EpiFi: An In-Home IoT Architecture for Epidemiological Deployments

Philip Lundrigan, Kyeong T. Min, Neal Patwari, Sneha Kumar Kasera, Kerry Kelly, Jimmy Moore, Miriah Meyer,
Scott C. Collingwood, Flory Nkoy, Bryan Stone, and Katherine Sward
University of Utah, USA
{philipbl@cs., kyeong.min@, npatwari@ece., kasera@cs., kerry.kelly@, jimmy@cs., miriah@cs.,
scott.collingwood@hsc., flory.nkoy@hsc., bryan.stone@hsc., kathy.sward@nurs.} utah.edu

***Abstract*—We design and build EpiFi, a novel architecture for in-home sensor networks which allows epidemiologists to easily design and deploy exposure sensing systems in homes. We work collaboratively with pediatric asthma researchers to design multiple studies and deploy EpiFi in homes. Here, we report on experiences from two years of deployments in 15 homes, of two different types of studies, including many deployments continuously monitored over the past year. Based on lessons learned from these deployments and researchers, we develop a new mechanism for sensors to bootstrap their connectivity to a subject's home WiFi router and implement data reliability mechanisms to minimize loss in the network through a long-term deployment.**

***Index Terms*—Internet of Things, Network security, Wireless networks**

I. Introduction

The holy grail of epidemiological research is to have continuous sensing of every person's exposures and activities, alongside data on their health outcomes. The conglomeration of the environmental effects on a person over their lifetime is called their *exposome* and, in interaction with their genome, plays a large part in their health [23]. Detailed exposome data from a segment of the population, including from sensors in their homes, could provide researchers new insights about the relationships between exposure and chronic diseases, such as heart disease, cancer, diabetes, and asthma, and how we can improve our health and reduce the incidence and costs of these diseases [22]. Low-cost internet-of-things (IoT) devices, including wireless and wearable sensors, are enabling the "quantified self" for those who are technologically skilled and interested in self-monitoring [21].

In contrast, for epidemiology researchers who wish to deploy, maintain, and obtain reliable data from a large population of volunteer subjects on a limited budget, several challenges must be addressed. The individual user is driven by the cost of devices; but researchers' costs are primarily comprised of the cost of study management, including a high personnel and travel cost any time it is required to meet at the subject's home. Inconvenienced subjects expect more compensation and may drop out of a study. Deployment at a subject's home must be fast and reliable, regardless of the particular configuration or robustness of the subject's home wireless network or the number of wireless sensors to be deployed. A researcher must analyze data from dozens or hundreds of subjects and large data loss from one subject may force them to throw out that subject entirely to preserve uniformity across subjects. Finally, medical researchers must ensure that any deployed system abides by laws regarding medical information privacy, such as the Health Insurance Portability and Accountability Act (HIPAA) in the US. In short, the costs and constraints for human subject research studies are fundamentally different from those of individuals who wish to monitor themselves. Today's networking tools are focused on the individual user, rather than the researcher, and new tools are needed to enable a new kind of epidemiology research.

In this paper, we present the design and development of *EpiFi*, a system that gathers data from home monitoring devices with a low implementation burden. We share experiences and problems we faced with deploying EpiFi in different homes and settings. EpiFi has been part of five different studies and been deployed in 18 different locations, ranging from homes to an Air Force hangar, and has collected over 210 million measurements. EpiFi includes novel tools to address the challenges of human subject IoT deployments. The goal of EpiFi is to allow epidemiologists to create comprehensive experiments from start to finish. In the process of building, deploying, and maintaining EpiFi, we encountered three key problems: *secure WiFi association bootstrapping*, *reliable data transfer*, and *WiFi disruptions*. We present solutions to the first two problems and outline future work for the third problem.

WiFi association bootstrapping is the process of connecting WiFi sensors to a participants home network. From our experience, current methodologies do not scale to multi-sensor deployments. To solve this problem, we build a novel protocol for an Ethernet connected device to securely share the subject's home WiFi network name and password to a group of unassociated WiFi sensors. We use key insights about how Ethernet frames are encapsulated, encrypted, and transmitted by wireless routers to encoded the credentials in the Ethernet source and destination address fields. The protocol protects against common threat vectors such as packet manipulation and replay attacks. It uses erasure coding to minimize the effects of wireless packet loss.

We learn through our deployments that care must be given to ensure that no data is lost even when using "reliable"

978-1-5386-5097-4/18/$31.00 ©2018 IEEE

protocols. To solve this problem, we create an approach for data persistence and a reliable protocol for sensors to transmit data. Our lightweight data persistence layer allows sensors to store data locally when there is a network outage or weak signal strength. This backlog of data protects against data loss when a sensor is disconnected from power. We build a reliable protocol for sensors to transmit data. This approach ensures that all data measured by sensors is transmitted to the cloud for storage without loss. Our reliable protocol performs as well as MQTT under normal network conditions and much better than MQTT when a sensor has a backlog of data. We reduce data loss from up to 25% to 0%.

When a sensor stops sending data, it is impossible to know remotely if it is from a WiFi disruption or something more serious, such as loss of power or hardware malfunction. Not knowing can lead to unnecessary data loss. We show the impact WiFi disruptions have on our deployments. We highlight the importance of support for lower data rates and look to new WiFi specifications or systems to address this problem.

