Design and Implementation of RAP - a Randomized Asynchronous Protocol for Data Aggregation in Wireless Sensor Networks

Jiahui Dai[1], Dmitry Degtyarev[1], Jingya Gao[1], Adrian Wang[1], Scott Burman[2], Ken Zillig[3], Dipak Ghosal[1]

***Abstract*— We consider a distributed deployment of sensors and a network of wireless relay nodes designed to operate in an environment in which there exists neither a stable energy source nor a global clock to which the network (sensor, relay, and gateway) nodes can synchronize. For such an environment, we design and implement a randomized asynchronous protocol (RAP) for aggregating data at the gateway. The protocol is based on the randomized sleeping of nodes in an asynchronous manner, taking advantage of the birthday paradox to ensure eventual communication. We present an approximate mathematical analysis and develop a simulation model to study the performance of the protocol under different sleep parameters: specifically, the duration of the random sleep times of the transmitter and the receiver. We show that a low average data transfer time can be achieved with low energy usage. We implemented the protocol on an Adafruit Feather 32u4 board, including a number of optimization to further improve the performance and minimize energy usage. Finally, we deployed a wireless sensor network for distributed temperature monitoring of aquaculture research at the Center of Aquatic Biology and Aquaculture (CABA) of University of California, Davis.**

***Index Terms*— Wireless Sensor Network, Randomized Protocol, Asynchronous Protocol, Simulation Analysis, Deployment**

I. INTRODUCTION

Wireless sensor networks (WSNs) [3], [24] are used in monitoring applications such as early detection of forest fires [8], actuating applications such as precision agriculture [12] or energy usage control in smart homes [19], [7], high resolution spatio-temporal monitoring in underwater environments [21], and tracking applications such as animal telemetry [10]. In a typical scenario, sensors are distributed over a geographical area and the data from these sensors are aggregated at a gateway node. This gateway then uploads the data to be processed and analyzed on a server. The focus of this study is on the data transfer protocol that aggregates the data from distributed sensors to the gateway node via the wireless mesh network of relay nodes.

Wireless sensor networks often operate in environments where there is no stable power source and no access to a network time server to synchronize their clocks. In such environments, a key requirement is to achieve high data fidelity while optimizing energy usage. Data fidelity depends on the application and could include (among other metrics) the fraction of the data generated by the sensors that is received at the gateway node and/or the data transfer time from the sensors to the gateway. In this study we consider a randomized and asynchronous data transfer protocol to address the environmental constraints of deploying a distributed monitoring system.

Randomized protocols have been studied in the context of WSNs. In a seminal paper [13], a randomized protocol based on the ”birthday paradox” was developed for ad hoc wireless networks to save energy during the deployment phase as well as to perform adjacent node discovery in an energy efficient manner. This paper analyzed a number of different birthday protocols both using abstract mathematical models as well as detailed simulation analysis. This work has over the years spurred a number of studies both in wireless ad hoc networks as well as in wireless sensor networks [15], [22], [16].

In this paper we adopt the birthday paradox to design and implement a randomized data transfer protocol to aggregate data at a gateway node. Unlike prior studies, we consider an asynchronous protocol. We consider a network of nodes with static routing which is achieved by pre-assigning channels on which nodes receive. Transmitters sleep for a duration that is sampled from a given distribution. Upon waking up the node transmits a packet (if a packet is present in its queue) and waits for an acknowledgement (ACK). The receiver also sleeps for a random time sampled from a given distribution. Upon waking up it listens for a specified amount of time for a data packet to arrive. Similar to the birthday paradox, we show that even when sleep duration is relatively large, the nodes can ”synchronize” relatively quickly, thereby achieving a low average data transfer time.

The asynchronous nature of the protocol eliminates the need for the nodes to synchronize to a global clock. This is an important feature as clocks in low-cost sensors and network nodes can have significant drifts. Furthermore, the asynchronous nature of the protocol also allows multiple senders to share a common channel to a receiver. The following are the main contributions of the present paper.

