Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2011, Article ID 195685, 19 pages
doi:10.1155/2011/195685

Research Article

Alert: An Adaptive Low-Latency Event-Driven MAC Protocol for Wireless Sensor Networks

Vinod Namboodiri[1] and Abtin Keshavarzian[2]

[1] *Department of Electical Engineering and Computer Science, Wichita State University, Wichita, KS 67260, USA*
[2] *Research and Technology Center, Robert Bosch Corporation, Palo Alto, CA 94304, USA*

Correspondence should be addressed to Vinod Namboodiri, vinod.namboodiri@wichita.edu

Received 18 March 2011; Accepted 29 August 2011

Academic Editor: Eduardo Da Silva

Collection of rare but delay-critical messages from a group of sensor nodes is a key process in many wireless sensor network applications. This is particularly important for security-related applications like intrusion detection and fire alarm systems. An event sensed by multiple sensor nodes in the network can trigger many messages to be sent simultaneously. We present Alert, a MAC protocol for collecting event-triggered urgent messages from a group of sensor nodes with minimum latency and without requiring any cooperation or prescheduling among the senders or between senders and receiver during protocol execution. Alert is designed to handle multiple simultaneous messages from different nodes efficiently and reliably, minimizing the overall delay to collect all messages along with the delay to get the first message. Moreover, the ability of the network to handle a large number of simultaneous messages does not come at the cost of excessive delays when only a few messages need to be handled. We analyze Alert and evaluate its feasibility and performance with an implementation on commodity hardware. We further compare Alert with existing approaches through simulations and show the performance improvement possible through Alert.

1. Introduction

With the transition of many automated tasks from a wired to a wireless domain, wireless sensor networks (WSNs) are increasingly being subjected to new application domains. Applications of critical nature have been the forte of wired networks due to their reliability. The ever increasing reliability of WSNs coupled with cost-effectiveness has led to their gradual adoption for such critical applications as well. The nature of such applications, however, require new MAC protocols for WSNs that meet the requirements and inspire sufficient confidence about their usage.

The requirements for applications of critical nature can be fundamentally different from the applications for which current MAC protocols are designed for. For example, energy is a valuable resource in sensor devices and most existing MAC protocols are optimized to conserve energy, trading off latency, throughput, and other similar performance metrics in the process. These same protocols are typically not suitable when the application demands better performance at the expense of some additional energy. If latency is to be minimized, with energy consumption only a secondary issue, protocols need to be redesigned from that application perspective.

In this paper, we consider applications that require all wireless sensors to convey urgent messages to a centralized base station (i.e., a single hop away) with minimum delay from the time they are generated. These messages are triggered by events detected by sensor nodes and their task is to inform the base station for possible action. Such messages are triggered very rarely, and the aim is to focus on minimizing latency when they are triggered, even if some additional energy is expended during those times. (We describe in Section 3.1 why energy is not a concern in this application scenario and can be ignored.) Intrusion detection and fire alarm applications are some examples which require such a solution. Even though the messages are typically correlated, the collection of all triggered messages as opposed to one of them provides valuable information which can be used for detection of false positives or postevent analysis. (When sensors cover a large area, only a subset of these nodes will detect events and trigger messages to be sent to the base station. We require that all messages generated due to event detection by this subset be reported.) For example, the European Standard EN 54-25 for fire alarm systems specifies the duration within which the first alarm should be reported and by when all alarms must be received at the base station [1]. The challenges in designing WSN MAC protocols for such applications are the handling of a number of simultaneous messages at the same time without knowledge of how many, and planning for possible interference. Additionally, it is also important to ensure implementation feasibility taking into account the additional constraints imposed on WSNs like time synchronization and limited computation and storage capabilities.

We present the Alert MAC protocol that is designed to minimize latency when collecting simultaneous urgent messages. Alert minimizes contention among nodes by using a combination of time and frequency multiplexing. Multiple frequency channels are used within time slots and contention is minimized by controlling the selection probability of each channel by the nodes. Note that in spite of the use of multiple channels, we assume the presence of *only one transceiver* in all nodes including the receiver. The important features of Alert are the following: (a) minimizes delay of collecting first message as well as all messages; (b) noncarrier sense protocol; it thus eliminates hidden terminal collision problems; (c) dynamic shifting of frequency channels to provide robustness against interference; (d) adaptive characteristic enables operation without knowledge of number of contending nodes.

We make the following additional contributions in this paper. (i) Theoretical justifications for the choice of different design parameters of Alert. Our analytical results are of a fundamental nature and should prove useful for the design of other MAC protocols as well. (ii) Detailed performance analysis of Alert by comparing it with other existing protocols through simulations. These evaluations examine the protocols from an implementation perspective and take into account low level details like degree of time synchronization available. This further provides great insight into the design criteria for event-driven MAC protocols that focus on minimizing latency. (iii) Demonstration of feasibility and validation of analytical results through an implementation on commodity hardware. This validation step inspires the necessary confidence to trust Alert with applications of critical nature.

The rest of this paper is organized as follows. Section 2 presents the design space of MAC protocols in Wireless Sensor Networks. Section 3 presents our Alert protocol and describes some of its features in more detail. Section 4 presents some theoretical considerations in the selection of design parameters for Alert and our analytical results. Section 5 describes an adaptive algorithm used with Alert to handle cases where the number of contending nodes is unknown. In Section 6, we demonstrate the feasibility of implementing Alert and validate our analytical results. We further compare Alert with two other existing MAC protocols to point out its advantages. Concluding remarks are made in Section 7.

2. Related Work

The related MAC protocols for wireless sensor networks can be mainly classified into contention-free, contention-based, and energy-saving protocols. Contention-free protocols are mainly the ones based on time-division multiple access (TDMA) where slots are assigned to each node by the base station and each node sends its message (if it has one) only during its assigned slot (e.g., GUARD [2]). Such TDMA-based protocols perform very poorly when the number of nodes contending is unknown or keeps varying. For the specific application targeted in this work, only a subset (of unknown size) of nodes may have events to report. This makes assigning slots to all nodes undesirable as it leads to a significant increase in delay to receive the first message. For example, consider the case of only one node having a message to report. The average delay incurred to collect this message would be half the TDMA cycle with the worst-case delay being the whole cycle. TDMA schemes are useful in cases where most of the nodes have events to report, a rare case for our scenario. Other contention-free approaches, for example, frequency-division multiple access (FDMA) [3] face similar limitations as outlined above in terms of unknown or varying number of nodes that make scheduling difficult. The work by Chintalapudi and Venkatraman [4] designs a MAC protocol for low-latency application scenarios similar to those considered in this paper, but with some important differences. They assume multiple base stations while our solution requires only one base station. Also in their work, many of the concepts are based on a TDMA schedule which has the same limitations as pointed out above. Finally, their model assumes that each message generated has its own deadline. In our scenario, we are indifferent to the order in which messages are received as long as constraints are met on the latency to receive the first message as well as all the messages.

Contention based protocols can be bifurcated into carrier sense multiple access- (CSMA-) based or non-CSMA-based. The IEEE 802.11 and 802.15.4 protocols are examples of CSMA protocols with the latter designed specifically for applications catered to by wireless sensor networks [5, 6]. They use a variable-sized contention window whose size is adjusted at each node based on the success of the node in sending its message, with each node picking a slot in this window using a uniform probability distribution. These protocols do a good job in handling scenarios with small number of nodes but do not handle a large number of simultaneous messages well. For a detailed performance evaluation on contention window-based schemes and description of deficiencies of the IEEE 802.11 protocol, refer to [7]. The Sift protocol was designed to overcome these deficiencies for WSN applications which need to handle such large number of event-driven spatially corelated messages [8, 9]. Sift is also CSMA based but uses a fixed-size contention window. Nodes pick slots from a geometric probability distribution such that only a few nodes contend for the first few slots, and thus handles a large number of messages easily. A variation based on replacing the uniform-distribution contention window of IEEE 802.11 with a p-persistent backoff protocol was presented in [10]. Protocols based on Aloha on the other hand do not sense the channel before transmission and rely on each node picking a slot to transmit on randomly, with the probability of transmission depending on number of messages contending [11, 12]. When this number is not known, these protocols do not adapt well. In general, CSMA-based protocols outperform Aloha-based protocols when the propagation time between nodes is small enough to make carrier sense useful. When the relative effectiveness of a CSMA-based and a TDMA-based scheme is unknown, a hybrid MAC protocol like Z-MAC can be used to adapt between these protocol types based on prevailing conditions [13].

Our protocol, Alert, is similar to Sift in that it uses a similar nonuniform distribution to control contention among nodes, but is non-CSMA based. Alert separates message transmissions across different frequency channels with this distribution while Sift does it over different time slots. This allows Alert to be free of hidden terminal issues while Sift is susceptible. (Aside from the performance perspective, operations performed at the sender side in Alert are comparatively much simpler than Sift or any other CSMA-based protocol. So fewer resources (less memory and less computational power) are required for implementation of Alert at individual nodes. This makes Alert a more cost-effective solution.) Alert extends this distribution to handle different interference levels as well. While Sift is also optimized for unknown number of messages, the adaptiveness of Alert allows it to perform well even when the number of contending messages is small.

The topic of energy-saving MAC protocols has been well researched for wireless sensor networks in the last few years primarily due to the limited energy supply in these devices [14–16]. A more general framework and survey of MAC protocols for wireless sensor networks can be found in [17]. Our work focuses on applications for which latency is the primary concern and the goal is to let all urgent messages reach a receiver node as soon as possible. Energy is a concern as well, but the rare occurrence of messages in these applications allows the MAC protocol to focus solely on the latency aspects. With such applications, energy efficiency should be built into more common tasks carried out by each node like time synchronization. Some researchers have focused on reducing latency in the collection of messages at a sink node from source nodes through duty cycling approaches (e.g., [18, 19]). These types of low-latency protocols solve a fundamentally different problem which is to sleep as often as possible while ensuring that messages reach the sink as fast as possible (with latencies in the order of seconds instead of milliseconds). Our work focuses on event that driven message generation where the receiver is always awaiting messages. The receiver is assumed to be wall powered or periodically rechargeable, and hence, its energy consumption is not an issue. The transmitters try to send messages when they have one, and the only latency that needs to be reduced is the one created due to contention between nodes trying to send messages at the same time.

In the preliminary version of this work in [20], we had presented some of the contributions above in terms of the Alert protocol. In this paper, there is additional emphasis on our theoretical results. We provide full mathematical proof of the optimal channel probabilities to use with Alert. Further, theoretical results on the success probability of sending a message in an Alert slot with number of nodes tending to infinity is given. Additionally, the optimal probability distributions for different optimization objectives are compared to each other.

3. Protocol Description

We present details of our Alert protocol in this section along with a description of the associated design parameters. Theoretical considerations for selecting values for these design parameters will be presented in the following section.

3.1. Alert Protocol Concepts. The protocol actions can be divided into those at a *sender*, a node that has a message to send, and a *receiver*, a node whose task is to collect messages from senders. The time is slotted into what we refer to as an Alert slot or simply a time slot. (We assume all the nodes in the network are time synchronized with each other. We use periodic broadcast beacons from our base station in our implementation in Section 6.) Each Alert slot can be used to exchange one data packet and its acknowledgment between a sender-receiver pair.

In each time slot, multiple frequency channels can be used by the senders or receiver. We denote the number of frequency channels in each slot by M. These channels have different priorities. (The topic of how priorities are calculated and assigned forms the basis for most of the analysis later in the paper.) The receiver samples them one by one based on their priority level and tries to receive a packet from one of the senders.

Each sender selects a frequency channel randomly and independently of all other senders, but the channels are *not selected with equal probability*. Less chance is given to select a higher-priority channel, that is, the selection probability decreases as we move toward higher priority channels. An example is shown in Figure 1 with $M = 3$ channels and channel selection distribution $(p_1, p_2, p_3) = (0.1, 0.3, 0.6)$. This nonuniform distribution is prespecified and known to all senders. It is designed to reduce the chance of collision among the senders. We discuss its effect and how to find the optimum distribution in detail in subsequent sections.