The problems of secure WiFi bootstrapping, reliable data transfer and WiFi disruptions have been studied to various degrees in previous work, however, we are not aware of any system that simultaneously addresses all of these problems. These are practical problems that we faced while building and deploying our system and receiving feedback from users of our system. EpiFi is open source and built on open source components. We highlight lessons learned from two of the five studies that use EpiFi. We describe the key challenges that should be considered in the deployment of sensor networks for purposes of epidemiology research and how EpiFi addresses these issues. We get input from pediatric asthma researchers on the design of EpiFi and realizations of the system we deploy in the homes of human subject study participants.

II. EpiFi Architecture

EpiFi consists of four components, as seen in Figure 1: database, gateway, home access point, and sensors. The gateway, home access point, and sensors are co-located in the home, and the database is in the cloud. The gateway acts as a central hub with all sensors and devices communicating with it. Having all the sensors communicate with the gateway, rather than directly with the database, provides important benefits. The gateway allows the sensors to be as simple as possible – all logic, configuration, and storage for a deployment is on the gateway. Also, the gateway provides local processing to add privacy protections, actuation, and user feedback. The gateway increases the variety of deployments that EpiFi supports, in part by allowing connections to Z-Wave, ZigBee, and Bluetooth devices.

To provide support for a wide variety of IoT devices, we build upon an open source project called Home Assistant [18], which runs on the gateway. It includes user-contributed components that allow it to interface with devices and web services. As of this writing, Home Assistant supports over 1,000 user-contributed components. It has a REST API and it supports HTTP, MQTT, raw TCP sockets, and custom components. Custom components allow users to add their own functionality without changing the core of Home Assistant. This makes it easy for new devices and sensors to integrate with Home Assistant. We develop several custom components for EpiFi.

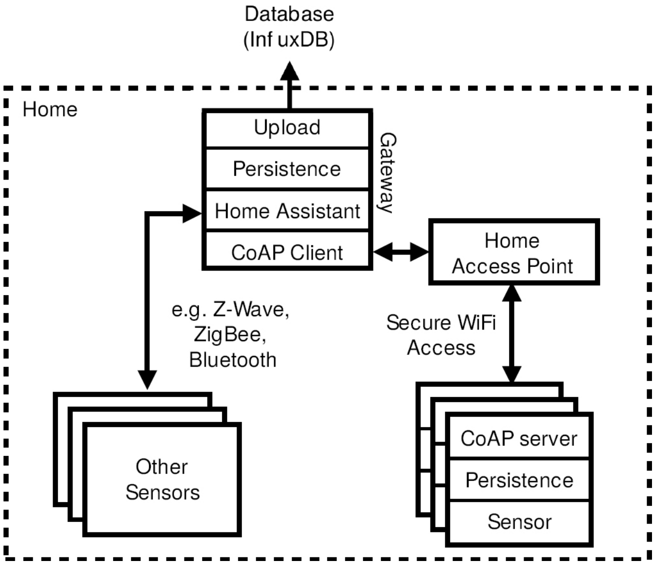
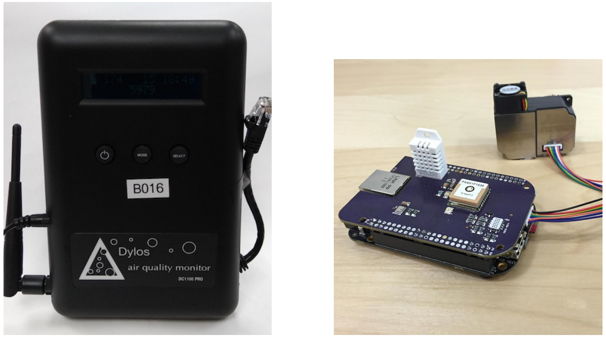


Fig. 1: General architecture of EpiFi.



(a) Modified Dylos air quality sensor

(b) Plantower air quality sensor

Fig. 2: Sensors used in deployments and evaluations.

For our remote database, we use InfluxDB [10], a database designed specifically for the kinds of time-series data epidemiologists want to collect. The database runs on a server in a protected environment that is HIPAA compliant. All data that is uploaded from the gateway to the server is encrypted using SSL.

Although the gateway supports a broad range of wireless protocols, for the remainder of this paper, we focus on WiFi sensors because 1) WiFi hardware is readily available and inexpensive, 2) WiFi is more widely deployed compared to other wireless protocols [20], and 3) a WiFi sensor can integrate with the rest of the home because it uses IP, making the sensor easier to remotely debug and monitor. EpiFi supports studies that use commercial sensors, but we feel that the most research-relevant studies will be done with custom sensors. We built two WiFi sensors to measure air quality, shown in Figure 2 and discussed in the next section.

III. Deployments

EpiFi has been part of five different studies and been deployed in 18 different locations. These locations include many different homes and even an Air Force hangar. In total, EpiFi has collected over 210 million measurements. In this paper, we highlight two studies, each deployed in multiple homes. We work with epidemiology researchers who study the relationship of exposure to indoor air pollution and the symptoms and treatment of children with asthma. We also work with air pollution scientists who understand the chemistry and appropriate sensors for indoor air quality measurements. Together, with these domain experts, we design and deploy the experiments. First, we describe the sensors we used, and then, each deployment.