1) Design of a randomized asynchronous data transfer protocol to aggregate data in a wireless sensor network. The underlying randomization is based on the ”birthday paradox.” However, unlike previous studies of birthday paradox in self-discovery of sensor nodes, we have adopted an asynchronous approach that does not require any clock synchronization among the nodes.
2) Development of a mathematical model and a detailed simulation model to study the parameter space of the protocol and determine the parameter settings that co-optimize the data transfer time and the energy usage.
3) Implementation of the protocol for a distributed tem-

[1] Department of Computer Science, University of California, Davis
[2] Department of Land, Air and Water Resources, University of California, Davis
[3] Department of Wildlife, Fish and Conservation Biology, University of California, Davis

978-1-7281-4905-9/20/$31.00 ©2020 IEEE

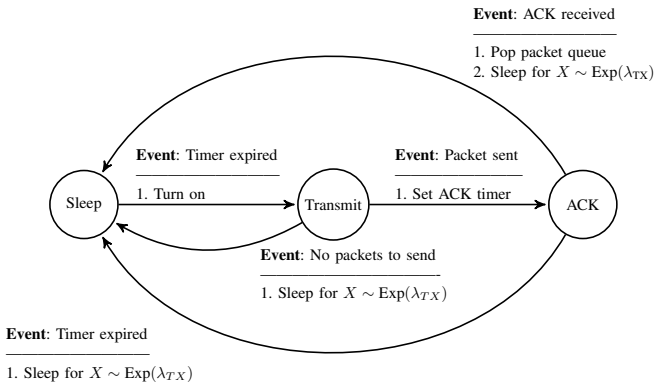

Fig. 1. The FSM representation of TX. The nodes represent the states and the arcs represent the transition which are triggered by events and invoke actions.

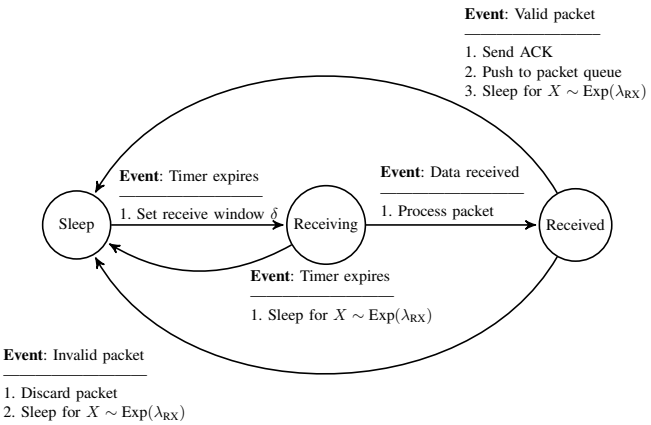

Fig. 2. The FSM representation of RX. The nodes represent the states and the arcs represent the transition which are triggered by events and invoke actions.

perature sensing application at the Center for Aquatic Biology and Aquaculture (CABA) at University of California, Davis (UC Davis). The protocol is implemented on a system with Adafruit Feather 32u4 boards programmed as relay nodes and a raspberry pi programmed as the gateway node. The results indicate that the implementation achieves the benefits predicted by the simulation and mathematical analyses of the protocol.

II. The Protocol

Wireless sensor networks often operate in environments without a stable power source. In these cases, a synchronous protocol would exhaust energy too quickly, as it requires either a reliable clock for consistent synchronization, or, network nodes remaining awake continuously, idly listening for a signal with which they establish a connection. With energy-saving in mind, a protocol which allows nodes to send messages to each other without having to synchronize would allow nodes to sleep, turning on only periodically to reduce energy usage. Through randomized transmissions, we constructed an asynchronous message transfer protocol to operate in low-energy environments.

A. Protocol Design

The protocol is separated into two modes — Transmit (TX) and Receive (RX) modes. Nodes in the network can have either or both modes running concurrently depending on if they are a source node (producing data), a relay node, or a sink node (the gateway). This overview focuses on how a transmitter transmits its message to a receiver, rather than how a message will be routed through a network to reach a destination node. With respect to routing, the easiest and most straightforward method by far is a static routing multi-hop architecture culminating in a single message sink, as described in [3]. However, with some further design, the protocol can be modified to support either static routing or dynamic routing methods, such as in a mesh network.

The Finite State Machine representations of TX and RX are shown in Figure 1 and Figure 2, respectively. The transmitter will transmit periodically, with a sleep time (denoted by X_{TX}) sampled from a negative exponential distribution with parameter λ_{TX} such that $X_{\text{TX}} \sim \text{Exp}(\lambda_{\text{TX}})$. Upon waking up, the transmitter will transmit a packet (if it has one) and transition to the waiting state for ACK. Similarly, the receiver periodically receives and then goes to sleep with a sleep time sampled from a negative exponential distribution with parameter λ_{RX}. Upon waking up, it listens for a transmission for a time denoted by δ. If it receives a valid packet (determined by a checksum that is appended with the transmitted packet), it sends an ACK and goes back to sleep for a random period of time. If a correct ACK is not received within 10 ms, the transmitter will reattempt the packet as a new packet.

