Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 706083, 15 pages
http://dx.doi.org/10.1155/2015/706083

Research Article

INDIGO: An In Situ Distributed Gossip Framework for Sensor Networks

Paritosh Ramanan, Goutham Kamath, and Wen-Zhan Song

Department of Computer Science, Georgia State University, Atlanta, GA 30303, USA

Correspondence should be addressed to Paritosh Ramanan; pramanan1@student.gsu.edu

Received 10 June 2015; Revised 20 September 2015; Accepted 27 September 2015

Academic Editor: Haiping Huang

With the onset of Cyber-Physical Systems (CPS), distributed algorithms on Wireless Sensor Networks (WSNs) have been receiving renewed attention. The distributed consensus problem is a well studied problem having a myriad of applications which can be accomplished using asynchronous distributed gossip algorithms on Wireless Sensor Networks (WSNs). However, a practical realization of gossip algorithms for WSNs is found lacking in the current state of the art. In this paper, we propose the design, development, and analysis of a novel in situ distributed gossip framework called INDIGO. A key aspect of INDIGO is its ability to perform on a generic system platform as well as on a hardware oriented testbed platform in a seamless manner allowing easy portability of existing algorithms. We evaluate the performance of INDIGO with respect to the distributed consensus problem as well as the distributed optimization problem. We also present a data driven analysis of the effect certain operating parameters like sleep time and wait time have on the performance of the framework and empirically attempt to determine a *sweet spot*. The results obtained from various experiments on INDIGO validate its efficacy, reliability, and robustness and demonstrate its utility as a framework for the evaluation and implementation of asynchronous distributed algorithms.

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], and 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 and keep communication costs as low 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 trends appear to be focusing on 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. Sensor nodes are heavily reliant on batteries. Wireless transmission of sensed data requires bandwidth which consumes considerably more energy than processing data locally [7, 8]. In some cases, for example, in seismic networks, the sensed data at a particular node does not vary drastically

in a spatial sense with respect to its neighbors. Therefore, in case of applications like seismic sensing, one would be more interested in obtaining the global picture with respect to the data obtained from the network rather than focusing on the fine grained nuance of the data pertaining to each node. This would typically involve the solving of a global optimization problem as a function of the sensed data. By adopting a distributed approach, we would also avoid loss of packets in and around the sink node owing to congestion. In the light of these observations, by using a decentralized, in-network approach, we could exploit the spatial corelation of data among neighbor nodes, avoid redundant transmission of data to a central entity by pushing the computation to the end nodes, and in turn hope to save on energy consumption owing to costly transmissions [9].

The consensus problem in sensor network epitomizes the abovementioned ideas. It deals with each node arriving at a consensus of a measured parameter solely on the basis of exchange of information with its neighbor nodes. As an extension of the distributed consensus problem, the distributed consensus optimization problem involves using consensus to propagate information to other nodes in the network and then solving a local optimization problem with constraints local to each particular 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.

The distributed gossip approach is a very promising one in the world of Cyber-Physical Systems [10]. In the recent past, a key implementation of CPS has been in the area of seismic monitoring [1, 11]. As an extension of the above work, research is being conducted for performing seismic tomography [12] in a distributed fashion. 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.

Currently, most tomography approaches use a centralized technique where information is relayed to a sink in order to solve a global optimization problem. 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 in theory, there is very little work done with respect to characterizing the behavior and performance of distributed gossip protocols on an actual WSN setup. There is also not much study done in terms of performance characterization in solving a distributed consensus optimization problem over a wireless network. In order to address these issues, we present INDIGO, a novel in situ distributed gossip framework aimed at solving distributed consensus optimization problems using distributed gossip techniques. INDIGO is a practical, flexible, and highly versatile framework, which can be seamlessly implemented on a specific hardware platform as well as on a generic TCP/IP based network. This feature enables a wide variety of use cases for INDIGO, from testing and evaluation of novel distributed approaches to actual field deployment. By incorporating two diverse gossip protocols, that is, broadcast and random, in its design, INDIGO also enables users to conduct a rich set of tests and compare results of distributed consensus optimization on an actual wireless network. Our results indicate INDIGO's strong performance with respect to real-world case studies in the field of seismic sensing and a strong corelation to the results as predicted by theory.

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 empirical 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 on both 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, that is, *broadcast, random, and geographic* [13–15]. Geographic gossip uses geographic routing, which is not preferable in the case of our sensor network as it is hard to implement on a proprietary hardware stack such as XBee. 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. [13] based on the asynchronous time model. Random gossip chooses nodes at random from its neighbors to exchange information and calculate the average. 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 paper also proves that the algorithm converges to the true average and further goes on to determine the convergence rate. It also provides upper bounds with respect to the averaging time of the algorithms. These conclusively provide sufficient evidence of the robust

nature of the random gossip algorithm. With respect to broadcast gossip however, the work done by Aysal et al. in [14] proves that the algorithm converges only in expectation. The paper also goes on to provide a comparison between different approaches (i.e., broadcast, random, and geographic) in terms of the variance as well as the mean squared error per node against the number of radio transmissions with respect to different network sizes. The work done in [16] provides a very good explanation of the rate of convergence in a more practical setting by assuming that each link has a fixed delay.

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. [17] 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 discusses gossip characteristics over wireless links. Further, the work done by Denantes et al. [18] 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 focusing on a time-invariant scenario, 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. [19] investigates 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. Paper [20] provides a very novel application of gossip protocols. By investigating the problem of consensus in a multiagent system, it demonstrates a practical application of gossip protocols towards a Distributed Flight Array (DFA). DFA is a set of multiple agents, which coordinate amongst themselves to arrive at a consensus and fly in a variety of combinations. While both works [13, 14] 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. [21] presents a practical approach for asynchronous gossip protocols but they do not use a bidirectional 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.

We now proceed to provide a detailed explanation of the problem to be solved coupled with an exhaustive overview of the INDIGO framework design.

3. Problem Formulation and Framework Design

3.1. Decentralized Consensus Optimization. A seismic tomography problem can be modeled as a linear least squares problem of the following form:

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

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. Equation (1) represents the global least squares problem that needs to be solved. If we let $F(x) = \min_x (1/2)\|Ax - b\|_2^2$, (1) transforms into an optimization problem of minimizing the objective function $F(x)$ with respect to x. Given the high-dimensional nature of seismic tomography, the global system of equations represented by (1) is very large and as a result the process of obtaining a good solution to the optimization problem is a tedious affair. Further, solving such a problem over a loosely connected and often unreliable Wireless Sensor Network where each of the nodes holds part of the global optimization puzzle is a challenge in its own right.