A. Sensors

To measure the air quality in homes, we use two air quality sensors (Figure 2): a modified Dylos DC1100 [7] sensor and the Plantower PMS3003 [15] sensor. Both of these sensors measure the concentration of airborne particulate matter. To integrate the Dylos into EpiFi, we modify its case to add a BeagleBone Black (BBB) [3], temperature and humidity sensor, and an LCD screen. For Plantower sensor, we use a BeagleBone cape, which also contains a temperature and humidity sensor.

Along with collecting air quality, temperature, and humidity, we collect network data. This includes whether or not the sensor is connected with a wireless network, wireless signal strength and noise power at the device, wireless data rate, wireless link quality, ping latency/loss to the gateway, and ping latency/loss to a remote server. Having these software sensors is an invaluable tool that provides insights into the challenges we are facing with EpiFi.

B. Multi-Room Deployment

For the first type of deployment, the experimenters wanted to study how air quality differs across space and time inside of a house. Most studies involving indoor air quality deploy one sensor per home. The goal of this experiment, for the domain experts, is to learn in more detail how the air quality is a function of a room, what caused the air pollution, and where that pollution originated. For this deployment, we set up eight Dylos sensors in various rooms in the house and one sensor outside. We deployed this system in three different homes.

1) Lessons Learned: The multi-room deployment was the first set of deployments of EpiFi. From these deployments, we learned valuable lessons on how to improve the architecture. We originally had the gateway set up as an access point that all sensors connected to. We found that sensors deployed in rooms far from the gateway could not connect to the gateway due to weak signal strength. We also saw packet loss much higher than we were expecting. Figure 3 shows an example of such a data collection. The graph shows the time difference between the gateway's received measurements. Data samples should be every minute. You can see that there is a great deal of variation, and there are times when no data is received for more than 20 minutes.

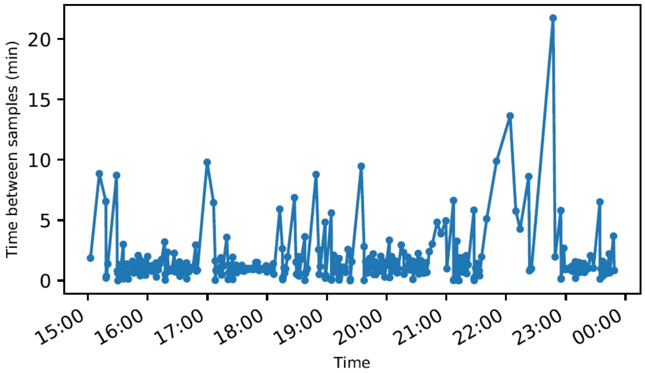


Fig. 3: Time difference between received samples. The time difference should be constant (1 minute).

From this experience, we learned two things. First, by switching to the home's wireless access point, the amount of packet loss is significantly reduced. This is because the gateway was not designed to be an access point and does not have many of the features of commercial access points. This is discussed in more detail in Section IV-A. Second, we learned the importance of distinguishing between packet loss and data loss. Packet loss can occur because of the nature of the wireless medium that we are using. Data loss is when a measurement collected by a sensor is lost completely. To eliminate data loss, we implemented persistence (Section IV-B1) and reliability (Section IV-B2). Before making these changes, we were seeing an average of 25.57% data loss for each sensor. After making these changes, we are seeing 0.0% data loss.

A more subtle problem we discovered in this deployment was the importance of recording a timestamp when the measurement was taken and sending it with the data. Even without any data loss in the network, we found that the measurements were not evenly spaced out as we expected. We originally thought that network latency would not be bad and our time stamps only needed to be accurate within a few seconds so we were sending the data without a timestamp and when the data got to the gateway, it would timestamp the data and upload it to the database. This had the advantage of only dealing with one clock, the gateway's clock, which we could ensure was accurate. However, we found that there is enough latency and jitter in the wireless network that it had an effect on the data. We also found that two measurements that were taken at the same time, such as the Dylos sensor returning temperature and humidity reading, would have different timestamps. This was due to the way Home Assistant processes incoming data. From a research point of view, this made it difficult to compare corresponding measurements, because two measurements that happened physically at the same time, would have slightly different timestamps. We fixed these problems by recording the timestamps when the measurements are being made and sending it with the measurement.

C. Clinical Deployment

The clinical deployment study is designed by pediatric asthma researchers. There is significant evidence that outdoor air pollution levels can be used to predict asthma exacerbation [16]. Further, incorporating outdoor levels into treatment, including decisions about when to talk with a nurse and the appropriate dosage of medication, can reduce the frequency of emergency department visits [14]. However, such studies do not have accurate measures of an individual asthma patient's exposure to pollution. These studies use only outdoor air measurements, and further, they use the measurements from outdoor monitoring stations, typically one sensor per city, which may be located far from the air to which an individual is exposed. The type of deployment study reported here is motivated and designed by pediatric asthma researchers interested in sensor data collected closer to the patient and its ability to improve treatment of asthma.