B. Mathematical Analysis

For data transfer to occur successfully, the receiver must be receiving at the same time that the sender is transmitting (i.e. they synchronize). The probability that they will synchronize is determined by the parameters λ_{TX}, λ_{RX}, and δ. The expected values of the TX and RX sleep times are λ_{TX}^{-1} and λ_{RX}^{-1}, respectively.

Low RX/TX sleep times will increase synchronization probability while high RX/TX sleep times will decrease synchronization probability. While low sleep times increase synchronization probability, for multiple senders transmitting to one receiver, the probability of collision, where two senders transmit at the same time, increases. This can be modeled following the birthday paradox, but further discussion is omitted for brevity. Another parameter is the ACK wait time, the duration for which the transmitter will wait for an ACK. However, the ACK wait time's effect on synchronization probability is negligible.

We can estimate the expected time for a successful transmission. Let the time taken for TX and RX to synchronize be given by the random variable T. For simplicity, we assume that if TX and RX are both awake at the same time a packet will be successfully transmitted. We consider the case of when RX becomes awake at time t for a window of time δ,

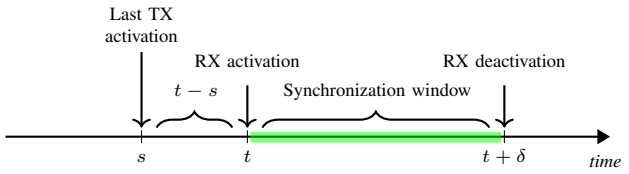

Fig. 3. This timeline shows the series of events that occur in the case mentioned in the mathematical motivation.

and TX was last active at time $s, s < t$. Any other scenario simplifies into this case.

Figure 3 shows the timeline of events. When RX activates, the sender has slept for $t - s$ seconds. Let X be the random variable that denotes the time for the transmitter to wake up after the receiver activates. Since the receiver activation is a random incidence on the sleep time of TX, which has a negative exponential distribution, then X has the same distribution as X_{TX} due to the memoryless property.

Synchronization occurs when the sender wakes up within the window of time that the receiver is active. Thus the probability of synchronization is given by

$$P(X < \delta) = 1 - e^{-\lambda_{\text{TX}}\delta} \tag{1}$$

Since the sleep times are independent and identical distributed, we make the assumption that each receiver activation is a trial with the probability of success $1 - e^{-\lambda_{\text{TX}}\delta}$. Let Y be a random variable that denotes the number of trials before the sender and receiver synchronize. This then takes the form of a Geometric distribution with parameter $(1-e^{-\lambda_{\text{TX}}\delta})$. Consequently, the expected number of trials to succeed is given by

$$E[Y] = \frac{1}{1 - e^{-\lambda_{\text{TX}}\delta}}. \tag{2}$$

We make another assumption that the time in between trials on average is equal to $\lambda_{\text{RX}}^{-1} + \delta$, which is the expected sleep time of the receiver plus its awake window. Let T be the random variable that denotes the time until synchronization. The expected value of T is then given by

$$E[T] = \frac{1}{1 - e^{-\lambda_{\text{TX}}\delta}}(\lambda_{\text{RX}}^{-1} + \delta) \tag{3}$$

We estimate energy usage with the assumption that each TX activation incurs a constant cost of a_{TX}, while each RX activation incurs a cost that is a function of its window size $a_{\text{RX}}\delta$. Let $B_{\text{TX}}, B_{\text{RX}}$ be the random variables representing the energy used by TX and RX to successfully transmit one packet. Then, the estimated expected energy used by TX and RX are

$$E[B_{\text{TX}}] = a_{\text{TX}}\lambda_{\text{TX}}\frac{1}{1 - e^{-\lambda_{\text{TX}}\delta}}(\lambda_{\text{RX}}^{-1} + \delta) \tag{4}$$

$$E[B_{\text{RX}}] = a_{\text{RX}}\delta\frac{1}{1 - e^{-\lambda_{\text{TX}}\delta}} \tag{5}$$

(4) follows from the fact that the number of TX activations is a Poisson process with rate λ_{TX}, and (5) follows from the fact that the number of trials in G is the number of RX activations.

In this section, we studied the impact of the parameters $(\lambda_{\text{TX}}, \lambda_{\text{RX}}, \delta)$ on T and the parameters $(a_{\text{TX}}, \lambda_{\text{TX}}, \lambda_{\text{RX}}, \delta), (a_{\text{RX}}, \lambda_{\text{TX}}, \delta)$ on $(B_{\text{TX}}, B_{\text{RX}})$ respectively. One feature of interest is that with respect to the parameter δ given $\lambda_{\text{TX}}, \lambda_{\text{RX}}$, there is an optimal (minimum) packet transfer time which can be determined.