To solve this issue, we construct a decentralized approach from the above system of equations, by partitioning A and b row wise over p nodes of the network to yield $A = \{A_1, A_2, \ldots, A_p\}$ and $b = \{b_1, b_2, \ldots, b_p\}$, respectively. The system of equations represented by $A_i \in \mathbb{R}^{m_i \times n}$ and $b_i \in \mathbb{R}^{m_i}$ form the subsystem at the ith node where each node holds a part of the input data. This decentralized version in turn leads to the formation of a relatively smaller, local optimization problem with the following individual objective function

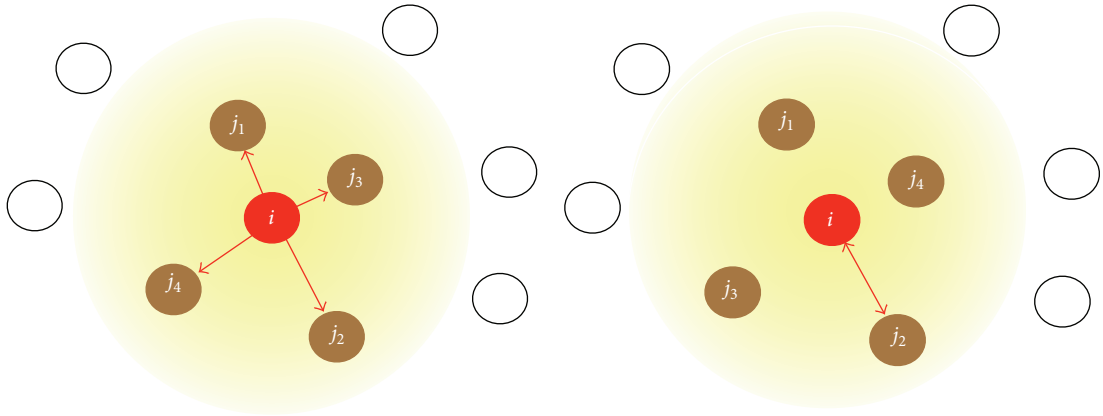
$$f_i(x) = \min_{x} \frac{1}{2} \left\|A_i x - b_i\right\|_2^2 \tag{2}$$

at the ith node. The work done in [22] proposes one such decentralized algorithm which aims to solve this consensus optimization problem. Hence, we obtain a decentralized consensus optimization of the following form:

$$\text{minimize } F(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x_i), \tag{3}$$

where x_i, χ_i, and f_i are the local estimate of the observed value and the local objective function on the ith node, respectively. The individual optimization is solved using Bayesian ART (Algebraic Reconstruction Technique) [23]. The Bayesian ART is an iterative technique used to solve a system of equations like those in (1) or (2) by driving the solution towards the minima as pointed to by the gradient of the objective function.

Therefore, we now only have to minimize each node's objective function independent of its peers and with the help of mutual exchange of information, that is, each node's own estimate of x, among neighbors we can expect a convergence among all the nodes to a solution of the global optimization problem in (1). Mutual exchange of information occurs among neighboring nodes with the help of the gossip algorithms mentioned in the previous section.



(a) Broadcast gossip (b) Random gossip

FIGURE 1: Illustration of random and broadcast gossip with respect to node i and its neighborhood $j_k \in \mathcal{N}_i, \forall k \in \{1, |\mathcal{N}_i|\}$.

At the very heart of each gossip algorithm is the intention to obtain global average of a measured parameter. Therefore, a gossip algorithm attempts to solve the following averaging problem in an entirely distributed way:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{4}$$

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 temperature sensing nodes measuring the temperature of a room may do so by relaying their measured values to a central sink or by exchanging information amongst themselves and arriving at an average which would be the consensus.

With the help of the INDIGO framework we can now apply the concepts illustrated above to solve our problem of seismic tomography. First, we construct a local optimization problem at each node in the network. Nodes then average their respective local estimate of x with their neighbors using the gossip protocols mentioned in the previous section. Once the averaging has taken place, the local optimization problem is solved by each individual node to obtain its own next estimate of x.

3.2. INDIGO Framework Overview and Design. As described in the previous section, gossip protocols can be broadly categorized into *random* and *broadcast* gossip protocols. In this section we present a novel and practical framework design that aims to bring forth the true spirit of the aforementioned protocols. INDIGO has the capability to be configured to execute either the broadcast or the random gossip protocol at run time. The idea is to create a flexible framework design, which can be extended into a platform on the basis of which various 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:

$$\mathcal{N}_i = \left\{ j \mid j \in V,\ \mathcal{W}_{ij} = 1 \right\}, \tag{5}$$

where $\mathcal{W}$ is the *adjacency matrix* of graph G.

The actual model followed by broadcast and random gossip algorithms is an asynchronous time model, which models a rate 1 Poisson clock on each node [13, 14]. We introduce the concepts of exclusivity and stochasticity of the framework as an approximation to enforce this behavior. 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 another 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. Figure 1

function RANDOM-GOSSIP ($\sigma, \rho, \mathscr{M}, x_{\text{self}}$)
  while updates < $\mathscr{M}$ **do**
    sleep for time t, s.t. $0 \le t \le \rho$
    if solicited by $j \in \mathscr{N}_{\text{self}}$ with value x_j **then**
      $x_{\text{self}} = \dfrac{(x_j + x_{\text{self}})}{2}$
      send(j, x_{self})
      updates ← updates + 1
    else
      pick random neighbor $j \in \mathscr{N}_{\text{self}}$
      send(j, x_{self}) and start timer for σ
      if recv(j, x_j)&!timer.expire() **then**
        $x_{\text{self}} = x_j$
        updates ← updates + 1
      end if
    end if
  end while
return x_{self}
end function

Algorithm 1: Random gossip algorithm.

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

Algorithm 2: Broadcast gossip algorithm.

provides the pictorial representation of both random and broadcast gossip protocols:

(i) x_{self}: the estimate of a node's measurement where $x_{\text{self}} \in \mathbb{C}^{n\times 1}$.

(ii) σ: the maximum duration of time after which a gossip exchange is deemed a failure.

(iii) ρ: the upper bound on the random interval of time a node sleeps before initiating gossip.

(iv) $\mathscr{M}$: the maximum number of gossip updates to be performed by all nodes.

(v) $\mathscr{N}_i$: the neighborhood of node i.

(vi) recv(k, x_k): an estimate x_k received from node k.

(vii) send(k, x_{self}): a node's self-estimate *unicasted* to node k.

(viii) $\chi_{\text{self}} = [x_j, \ldots, x_{j+m}]$: matrix of values received from m nodes to be averaged where $\chi \in \mathbb{C}^{n\times m}$.

(ix) broadcast(x_{self}): a node's self-estimate broadcasted to all neighbors.