This was the first study that involved deployers who were not familiar with the EpiFi architecture. The participants of this study are families that suffer from asthma. This study has been going on for eleven months, has been deployed in ten homes, and we have collected over 111 million measurements.

1) Lessons Learned: As part of these deployments, epidemiologists involved with the study deployed our system in homes. This gave us valuable insights into how we can make deployments easier and what was not working with early versions of EpiFi.

After talking with these deployers, we learned the importance of speeding up the process of deployment. It is important to make deployments as quick and efficient as possible so as to not inconvenience a participant. Deployments were taking hours and participants were feeling inconvenienced because of the length of time. A large amount of the time deploying was spent connecting the sensors to WiFi and setting them up. This feedback led to a new problem of quick WiFi bootstrapping which we address in Section IV-A.

Talking with deployers and others involved with the study highlighted the importance of having a set of tools that allows deployers and researcher coordinators to check on the status of the sensors easily. To solve this problem, we built a one page status board that shows the current state of each sensor, as well as the last time data was received. Lastly, we learned the importance of remote debugging of the sensors. Being able to check on the state of the sensor, debug a problem, and update code if necessary became an invaluable tool.

IV. Challenges

We faced many challenges while building and deploying EpiFi in numerous homes. We highlight three of these challenges. We think these challenges extend to all types of deployments and are not just limited to epidemiological research deployments.

A. Secure WiFi Association Bootstrapping with STRAP

In order for a WiFi device to connect to a home's network, it must have the network name (SSID) and password. Getting this information to devices without keyboards and screens is time-consuming, particularly when there are several devices to be connected. We had a few deployments with 15 sensors that needed to be connected to WiFi. The process of connecting each sensor was monotonous. Typically deploying sensors in a home involves more than just connecting up the sensors. It can include things like interviewing participants or doing a home dwelling survey. To minimize the time a deployer needs to be in a home, we look at reducing the time it takes to connect each sensor to WiFi.

One solution is to provide an access point to a deployment with the sensors already preconfigured with the network name and password. With early deployments, we made the gateway an access point. However, our deployment experiences showed many drawbacks. First, the commodity hardware available is not designed to be used as an AP. Some drivers support this option, but using it proved to be unreliable. Second, a USB-connected WiFi dongle is unable to take advantage of many of the features that come standard on today's APs, such as beamforming and MIMO. We also found that homes we deployed in had WiFi networks customized to the needs of the home. To ensure complete coverage of the deployed sensors, we would have to essentially duplicate their network by placing multiple gateways throughput the house. These experiences, shared in more detail in Section III-B, led us to design EpiFi to use the home's WiFi.

Another approach is to configure each sensor with the home's network name and password before connecting them. Since participants were part of a human subjects research study, they were asked to complete an online, paper, or phone survey with information about themselves. Receiving network information at this time was ideal because we could load this information onto the devices before deploying them to the participant's home. However, we found that collecting network information through the survey resulted in the following issues: 1) participants were uncomfortable entering their password into a remote database; 2) participants often do not know their network name and password when away from their home; and 3) there are a high rate of errors when communicating the network name and password via survey. This may be due to default long and difficult passwords on home wireless routers. These passwords may be challenging to relay accurately over the phone or on a paper survey (e.g., mistaking a capital "O" for a zero). An incorrect network name or password is only detected after trying to deploy the sensors. In this case, the sensors have to be returned to the research facility and loaded with the correct network name and password. This experience highlights the importance of providing participants with instant feedback when entering network names and passwords.

The commercially accepted solution to this problem for IoT devices is for a device to create its own temporary wireless network. This allows the person setting up the IoT device to connect to the temporary wireless network via a smartphone, select the network name they want the device to connect to, and enter the password, which then provides the IoT device the information it needs to connect to the WiFi network. This process works well with one or two new devices, but it becomes a nuisance when deploying multiple devices and increasingly untenable as the number of devices increases. Setting up each device one at a time is unacceptably long for our deployers, for whom deploying sensors in a participant's home is only one of the tasks they must accomplish in a short period of time. The more time researchers take in a person's home, the less happy they are to participate, and the more they expect to be compensated. Moreover, the exchanging of network name and password is a general problem for all IoT system deployments that require a deployer to place multiple sensors in a home, not just unique to our study.

To solve this problem, we created a novel approach for securely sending the network name and password to the devices from the gateway using the home WiFi, which we call *Secure TRansfer of Association Protocol* (STRAP) [13]. STRAP allows a device to bootstrap its connection to the home's wireless network with the help of the gateway. This is a challenge because there is no direct way to send secure information to the device before it is connected to any network and, as a result, cannot decrypt WiFi frames. To make this task more difficult, the gateway is connected via Ethernet to the home WiFi router, such that, any data it sends will be encapsulated in an 802.11 frame and encrypted by the wireless router. A WiFi device can be in monitor mode, allowing it to see the frames that are being sent, but not decrypt them.