C. Importance of the Asynchronous Property

The asynchronous property of the RAP protocol gives three main advantages over a synchronous protocol: avoidance of continuous clock misalignment, skewed clock independence, and collision avoidance.

1) A synchronous protocol with a set time period for advertisement and synchronization can lead to continuous clock misalignment (or alignment) between two nodes, where the communication is constantly failing or constantly succeeding (thus blocking other nodes on the channel). An asynchronous protocol naturally resolves this potential issue by producing random synchronizations that are independent and unlikely to form persistent timing patterns.
2) More generally, because of the random timing of the asynchronous property, each node has no timing reliance on its previous state. The overall network remains unaffected by individual clock skews, whereas a synchronous network can be blocked by continuous clock misalignment.
3) Synchronization to a global clock can increase the chances of collision when two or more nodes attempt to transmit on the same channel. An asynchronous protocol naturally avoids repeated collisions on the same channel by randomizing transmission and receiving timings.

III. Simulation Analysis

To model the protocol, we developed an event based simulation environment, used to simulate networks running the protocol (Section V). The simulation model allowed us to explore the impact of various protocol parameters and network tree configurations. The simulation is written in Python and uses the SimPy library [20]. SimPy is an event simulation framework which lets us create interacting processes. The simulation mirrors the nodes in the deployment by creating a process for each node and making them communicate through timed events. The code of the simulation is modeled after the deployment code, there are state changes, similar timings and similar packet queue behavior.

The key input parameters of the simulation are the two sleep rates, λ_{RX} and λ_{TX}. The simulation outputs all of the packets that have been transmitted. Each packet has the following information: when it was created, which node created it, when it reached the gateway, and how long it took for the packet to reach the gateway (packet age). These outputs are processed to measure the distribution of the packet age. In addition, simulation also records the metadata of each node. Specifically, it records how long the node sleeps, how long it spends transmitting packets, and how

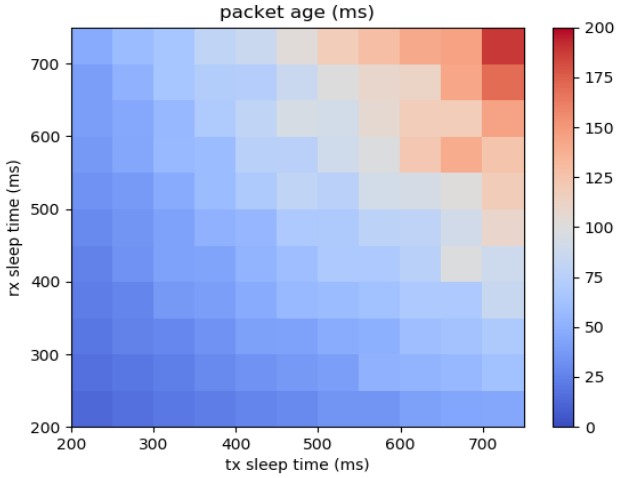

Fig. 4. The packet age as a function of λ_{RX}^{-1} and λ_{TX}^{-1}. Parameter: $\delta = 10\ ms$

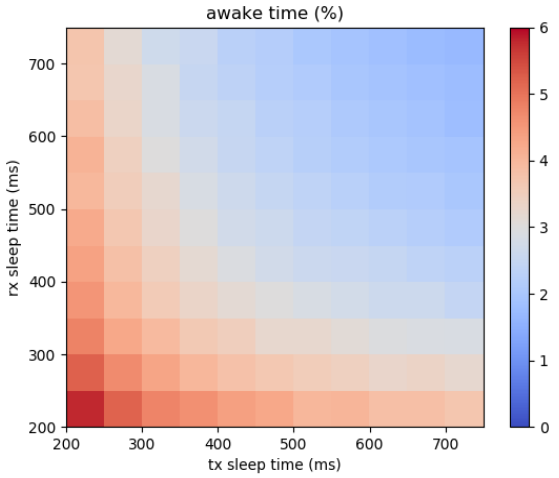

Fig. 5. Percentage of time awake as it relates to the λ_{RX}^{-1} and λ_{TX}^{-1}. $\delta = 10\ ms$

long it spends receiving packets. The output is a percentage of total time for each type of activity. These statistics serve as approximations of the power usage of nodes. Since the actual power usage for each type of activity depends on factors such as the hardware used and the physical environment, we use the sum of time spent for transmitting and receiving as a proxy for actual power usage.