3.3. 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 $t \le \rho$. A node wakes up from sleep and chooses a random peer from its routing table and *solicits* an average. It starts a timer for $t \le \sigma$ 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 $t \le \rho$. The solicited node responds before timer expires. It updates its current value with the newly received value and goes to sleep for time $t \le \rho$. A node wakes up from sleep and finds that there is already a request for average by one of its peers. In such a case the node performs the average and sends back the result to the solicitor node. This process is summarized in Figure 2(b) which summarizes the sequence of events discussed in Algorithm 1.

3.4. 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 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 cutoff 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 $t \le \rho$. A node that has just woken up from sleep broadcasts its value to neighbors. It then waits for interval of time $t \le \sigma$. It performs the average with whatever values have been received in the interim period and again goes to sleep for random interval of time $t \le \sigma$.

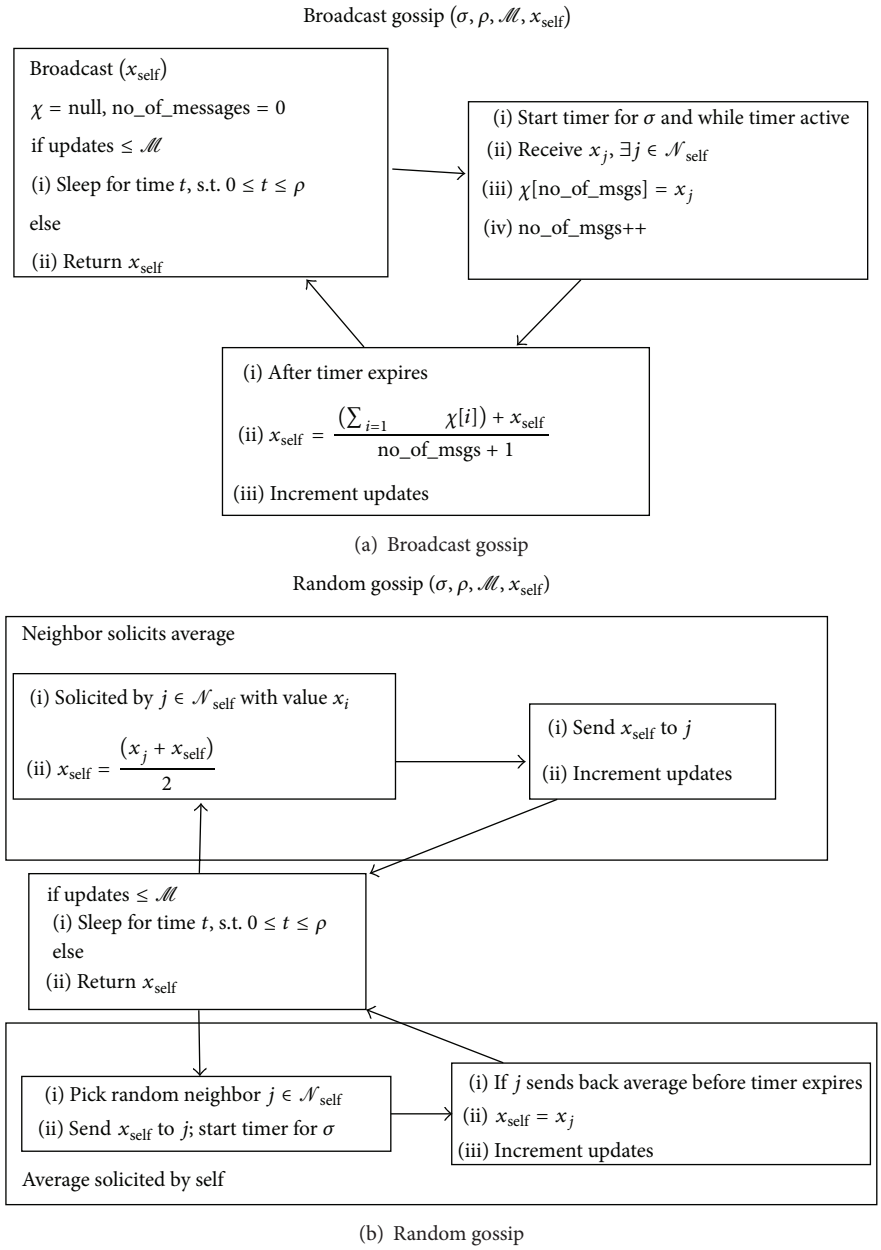


Figure 2: Flow diagram depicting broadcast and random gossip algorithm.

Figure 2(a) summarizes the sequence of events discussed in Algorithm 2. We will now turn our attention to the effects that ρ and σ have on the gossip performance.

3.5. 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*, that is, a range of values of ρ and σ 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 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 following relation:

$$p_s = \sum_{i=1}^{n} \frac{N_{s_i}}{N_{t_i}}, \tag{6}$$

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 ith 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

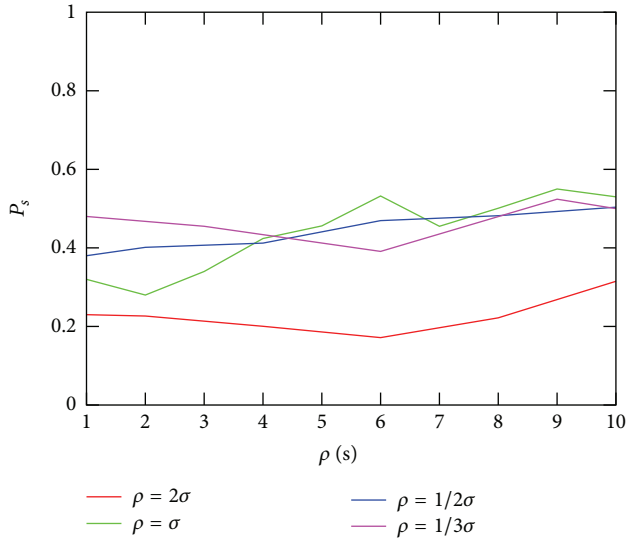


Figure 3: Sweet spot analysis.

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 suboptimal. 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.

The sweet spot value depends quite heavily on the underlying graph characteristics. With the use of linear regression and hypothesis testing methods, one could determine the optimum value based on empirical data pertaining to a given network graph.

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, that is, 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 BeagleBone Black coupled with 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. Figure 4 depicts a schematic comparing the design of the testbed and the system platforms.

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 connectivity 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) [24]. 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 ensures reliable transmission of messages between source and destination.