STRAP overcomes these obstacles using the gateway to encode data into the source and destination address fields of an Ethernet frame, setting the destination address such that the frame is broadcast to all wireless devices. When an Ethernet frame gets translated into an 802.11 frame by a wireless router, the wireless router directly copies the source and destination address of the Ethernet frame into the source and destination address of the 802.11 frame. A wireless router does not encrypt the 802.11 header, which includes the destination and source addresses. If such a frame gets sent out wirelessly, a device in monitor mode can receive the frame and decrypt the data in the source and destination address fields. To ensure that a frame gets broadcast on the wireless routers wireless interface, a special destination address must be used. Such addresses include broadcast (`FF:FF:FF:FF:FF:FF`), IPv4 multicast (`01:00:5E:xx:xx:xx`) [6], and IPv6 multicast (`33:33:xx:xx:xx:xx`) [5], where the `x`'s of the addresses can be replaced with a unique identifier for that multicast group. STRAP uses the IPv6 multicast address because it provides more unused bytes compared to the other addresses. This allows STRAP to encode 10 bytes of data inside each Ethernet frame.

The flow of data through STRAP is shown in Figure 4. A deployer enters the network name and password into the gateway. This information is encrypted using a shared key between the gateway and sensors. A message authentication code is created to ensure the data is not modified. The encrypted data, message authentication code, IV, and a global sequence number are combined together. Erasure coding is added to the data so that if any part of the data is lost, it can still be decoded. This is important for STRAP because we are using broadcast to send the frames, so there are no link layer acknowledgments. Last, the data is packetized and a header is put on each packet. An unassociated sensor scans through the WiFi channels looking for STRAP frames. Once it has received enough STRAP frames, it decodes the data and connects to the home's network. Using STRAP, we are able to greatly speed up the deployment of multiple sensors in a home.

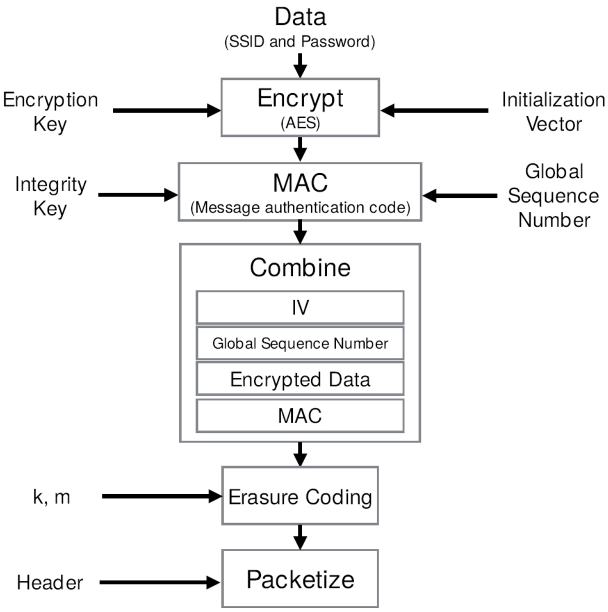


Fig. 4: The flow of data through STRAP.

B. Reliable Data Transfer

Early in our deployment experience, we realized the difficulty of obtaining reliable data transfers. Initially, we transferred data from the sensors to the gateway using TCP without any form of application-level acknowledgments because TCP typically results in reliable data transfers. We also assumed that a sensor would reliably stay connected to WiFi. We found that even when the sensor was connected to WiFi, occasionally TCP could not transfer data because of packet loss and eventually time out. Additionally, sensors would disconnect from WiFi for long periods of time. During our first deployment (Section III-B) we saw an average of 25.57% data loss for each sensor.

This experience highlighted two important aspects of reliable data transfer: 1) data must be saved regularly once it has been collected and 2) data must be confirmed as successfully transferred before deleting it from the sensor. There might be times where the sensor cannot reliably transfer data due to a loss of connection with the wireless router, the link quality being poor or the gateway being down because it was unintentionally unplugged. As said in [9], "homes are hazardous environments" for sensors because random power outages and Internet disconnections are very common. Our experience agrees with this. The sensors that we deployed disconnected from the Internet on average 37 times per month per sensor with the disconnection lasting on average 9.9 minutes. To protect against any data loss, data from sensors needs to be persistently stored. To solve these two problems we build a lightweight persistence storage system to safely store data on the sensor and a low overhead transport protocol on top of CoAP [19] that transfers data reliably. Our reliable transport protocol, which we call Reliable Pull over CoAP (RePoC), has the same efficiency as MQTT [2] but performs much better if a sensor has a backlog of data stored. With these two building blocks, we are able to transfer all data that is collected from a sensor without any loss.

1) Data Persistence: In our own deployments and through the experiences of others [9], we learn power loss to sensors is inevitable. Data loss can have a negative impact on a study and the trust of a system. Under these circumstances, data persistence is important to ensure that no data is lost when a sensor is accidentally powered down. Using a database is a possible solution because it provides the data integrity that we need. However, using a database for an application like this is heavy and cumbersome. Also, a database is not easily implemented in an embedded system environment.