A. Simulation Results

For our simulation analysis we first consider a three node network consisting of a source node, a relay node, and a sink node. The source node generates packets whenever its queue is empty, then packets travel to the relay node, and finally to the sink node. We vary TX and RX sleep times throughout our simulation. Note that sleep times are inverses of the sleep rates, λ_{TX} and λ_{RX}.

a) Packet Age: The graph of packet ages (Figure 4) looks like a slope with a peak at highest sleep times and the lowest point at lowest sleep times. Therefore, packet age is proportional to sleep time. This is as expected; if the nodes sleep for shorter periods of time, they will transmit and receive more often. The slope is symmetrical along TX/RX axes, so we can conclude that contributions from RX/TX activity to the overall performance of the network are roughly the same.

b) Total Awake Time: Awake time graph (Figure 5) is also a sloped mesh. In the case of awake time, the lowest point happens at highest sleep times, the highest point happens at lowest sleep times. Awake time is inversely proportional to average sleep time; as nodes sleep more, the time and energy spent transmitting and receiving goes down.

c) Multiple Sender with Single Receiver: We expand upon our prior simulation with a network consisting of a single sink node with N source nodes attached to it, where N varies and λ_{TX} and λ_{RX} parameters are fixed at 1/400ms. The purpose of this simulation is to find out how much the network performance performs as the number of children

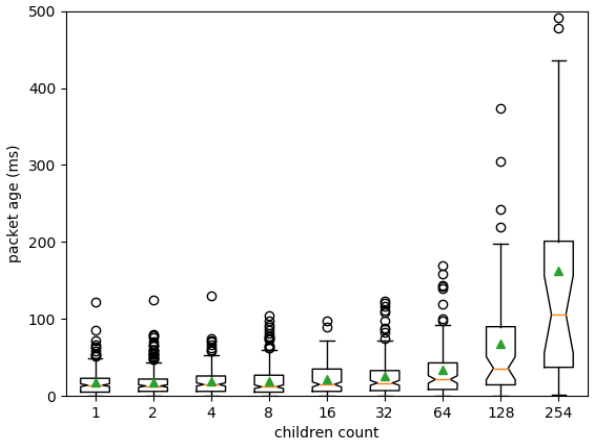

Fig. 6. The boxplot of the packet age as the number of children increases. Parameters: $\lambda_{\text{TX}} = \lambda_{\text{RX}} = 1/400$ ms $\delta = 10\ ms$.

increases. According to the simulation results (Fig. 6), the performance doesn't decline significantly and is almost stable within 1 to 32 children. Packet age start to increase when the number of children exceeds 128.

IV. Implementation

A. Libraries and environment

We chose to implement RAP in the open source Arduino environment, using the interactive Adafruit Feather 32u4 Radio (RFM69HCW) 433MHz [1] as the backbone of the implementation, which can be programmed to read input from sensors as well as communicate on radio signals. The Arduino IDE, which includes a user interface and a serial monitor, is used for programming and debugging the data transfer application.

The RadioHead Packet Radio library [14]is an object oriented library for sending and receiving formatted data on

a range of embedded processors. It abstracts packet transmission and reception processes, including packet creation, CRC checking, transmission-related timing setup, and acknowledgement into simple functions. The Arduino SleepyDog library [2]configures the sleep behaviors of Adafruit Feather 32u4, allowing the minimization of energy usage in the inactive state.

B. Process structure

The master process is controlled by a finite state machine that schedules sub-processes based on a master timer. Sub-processes such as transmit (TX), receive (RX), data packet creation, and health packet creation are executed via corresponding scheduled awake time. The finite state machine determines the next state based on the shortest time to the next scheduled sub-process. For example, the transmit process and the receive process are both scheduled via a negative exponential function, while the packet generation sub-processes are set based on specific deployment requirements. As a result, the transmit and the receive sub-processes are not guaranteed to be in order and varies due to their corresponding sampled sleep times. A timer based state machine is preferred over a linear process because it allows an integrated TX and RX process. While the queue is non-empty, unsuccessful TX transmissions do not affect or block the RX process.

C. Optimizations

After the initial implementation of RAP with simple transmit and receive functionality, several additional features were implemented in order to further improve performance.