Once a sender selects a frequency channel randomly based on the prespecified channel selection probabilities, it switches to the selected frequency and sends a long preamble before sending its data packet. (Note that the diagrams in Figures 1 and 2 do not represent the actual timing scales within a time slot. To present the idea, the sampling and preamble durations are shown much longer than their actual value compared to the packet and ack exchange duration.) After the data packet, the node expects an acknowledgment packet (ack) from the receiver. If the ack packet is received

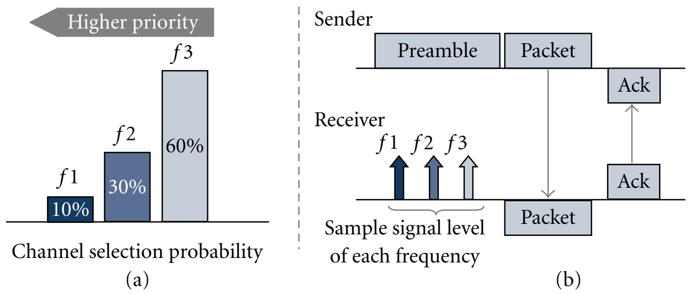


Figure 1: (a) Channel selection probability. (b) One Alert time slot.

correctly, the sender stops, otherwise, it tries to send its message again in the next time slot.

At the beginning of each time slot, the receiver samples the signal level on each of the M frequency channels starting with the highest priority channel. If high signal level (high RSSI (RSSI stands for received signal strength indicator)) is sensed by the receiver, it stays on the same frequency (locking to that channel) and stops sampling any more channels. Then, the receiver waits to receive a packet. If a packet is received correctly, it sends an acknowledgment packet back in response, otherwise, after some fixed timeout period, the receiver stops and continues to the next Alert slot. If the sensed high signal on a channel is due to the simultaneous transmission of preambles by more than one sender, then it is very likely that the received packets are corrupted (Note that a packet may still be received correctly due to capture effect.) If the high signal is due to interference or noise, a packet never arrives from any node, and the receiver simply repeats the procedure in the following slot.

Note that a transmitter is not aware if a receiver has successfully "locked" onto its selected frequency and will thus always transmit the packet even if the receiver is waiting on some other channel. For the applications under consideration, a sender has a message to send very infrequently. For example, a typical system may encounter a situation that requires sending fire alarms only once or twice a year. The rarity of alarms allows for greater effort to be put into reducing latency even at the cost of some extra energy. The receiver, or centralized base station, is typically wall powered (AC-outlet) and its energy consumption is not an issue as mentioned in Section 2. This allows the focus on reducing contention among nodes reporting messages without worrying about receiver wakeup schedules.

While the number of channels, M, remains the same, the frequency of channels (and their priority) can change across time slots. This is illustrated in Figure 2. The frequency table shown in this example is from a simple expression where we have 16 channels numbered from 0 to 15 and $f_m(k)$ represents the mth frequency channel in k th Alert slot:

$$f_m(k) = [5k + 9(m-1)] \mod 16, \text{ for } m = 1, 2, 3. \quad (1)$$

We assume that the frequency table is prespecified, and all the nodes in the network know this pattern. Varying frequency channels after each time slot increases the reliability and robustness against channel fading and interference.

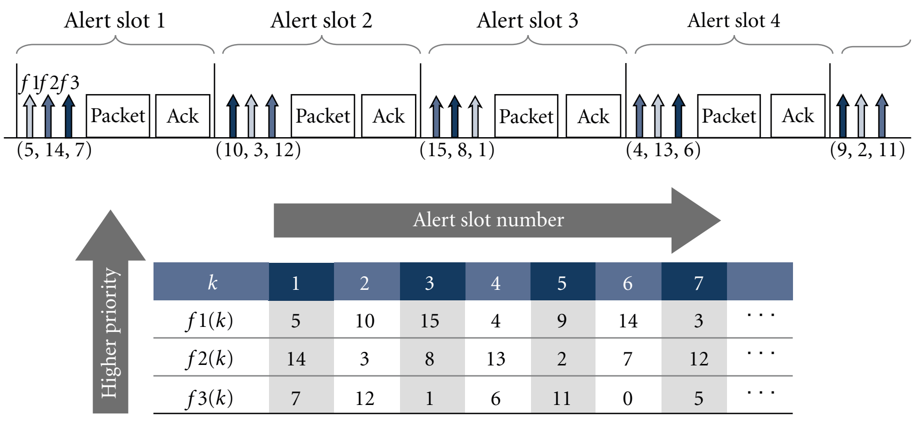
Table 1: Summary of features and assumptions of the Alert Protocol.

Feature	Assumption or detail
Architecture	Single hop with a centralized base station. Base station is AC Powered
Transceiver	Only one transceiver in each node capable of switching frequency channels
Messages	Only one message by a node reporting an event
	Number of nodes reporting events likely a subset of entire deployment
Events	Very rare, but correlated among neighboring nodes
	Only one event report needs to reach the base station
Frequency table	Prespecified among nodes and switching pattern known to all nodes

A summary of the protocol design space with all our assumptions is provided in Table 1 for convenience.

3.2. Collision Avoidance. Alert is designed to avoid collision among senders such that in most time slots (with high probability), one message is received correctly. Therefore, the protocol can collect messages from all senders in as few time slots as possible. If there is only one sender, there will be no collision and the frequency channel selected by the sender does not matter as the receiver will find and lock to the frequency picked by the sender. If two nodes are contending to send their messages, the node that selects the higher priority channel will be successful since the receiver will hear its preamble/tone first and stay in the channel awaiting its packet. If both select the same channel, a collision occurs. The channel selection probabilities are designed to reduce the chance of collision. As the number of senders increases, it becomes more probable that one of the senders selects a higher priority channel, and if only one node selects the higher priority channel, the message from that sender will be successfully received. The illustration in Figure 3 shows three example cases where there are different number of senders.

3.3. Design Parameters. Based on the protocol description, it should be clear that the two key design parameters are the *number of frequency channels* to use in the protocol and



k	1	2	3	4	5	6	7	
$f1(k)$	5	10	15	4	9	14	3	⋯
$f2(k)$	14	3	8	13	2	7	12	⋯
$f3(k)$	7	12	1	6	11	0	5	⋯

FIGURE 2: Frequency channels in each Alert slot.

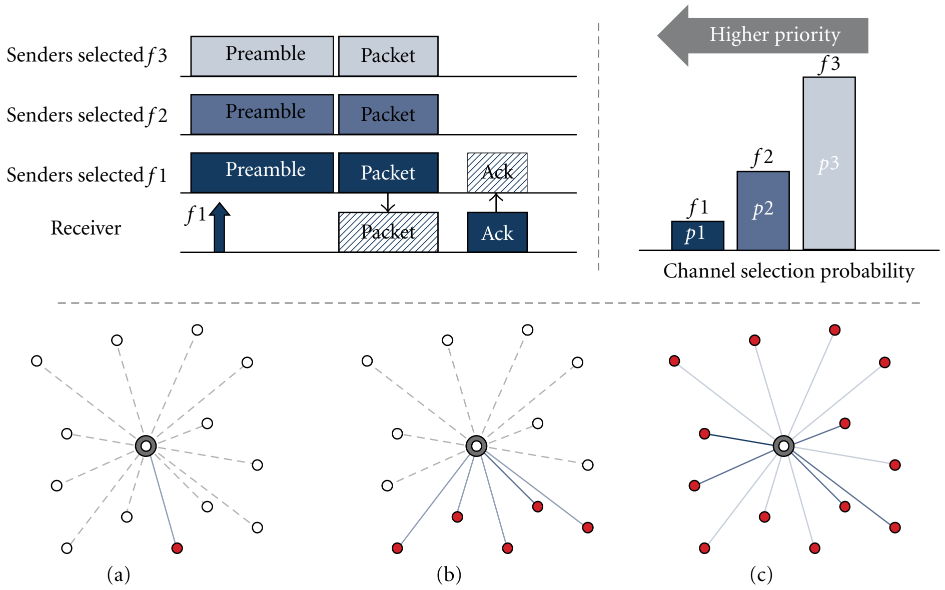


FIGURE 3: Alert protocol illustration. (a) If there is only one sender, it does not matter which channel it picks. (b) With 5 nodes sending, with high probability only one picks the second channel. (c) With many senders, one is very likely to pick the first channel and succeed.

the *probability distribution* over the channels that minimizes the overall time to read all messages. A larger number of channels should decrease contention among the nodes that have messages to send. But this increases the size of a time slot and results in fewer time slots within a given period of time, exposing a tradeoff. The channel probability distribution controls the contention among the nodes. When the number of nodes with messages is large, assigning small probabilities to the higher priority channels ensures lower contention for those channels. This increases the chance that only one node chooses that channel. On the other hand, when the number of simultaneous messages is small, that is, the load is low, assigning small probabilities to the higher priority channels could lead to under utilization of these channels and higher utilization on the lower priority channels resulting in collisions and an increase in the overall latency. In the following section, we construct a theoretical framework to select these design parameters to optimize the performance of our protocol.

For final deployment, each node should be preloaded with information about the number of channels that are to be used in a slot and the probability distribution from which to select them. These should be designed taking into account the application under consideration which might give some information about the expected number of messages and interference levels it should be able to handle.

4. Analysis and Design

For selection of the right design parameters to use in the Alert protocol to minimize latency, we need to consider the probability of success that a message is sent successfully in a single slot and extend that to quantify the number of slots that will be required to read all outstanding messages. We begin our analysis by obtaining an expression for the success probability of a single slot.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_M)$ represent a row vector of channel probabilities corresponding to each channel $1, 2, \ldots, M$. In this section, we assume that the same channel probabilities $\mathbf{p}$ will be used by each node in all slots; that is, $\mathbf{p}$ is not updated during the protocol execution. This allows for a simpler implementation. Also, the results of this section can be built upon by more sophisticated approaches (we show one such approach in the next section).

4.1. Probability of Success in a Single Slot. We assume n nodes are contending to send their message across in the slot (each node is assumed to have only one message to send). They select M channels based on the probability distribution $\mathbf{p} = (p_1, p_2, \ldots, p_M)$. Let random variables X_m (for $m = 1, \ldots, M$) denote the number of nodes deciding to transmit on channel m. The variables $(X_1, \ldots, X_M)$ will have the joint multinomial distribution

$$(X_1, \ldots, X_M) \sim \text{Multinomial}(n, \{p_1, \ldots, p_M\}), \tag{2}$$

$$\mathbb{P}(X_1 = x_1, \ldots, X_M = x_M) = \begin{cases} \dfrac{n!}{x_1! \cdots x_M!}\, p_1^{x_1} \cdots p_M^{x_M}, & \text{if} \sum_{m=1}^{M} x_m = n, \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

for nonnegative integers $x_1, \ldots, x_M$.

In order to model interference, we use an indicator random variable Y_m to specify if channel m has been interfered with or not, if and when the receiver samples the channel. We assume Y_ms are independent and have the Bernoulli distribution:

$$Y_m \sim \text{Bernoulli}(1 - Q), \tag{4}$$

that is,

$$\mathbb{P}(Y_m = y) = \begin{cases} Q, & \text{if } y = 0 \text{ (no interference)}, \\ 1 - Q, & \text{if } y = 1. \end{cases} \tag{5}$$

Parameter Q represents the probability that a channel sees no interference. The value of $Q = 1$ corresponds the ideal case of no interference on any channel. The assumption of independent interference on different channels is strengthened by the periodic switching of channels by all nodes in Alert protocol. (Our analysis can be easily extended to consider a correlated model for interference on all channels.)