A more lightweight approach would be to use some kind of queue that also persists the data. After investigating this option, we could not find a persistent queue that supports the workflow we were looking for. The required workflow is to "peek" at the data on the queue (copy the data *without* removing it from the persistent storage), transfer the data, confirm that the data has been transferred properly, and delete the data from the queue. All queue implementations that we found only supported pop/push semantics, but did not support peeking (reading data without deleting it from the queue) and deleting (removing data from the queue without reading it). We feel like this workflow is universal enough that we create our own persistent queue [12]. We use our persistent queue in the following manner. The sensor has two main processes: recording data from the sensors into the persistent queue and sending data to the gateway when requested. When a request for data comes in from the gateway, the sensor reads data from the persistent queue, sends it to the gateway, and deletes the data once receipt of data has been acknowledged. By using a persistent queue, we ensure that all data collected will be safe, regardless of application failure or power loss.

2) Reliable Pull over CoAP (RePoC): Even with data persistence, loss can occur when transferring between sensor and gateway. To ensure that data has been properly transferred, application level acknowledgments are needed. We design a lightweight protocol on top of CoAP to pull data from sensors. Without such a protocol, there is no way for the sensor to know if the gateway received *and* handled the data properly. We built RePoC to be optimized for a backlog of data, by adding the ability to request multiple samples at a time. We find in our deployments that backlogged data is common. The sensors that we have deployed have a backlog of data 6.11% of the time, with some sensors having a backlog 40% of the time. RePoC allows the gateway to request multiple samples at once, improving the efficiency of our protocol.

RePoC fulfills three important aspects. First, the protocol ensures that data is reliably transferred. It does this by putting the burden on the gateway to tell the sensor how many data points have been received and can be deleted by the sensor. This allows the gateway to ensure that the data is safe before it tells the sensor to delete the data. Second, the protocol requires no configuration on the sensor. Instead, the configuration goes into the gateway. The gateway is responsible for finding the sensors, determining how much data to request from each sensor, and how often to request data from the sensor. The sensor acts as a CoAP server, waiting for requests from the gateway. With other protocols, like MQTT, the sensor would need to be configured with the information of a broker. Last, RePoC is scalable because the gateway is pulling data from the sensors, rather than having the sensors push data to the gateway. This allows the gateway to act as a coordinator, controlling all transmissions.

Our process for retrieving data from a sensor works in the following manner. After the gateway has discovered a set of sensors, it requests data from them in a round-robin approach. The gateway will transmit a confirmable CoAP GET request to a sensor for its data. The sensor will respond to the confirmable GET request with an ACK that also contains the data the gateway requested. The GET request has two important pieces of information: the number of data points the gateway wants from the sensor and the number of data points the gateway is acknowledging from the previous request.

By having the gateway choose how many data points it wants from a sensor, the gateway is able to balance getting data quickly and in a scalable manner. The sensor sends, at most, the number of data points requested. If it does not have that many data points to give, it gives the maximum it has. If the sensor responds with the same amount of data points as the gateway requested, the gateway can infer there is a good chance the sensor has more data to send and will poll it for more data. If the sensor responds with fewer data points than the gateway requested, the gateway knows the sensor has given all of its data points and can move to the next sensor.

The second portion of the request is the number of data points being acknowledged. The sensor has no way of knowing if the gateway has received *and* processed the data properly. We use CoAP confirmable messages, but this only tells the sensor if the message was received properly and not if it was processed. The number of data points acknowledged will typically be the number of data points received in the previous request, but if an error occurs on the gateway or a CoAP transmission goes unacknowledged, the number gets set to zero. This ensures that no data will be acknowledged and deleted without the gateway processing it first. A trade-off of this approach is that duplicate data can be received from

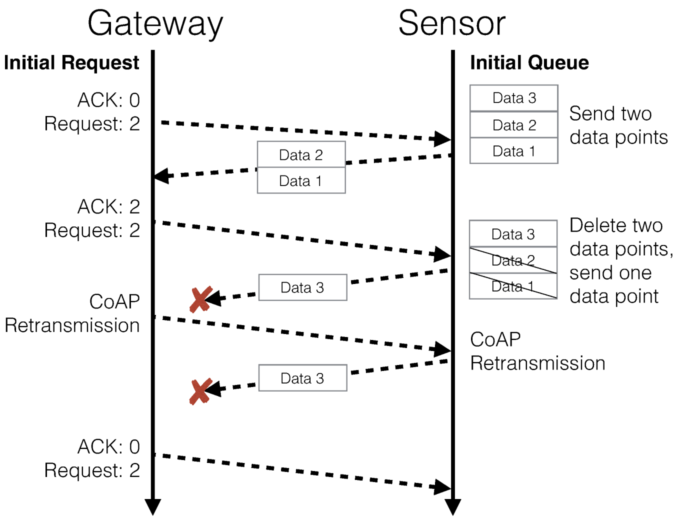


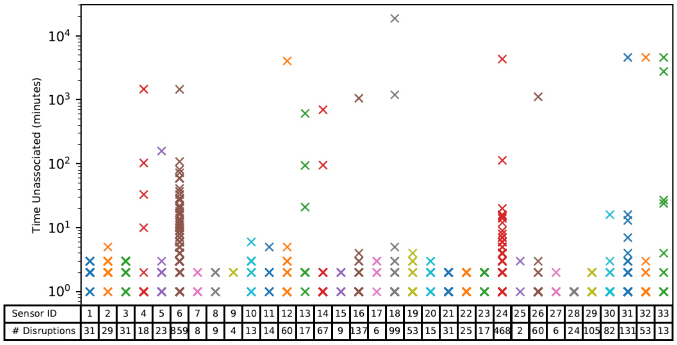
Fig. 5: An example of packets exchanged between gateway and sensor using RePoC.

a sensor. This is fine given duplicate data is detected by the database, InfluxDB, and ignored [11]. When the sensor receives the number of data points acknowledged, it is free to delete those data points.