1) Packet Chaining: RAP creates a synchronization process that is completely random, and thus intermittent and unpredictable by nature. In order to maximize a successful communication, upon successful receipt of a message, both the sender and receiver enter a synchronized state where multiple messages can be transmitted without additional synchronization. After successfully receiving an acknowledgement, the sender will attempt to send another packet (if it has one) and wait for another acknowledgement. Similarly, the receiver will wait for another message after successfully receiving a packet and acknowledging. If it receives another message, this process is repeated.
   In networks where messages can accumulate in nodes, the efficacy of one successful synchronization can be significantly increased. This lessens the number of independent communications between two nodes and consequently lessens the overall networking load at the trade off of short bursts of network monopolization by a single connection.
2) Concurrent RX/TX: In this implementation, a node runs both the TX and RX processes simultaneously; that is, the TX and RX processes are independent and concurrent. Two timers, for TX and RX respectively, are both set by inverse exponential functions. In a basic implementation of the protocol, the RX process is run after the TX process, creating a pseudo order that prioritizes the TX process (given a non-empty queue), obstructing the random property of the RAP protocol. With concurrent RX and TX process, a node can interchange between TX and RX process solely based on their individual timer.
3) Duplicate Detection: If a packet successfully reaches the receiving node but the overall transmission fails – that is, if the acknowledgement fails to return to the sender – the receiving node saves the packet and filters out any duplicates received in later transmissions. A packet from any child is considered unique if its origin ID and sequence number are jointly different from the saved record in the respective hash tables of the parent. The parent will send acknowledgement in response to duplicates to prevent further repeated transmission.

V. Deployment and Results

The deployment has three kinds of nodes: sensor node, relay node, and gateway node, whose components are respectively Adafruit feather 32u4, temperature sensor DS18B20 [6], and raspberry pi. Adafruit feather 32u4 is a portable microcontroller board with radio module from Adafruit. The feather board sends and receives packets via radio. The temperature sensor is the DS18B20 digital waterproof thermometer from Sparkfun, whose 1-Wire interface allows multiple sensors to connect to a single pinout on a broad. Raspberry Pi is a single-board computer that uploads data to the internet. All nodes are sealed in waterproof cases.

The CABA facility at UC Davis (Figure 7) was chosen for the in-field deployment and testing of the wireless sensor network. In this context our network continuously monitors and reports the water temperatures of 20 tanks containing Chinook salmon (*Oncorhynchus tshawytscha*). This application allows for continuous temperature monitoring and real-time data collection on tank temperature.

a) Sensor Nodes: A sensor node contains one Adafruit Feather board and four sensors. We design a node that can connect up to six sensors, but four sensors were best suited for this application setup. All sensor probes were submerged in the tanks. The node processes the four temperature readings from sensors into a packet and sends the packet to its parent node using the wireless interface.

b) Relay Nodes: A relay node contains an Adafruit Feather board. It receives packets from its children nodes and sends packets to its parent node. A relay node can have multiple children nodes but only one parent node.

c) Gateway Node: A gateway node is similar to a relay node, except it connects to a Raspberry pi via USB. It receives packets and sends the data to raspberry pi, which then uploads the data to Google Drive using WiFi.

d) Packet Structure: A packet consists of the following fields: packet age(0-65536 seconds), previous child node id(0-254), packet number(0-254), origin node id(0-254), sequence of readings(0.1-99.0 floats). The packet age denotes the cumulative time the packet spent in the path to the Gateway node. For example, consider a packet with the

Fig. 7. Aerial view of CABA. Nodes (35, 36, and 131) are inside the building (shelter 3). Nodes (69, 70, 71, 133, and 254) are outside of buildings.

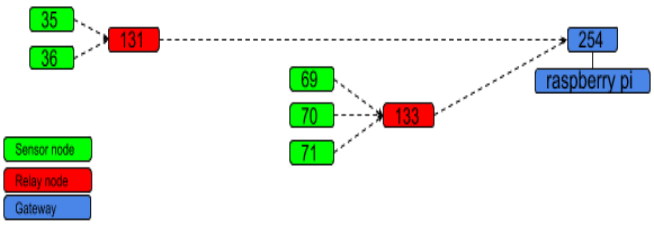

Fig. 8. The topology of the wireless sensor network deployed at CABA.

following data values: ”45, 3, 120, 7, 20.3, 25.2, 31.0, 15.4, 0.0, 0.0”. This means the packet has cumulatively spent 45 sec in all the previous node, it came from node 3, the packet number is 120, and the first node, where the packet was created, is node 7. The last six float digits are temperature readings.

e) Health packet: Health packets are generated at all relay nodes and the gateway node. These packets are used to confirm the status of the nodes.

