 represents an experiment involving random and broadcast gossip while



(a) Random Gossip Trends

(b) Broadcast Gossip Trends

Figure 9: System testbed results of Distributed Event location using random and broadcast gossip

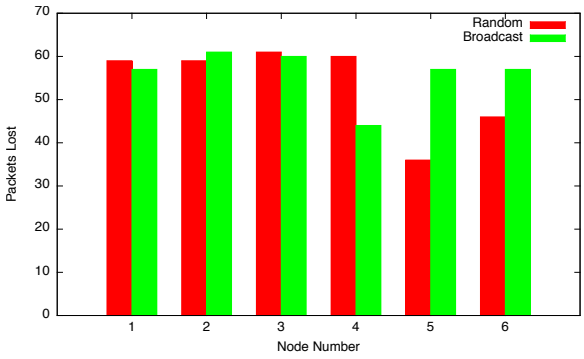


Figure 10: Packet Loss of Random and Broadcast Gossip while performing Distributed Event Location with 100 iterations

performing distributed event location for one particular event where the y-axis represents the relative error η

$$\eta_i = \frac{||x_i - x^*||}{||x^*||}, \tag{8}$$

where i is the iteration number and x^* is the ground truth. As a result, each node solves its local system of equations referred to by Equation 7 by using an initial guess. Next,

it generates the new x value and performs gossip with some other of its neighbor node. After the completion of this gossip exchange, it uses the obtained x value as basis to again generate a new estimate of x and the process continues till a given tolerance is reached or the maximum number of iterations are reached. This technique embodies a true *asynchronous* gossip approach as the objective function being solved is directly coupled with exactly one gossip update. With this result, it becomes apparent that distributed event location can be fruitfully applied to INDIGO.

5.3 Distributed Seismic Tomography

Another application of INDIGO is to perform distributed seismic tomography [23] which is a TYPE 2 problem and can be modelled as a distributed consensus optimization problem. Centralized seismic tomography involves solving an objective function of the type,

$$\begin{aligned} &\text{minimize } ||x|| \\ &\text{subject to } Ax = b \end{aligned} \tag{9}$$

where $x \in \mathbb{C}^n$, $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^m$ and $i \in \{0, n\}$. In distributed seismic tomography, k^{th} node has its own b^k and A^k and an initial x^k_{init} which it uses to solve a *local optimization problem* (LOP). referred to by Equation 9. However, in the distributed scenario, the k^{th} node performs a gossip update with its neighbor(s) to obtain a new estimate of its value x^k. This value is inturn used to solve the local optimization problem and the process repeats till a threshold is reached. In other words, the distributed gossip and the LOP are tightly coupled leading to true *asynchronous* behaviour.

To execute this problem on INDIGO, we used a synthetic data model. Our resolution was 16×16, which meant that our x matrix was of size 256. Our setup was simulated on a network comprising of 49 nodes, arranged in a grid topology. The key idea being that a node initially generates an estimate of vector x using Bayesian ART to solve the LOP. It performs performs gossip with neighbor(s) and obtains a new value of x. This value is then

used as a basis for computing the next estimate of x and the process repeats.

We evaluate our results based on two parameters, η being the relative residual and β being the relative error with respect to the ground truth. For the ground truth, we use the least square solution of the centralized form of $Ax = b$, denoted by x^{gt}

$$\eta_i = \frac{||Ax^i - b||}{||b||} \tag{10}$$

$$\beta_i = \frac{||x^i - x^{gt}||}{||x^{gt}||} \tag{11}$$

Figure 11 depicts the error bar of the relative residual value η for both the random and

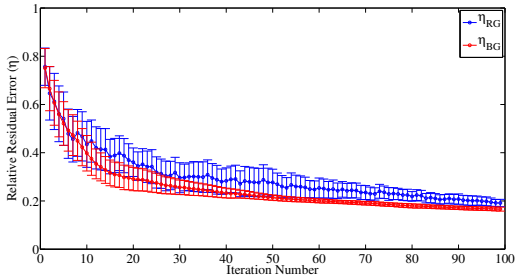


Figure 11: Distributed Seismic Tomography relative residual(η)

broadcast gossip experiments each of which have performed 100 successful gossip updates. Figure 12 depicts the error bar of the relative error value β for both types of gossip, compris-