Figure 5 illustrates an example of RePoC in use. In this example, the gateway starts by requesting two data points and acknowledging zero data points. The sensor has three data points stored, so it sends the first two data points to fulfill the request. The gateway responds with a request for two more data points and acknowledges that it received the previous two data points. The response of the sensor is not received, so the gateway retransmits its request and the sensor retransmits its response. Eventually, CoAP will give up, in which case the gateway will wait a period of time and try again, this time, setting the number of data points acknowledged to zero. If the gateway had kept its acknowledgment at two, data point three would have been deleted without the gateway receiving it. By using RePoC, even if the request or response are lost, data will be reliably transferred and no data will accidentally be deleted.

C. WiFi Disruptions

Occasionally, sensors stop sending data. This can happen for many reasons, some more severe than others. These reasons range from momentary WiFi disruptions to accidental power outages or hardware malfunctions. For example, from our deployments, sensors were either unplugged or power was lost on average 1.99 times per month per sensor. As mentioned before, homes can be a hazardous environment with poor wireless connectivity, increased power interruptions, and occupants (including children and animals) interacting with the devices. A temporary WiFi disruption is the least problematic because the sensor is still collecting data and all data will be uploaded when the sensor comes back online. WiFi disruptions can last from a few minutes to a few days. On the other hand, a sensor being unplugged or a hardware malfunction leads to lost data which would have negative impacts on a study.



Sensor ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
# Disruptions	31	29	31	18	23	859	8	9	4	13	14	60	17	67	9	137	6	99	53	15	31	25	17	468	2	60	6	24	105	82	131	53	13

Fig. 6: Length of WiFi disruptions and the number of disruptions of sensors seen in our deployments over eleven months.

Even though a researcher might find out the cause when the sensor has come back online, they do not know the cause of the disruption at the moment. When the disruption may be causing data loss, the researcher may contact a participant to have them check on the sensor, or a deployer goes out to fix the sensor. For a researcher, when a sensor goes offline, it becomes a balance of possibly losing data and burdening a participant by sending someone to check on the sensor. In our experience deploying EpiFi, there were times where sensors would go offline and we assumed it was a temporary WiFi disruption. It was only after a few days and having a deployer check on the sensor, that we learned the hardware had malfunctioned and data had not been collected. A researcher must decide how much data they are potentially willing to lose before they intervene with a participant and their sensor. To make matters worse, from our experience, it can take days to weeks to coordinate with a participant to find a time to meet at their home to investigate a sensor. Also, each visit is an inconvenience to a participant, making them more likely to drop out of the deployment study. We had patients drop out of the study during both studies. Each loss of a participant was reported on and extensively discussed by the researchers as a significant event. These factors make it that much more important to gain insights into the state of the sensor remotely.

Figure 6 shows the length of time (in minutes) that each sensor was disrupted from WiFi and the number of disruptions that occurred for that sensor. Based on this data, we learn that all sensors deal with this problem, however, some sensors have a lot more disruptions compared to others. These times where a sensor was disconnected from WiFi represent times when the researcher did not know the state of the sensor.

We focus on WiFi disruptions because we think that it is a solvable problem. These disruptions are caused by the wireless signal to noise ratio falling below a threshold such that the lowest data rate WiFi supports, 1 Mbps, cannot be achieved. If a wireless device is unable to communicate at the lowest data rate, then it becomes disconnected from the network. In our context, we find the lack of lower data rates of WiFi to be a hindrance. Since most exposure sensors are producing at relatively low data rates (e.g., a few bits per minute), we would gladly trade a faster data rate for longer range communication. However, with current WiFi, that is not possible.

Two WiFi standards already exist that support lower data rates and longer ranges: HaLow (802.11ah) [1] and White-Fi (802.11af) [8]. Both of these standards are designed for applications that need longer ranges with a decrease in data rate. Both are able to increase their range, in part, because of the lower frequencies they run on. HaLow runs 902 to 928 MHz and White-Fi runs on the TV white space frequencies, 54 to 698 MHz. However, the new standards require a different transceiver – they are incompatible with the standard WiFi used in people's homes. As a result, this is not a viable solution for current deployments and will not be viable until these protocols are built into consumer wireless routers. Until this happens, the current 802.11 standard is updated to support lower data rates, or another approach is taken, this problem will continue to exist.

V. Related Work

Work done by T. Hnat et al. [9] shares valuable insights and lessons learned through their many experiences doing home deployments. They discuss common problems when dealing with sensors in homes, such as power loss and wireless connectivity issues. Such insights point to the importance of EpiFi and the numerous problems that it addresses.