A. Deployed Network

We can only use the radio frequency from 433 MHz to 434.5 MHz in the area around CABA. The Adafruit Feather can distinguish radio channels that are 0.25 MHz apart, thus the deployed WSN has 6 radio channels, 433.00 MHz, 433.25 MHz, 433.50 MHz ... 434.50 MHz. As shown in Figure 8, the WSN has three layers: 1. the gateway node (254), 2. the two relay nodes (129 & 130), and 3. the five sensor nodes (35, 36, 69, 70, and 71). Each layer uses one channel to prevent transmission to incorrect nodes.

B. Deployment Result

Figure 9 shows the box plots of the packet age for packets generated by the different sensors in the deployment. Using

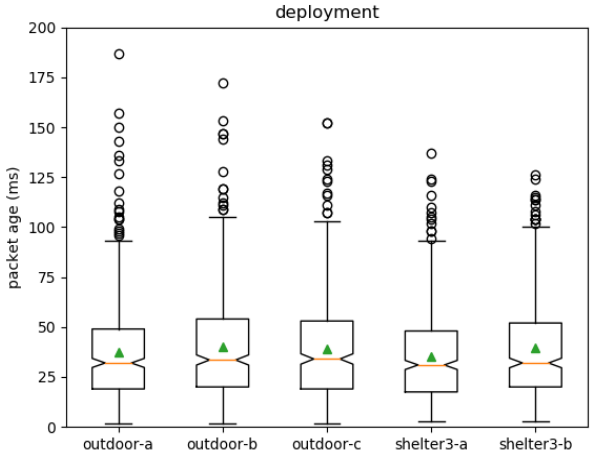

Fig. 9. The box plots of the packet age for the different sensors from the deployment. Parameters: $\lambda_{\text{TX}} = 1/200$ ms, $\lambda_{\text{RX}} = 1/600$ ms, and $\delta = 10ms$.

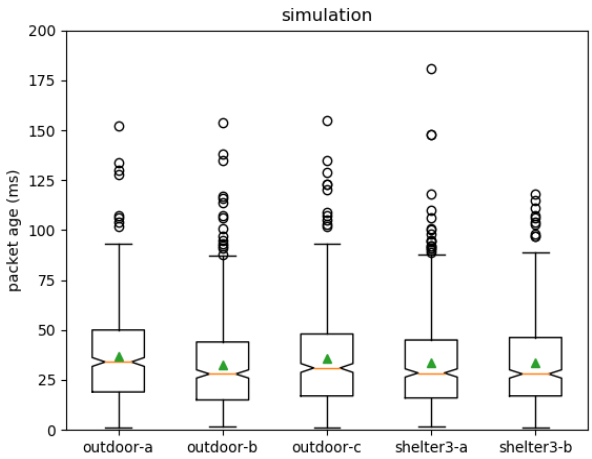

Fig. 10. The box plots of the packet age for the different sensors from the simulation of the deployment topology. Parameters: $\lambda_{\text{TX}} = 1/200$ ms$\lambda_{\text{RX}} = 1/600$ ms $\delta = 10ms$.

simple grouping, the graphs show few outliers and are within reasonable limits. The ages are similar for all nodes, meaning that the randomized protocol spreads performance across nodes evenly. The ages from outdoor nodes (a group of 3) are slightly higher than ages from shelter3 (a group of 2), as expected, because higher number of children connected to the same relay node increases packet age slightly due to collisions.

Figure 10 shows the simulation results of deployment topology. The simulation results match the deployment results, thus validating the simulation model.

VI. Related Work

Optimizing energy efficiency of wireless sensor networks has been studied extensively [9]. A Sparse Topology and

Energy Management (STEM) [18] algorithm was proposed to efficiently wake up nodes from a deep sleep state without the need for an ultra low-power radio. This allows a tradeoff between energy efficiency and the latency that is incurred due to waking up the node. The tradeoff between data fidelity and energy efficiency of the sensor network has been investigated in [4], Furthermore, energy efficient routing has been studied in [17], [11]

The application of the birthday paradox in wireless sensor networks was reported in [13]. The paper adopted the birthday paradox to develop a suite of protocols (referred to as birthday protocols) that saves energy during the deployment phase of the sensors as well as the node discovery phase. The approach has been been adopted in a number follow-up studies [15], [23], [5]. In this paper we adopt the birthday paradox and develop data transfer protocol for static wireless sensor network. We also implement the protocol on a wireless device and demonstrate the advantage of RAP both in terms of energy savings and in terms of having multiple source share a channel to a single receiver.