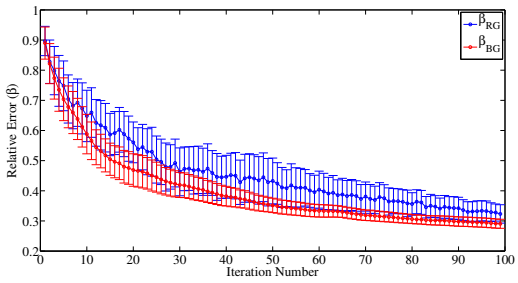
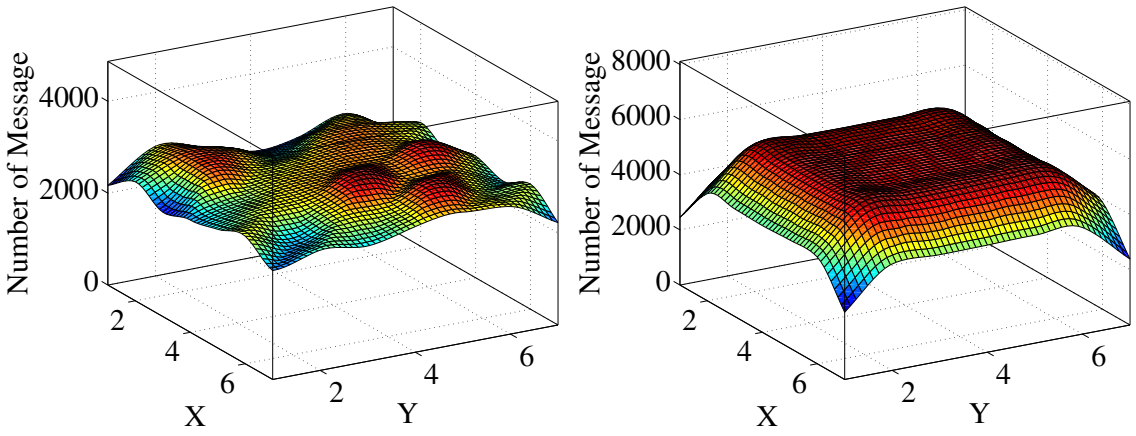


Figure 12: Distributed Seismic Tomography relative error(β)

ing of 10 successful gossip updates. From both figures it is apparent that distributed seismic tomography on INDIGO yields very good and clear results which are expected. Lastly we



(a) Communication Cost Random Gossip (b) Communication Cost Broadcast Gossip

Figure 13: Distributed Seismic Tomography Communication Cost

examine the communication cost, depicted in Figure 13. While random gossip exhibits a relatively uneven surface in Figure , broadcast gossip has a highly consistent communication cost among nodes as depicted in 13(b). This fact can be attributed to the relatively higher stochastic nature of random gossip as compared to broadcast gossip.

From the above discussions on the various applications and investigations into the behavior of gossip protocols in each, it becomes apparent that INDIGO is indeed a versatile framework capable of providing an evaluation platform for a myriad of algorithms and problems.

6 Conclusion

This work focusses on the design, development and evaluation of INDIGO, a distributed gossip protocol design for sensor networks. Drawing from the strong theoretical analysis and study found in existing literature, this work attempts to design a practical and highly useful gossip protocol framework. It enumerates certain presumptions which are made by existing theoretical works in their analysis and which may not necessarily hold good for practical implementations. In order to address these issues, we introduce parameters like *wait time* and *sleep time* which serve as a practical way of realizing the true nature of

asynchronous gossip protocols. We further go on to analyze the effect of these parameters on the performance of INDIGO and endeavor to find of sweet spot with respect to these.

INDIGO further provides a versatile design for performing distributed consensus and consensus optimization. With the help of gossip protocols which have been implemented on a generic system platform as well as on a testbed platform, we ensure seamless portability of algorithms. Further, INDIGO is evaluated on the basis of various case studies, where its efficacy is demonstrated. Mainly, three different case studies are evaluated. Firstly, INDIGO's performance on simple distributed consensus is demonstrated to yield expected results. Next, INDIGO is applied to the problem of distributed event location. Lastly, we apply INDIGO to the domain of distributed seismic tomography, where we get good results as well.