There are commercial systems, like those sold by Qualcomm Life [17], that provide similar benefits as EpiFi. This system has the appeal of being backed by a company. The use of cellular connectivity increases the cost compared to piggybacking onto a home's internet connection, and researchers are typically budget-limited. Further, support for devices is limited to what a company will integrate into their system. We feel that some of the best innovations will be from new sensors that are developed and believe in the importance of open source software that can be improved upon by the community.

Microsoft's Lab of Things [4] provides a testbed for researchers to conduct home field deployment. Though some of its goals overlap with EpiFi, its focus is on allowing researchers to test new home technologies in a shared home testbed environment. Our focus is on enabling epidemiologists to more easily build and deploy sensors for their studies.

VI. Conclusions

In this paper, we described EpiFi, an architecture for epidemiologists to use to build and deploy experiments. The design of EpiFi is targeted to address issues that are unique to epidemiologists. We use extensive long-term deployment experiences and collected data to observe and find real problems and improve our system. The components we build address real and important problems when dealing with deployments in study participants' homes, such as WiFi association bootstrapping, reliable data transfer, and WiFi disruption tolerance. These components work together to create a robust system, allowing epidemiologists to focus on epidemiology, rather than networking.

Acknowledgments

Research reported in this publication was supported by NIBIB of the US NIH under award number 1U54EB021973-01.

References

[1] S. Aust, R. V. Prasad, and I. G. Niemegeers, "IEEE 802.11ah: Advantages in standards and further challenges for sub 1GHz Wi-Fi," *IEEE Intl. Conf. on Communications (ICC)*, 2012.

[2] A. Banks and R. Gupta, "MQTT version 3.1.1," *OASIS Committee Specification Draft 02 / Public Review Draft 02*, Apr. 2014.

[3] BeagleBoard, "Beaglebone black," https://beagleboard.org/black, 2017.

[4] A. B. Brush, E. Filippov, J. Huang, Danny amd Jung, R. Mahajan, F. Martinez, K. Mazhar, A. Phanishayee, A. Samuel, J. Scott, and R. P. Singh, "Lab of things: A platform for conducting studies with connected devices in multiple homes," ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 35–38.

[5] M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks," Internet Requests for Comments, RFC Editor, RFC 2464, December 1998. [Online]. Available: https://tools.ietf.org/html/rfc2464

[6] S. Deering, "Host Extensions for IP Multicasting," Internet Requests for Comments, RFC Editor, RFC 1112, August 1989. [Online]. Available: https://tools.ietf.org/html/rfc1112

[7] Dylos, "DC1100 air quality monitor," http://www.dylosproducts.com, 2017.

[8] A. B. Flores, R. E. Guerra, E. W. Knightly, P. Ecclesine, and S. Pandey, "IEEE 802.11af: A standard for TV white space spectrum sharing," *IEEE Communications Magazine*, vol. 51, pp. 92–100, 2013.

[9] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse, "The hitchhiker's guide to successful residential sensing deployments," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11. New York, NY, USA: ACM, 2011, pp. 232–245. [Online]. Available: http://doi.acm.org/10.1145/2070942.2070966

[10] InfluxData, "Influxdb," https://github.com/influxdata/influxdb, 2013.

[11] ——, "Frequently asked questions," https://docs.influxdata.com/influxdb, 2017.

[12] P. Lundrigan. (2017) A persistent queue for python. [Online]. Available: https://github.com/philipbl/python-persistent-queue

[13] P. Lundrigan, S. K. Kasera, and N. Patwari, "Strap: Secure transfer of association protocol," *27th International Conference on Computer Communications and Networks*, 2018.

[14] F. L. Nkoy, B. L. Stone, B. A. Fassl, D. A. Uchida, K. Koopmeiners, S. Halbern, E. H. Kim, A. Wilcox, J. Ying, T. H. Greene, D. M. Mosen, M. N. Schatz, and C. G. Maloney, "Longitudinal validation of a tool for asthma self-monitoring," *Pediatrics*, vol. 132, no. 6, pp. e1554–e1561, 2013. [Online]. Available: http://goo.gl/69vzAK

[15] Plantower, "Pms 3003-pm2.5-plantower technology," http://www.plantower.com, 2018.

[16] C. A. Pope III, "Epidemiology of fine particulate air pollution and human health: biologic mechanisms and who's at risk?" *Environmental Health Perspectives*, vol. 108, no. Suppl 4, pp. 713–723, 2000.

[17] Qualcomm, "Qualcomm life," http://www.qualcommlife.com, 2017.

[18] P. Schoutsen, "Home Assistant," https://home-assistant.io, 2017.

[19] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)," Internet Requests for Comments, RFC Editor, RFC 7252, Dune 2014.

[20] E. Smith, "Global Broadband and WLAN (Wi-Fi) Networked Households Forecast 2009-2018," https://www.strategyanalytics.com, Oct. 2014.

[21] M. Swan, "Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0," *Journal of Sensor and Actuator Networks*, vol. 1, no. 3, pp. 217–253, 2012.

[22] M. van Tongeren and J. W. Cherrie, "An integrated approach to the exposome," *Environmental Health Perspectives*, vol. 120, no. 3, p. A103, 2012.

[23] C. P. Wild, "The exposome: from concept to utility," *International Journal of Epidemiology*, vol. 41, no. 1, pp. 24–32, 2012.