Assume that c is a non-idle channel that is selected by at least one of the nodes and node u is transmitting on this channel. If any other node had transmitted (or interference had caused a high signal) on another channel before c (in order of priority), the receiver would never have been waiting on c to receive u's packet. If another node v also selects channel c, the message from these two nodes will collide at the receiver. Ignoring the possibility of capture effect, the collision results in a slot failure with no message succeeding in the whole slot since the receiver will remain on channel c for the whole slot. Thus, u succeeds on channel c if it is the only node transmitting on that channel (with no interference as well). So we have the following rule: *if there is only one node transmitting in the first non-idle channel (in order of priority), an Alert time slot will be successful.*

So a time slot is successful if for any value of m (m between 1 to M), only one node selects the mth channel (event $\mathcal{E}_1 = \{X_m = 1\}$) and no node selects any of the first higher priority $(m-1)$ channels (event $\mathcal{E}_2 = \{X_1 = \cdots = X_{m-1} = 0\}$), and there is no interference on any of the m channels, that is, $\mathcal{E}_{\text{Int}} = \{Y_1 = \cdots = Y_m = 0\}$. So the probability of successful message delivery in a slot is

$$\mathcal{P}_n^{(\mathbf{p})} = \sum_{m=1}^{M} \mathbb{P}(\mathcal{E}_1 \cap \mathcal{E}_2)\mathbb{P}(\mathcal{E}_{\text{Int}}), \tag{6}$$

where we assume that interference is independent of the activities of the node, and $\mathbb{P}(\mathcal{E}_{\text{Int}}) = \mathbb{P}(Y_1 = \cdots = Y_m = 0) = Q^m$. Combining this with (3), we find the the probability of successful message delivery in a slot for a given Q, M, for known number of senders n and channel probability distribution $\mathbf{p} = (p_1, \ldots, p_M)$ as

$$\mathcal{P}_n^{(\mathbf{p})} = n \sum_{m=1}^{M} \left[p_m Q^m \left(1 - \sum_{k=1}^{m} p_k\right)^{n-1} \right]. \tag{7}$$

4.2. Number of Time Slots. We use random variable T_n to denote the number of time slots required to collect n messages. Next, we find the distribution of T_n for a given probability distribution $\mathbf{p}$.

If we start with n nodes, in first time slot, with probability $\mathcal{P}_n$, one message is successfully received and $(n-1)$ nodes remain to go in the next time slots, or with probability $(1 - \mathcal{P}_n)$ no node is successful and we still have n messages left. Hence, we can write the following recursive expression:

$$\xi(n,k) = \mathcal{P}_n \xi(n-1, k-1) + (1 - \mathcal{P}_n)\xi(n, k-1), \tag{8}$$

where $\xi(n,k) \triangleq \mathbb{P}(T_n = k)$ is defined as the probability that it takes k time slots to collect all messages from n nodes. We can solve (8) numerically using the following initial condition $\xi(0,i) = \xi(i,0) = 0$ for all $i = 1, 2, \ldots$, and $\xi(0,0) = 1$.

We define the moment generating function (MGF) of T_n as

$$\Phi_n(z) = \mathbb{E}\left(z^{T_n}\right) = \sum_{k=0}^{\infty} \mathbb{P}(T_n = k) z^k, \tag{9}$$

then from (8), we get

$$\Phi_n(z) = \prod_{k=1}^{n} \frac{\mathcal{P}_k z}{1 - (1 - \mathcal{P}_k) z}. \tag{10}$$

This shows that the random variable T_n is the sum of n independent Geometrically distributed random variables with parameter $\mathcal{P}_n$, that is,

$$T_n \sim \sum_{k=1}^{n} \text{Geom}(\mathcal{P}_k), \tag{11}$$

where the Geometric distribution ($X \sim \text{Geom}(\alpha)$) shows the number of Bernoulli trials needed to get the first success and is defined as

$$\mathbb{P}(X = k) = \alpha(1-\alpha)^{k-1} \quad \text{for } k = 1, 2, \ldots,$$
$$\mathbb{E}(X) = \frac{1}{\alpha}, \qquad \text{Var}(X) = \frac{1-\alpha}{\alpha^2}, \tag{12}$$
$$\Phi_X(z) = \mathbb{E}\left(z^X\right) = \frac{\alpha z}{1-(1-\alpha)z}.$$

So from (11), we have

$$\mathbb{E}(T_n) = \sum_{k=1}^{n} \frac{1}{\mathcal{P}_k}, \quad \text{Var}(T_n) = \sum_{k=1}^{n} \left(\frac{1-\mathcal{P}_k}{\mathcal{P}_k^2}\right). \tag{13}$$

There is a simple intuitive explanation behind the distribution in (11): when we start with n nodes, the chance of success in each slot is $\mathcal{P}_n$, so it takes $\text{Geom}(\mathcal{P}_n)$ trials/slots for the first message to go through. With the first message received, there remain $(n-1)$ senders/messages, so the second message requires an extra $\text{Geom}(\mathcal{P}_{n-1})$ slots. So, in general, the kth message requires $\text{Geom}(\mathcal{P}_{n-(k-1)})$ slots.

4.3. Optimum Probability Distribution. The optimum distribution **p** depends on the available information about number of nodes and interference level, and the performance metric that is optimized. In this subsection, we present different methods to find the distribution **p** and rationale behind each case.

4.3.1. Maximizing $\mathcal{P}_n$ for a Given $n = N$. If we know (or estimate) the number of senders to be N, we can select **p** such that the probability of successful message delivery in one time slot is maximized for the given value of $n = N$. By maximizing $\mathcal{P}_N$, we guarantee that the average delay of receiving *the first message* is minimized when N nodes are contending to send. The number of time slots needed to successfully receive the first message is a random variable with geometric distribution and mean $1/\mathcal{P}_N$. The optimum distribution **p** in this case can be found using the following recursive expression:

$$p_m = \left(\frac{Q-\gamma_m}{NQ-\gamma_m}\right)\left(1 - \sum_{k=1}^{M-m-1} p_k\right), \tag{14}$$

where $\gamma_1 = 0$, and for $m = 2, \ldots, M-1$, we have

$$\gamma_m = Q^{N+1}\left(\frac{N-1}{NQ-\gamma_{m-1}}\right)^{N-1}. \tag{15}$$

This case is very similar to the results in Sift [8] and our result is a generalization of their solution with addition of the effect of interference through parameter Q. The proof is given by the following lemma.

Lemma 1. *Let $\gamma_1 = 0$ and, $\gamma_m = Q^{N+1}[(N-1)/(NQ-\gamma_{m-1})]^{N-1}$. Then given a probability distribution p, if $(\partial/\partial p_j)(\Pi_p(N)/N) = 0$ for $j = 1, \ldots, M-1$, then*

$$(NQ-\gamma_i)p_{M-i} = (Q-\gamma_i)\left(1 - \sum_{m=1}^{M-(i+1)} p_m\right). \tag{16}$$

Proof. We have the success probability:

$$\Pi_p(N) = N\sum_{s=1}^{M-1} p_s\left(1 - \sum_{m=1}^{s} p_m\right)^{N-1} Q^s. \tag{17}$$

Now,

$$\frac{\partial}{\partial p_j}\left(\frac{\Pi_p(N)}{N}\right) = \left(1 - \sum_{r=1}^{j} p_m\right)^{N-1} Q^j - \sum_{s=j}^{M-1} p_s(N-1)\left(1 - \sum_{m=1}^{s} p_m\right)^{N-2} Q^s. \tag{18}$$

Equating the left-hand side to 0, we have

$$(N-1)\sum_{s=j}^{M-1} p_s\left(1 - \sum_{r=1}^{s} p_m\right)^{N-2} Q^s = \left(1 - \sum_{r=1}^{j} p_m\right)^{N-1} Q^j. \tag{19}$$

Now, we will use induction to prove (16).

For $i = 1$, set $j = M-1$ in (19) to get

$$\begin{aligned}(N-1)p_{M-1}\left(1 - \sum_{r=1}^{M-1} p_m\right)^{N-2} Q^{M-1} \\ = \left(1 - \sum_{r=1}^{M-1} p_m\right)^{N-1} Q^{M-1},\end{aligned} \tag{20}$$

which can be simplified as $Np_{M-1} = 1 - \sum_{r=1}^{M-2} p_m$ which proves Lemma 1 for $i = 1$ since $\gamma_1 = 0$.

Next, assume Lemma 1 true for $i = l$. That is,

$$(NQ-\gamma_l)p_{M-l} = (Q-\gamma_l)\left(1 - \sum_{m=1}^{M-(l+1)} p_m\right), \tag{21}$$

which can be rearranged as

$$(N-1)Qp_{M-l} = (Q-\gamma_l)\left(1 - \sum_{m=1}^{M-l} p_m\right). \tag{22}$$

Now, we set $j = M - (l+1)$ in (19) which gives (along with splitting the left-hand side into two terms)

$$\begin{aligned}(N-1)p_{M-(l+1)}\left(1-\sum_{m=1}^{M-(l+1)}p_m\right)^{N-2}&Q^{M-(l+1)}\\ +(N-1)\sum_{s=M-l}^{M-1}p_s\left(1-\sum_{m=1}^{s}p_m\right)^{N-2}&Q^s\\ =\left(1-\sum_{m=1}^{M-(l+1)}p_m\right)^{N-1}&Q^{M-(l+1)}.\end{aligned}\tag{23}$$

Using (19), we can write the above equation as

$$\begin{aligned}(N-1)p_{M-(l+1)}\left(1-\sum_{m=1}^{M-(l+1)}p_m\right)^{N-2}&Q^{M-(l+1)}\\ +\left(1-\sum_{m=1}^{M-l}p_m\right)^{N-1}&Q^{M-l}\\ =\left(1-\sum_{m=1}^{M-(l+1)}p_m\right)^{N-1}&Q^{M-(l+1)}.\end{aligned}\tag{24}$$

It follows from (21) and (22) that

$$\begin{aligned}(N-1)p_{M-(l+1)}\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]^{N-2}&p_{M-l}^{N-2}Q^{M-(l+1)}\\ +p_{M-l}^{N-1}Q^{N-1}\left[\frac{N-1}{Q-\gamma_l}\right]^{N-1}&Q^{M-l}\\ =\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]^{N-1}&Q^{M-(l+1)}p_{M-1}^{N-1}.\end{aligned}\tag{25}$$

Dividing through by $p_{M-(l+1)}Q^{M-(l+1)}[(NQ-\gamma_l)/(Q-\gamma_l)]^{N-2}$ and simplifying eventually gives

$$\begin{aligned}(N-1)p_{M-(l+1)}+\left[\frac{N-1}{NQ-\gamma_l}\right]^{N-1}Q^{N+1}\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]\frac{1}{Q}\\ =\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]^{N-1}p_{M-l}.\end{aligned}\tag{26}$$

But from Lemma 1, $\gamma_{l+1} = [(N-1)/(NQ-\gamma_l)]^{N-1}Q^{N+1}$. Thus, with some rearranging, we get

$$(N-1)p_{M-(l+1)} = \left(1-\frac{\gamma_{l+1}}{Q}\right)\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]p_{M-l} \tag{27}$$

or

$$(N-1)Qp_{M-(l+1)} = (Q-\gamma_{l+1})\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]p_{M-l}. \tag{28}$$

From (21), we have

$$\left[\frac{NQ-\gamma_l}{Q-\gamma_l}\right]p_{M-l} = \left(1-\sum_{m=1}^{M-(l+1)}p_m\right). \tag{29}$$

Thus, (28) becomes

$$(N-1)Qp_{M-(l+1)} = (Q-\gamma_{l+1})\left(1-\sum_{m=1}^{M-(l+1)}p_m\right), \tag{30}$$

which is the same as (22) for $l+1$ instead of l. Thus, by rearranging, we can get (21) for $l+1$ as well. That is,

$$(NQ-\gamma_{l+1})p_{M-(l+1)} = (Q-\gamma_{l+1})\left(1-\sum_{m=1}^{M-(l+2)}p_m\right). \tag{31}$$

Thus, the lemma is true for $i = l+1$ as well, completing the proof by induction. □

Note that for $N = 1$, the optimal probabilities are $p(1) = 1$ and all $p(m) = 0$, $m = 2,\ldots,M$ since success in the later channels after first is smaller since they require all previous channels to be interference free as well. With these probabilities, the probability of success for $N = 1$ becomes $p(1)\times Q = Q$.