4.2. Testbed Design. Our testbed setup comprises 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 1 GHz. 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 functionalities have been abstracted by XBee including routing and mesh network capability. The programmable control allows us to operate the 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 predecided 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 blowup of the different components which go into

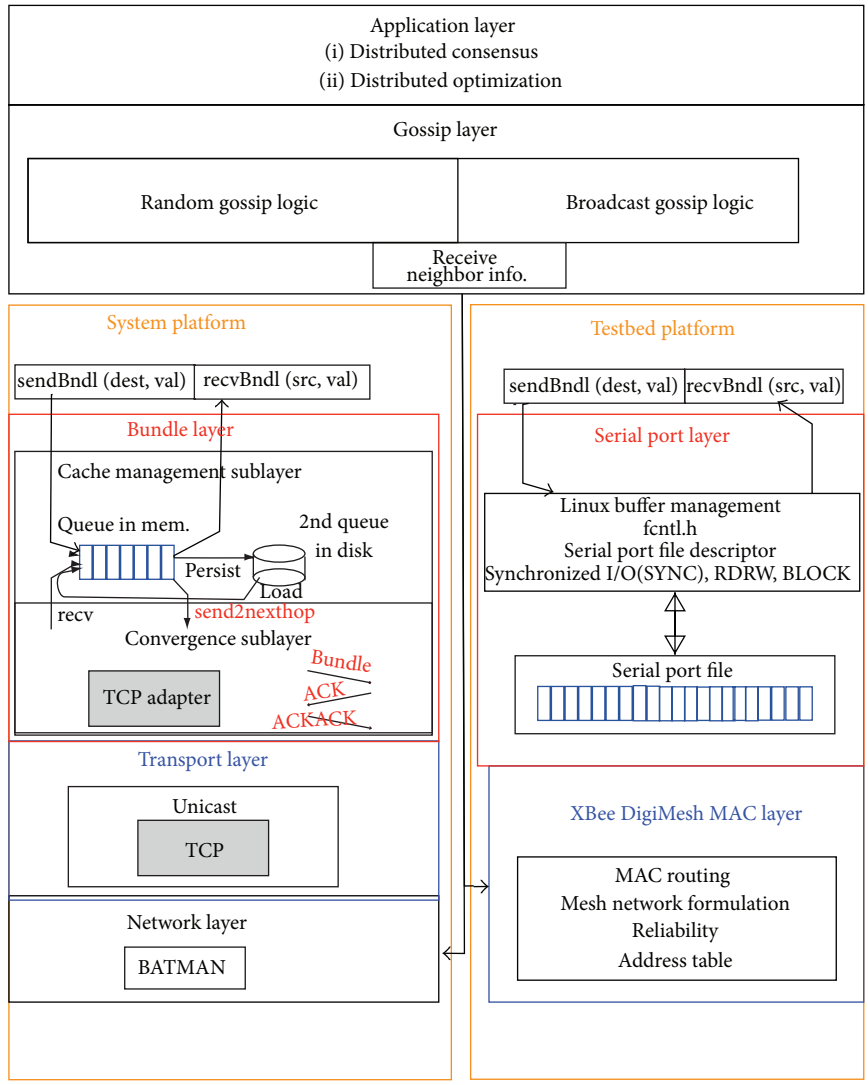
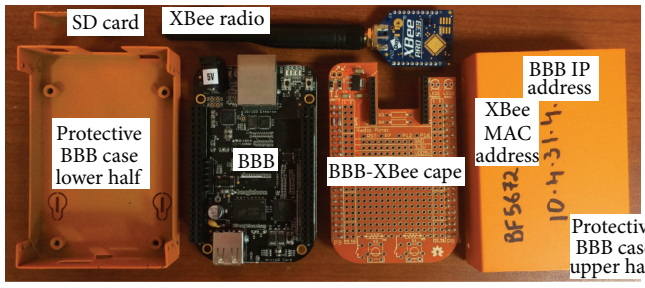
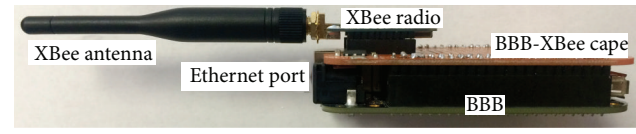


Figure 4: System design and testbed design: a comparison.



(a) Blowup of the BBB-XBee node



(b) Interfacing of BBB with XBee

Figure 5: Actual testbed node setup involving BBB and XBee.

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

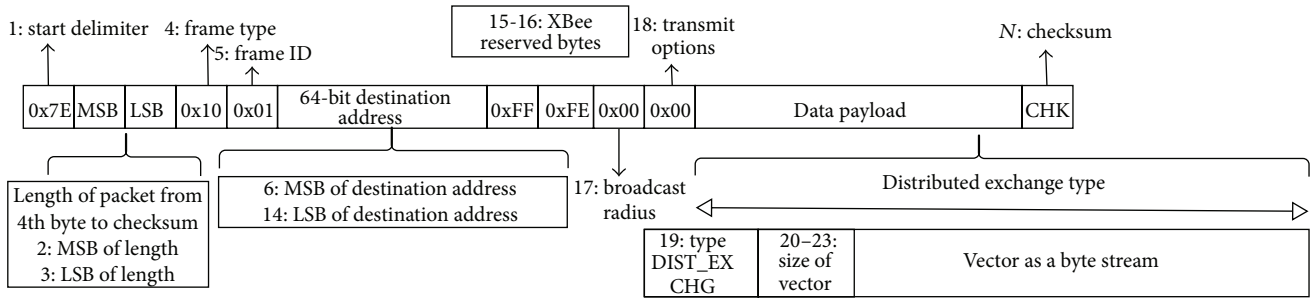


Figure 6: Distributed gossip XBee message structure.

of XBee specific functions for sending and receiving 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.

5. Case Studies

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

(i) TYPE 1: *distributed consensus* gathering of the form

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{7}$$

(ii) TYPE 2: *distributed consensus optimization* of the form

$$\begin{aligned} &\text{minimize} \quad F(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) \\ &\text{subject to} \quad x_i \in \chi_i. \end{aligned} \tag{8}$$