This paper demonstrates that INDIGO is indeed an efficient and robust gossip framework and can be applied practically to any scenario which warrants asynchronous distributed consensus or distributed consensus optimization and get reliable results.

References

[1] W.-Z. Song, R. Huang, M. Xu, A. Ma, B. Shirazi, and R. LaHusen, "Air-dropped sensor network for real-time high-fidelity volcano monitoring," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, MobiSys '09, (New York, NY, USA), pp. 305–318, ACM, 2009.

[2] L. Shi, W.-Z. Song, M. Xu, Q. Xiao, J. Lees, and G. Xing, "Imaging seismic tomography in sensor network," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2013 10th Annual IEEE Communications Society Conference on*, pp. 327–335, June 2013.

[3] D. Anthony, W. Bennett, M. Vuran, M. Dwyer, S. Elbaum, A. Lacy, M. Engels, and W. Wehtje, "Sensing through the continent: Towards monitoring migratory birds using cellular sensor networks," in *Information Processing in Sensor Networks (IPSN), 2012 ACM/IEEE 11th International Conference on*, pp. 329–340, April 2012.

[4] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292 – 2330, 2008.

[5] E. Lee, "Cyber physical systems: Design challenges," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pp. 363–369, May 2008.

[6] C. Li and H. Dai, "Efficient in-network computing with noisy wireless channels," *Mobile Computing, IEEE Transactions on*, vol. 12, pp. 2167–2177, Nov 2013.

[7] Z. Zhang, N. Rahbari-Asr, and M.-Y. Chow, "Asynchronous distributed cooperative energy management through gossip-based incremental cost consensus algorithm," in *North American Power Symposium (NAPS), 2013*, pp. 1–6, Sept 2013.

[8] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, pp. 18–25, Mar. 2006.

[9] G. Kamath, L. Shi, and W.-Z. Song, "Component-average based distributed seismic tomography in sensor networks," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pp. 88–95, May 2013.

[10] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *Information Theory, IEEE Transactions on*, vol. 52, pp. 2508–2530, June 2006.

[11] T. Aysal, M. Yildiz, and A. Scaglione, "Broadcast gossip algorithms," in *Information Theory Workshop, 2008. ITW '08. IEEE*, pp. 343–347, May 2008.

[12] A. Dimakis, A. Sarwate, and M. Wainwright, "Geographic gossip: efficient aggregation for sensor networks," in *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*, pp. 69–76, 2006.

[13] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, pp. 1847–1864, Nov 2010.

[14] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should i choose for my sensor network?," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. –, April 2008.

[15] P. Braca, S. Marano, and V. Matta, "Running consensus in wireless sensor networks," in *Information Fusion, 2008 11th International Conference on*, pp. 1–6, June 2008.

[16] M. Kriegleder, R. Oung, and R. D'Andrea, "Asynchronous implementation of a distributed average consensus algorithm," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1836–1841, Nov 2013.

[17] K. Tsianos, S. Lawlor, and M. Rabbat, "Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 1543–1550, Oct 2012.

[18] D. Seither, A. Konig, and M. Hollick, "Routing performance of wireless mesh networks: A practical evaluation of batman advanced," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pp. 897–904, Oct 2011.

[19] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "Core: A real-time network emulator," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7, Nov 2008.

[20] Y. Y. Jun and M. Rabbat, "Performance comparison of randomized gossip, broadcast gossip and collection tree protocol for distributed averaging," in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2013 IEEE 5th International Workshop on*, pp. 93–96, Dec 2013.

[21] L. Geiger, "Probability method for the determination of earthquake epicenters from the arrival time only," *Bull.St.Louis.Univ*, vol. 8, pp. 60–71, 1912.

[22] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Springer Publishing Company, Incorporated, 2nd ed., 2009.

[23] G. Kamath, P. Ramanan, and W.-Z. Song, "Distributed randomized kaczmarz and applications to seismic imaging in sensor network," in *The 11th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, (Fortaleza, Brazil), 2015.