4.3.2. Minimizing $\mathbb{E}(T_n)$ for a Given $n = N$. Here, we find the probability distribution $\mathbf{p}$ such that the expected number of time slots needed to collect from $n = N$ senders is minimized. In this way, we guarantee that *all messages* are collected in as few time slots as possible. The optimization problem can be solved by using numerical methods and gradient descent techniques [21]. The gradient of the $\mathbb{E}(T_n)$ is

$$\nabla\mathbb{E}(T_n) = \left[\frac{\partial\mathbb{E}(T_n)}{\partial p_1}\ \frac{\partial\mathbb{E}(T_n)}{\partial p_2}\ \cdots\ \frac{\partial\mathbb{E}(T_n)}{\partial p_{M-1}}\right]^T, \tag{32}$$

and we have

$$\begin{aligned}\frac{\partial\mathbb{E}(T_n)}{\partial p_j} &= \sum_{k=1}^{n}\frac{-1}{\mathcal{P}_k^2}\frac{\partial\mathcal{P}_k}{\partial p_j},\\ \frac{\partial\mathcal{P}_n}{\partial p_j} &= nQ^j\left(1-s_j\right)^{n-1}-\sum_{k=j}^{M-1}n(n-1)Q^kp_k(1-s_k)^{n-2},\end{aligned}\tag{33}$$

for $n \geq 2$ and $s_m \triangleq \sum_{k=1}^{m}p_k$. For $n = 1$, we have

$$\frac{\partial\mathcal{P}_1}{\partial p_j} = Q^j - Q^M. \tag{34}$$

4.3.3. Maximizing min $\mathcal{P}_n$ *for a Given Range for $n \leq N$.* If we do not know the number of nodes in the system, we can select the distribution $\mathbf{p}$ such that the probability of successful message delivery in a time slot is high across a given range.

Essentially, we try to find the solution which maximizes the minimum value of $\mathcal{P}_n$ in the given range, that is,

$$\text{maximize} \ \min_{n=1}^{N} \mathcal{P}_n. \tag{35}$$

We used the subgradient method [21] to solve this optimization problem numerically.

4.4. Comparison of Optimal Distributions. We will present numerical examples and compare the different methods introduced in the previous subsection used to select the probability distribution $\mathbf{p}$.

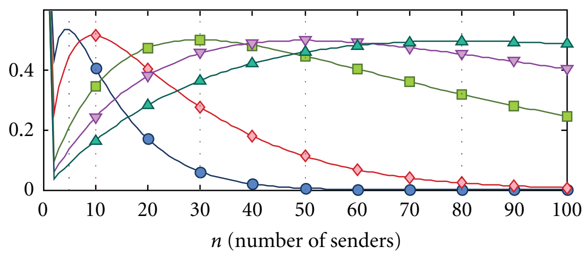
Figure 4 shows the success probability of an Alert slot, $\mathcal{P}_n$, as a function of number of senders, n, for different values of N, with $M = 3$ channels and $Q = 0.95$. In Figure 4(a) (case 1), the probability distribution $\mathbf{p}$ is calculated to maximize $\mathcal{P}_N$. In Figure 4(b) (case 2), the probability distribution $\mathbf{p}$ is found to minimize the expected number of time slots required to receive from N nodes. Figure 4(c), (case 3) corresponds to case where $\mathbf{p}$ is found such that minimum of $\mathcal{P}_n$ over the range $1 \leq n \leq N$, is maximized.

Note that there are two variables representing number of nodes or senders: N denoting the estimated number of senders based on which the distribution $\mathbf{p}$ is calculated and n which is the actual number of senders (the horizontal axis).

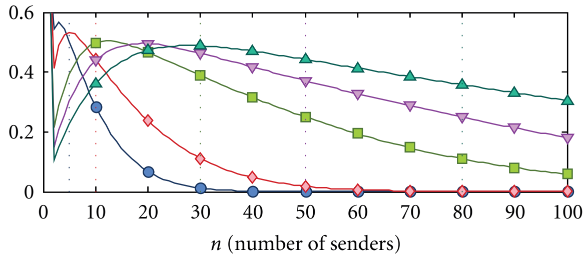
For the first case (Figure 4(a)) the probability of success is maximized if the estimated n is correct, that is, for $n = N$, but for smaller values of $n < N$, the probability of success decreases. So for this case, the delay of getting the first message is smallest. The second case on the other hand puts emphasis on collecting all messages as fast as possible. The third case provides a more flat $\mathcal{P}_n$ for all values.

Figure 5 shows the average number of time slots required to receive the first message or all the messages in the network for the three cases. The results are calculated for $N = 50$ messages, with $M = 3$ channels, and $Q = 0.95$ (interference probability of 5%). For correct number of senders $n \approx N$, we see that case 2 (minimizing $\mathbb{E}(T_N)$) achieves the smallest number of time slots (on average) to collect all messages; case 3 follows closely, then we have case 1. This is expected as the optimization parameter in case 2 is the delay to collect all nodes. Case 1 (maximizing $\mathcal{P}_N$) gives the best delay for collecting the first message at $n \approx N$ and for $n > N$, but for smaller values $n < N$, the delay of the first message and the overall delay to collect all messages are worse than the other two cases. Case 3 (maximizing $\min_{n=1}^{N} \mathcal{P}_n$) gives an overall good performance between the two other cases and tries to keep delay of collecting all messages and the delay of the first message low for the whole range of $1 \leq n \leq N$.

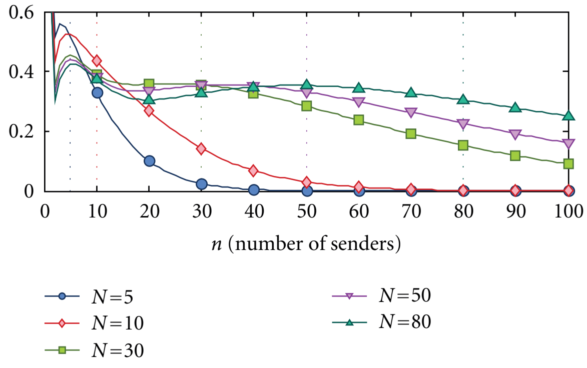
4.4.1. Asymptotic Limit of $\mathcal{P}_n$ for $n \to \infty$. It is interesting to see if $n \to \infty$, how the probability of successful message delivery in a time slot scales. If we fix the probability distribution $\mathbf{p}$, it is easy to see that $\lim \mathcal{P}_n = 0$ as $n \to \infty$. However, if we let the distribution $\mathbf{p}$ to change with n, we can get nonzero limits for the probability of success. These limits represent the asymptotic bounds on the performance of Alert. The distribution $\mathbf{p}_n = (p_1^{(n)}, \ldots, p_M^{(n)})$ which gives



(a) Case 1: maximum success probability for known N



(b) Case 2: minimum average delay



(c) Case 3: maximize min of success probability for a given range for n

Figure 4: Probability of success of one Alert slot for different cases: case 1 (a) maximizing the success probability for a given N, case 2 (b) Minimizing the average number of time slots, case 3 (c) Maximizing minimum of success probability over $1 \leq n \leq N$.

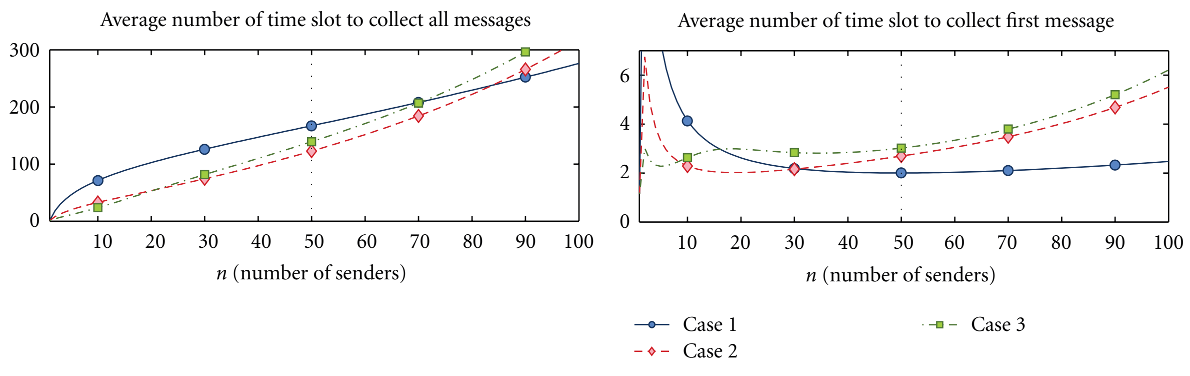
a nonzero $\lim \mathcal{P}_n^{(\mathbf{p}_n)}$ should have the following form: $p_m^{(n)} = \alpha_m/n$ for $m = 1, \ldots, M-1$, and $p_M = 1 - \sum_{k=1}^{M-1} p_k$, which gives

$$\mathcal{P}_\infty \triangleq \lim_{n\to\infty} \mathcal{P}_n^{(\mathbf{p}_n)} = \sum_{m=1}^{M-1} \alpha_m Q^m \exp\left(-\sum_{k=1}^{m} \alpha_k\right). \tag{36}$$

We claim that the values of α_m which maximize $\mathcal{P}_\infty$ can be found from the recurrence $\alpha_m = 1 - Qe^{-\alpha_{m+1}}$ for $m = 1, \ldots, M-2$, with $\alpha_{M-1} = 1$. These values give the following asymptotic (upper) bound on the probability of successful message delivery in a time slot:

$$\mathcal{P}_\infty = Qe^{-\alpha_1} = \frac{Q}{e} \underbrace{e^{(Q/e)e^{(Q/e)\cdots e^{Q/e}}}}_{(M-2)\text{ times}}. \tag{37}$$

The proof of above result can be obtained as follows.



(a) (b)

FIGURE 5: Expected number of time slots to collect all message or first message for $N = 50$, $M = 3$, and $Q = 0.95$.

From (7) for $n > 1$, we have

$$\mathcal{P}_n = \sum_{m=1}^{M-1} \left[p_m Q^m n \left(1 - \sum_{k=1}^{m} p_k \right)^{n-1} \right]. \tag{38}$$

If $\mathbf{p}$ is fixed and $p_1 > 0$, then the limit of $\mathcal{P}_n$ as $n \to \infty$ will be

$$\lim_{n\to\infty} \mathcal{P}_n = \sum_{m=1}^{M-1} \left(p_m Q^m \times \lim_{n\to\infty} n \left(1 - \sum_{k=1}^{m} p_k \right)^{n-1} \right) = 0. \tag{39}$$

This follows from the fact that

$$\lim_{n\to\infty} n \left(1 - \sum_{k=1}^{m} p_k \right)^{n-1} = 0 \qquad m = 1, 2, \ldots, M-1. \tag{40}$$

Note that with $p_1 > 0$, it is guaranteed that $(1 - \sum_{k=1}^{m} p_k) < 1$ for all $m = 1, \ldots, M-1$.