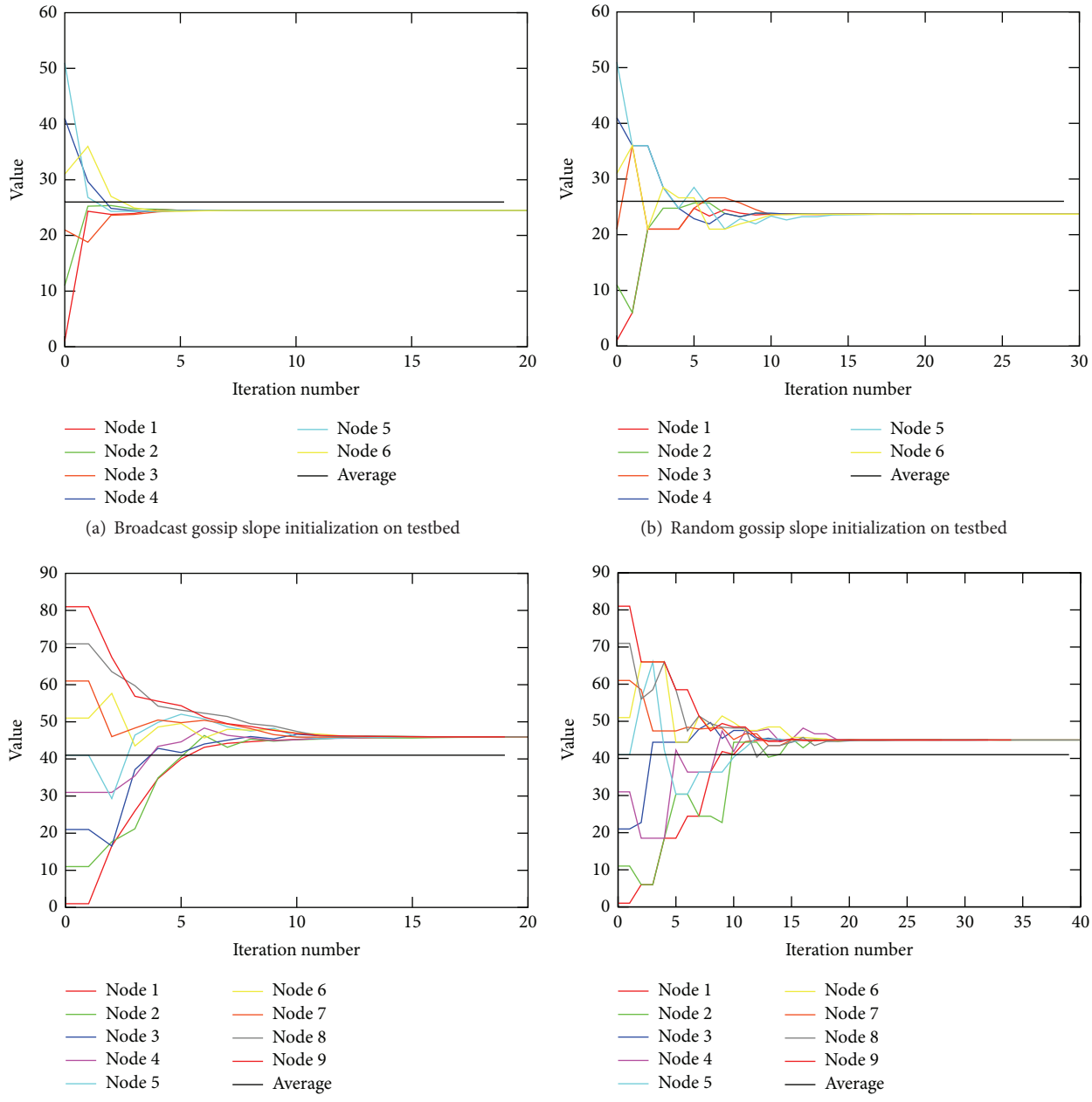
We start with the simple case of distributed consensus gathering, which is of TYPE 1 in both the system and the testbed platform. Then we move to more complex cases like distributed event location on the testbed and finally to distributed tomography computation on a simulation setup which are problems of TYPE 2. For the system evaluation platform we employ a network emulator named CORE [25]. 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 [26] on the basis of their ability to converge to consensus based on two different types of initializations of data, that is, slope and spike initialization. We plot the values on each node in the experiment at each iteration to track and demonstrate convergence. Since there are two setups, the system and the testbed, we conduct experiments relating to each of the initializations on each of the setups leading to a total of 8 combinations as depicted in Figures 7 and 8. All the experiments were performed on the testbed platform using 6 BeagleBone Blacks and XBees and on system emulation platform comprising 9 nodes with $\rho = 3$ and $\sigma = 3$.

5.1.1. 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. Figure 7 depicts the gossip trends arising out of slope initialization on the testbed platform and the system platform. 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. All but one of the nodes are 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. Figure 8 depicts the gossip trends using a spike initialization on. 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 is not close to the actual average. This is expected behavior as it has been anticipated in [14] that broadcast gossip only converges to average consensus in expectation.

The results obtained in this section demonstrate the robustness of the INDIGO framework in successfully realizing the gossip algorithms with respect to a real-world scenario and are commensurate with what was expected in theory.



(a) Broadcast gossip slope initialization on testbed

(b) Random gossip slope initialization on testbed

(c) Broadcast gossip slope initialization on system emulation

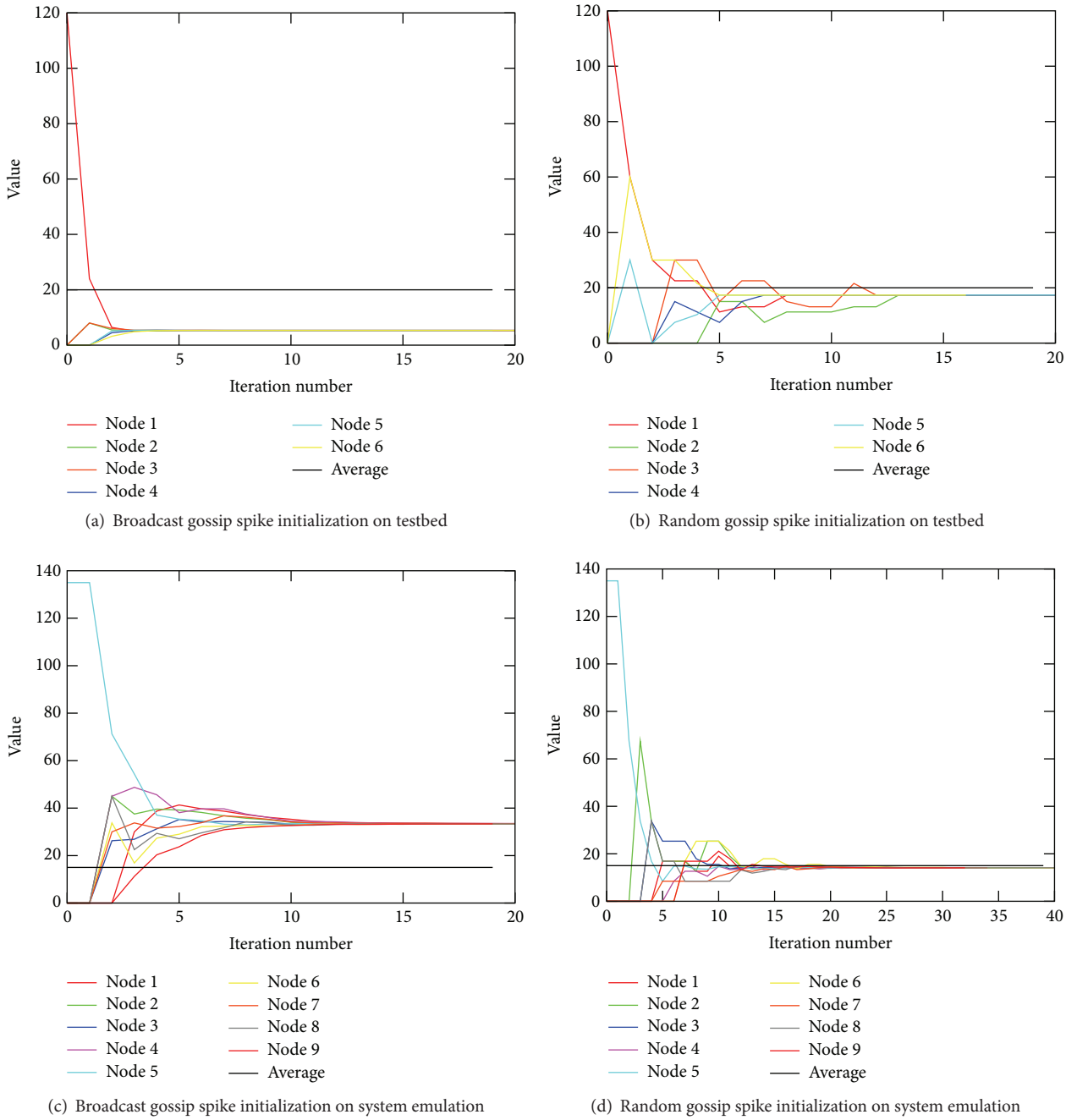
(d) Random gossip slope initialization on system emulation