VII. Conclusions

In this paper we designed and implemented a randomized and asynchronous protocol to aggregate data in a wireless sensor network. Both the randomized and asynchronous nature of the protocol is important for energy efficiency of the network nodes, ensuring their functionality without requiring central clock synchronization. We analyzed the protocol using mathematical model and detailed simulation, demonstrating that RAP achieves low mean latency of data transfer while minimizing the energy usage. Finally, we implemented the protocol and deployed it for distributed temperature monitoring at CABA of UC Davis.

In our current deployment, our sensor network adopts a fixed tree network with static routing, which can be extended to dynamic neighbour discovery in the future. Additionally, we plan to develop an adaptive algorithm to adjust the sleep rates to node and network states. For example, the node can automatically adjust the sleep rate according to the number of packets it has received and is going to send, or battery life if batteries power the nodes. The extension can further improve the efficiency of packet transmission and increase the lifetime of the nodes. Finally, we plan to improve network monitoring. In our current deployment, the health packet contains the number of packets waiting in the queue only. More node information can be added, such as battery life, number of send or receive packets, number of failure transmission, and other data that are related to the status of nodes. The extension can help us better understand the status of the network.

ACKNOWLEDGMENTS

This work was supported by grants from the United States Fish and Wildlife Service and the National Science Foundation (NSF) grant CNS-1528087.

References

[1] Adafruit Industries. Adafruit feather 32u4 radio with RFM69HCW module. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-32u4-radio-with-rfm69hcw-module.pdf.

[2] Adafruit Industries. Adafruit sleepydog arduino library. https://github.com/adafruit/Adafruit_SleepyDog.

[3] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[4] Athanassios Boulis, Saurabh Ganeriwal, and Mani B Srivastava. Aggregation in sensor networks: an energy–accuracy trade-off. *Ad hoc networks*, 1(2-3):317–331, 2003.

[5] Tingjun Chen, Javad Ghaderi, Dan Rubenstein, and Gil Zussman. Maximizing broadcast throughput under ultra-low-power constraints. *IEEE/ACM Transactions on Networking*, 26(2):779–792, 2018.

[6] Dallas Semiconductor. DS18B20 programmable resolution 1-wire digital thermometer. https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf.

[7] Dae-Man Han and Jae-Hyun Lim. Smart home energy management system using ieee 802.15. 4 and zigbee. *IEEE Transactions on Consumer Electronics*, 56(3):1403–1410, 2010.

[8] Mohamed Hefeeda and Majid Bagheri. Forest fire modeling and early detection using wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 7(3-4):169–224, 2009.

[9] Holger Karl and Andreas Willig. *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[10] Roland Kays, Margaret C Crofoot, Walter Jetz, and Martin Wikelski. Terrestrial animal tracking as an eye on life and planet. *Science*, 348(6240):aaa2478, 2015.

[11] Arvind Kumar et al. *Energy Efficient Clustering Algorithm for Wireless Sensor Network*. PhD thesis, Lovely Professional University, 2017.

[12] Subramania Ananda Kumar and Paramasivam Ilango. The impact of wireless sensor network in the field of precision agriculture: a review. *Wireless Personal Communications*, 98(1):685–698, 2018.

[13] Michael J McGlynn and Steven A Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145. ACM, 2001.

[14] Mike McCauley. Radiohead packet radio library for embedded microprocessors. https://www.airspayce.com/mikem/arduino/RadioHead/.

[15] Santashil PalChaudhuri and David B Johnson. Birthday paradox for energy conservation in sensor networks. *Sleep*, 9:14mA, 2002.

[16] Bryan Parno, Adrian Perrig, Virgil Gligor, et al. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy*. Oakland, CA, USA, 2005.

[17] Venkatesh Rajendran, Katia Obraczka, and Jose Joaquin Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless networks*, 12(1):63–78, 2006.

[18] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE transactions on mobile computing*, 1(1):70–80, 2002.

[19] Biljana L Risteska Stojkoska and Kire V Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464, 2017.

[20] Team Simpy. Documentation for simpy. c2002-2019. https://simpy.readthedocs.io/en/latest/contents.html.

[21] J. Wang, D. Li, M. Zhou, and D. Ghosal. Data collection with multiple mobile actors in underwater sensor networks. In *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pages 216–221, June 2008.

[22] Qiang Wang, Andreas Terzis, and Alex Szalay. A novel soil measuring wireless sensor network. In *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*, pages 412–415. IEEE, 2010.

[23] Qiang Wang, Andreas Terzis, and Alex Szalay. A novel soil measuring wireless sensor network. In *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*, pages 412–415. IEEE, 2010.

[24] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.