In order to get a nonzero limit, we need to let the vector $\mathbf{p}$ be a function of n as well. The form $p_m = \alpha_m / n$ gives a simple expression for $\mathcal{P}_n$ which has a nonzero limit:

$$\mathcal{P}_n = \sum_{m=1}^{M-1} \alpha_m Q^m \left(1 - \frac{\sum_{k=1}^{m} \alpha_k}{n} \right)^{n-1}, \tag{41}$$

with the following limit as $n \to \infty$:

$$\begin{aligned} \mathcal{P}_\infty \triangleq \lim_{n\to\infty} \mathcal{P}_n &= \sum_{m=1}^{M-1} \alpha_m Q^m \lim_{n\to\infty} \left(1 - \frac{\sum_{k=1}^{m} \alpha_k}{n} \right)^{n-1} \\ &= \sum_{m=1}^{M-1} \alpha_m Q^m e^{(-\sum_{k=1}^{m} \alpha_k)}. \end{aligned} \tag{42}$$

The value of α_j which maximizes $\mathcal{P}_\infty$ should satisfy

$$\frac{\partial \mathcal{P}_\infty}{\partial \alpha_j} = 0, \quad \text{for } j = 1, 2, \ldots, M-1. \tag{43}$$

we have

$$\begin{aligned} \frac{\partial \mathcal{P}_\infty}{\partial \alpha_j} &= Q^j e^{(-\sum_{k=1}^{j} \alpha_k)} - \sum_{m=j}^{M-1} \alpha_m Q^m e^{(-\sum_{k=1}^{m} \alpha_k)} \\ &= Q^j e^{(-\sum_{k=1}^{j} \alpha_k)} \left(1 - \sum_{m=j}^{M-1} \alpha_m Q^{m-j} e^{(-\sum_{k=j+1}^{m} \alpha_k)} \right), \end{aligned} \tag{44}$$

which gives the following conditions for the optimum values of $(\alpha_1, \ldots, \alpha_{M-1})$:

$$\sum_{m=j}^{M-1} \alpha_m \, Q^{m-j} \, e^{(-\sum_{k=j+1}^{m} \alpha_k)} = 1, \qquad \text{for } j = 1, 2, \ldots, M-1. \tag{45}$$

For $j = M-1$, we can simplify previous condition to get

$$\alpha_{M-1} = 1, \tag{46}$$

and for $j = 1, 2, \ldots, M-2$, we expand the sum and take out the first term (the term for $m = j$) and factor out $Qe^{-\alpha_{j+1}}$ from the remaining terms in the sum:

$$\begin{aligned} 1 &= \sum_{m=j}^{M-1} \alpha_m \, Q^{m-j} \, e^{(-\sum_{k=j+1}^{m} \alpha_k)} \\ &= \alpha_j + \sum_{m=j+1}^{M-1} \alpha_m \, Q^{m-j} \, e^{(-\sum_{k=j+1}^{m} \alpha_k)} \\ &= \alpha_j + Qe^{-\alpha_{j+1}} \underbrace{\left(\sum_{m=j+1}^{M-1} \alpha_m \, Q^{m-j-1} \, e^{(-\sum_{k=j+2}^{m} \alpha_k)} \right)}_{=1}. \end{aligned} \tag{47}$$

Note that the expression in parenthesis is the condition (45) for $(j+1)$ and, therefore, should be equal to one. So, we obtain the following recursive expression to solve for α_j in terms of $\alpha_j + 1$ (with initial value of $\alpha_{M-1} = 1$):

$$\alpha_j = 1 - Qe^{-\alpha_{j+1}} \quad \text{for } j = 1, 2, \ldots, M-2. \tag{48}$$

The value of $\mathcal{P}_\infty$ for the optimum $(\alpha_1,\ldots,\alpha_{M-1})$ can be found from (42) and (45) as follows:

$$\begin{aligned}\mathcal{P}_\infty &= \sum_{m=1}^{M-1} \alpha_m\, Q^m\, e^{(-\sum_{k=1}^{m}\alpha_k)} \\ &= Qe^{-\alpha_1} \underbrace{\sum_{m=1}^{M-1} \alpha_m\, Q^{m-1}\, e^{(-\sum_{k=2}^{m}\alpha_k)}}_{=1 \text{ from (45) for } j=1}.\end{aligned} \tag{49}$$

We get close to this asymptotic bound for relatively, small values of n, for example, with $Q = 0.9$ and $M \geq 3$ the maximum possible $\mathcal{P}_n$ is close to $\mathcal{P}_\infty$ for $n \geq 20$.

4.5. Optimum Number of Channels. When multiple nodes contend to send messages in a slot, a larger value of M (larger number of channels) *decreases the contention* among them by increasing the likelihood that they select different channels. Thus, one would think that we should use as many channels per slot as we can. However, there are practical considerations that present a tradeoff. For each channel used, the receiver has to sample it, and switch to the next channel. Thus, for each channel used, there is a sensing plus switching delay added to the size of a time slot. On one hand, as we increase M, the length of each time slot is increased, on the other hand, with larger M, we can get better success probabilities $\mathcal{P}_n$ and, therefore, we can collect all messages in fewer time slots. Thus, selecting M poses a tradeoff. We select M to optimize this tradeoff and minimize the overall delay.

To find the optimum M, we need to have some timing constants from the radio. In general, we can write the length of a time slot as $\tau_{\text{slot}}(M) = M\tau_1 + \tau_2$, where τ_1 specifies the time duration required by the receiver to sample the presence of tone/preamble on one channel and switch to the next channel, and τ_2 is the time for completion of the packet, ack message, and other constant times in one time slot (see Figure 11 for more details). Since the number of channels is a positive integer value and bounded by a small number (total number of channels), we elected to calculate the optimum probabilities $\mathbf{p}$ for each value of M and pick the optimum M_{opt} (and corresponding $\mathbf{p}_{\text{opt}}$) as the one which minimizes the expected value of overall delay to collect from N nodes, that is, $\mathbb{E}(\mathcal{D}_N) \triangleq \mathbb{E}(T_N) \times \tau_{\text{slot}}$. (Considering each value of M has additional benefits as explained in the following section.)

Figure 6 shows the normalized delay $(\mathbb{E}(D_N)/N)$ as a function of M for $Q = 0.95$. For timing constants, we used $\tau_1 = 0.4$ ms and $\tau_2 = 6.0$ ms which are representative values based on our measurements with the implementation on Bosch CC2420 node boards (see Section 6). The optimum M which minimizes the delay is shown on each graph by a filled marker. The optimum value of M depends on value of N and Q. However, it is not very sensitive to N as we see that close to M_{opt}, the curves are flat. So we can increase or decrease M by one or two and expect to get almost the same performance.

Here we find out the optimal channel probabilities P to be used by nodes for known values of N and Q. This can be useful if the application is such that every node can fairly accurately estimate the number of nodes that will respond to any event it detects itself within an environment with known, unvarying interference levels.

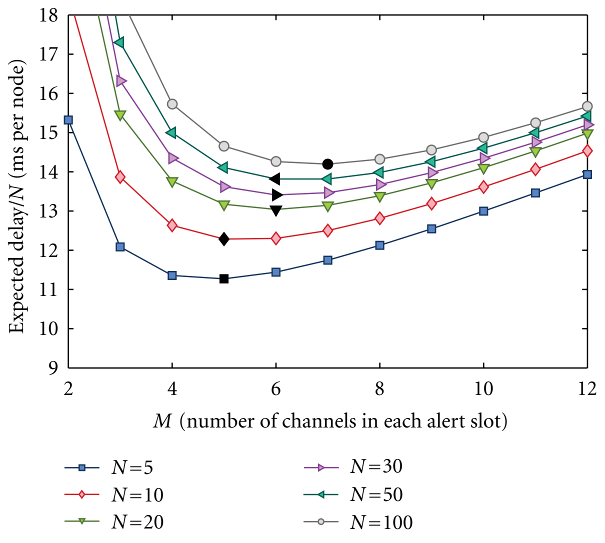


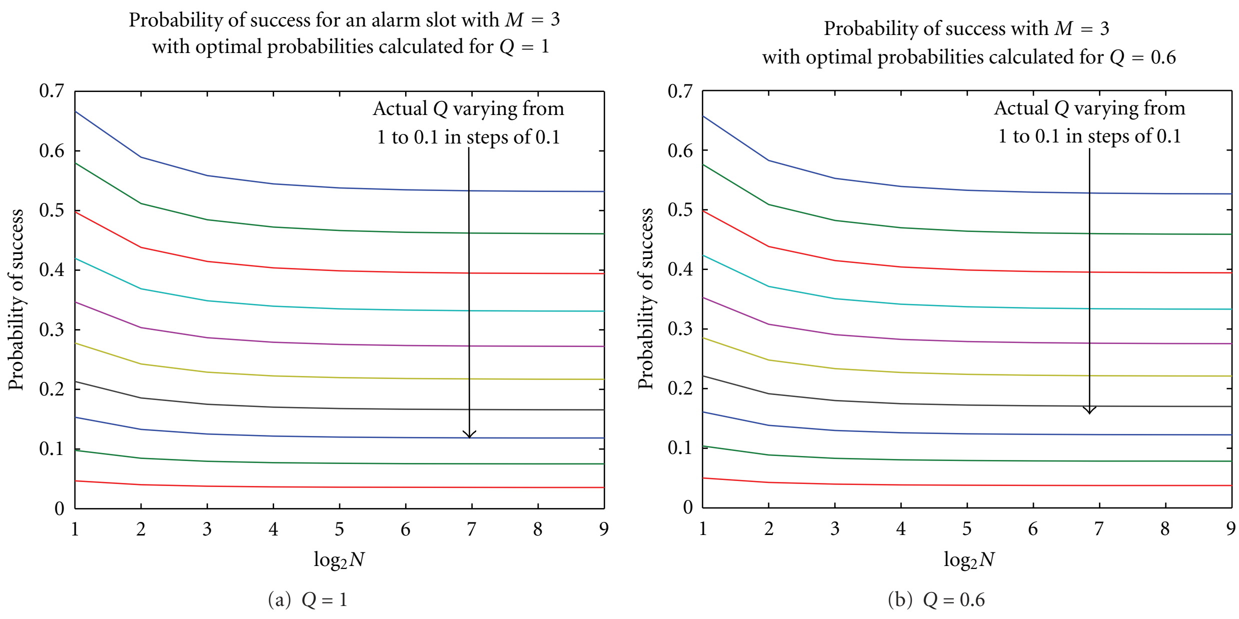
Figure 6: Normalized delay (ms per node) versus number of channels in each Alert slot.

4.6. Impact of Q on Probability of Successful Message Delivery in a Slot. Given N, Q, and M, the optimal channel probabilities can be calculated based on (14). In this subsection, we evaluate numerically the impact various values of Q and N have on the probability of successful message delivery in a slot. These results are important because in subsequent sections, we will assume we do not know the level of interference Q during protocol execution. Using (14), we study the sensitivity of optimum channel probabilities $\mathbf{p}$ to the interference level Q. The aim was to see if an estimate of Q would be sufficient to compute close to optimal $\mathbf{p}$. We calculated the optimal probabilities for three values of Q; $Q = 1$, $Q = 0.6$, and $Q = 0.1$ for various values of N keeping $M = 3$. The results are plotted in Figure 7. (For $Q = 0.1$, for large N, some values are undefined and hence those data points are missing.)

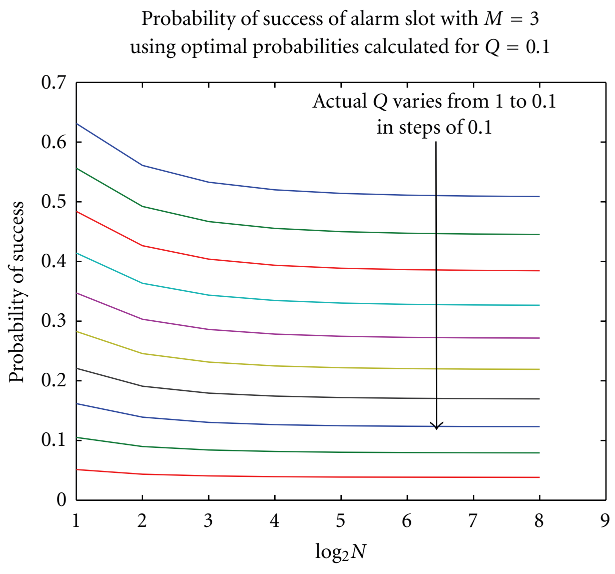
It can be seen that the probability of successful message delivery in a slot, for all values of Q on which channel probabilities were calculated, does not change much. This is because the computed channel probabilities themselves do not vary much across different values of Q. This does not mean that Q has no effect on probability of success in a slot; as actual Q varies, the slot success probability decreases as shown in the plots. We saw similar results for other values of M as well. Thus, we can conclude that a reasonable estimate on Q is good enough for calculating optimal probabilities.