FIGURE 7: Results of slope initialization.

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 [27] wherein a system of equations of the form represented in (8) is solved. Therefore, distributed event location falls under TYPE 2. We can solve this system of equations using any least squares technique like Bayesian ART [23]. The whole idea behind this experiment is to make the process of event location as mentioned in [27] distributed. We are primarily interested in the rate of decrease of error of the location vector as an indication of success in localizing the seismic event. Therefore, we plot the relative error against the iterations to demonstrate effective event location. The relative error η is calculated as

$$\eta_i = \frac{\|x_i - x^*\|}{\|x^*\|}, \tag{9}$$

where i is the iteration number and x^* is the ground truth and $\|\cdot\|$ is the 2-norm of the vector. For performing this experiment we used the system testbed which comprised 6 BeagleBone Blacks communicating with each other using the XBee radio.



(a) Broadcast gossip spike initialization on testbed

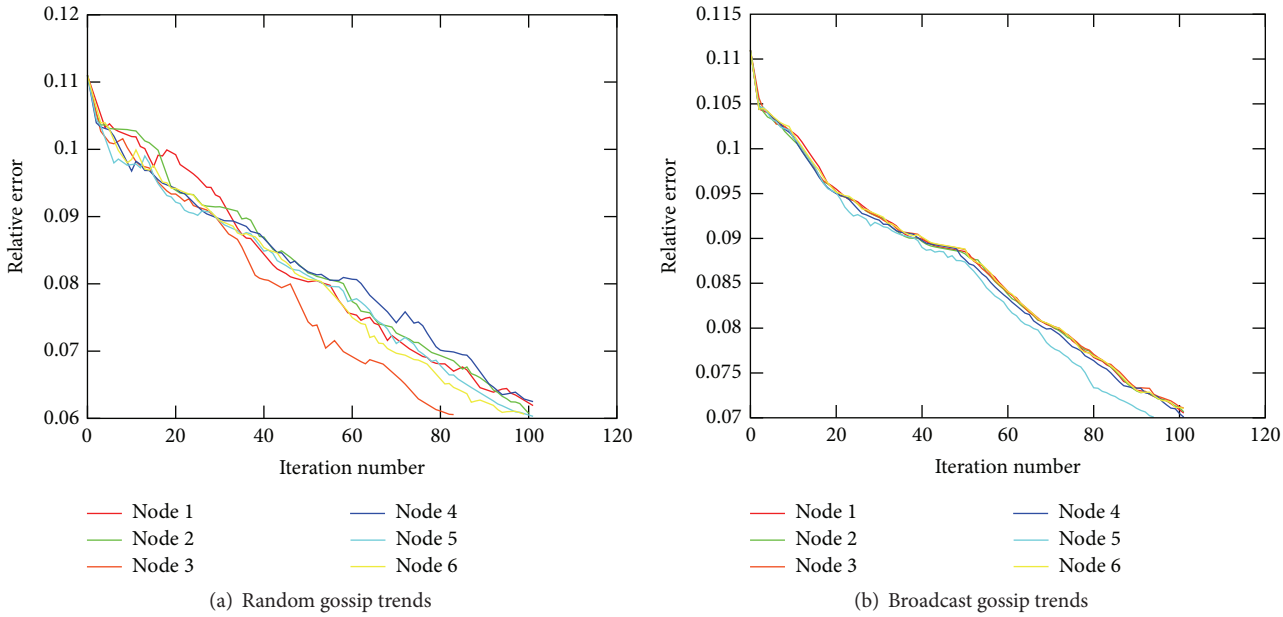
(b) Random gossip spike initialization on testbed

(c) Broadcast gossip spike initialization on system emulation

(d) Random gossip spike initialization on system emulation

FIGURE 8: Results of spike initialization.

Figure 9 represents the experiment involving random and broadcast gossip while performing distributed event location for one particular event where the y-axis represents the relative error. We observe a monotonously decreasing error trend in Figures 9(a) and 9(b) before reaching an acceptable error margin in both the random and broadcast gossip case. Figure 10 shows the number of packets lost while performing distributed event location among the different nodes in both cases. From Figures 9 and 10, we can safely assert that the framework can tolerate packet losses observed in the network. As a result, each node solves its local system of equations referred to by (8) by using an initial guess. Next, it generates the new x value and performs gossip with another 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 is reached. This technique embodies a true *asynchronous* gossip approach as the objective function being solved is directly coupled with exactly one gossip update.



(a) Random gossip trends

(b) Broadcast gossip trends

FIGURE 9: Results of distributed event location using random and broadcast gossip performed on the testbed.

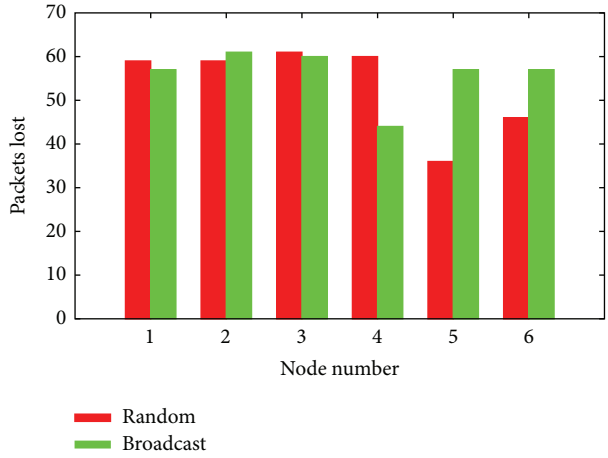


FIGURE 10: Packet loss of random and broadcast gossip while performing distributed event location with 100 iterations.

With this result, it becomes apparent that INDIGO can be fruitfully applied to solve the event location problem in a distributed way.

5.3. Distributed Seismic Tomography. Another application of INDIGO is to perform distributed seismic tomography [28] which is a TYPE 2 problem and can be modeled as a distributed consensus optimization problem. Centralized seismic tomography involves solving an objective function of the type

$$\begin{aligned} &\text{minimize} \quad \|x\| \\ &\text{subject to} \quad Ax = b, \end{aligned} \tag{10}$$

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, kth 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 (10). However, in the distributed scenario, the kth node performs a gossip update with its neighbor(s) to obtain a new estimate of its value x^k. This value is in turn 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* behavior.

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 49 nodes, arranged in a grid topology. The key idea is that a node initially generates an estimate of vector x using Bayesian ART to solve the LOP. It 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.

Our objective is to show that, by using distributed gossip algorithms, the system converges to a solution obtained by solving the centralized form of the same problem. For this reason, we use the least squares solution of the centralized form of $Ax = b$, denoted by x^{gt}, as our ground truth. 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:

$$\begin{aligned} \eta_i &= \frac{\left\|Ax^i - b\right\|}{\|b\|}, \\ \beta_i &= \frac{\left\|x^i - x^{\text{gt}}\right\|}{\|x^{\text{gt}}\|}. \end{aligned} \tag{11}$$

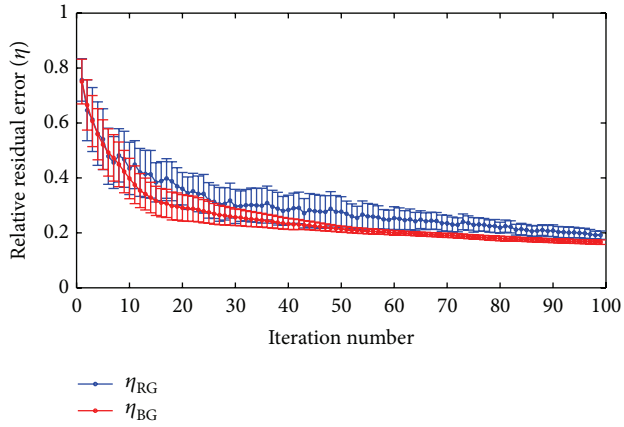


Figure 11: Distributed seismic tomography relative residual (η).

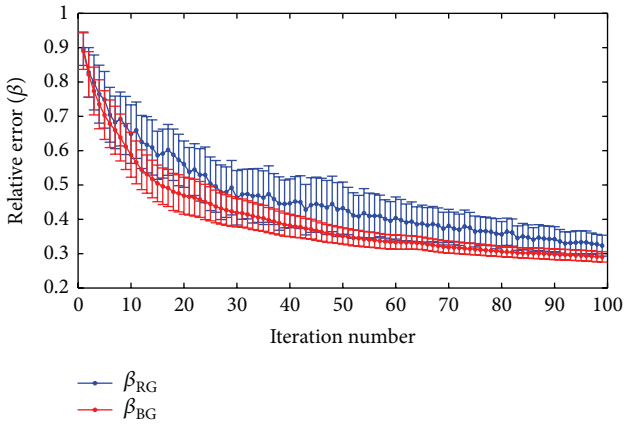


Figure 12: Distributed seismic tomography relative error (β).

Relative residual at each iteration helps in determining how close the system is to the actual observed parameters denoted by the vector b. The relative error helps determine how close we are to the actual ground truth. In order to depict the uniformity in convergence among all nodes, we employ an error bar technique of plotting the results. The points on the curve denote the mean relative residual and the mean relative error in Figures 11 and 12, respectively, among the 49 nodes in the network while the vertical bars denote the standard deviation observed at each iteration. We are able to assert with certainty that the behavior of all the nodes is monotonously decreasing, consistent with theory, and there were no rapid deviations in any node at any point of time. The figures therefore provide a holistic picture in relation to convergence to the centralized solution without loss or underrepresentation of any facet of the experiment.

Figure 11 depicts the error bar of the relative residual value η for both the random and broadcast gossip experiments each of which has performed 100 successful gossip updates. Figure 12 depicts the error bar of the relative error value β for both types of gossip, comprising 100 successful gossip updates.

Observing both figures, one can instantly notice a healthy converging trend with respect to relative residual norm and the relative error norm. There is a slight jitter in case of random gossip as against broadcast gossip due to the fact that random gossip needs to maintain exclusivity with respect to averaging. The standard deviation of broadcast gossip seems to reduce much more drastically owing to a higher degree of mixing among neighbors leading to a higher flow of information through the network. This experiment conclusively demonstrates a real-world working implementation of the INDIGO framework for solving the decentralized consensus optimization problem with the help of distributed gossip protocols.

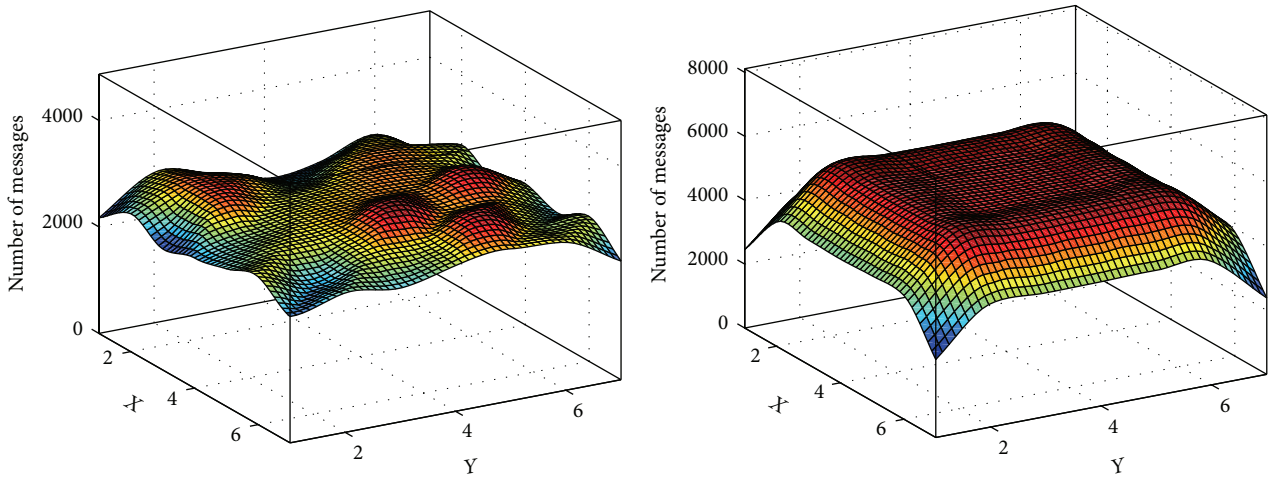
Lastly we examine the communication cost in terms of the number of messages sent and received, depicted in Figure 13. The number of messages is plotted as a function of the grid points on the XY plane representing the nodes. These messages include the total number of incoming and outgoing messages handled by the wireless radio. While random gossip exhibits a relatively uneven surface in Figure 13(a), broadcast gossip has a highly consistent communication cost among nodes as depicted in Figure 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 focuses on the design, development, and evaluation of INDIGO, a distributed gossip framework design for sensor networks. Distributed gossip has been proposed as a more efficient way for solving a global optimization problem with respect to spatially corelated data. Distributed consensus optimization employs gossip techniques to solve local optimization problems as a precursor to solving the global one. We incorporate the random and broadcast gossip models in our framework owing to their high suitability to our application domain of seismic sensing. We present a practical framework design which realizes the true nature of asynchronous gossip and serves as a highly versatile setup for testing and evaluation for distributed algorithms. We show that, using INDIGO, we could perform distributed consensus optimization to solve real-world practical problems in seismic domain. We characterize the effect of our framework parameters on the chance of success of gossip attempts before moving on to evaluation of INDIGO in the domain of seismic sensing.