5. Adaptive Algorithm for Alert

The previous section presented different theoretical approaches to minimize the number of time slots to collect all messages. These approaches focused on using a single



(a) $Q = 1$ (b) $Q = 0.6$



(c) $Q = 0.1$

Figure 7: Effect of Q on which probabilities are calculated as a function of probability of success in an alarm slot.

set of channel probabilities $\mathbf{p}$ throughout the execution of Alert. Though this results in a simpler implementation, we can do better by adapting these probabilities as the protocol executes. We present one such approach in this section where the probabilities to use are updated every time unit t. We thus seek a set of channel probabilities used throughout the protocol execution represented as a list of vectors $\mathbf{P} = (\mathbf{p}^{(1)}; \ldots ; \mathbf{p}^{(F)})$, or equivalently in matrix form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{p}^{(1)} \\ \vdots \\ \mathbf{p}^{(F)} \end{pmatrix} = \begin{pmatrix} p_1^{(1)} & \cdots & p_M^{(1)} \\ . & \cdots & . \\ . & \cdots & . \\ p_1^{(F)} & \cdots & p_M^{(F)} \end{pmatrix}, \tag{50}$$

where F is the number of time units required for reading all messages, with a new set of channel probabilities calculated for each unit. A time unit can be either a single time slot or a whole frame of time slots in the context of Alert. (The actual implementation of Alert is based on the concept of frames. A specific amount of time per frame is allocated for reading messages, with the rest used for other operations like time synchronization, maintenance among others. This time can be divided up into any number of slots depending on the size of a slot.) In this section, we specifically look at the case where no knowledge of number of messages is assumed and present an adaptive algorithm (Algorithm 1). We believe this to be the most important case to handle for the protocol. For simplicity, we assume that all initial messages arrive simultaneously. This can be justified when

```
(1) Get $M_{\text{opt}}$ for some guess on number of nodes $N_{\text{guess}}$,
    $\langle M_{\text{opt}} \rangle = \text{get } M(N_{\text{guess}})$.
(2) Start with some initial value of $N = N_{\text{init}}$
(3) Get some estimate on value of $Q$.
(4) Get $\mathbf{P}$, and expected number of frames needed, $F$,
    based on current estimate $N$ and for $M_{\text{opt}}$,
    $\langle \mathbf{P}, F \rangle = \text{get } \mathbf{P}(N, M_{\text{opt}})$
(5) while message not read do
(6)     for frame $t = 1$ to $F$ do
(7)         for all slots in frame $t$ do
(8)             Execute protocol for calculated $\mathbf{p}^{(t)}$ for frame $t$
(9)             Terminate execution if message read and reset
                to initial state
(10)        end for
(11)    end for
(12)    Increase estimate by a constant factor $N = N + C_{\text{incf}}$
(13)    Get $\mathbf{P}$, and expected number of frames needed, $F$,
        based on current $N$ for $M_{\text{opt}}$
        $\langle \mathbf{P}, F \rangle = \text{get } \mathbf{P}(N, M_{\text{opt}})$
(14) end while
```

Algorithm 1: Adaptive alert protocol.

messages are rare, but correlated, and usually occur due to some event observed by a subset of the nodes resulting in the generation of simultaneous messages. Generation of alarms by surveillance systems is such an example.

We begin by describing how we calculate $\mathbf{P}$ for our proposed adaptive algorithm. Then, we give the details of our algorithm along with our strategy to calculate the number of channels to be used per slot considering we do not know the number of nodes sending messages.

5.1. Calculation of Channel Probabilities. The distribution $\mathbf{p}^{(t)}$ controls the contention of nodes with messages to send in a time unit t. Given M, the $\mathbf{p}^{(t)}$ used for each time unit must control contention such that the probability of successful message delivery in that time unit is maximized. As the protocol progresses, maintaining our assumption of simultaneous arrivals of messages, the number of remaining messages, n, decreases as they are successfully received by the base station. Thus, for subsequent slots, after the first slot, a new value of $\mathbf{p}^{(t)}$ must be calculated before each slot t taking into account the current value of n. For large initial n, the time to read all messages can be quite large, and the requirement of recalculation of $\mathbf{p}^{(t)}$ before each slot t can prove infeasible. Moreover, we would prefer that the values of $\mathbf{p}^{(t)}$ used in each slot be precalculated and stored in memory. Thus, due to the computational and memory limitations on wireless sensor devices, we adopt a *perframe* recalculation strategy.

Let R_n be the number of messages that go through in frame t with n messages contending at the beginning of the frame. We seek for frame t the channel probabilities $\mathbf{p}^{(t)}$ that maximize the number of messages read, that is, the channel probabilities to use in a frame satisfies

$$\underset{\mathbf{p}^{(t)}}{\text{maximize}} \ \mathbb{E}(R_n). \tag{51}$$

This can be solved through numerical methods using a recursive algorithm or through Markov chains in conjunction with (14). The distribution $\mathbf{p}^{(t)}$ is updated at the beginning of each frame t. The number of remaining messages is updated at the end of each frame by subtracting the expected number of messages read in the frame for the $\mathbf{p}^{(t)}$ used in the frame. Thus, if M is the number of channels used in a slot and F is the number of frames that are expected to be required by the protocol to read all messages, the distribution $\mathbf{P}$ is in the form of a $F \times M$ matrix. Each row of $\mathbf{P}$ is the distribution to use in the frame corresponding to that row. The steps to calculate $\mathbf{P}$ are given in Algorithm 2.

This procedure calculates the number of frames required to read all messages up to some small threshold, N_{thr}, and then repeats the last $\mathbf{p}^{(t)}$ in the final frame F. The constant N_{thr} is used to ensure that an underestimate of number of messages remaining (compared to the actual value) does not result in using channel probabilities that will make it almost impossible for further messages to go through. (For example, channel probabilities chosen when estimated number of messages is 2, but actual number of messages is 10, will result in probabilities that does not allow any of the 10 messages to go and may delay reading any message for next 100–500 attempts. On the other hand if estimated number is 10 and actual number is 2, the additional delay is much smaller and is of the order of 1–10 extra attempts. So by setting N_{thr} to some small value, say 10, we ensure that the estimated number left allows further messages to go through still with a high probability.)

5.2. Adaptive Algorithm Details. When the actual number of messages to be sent is unknown, we desire that the protocol itself try to estimate it and calculate the corresponding design parameters for this estimate. We employ an adaptive approach where we vary a node's estimate of number of senders, N, until it succeeds in sending its message. The algorithm starts out by selecting the number of channels to use per slot, M_{opt}, based on some guess on the initial number of nodes (or messages with our assumption of one message per node), N_{guess}, which need not be accurate due to insensitivity of M to number of nodes. Further details of calculation of M_{opt} are explained in Section 5. The algorithm sets its estimate of number of messages, N, to a small value N_{init} as the starting point of its *upward* adaptation to larger values. Upward adaptation is chosen because changing to a different estimate only requires waiting a small time before realizing that the estimate may be incorrect. Q is a deployment environment specific parameter that is measurable, and, hence, can be estimated. As mentioned in Section 4.6, a reasonable estimate is enough. In the next step, the algorithm finds the optimal channel probabilities, $\mathbf{P}$, and the expected maximum number of frames F within which the message will be read for this estimate. If, after F frames of protocol execution, the node's message is still not successful, it increases the estimate N by a constant additive factor C_{incf} ($C_{\text{incf}} > 0$) and recalculates F and $\mathbf{P}$ to use in the subsequent frames. The algorithm continues until the message is finally read, after which the node resets back to the initial state and on the next event will reexecute the algorithm. The algorithm

```
(1) for each frame $t$ do
(2)     Calculate (and store) $\mathbf{p}^{(t)}$ for the given $M$ with
        current value of $n$ using (51)
(3)     Reduce $n$ by expected no. of messages read so far,
        that is,
        $n = n - \mathbb{E}(R_n)$
(4)     if $n \leq N_{\text{thr}}$ then
(5)         Store number of frames $F = t + 1$
(6)         Repeat last $\mathbf{p}^{(t)}$, that is, $\mathbf{p}^{(F)} = \mathbf{p}^{(t)}$.
(7)         Break from for loop
(8)     end if
(9) end for
(10) Return $F$ and $\mathbf{P} = (\mathbf{p}^{(1)}; \dots; \mathbf{p}^{(F)})$
```

Algorithm 2: Procedure get $\mathbf{P}(n, M)$.

```
(1) Use timing constants $\tau_1$ and $\tau_2$ of specific radio used
(2) for each $M \leq M_{\max}$ do
(3)     $\tau_{\text{slot}}(M) = M\tau_1 + \tau_2$
(4)     Get the expected number of frames needed, $F(M)$,
            based on value of $N_{\text{guess}}$:
            $\langle \mathbf{P}, F(M) \rangle = \text{get}\,\mathbf{P}(N_{\text{guess}}, M)$
(5) end for
(6) Get optimal $M$ for reading all messages:
            $M_{\text{temp}} = \arg\min_M F(M) \times \tau_{\text{slot}}(M)$
(7) Get associated time to read all messages:
            $t_{\text{val}}^{M_{\text{temp}}} = F(M_{\text{temp}}) \times \tau_{\text{slot}}(M_{\text{temp}})$
(8) Find largest $M$ within some factor $\beta$ of $t_{\text{val}}^{M_{\text{temp}}}$:
            $M_{\text{opt}} = \max\{M \mid t_{\text{val}}^{M} \leq (1+\beta) t_{\text{val}}^{M_{\text{temp}}}\}$
(9) Return $M_{\text{opt}}$
```

Algorithm 3: Procedure get $M(N_{\text{guess}})$.

is designed such that all the calculations of M_{opt} and $\mathbf{P}$ can be prestored and used from memory. The same initial N and constant increase factor C_{incf} ensures that we just need to store M_{opt} calculated for some constant N_{guess}, and $\mathbf{P}$, F for all estimates N which can only have the following values: $\{N_{\text{init}}, N_{\text{init}} + C_{\text{incf}}, N_{\text{init}} + 2C_{\text{incf}}, \dots\}$ up to some maximum limit on the value of N or memory capacity available.

5.3. Number of Channels Per Slot.

When the Alert protocol is deployed, we desire that only a single value of M be used. Changing the number of channels per slot dynamically as number of messages N that remain to be read changes is difficult to implement in practice. Next, we describe our method to derive M_{opt} given in Algorithm 3. It begins with some initial guess of N, N_{guess}, not necessarily close to the actual N, but not a small value. The optimal number of channels M_{opt} to use is calculated based on this N_{guess}. Because we do not recalculate M_{opt}, we need to ensure that the initial value of N_{guess} used gives us a good M_{opt} to use for the rest of the protocol execution, as adaptive estimates on N changes. From our earlier analysis (mentioned in Section 4.5), it was found that optimal M is quite insensitive to N.

The calculation of M_{opt} uses the same method explained in Section 4.5 but with a small difference. Assume M_{temp} is the optimal value of M for reading all messages with minimum delay. But this value may not be good for getting the first message through with minimum delay, for which a larger number of channels might be better. So we introduce a design factor $\beta, \beta \geq 0$ by which we look for possibly larger number of channels to use without incurring an expected time penalty greater than $(1+\beta)t_{\text{val}}^{M_{\text{temp}}}$, where $t_{\text{val}}^{M_{\text{temp}}}$ is the expected time to read all messages using M_{temp}. Parameter β allows us to control our optimization criteria: $\beta = 0$ specifies selection of M_{opt} that optimizes the time to read all messages, while larger values of β increasingly look to optimize the time to send the first message by considering usage of larger number of channels.

6. Protocol Evaluation

In this section, we evaluate the feasibility and performance of Alert through an implementation on commodity hardware and also simulations. We begin with our implementations focusing on the feasibility of Alert.

6.1. Feasibility of Alert.

We implemented the Alert protocol on Bosch CC2420-based wireless nodes. The Bosch boards use Chipcon/TI CC2420 radio which is an IEEE 802.15.4 compliant transceiver operating at 2.4 GHz band with direct sequence spread spectrum (DSSS) O-QPSK modulation and 250 Kbps data rate. An external power amplifier (max transmit power 10 mW) is used to improve the communication range.

In the first experiment setup, we deployed 15 senders in an office in Palo Alto, Calif, as shown in Figure 8. The receiver (base station) is in communication range of all nodes and it kept them in sync by sending periodic beacon messages. Every second, all the nodes sent a message simultaneously using Alert protocol with the following fixed probability distribution:

$$\mathbf{p} = (0.05, 0.063, 0.092, 0.182, 0.613). \tag{52}$$

The receiver measured the number of time slots to receive the first message and number of time slots to collect all the 15 messages. Each time slot is 8 ms long.

Figure 10 shows the measured distribution of the number of time slots (for both the first and all messages). We see that the Alert protocol is performing *better* in real deployment (the experiment setup) than what the analysis predicts. The calculations show that in average we need 24.82 time slots to collect from all 15 nodes, but our experiment gives an average of 17.60 time slots. This improvement in performance is mainly due to *Capture effect*. When two nodes are sending simultaneously, in our analysis, we assume that there will be a failure, but in many cases the receiver can correctly decode one of the packets while treating the signal from the other sender as noise. Since the CC2420 radio employs

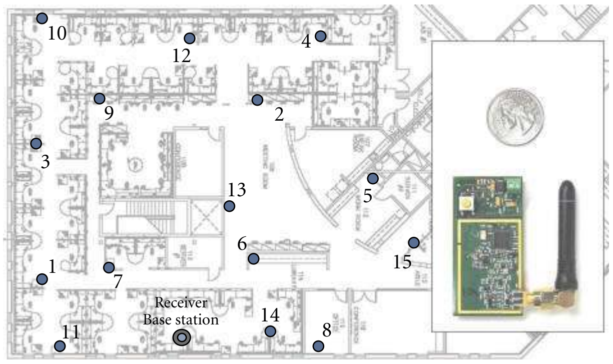


Figure 8: Experiment setup number 1: 15 Bosch CC2420 nodes deployed in an office building.

Figure 9: Experiment setup number 2: nodes on a table close to base station.

spread spectrum techniques, it can tolerate higher level of interference and this helps increase the chance of capture effect. Note that Alert, by reducing the number of contending nodes at higher priority channels, increases the likelihood of capture effect.

To validate our analysis we reduced the chance of the capture effect by repeating the experiment with a different setup. In the second setup, all the 15 nodes were placed close to each other and close to the receiver on a table. The network configuration is shown in Figure 9. Since all nodes are close to one another, the received power at the receiver from all senders is high and equal. This reduces the chance of capture effect. The results are shown in Figure 10. We see that the measurements distribution with the second setup matches very closely to what the analysis predicts.

6.2. Simulation Setup. Using simulations enables us to evaluate the performance of Alert with a large number of nodes with messages to send, that is, for scalability, and also compare against other protocols. We compare Alert with two other contention-based MAC protocols—Sift [8] and Slotted Aloha (S-Aloha) [11]. Sift was chosen because it is a CSMA-based protocol (unlike Alert) and previously shown to do better than variable contention window protocols like 802.11 for the target application scenario (refer to Section 2 for more details). S-Aloha is a simple protocol, allowing each time slot to be very small, possibly providing advantages in reducing time required to read messages. We believe comparisons of Alert with these two protocols covers a wide design space for MAC protocols for the target application. We had discussed the infeasibility of other possible MAC protocols (e.g., TDMA) in Section 2.

For the evaluations, we wrote a simulator in MATLAB. The important abstraction was the concept of time across different protocols from an implementation perspective. The interference was modeled as pointed out in Section 4. All protocols send messages to the receiver (or base station) in fixed time slots. (The 802.15.4 protocol also has a fixed slot structure with both a contention access period and a contention-free period within each frame [6].) The size of a time slot for each protocol is different (but of *fixed size*) based on how it is used. The timing of all three protocols as used in our simulations are shown in Figure 11. In this figure, t_1 is the guard time plus the rx/tx switching time, t_{Skew} is the maximum clock skew, t_2 is the channel sensing time, t_3 is the channel switching time, and t_4 is the total time to exchange a packet and ack. Based on the implementation of Alert and measurement on the CC2420 radio, the values used for these constants are $t_1 = 0.5\,\text{ms}$, $t_2 = 0.1\,\text{ms}$, $t_3 = 0.3\,\text{ms}$, $t_4 = 2.5\,\text{ms}$.

In the S-Aloha protocol, each node tries to send its message in a slot with probability $1/N$ until it finally succeeds in doing so, where N is the current estimate of number of messages. We let the S-Aloha protocol use the same methodology of adapting N as Alert, and chose an initial value and increment factors that gave best results; this was initial $N = 10$ with additive increments $C_{\text{incf}} = 50$. For S-Aloha, a slot duration consists of the guard time plus RX/TX switching time t_1, a single adjustment for clock skew t_{Skew} and the time to exchange a Packet and Ack.

Sift uses a fixed contention window (CW) size and relies on a geometric probability distribution with which nodes pick a backoff slot for transmission. Once a node counts down to its chosen backoff slot and is the only one that has chosen that slot, it completes the packet and ack exchange with the receiver and all nodes move onto the beginning of the next protocol time slot. For simplicity, we do not implement RTS/CTS with Sift and do not consider the hidden terminal effect in our evaluations. The Sift slot duration consists of the guard time t_1, the length of each backoff slot which is the sum of adjustment for clock skew and the channel sensing time, $t_{\text{Skew}} + t_2$, and the time to exchange a packet and ack, t_4. (We will mention how that consideration would effect the comparison between the protocols when we present our results.) Since a node might capture the channel in some backoff slot within the CW and begin packet transmission at that time, there is the possibility of some time left over after the ack is sent back until the next slot begins. This limitation is due to implementation considerations for which a fixed slot duration is highly desirable, and often, the most practical.

For Alert, a slot duration consists of the guard time t_1, adjustment for possible clock skew t_{Skew} both before and after the sampling time by receiver, multiple copies of time to sample channel plus channel switching time, $t_2 + t_3$, and the time to exchange a packet and ack, t_4. Because each transmitter is sending a continuous tone the whole time

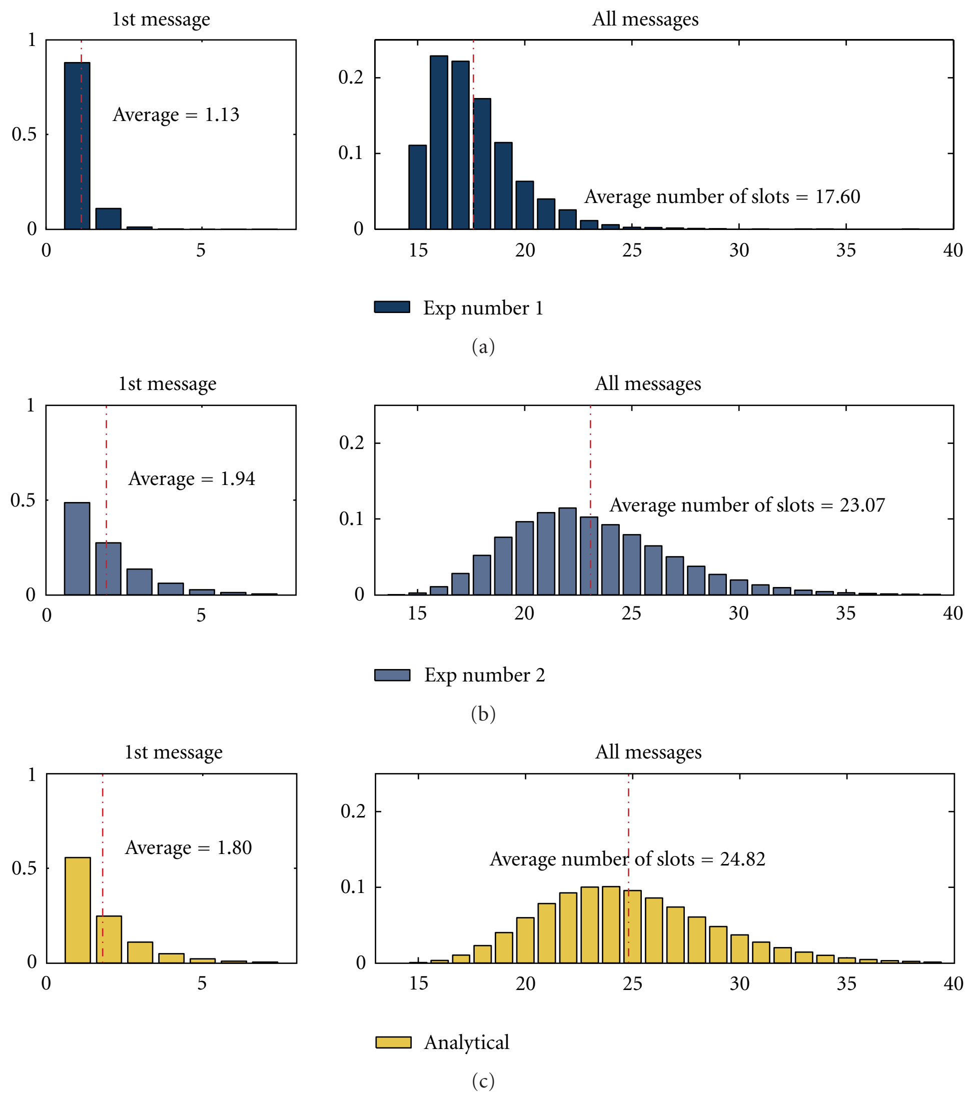


Figure 10: Experiment results: measured distribution of number of time slots needed to send the first message (left) and all messages (right) for experiment setup number 1 (a) and number 2 (b) and comparison with analytical results (c).

the receiver is sampling channels, we do not need to adjust for clock skew (t_{Skew}) anytime except before and after the sampling is done when the transmitter is not sending the tone. Alert does not have any spare time left over in a slot-like Sift because the packet and Ack exchange takes place at a specified time in the slot regardless of which channel is used. The receiver simply waits on that channel at the time to receive a packet and return an ack. In our simulations, N_{init} was taken as 10 with additive increments $C_{\text{incf}} = 50$. The value of M was calculated for $N_{\text{guess}} = 50$ and $N_{\text{thr}} = 10$. The value of β was set to 0.1 which provided a good balance between minimizing delay of collecting first message and collecting all messages.

Two levels of time synchronization were considered; *tight* and *loose*. Note that these are relative terms that are used to convey the compensation required for expected clock skew within slots. The tight synchronization allows smaller compensation times to be used with all protocols, but can prove to be a heavy burden on the higher level protocol that is responsible for it. Tight synchronization would require all nodes to participate in the synchronization protocol more frequently and would consume a lot of energy, even when the nodes have no messages to send. Thus, for applications targeting rare events, tight synchronization may not be feasible and a "looser" form of synchronization may be more desirable. We use the values $t_{\text{Skew}} = 0.7$ or 0.2 ms for loose and tight synchronization, respectively. (Note that, for tight synchronization, the values used ($t_{\text{Skew}} = 0.2$, $t_2 = 0.1$ ms) give roughly the specified backoff slot duration of 0.32 ms in the IEEE 802.15.4 standard [6].)

To get a sense of the effectiveness of the adaptive version of Alert, we also show a plot of the expected number of slots required to read all messages when exact value of N is known throughout the protocol execution. This is calculated theoretically using (13) for a known N. The scheme is termed *ExOptAlert*. Each data plot shown is the average of 150 runs

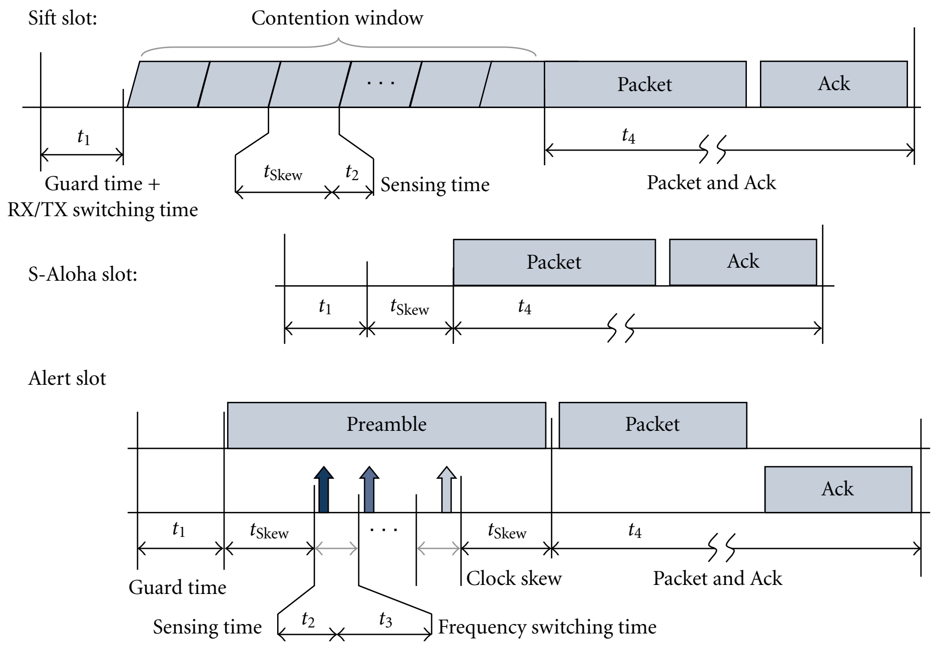


FIGURE 11: Timing within slots of Sift, S-Aloha, and Alert protocols.