We apply INDIGO to solve two significant problems in seismic sensing, event location, and distributed tomography. We demonstrate the flexibility of INDIGO by yielding concurrent results on both the system implementation and the testbed setup. The results indicate a strong performance of INDIGO even on a testbed comprising low-powered devices like the BBB and XBee. The results obtained on the system implementation go on to show that INDIGO has the capability to perform well even on a standard TCP/IP stack based wireless network. By ensuring seamless portability of algorithm between the two setups, INDIGO can be used in a wide



(a) Communication cost random gossip

(b) Communication cost broadcast gossip

Figure 13: Distributed seismic tomography communication cost.

variety of ways starting from testing and evaluation of different seismic algorithms all the way to actual field deployment.

Future work in this domain involves deeper analysis of the effect of framework parameters on the convergence of the optimization algorithms. We are also investigating the extension of INDIGO to construct an asynchronous and purely decentralized MPI like version for sensor network.

In conclusion it can be said 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 gets reliable results.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

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 ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 305–318, ACM, Kraków, Poland, June 2009.

[2] L. Shi, W.-Z. Song, M. Xu, Q. Xiao, J. M. Lees, and G. Xing, "Imaging seismic tomography in sensor network," in *Proceedings of the 10th Annual IEEE Communications Society Conference on Sensing and Communication in Wireless Networks (SECON '13)*, pp. 327–335, New Orleans, LA, USA, June 2013.

[3] D. Anthony, W. P. Bennett Jr., M. C. Vuran et al., "Sensing through the continent: towards monitoring migratory birds using cellular sensor networks," in *Proceedings of the 11th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '12)*, pp. 329–340, ACM, Beijing, China, 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. A. Lee, "Cyber physical systems: design challenges," in *Proceedings of the 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC '08)*, pp. 363–369, IEEE, Orlando, Fla, USA, May 2008.

[6] C. Li and H. Dai, "Efficient in-network computing with noisy wireless channels," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2167–2177, 2013.

[7] V. Shnayder, M. Hempstead, B.-R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 188–200, ACM, New York, NY, USA, November 2004.

[8] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[9] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 20–27, April 2004.

[10] Z. Zhang, N. Rahbari-Asr, and M.-Y. Chow, "Asynchronous distributed cooperative energy management through gossip-based incremental cost consensus algorithm," in *Proceedings of the 45th North American Power Symposium (NAPS '13)*, Manhattan, Kan, USA, September 2013.

[11] G. Werner-Allen, K. Lorincz, M. Welsh et al., "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, 2006.

[12] G. Kamath, L. Shi, and W.-Z. Song, "Component-average based distributed seismic tomography in sensor networks," in *Proceedings of the 9th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '13)*, pp. 88–95, IEEE, Cambridge, Mass, USA, May 2013.

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

[14] T. C. Aysal, M. E. Yildiz, and A. Scaglione, "Broadcast gossip algorithms," in *Proceedings of the IEEE Information Theory Workshop (ITW '08)*, pp. 343–347, IEEE, Porto, Portugal, May 2008.

[15] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: efficient aggregation for sensor networks," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 69–76, IEEE, Nashville, Tenn, USA, April 2006.

[16] K. I. Tsianos and M. G. Rabbat, "Distributed consensus and optimization under communication delays," in *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton '11)*, pp. 974–982, IEEE, Monticello, Ill, USA, September 2011.

[17] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[18] P. Denantes, F. Bénézit, P. Thiran, and M. Vetterli, "Which distributed averaging algorithm should I choose for my sensor network?" in *Proceedings of the 27th IEEE Conference on Computer Communications*, pp. 1660–1668, IEEE, Phoenix, Ariz, USA, April 2008.

[19] P. Braca, S. Marano, and V. Matta, "Running consensus in wireless sensor networks," in *Proceedings of the 11th International Conference on Information Fusion*, pp. 1–6, Cologne, Germany, July 2008.

[20] M. Kriegleder, R. Oung, and R. D'Andrea, "Asynchronous implementation of a distributed average consensus algorithm," in *Proceedings of the 26th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '13)*, pp. 1836–1841, IEEE, Tokyo, Japan, November 2013.

[21] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Consensus-based distributed optimization: practical issues and applications in large-scale machine learning," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton '12)*, pp. 1543–1550, IEEE, Monticello, Ill, USA, October 2012.

[22] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: an exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[23] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*, Springer, 2nd edition, 2009.

[24] D. Seither, A. König, and M. Hollick, "Routing performance of wireless mesh networks: a practical evaluation of batman advanced," in *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks (LCN '11)*, pp. 897–904, IEEE, Bonn, Germany, October 2011.

[25] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "CORE: a real-time network emulator," in *Proceedings of the IEEE Military Communications Conference (MILCOM '08)*, pp. 1–7, IEEE, San Diego, Calif, USA, November 2008.

[26] J. Y. Yu and M. Rabbat, "Performance comparison of randomized gossip, broadcast gossip and collection tree protocol for distributed averaging," in *Proceedings of the 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP '13)*, pp. 93–96, December 2013.

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

[28] G. Kamath, P. Ramanan, and W. Song, "Distributed randomized kaczmarz and applications to seismic imaging in sensor network," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS '15)*, pp. 169–178, Fortaleza, Brazil, June 2015.