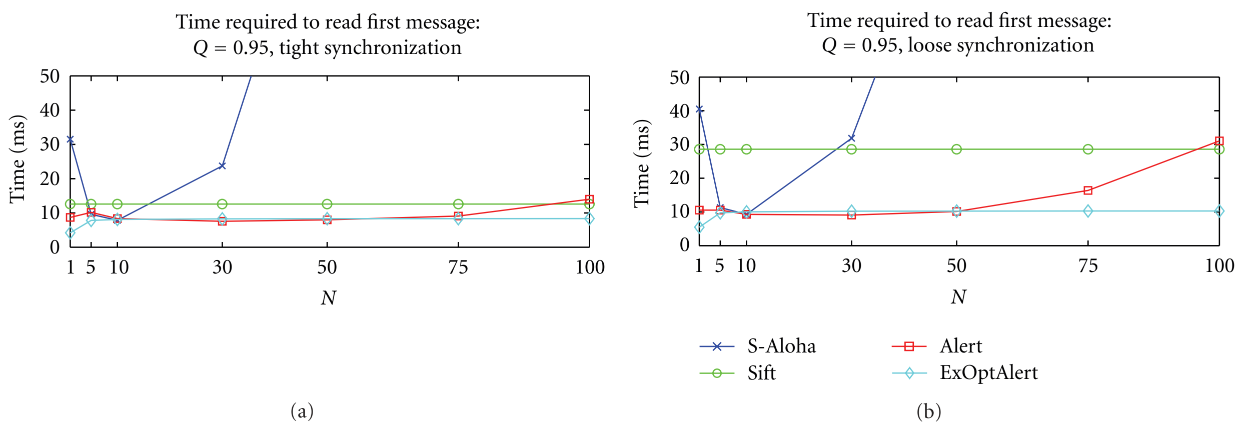


FIGURE 12: Average time required to receive first message with $Q = 0.95$.

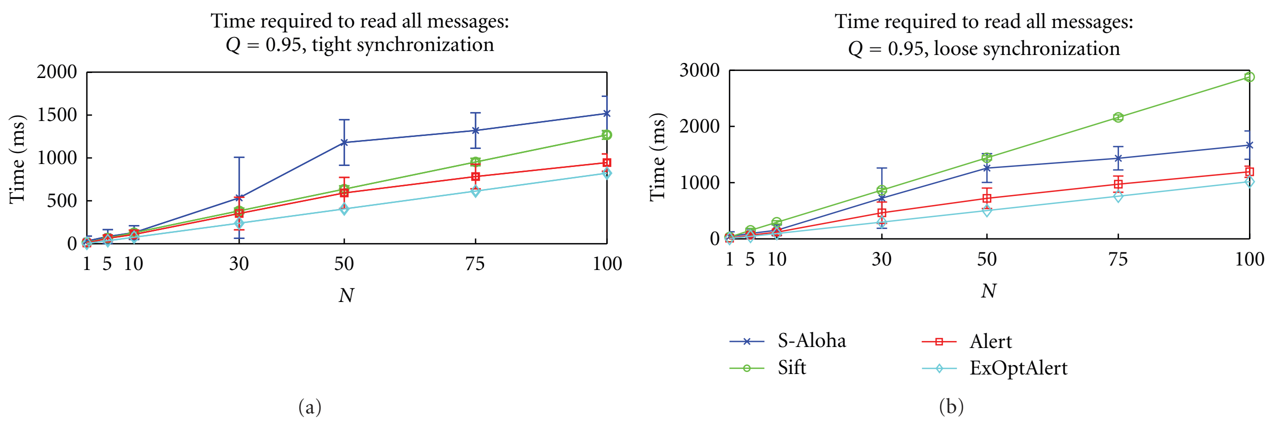


FIGURE 13: Average time required to receive all messages with $Q = 0.95$.

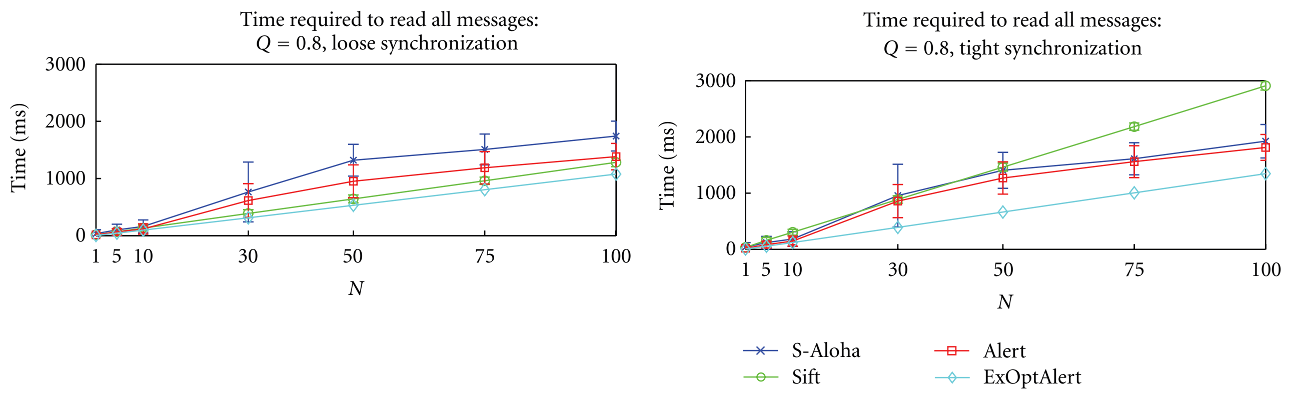


Figure 14: Average time required to receive all messages with $Q = 0.8$.

and 95% confidence intervals are shown for plots that show time required to read all messages. The maximum number of nodes was set at 100 which is a reasonably large number for a one hop centralized network.

6.3. Comparisons Through Simulations. Figure 12 shows the comparison between all 4 schemes for the average time required to send the first message. (Confidence intervals are not shown for this plot to allow a close up snapshot of the schemes other than S-Aloha.) It can be seen that Alert manages to send the first message out far earlier than Sift and S-Aloha, and is quite close to its optimal expected performance ExOptAlert. The chosen channel probabilities of Alert allow the first message to go through in the initial few slots. The same happens for Sift, but more backoff slots in its contention window mean it takes more time to send the first message even though it may be successful in the first time slot. A smaller contention window for Sift could be useful here, but that could have a negative impact on success probability of a single slot and, hence, the delay to send all messages. S-Aloha seems to have the most delay since the random slots picked by nodes to send may not be the initial slots, or if they are, may not be successful due to collisions. The small slot time does not seem to have helped S-Aloha in this case.

Figure 13 shows the comparisons for all schemes to read all messages when $Q = 0.95$ (interference level of 5%). For the tight synchronization case, we see that Alert does slightly better than Sift. Note that this result does not take into account additional procedures like RTS-CTS which Sift might need to employ to handle hidden terminal collisions. Alert being a noncarrier sense protocol does not suffer from such issues. When loose time synchronization is used, the difference between Alert and Sift increases; in fact, S-Aloha does better than Sift now due to the much smaller slot structure it uses. When a higher level of interference ($Q = 0.8$) is taken into consideration, Sift does better than Alert for the tight synchronization case because the latter has a higher possibility to be affected due to its use of multiple frequency channels (see Figure 14). The possibility of such high levels of interference ($Q = 0.8 = 20\%$ interference) is, however, very unlikely. In practice, Alert switches the frequency channels it uses periodically so that an interference source on some channel does not affect performance for long.

7. Conclusions

We presented Alert, a MAC protocol to collect rare event-driven messages from multiple wireless sensor nodes with low latency. The protocol uses a novel time slot structure with nodes separated by prioritized frequency channels, which allows one node to succeed per slot with high probability. We provided extensive theoretical justifications for selecting values for the design parameters involved, and designed an adaptive algorithm for Alert to adjust parameter values based on the level of contention in the network. The feasibility and effectiveness of the protocol were demonstrated through both an implementation as well as extensive simulation-based comparisons with other protocols.

Disclosure

A preliminary version of this paper appeared in proceedings of ACM/IEEE International conference on Information Processing in Sensor Networks (ACM/IEEE IPSN), April 2008.

References

[1] "Fire detection and fire alarm systems. part 25. components using radio links and system requirements," Tech. Rep. EN54-25, European Committee for Standardization, 2005.
[2] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 23–29, 1994.
[3] A. Goldsmith, *Wireless Communications*, Cambridge University Press, 2005.
[4] K. K. Chintalapudi and L. Venkatraman, "On the design of MAC protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 356–367, April 2008.

[5] IEEE802.11, “Wireless medium access control (MAC) and PHY) specifications for low rate wireless personal area networks,” IEEE Standard 802, part 15.4, (WPANs), 1999.

[6] IEEE802.15.4, “Wireless medium access control (MAC) and PHY) specifications for low rate wireless personal area networks,” IEEE Standard 802, part 15.4, (WPANs), 2003.

[7] A. Woo and D. Culler, “A Transmission control scheme for media access in sensor networks,” in *Proceedings of the 7th International ACM Conference on Mobile Computing and Networking (MOBICOM ’01)*, Rome, Italy, 2001.

[8] K. Jamieson, H. Balakrishnan, and Y. C. Tay, “Sift: a MAC protocol for event-driven wireless sensor networks,” in *Proceedings of the European Workshop on Wireless Sensor Networks (EWSN ’06)*, 2006.

[9] Y. C. Tay, K. Jamieson, and H. Balakrishnan, “Collision-minimizing CSMA and its applications to wireless sensor networks,” *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1048–1057, 2004.

[10] F. Calì, M. Conti, and E. Gregori, “Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, pp. 785–799, 2000.

[11] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, 2nd edition, 1992.

[12] N. Chirdchoo, W. S. Soh, and K. C. Chua, “Aloha-based MAC protocols with collision avoidance for underwater acoustic networks,” in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM ’07)*, pp. 2271–2275, May 2007.

[13] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, “Z-MAC: a hybrid MAC for wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 511–524, 2008.

[14] W. Ye, J. Heidemann, and D. Estrin, “Medium access control with coordinated adaptive sleeping for wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[15] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor netwo%rks,” in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 95–107, November 2004.

[16] T. Van Dam and K. Langendoen, “An adaptive energy-efficient MAC protocol for wireless sensor networks,” in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 171–180, Los Angeles, Calif, USA, November 2003.

[17] I. Demirkol, C. Ersoy, and F. Alagöz, “MAC protocols for wireless sensor networks: a survey,” *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

[18] G. Lu, B. Krishnamachari, and C. S. Raghavendra, “An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks,” in *Proceedings of the18th International Parallel and Distributed Processing Symposium (IPDPS ’04)*, pp. 3091–3098, Los Alamitos, Calif, USA, April 2004.

[19] M. Strasser, A. Meier, K. Langendoen, and P. Blum, “Dwarf: delay-aWAre robust forwarding for energy-constrained wireless sensor networks,” in *Proceedings of the 3rd International Conference on Distributed Computing in Sensor Systems (DCOSS ’07)*, Santa Fe, NM, USA, 2007.

[20] V. Namboodiri and A. Keshavarzian, “Alert: an adaptive low-latency event-driven MAC protocol for wireless sensor networks,” in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pp. 159–170, April 2008.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, NY, USA, 2004.