scientific reports

Check for updates

OPEN

A data-driven approach to increasing the lifetime of IoT sensor nodes

Shikhar Suryavansh[1], Abu Benna[2], Chris Guest[3] & Somali Chaterji[4]✉

Data transmission accounts for significant energy consumption in wireless sensor networks where streaming data is generated by the sensors. This impedes their use in many settings, including livestock monitoring over large pastures (which forms our target application). We present Ambrosia, a lightweight protocol that utilizes a window-based timeseries forecasting mechanism for data reduction. Ambrosia employs a configurable error threshold to ensure that the accuracy of end applications is unaffected by the data transfer reduction. Experimental evaluations using LoRa and BLE on a real livestock monitoring deployment demonstrate 60% reduction in data transmission and a 2 x increase in battery lifetime.

There has been a tremendous growth in the number of IoT devices deployed worldwide. These IoT devices comprise of a network of dedicated physical objects (things) that contain embedded technology to interact with the external environment. According to Cisco, the number of networked devices is expected to reach 27.1 billion in 2021, i.e., about 3.5 devices per human on the planet. These IoT wireless nodes typically consist of a microcontroller, transceiver, memory unit, power source, and one or more sensors for sensing the ambient environment. Examples of such sensors include accelerometers, and sensors for vibration, temperature, and humidity. Wireless sensor networks are widely used in environmental monitoring[1–3], industrial control[4–7], infrastructure security, and other fields[8,9]. Data collected by the sensors is either processed locally or sent to a server for analysis[10]. This is meant to support algorithmic processing of sensor data for varied use cases like anomaly detection and object detection for end user applications in self-driving cars[11], digital agriculture[12], smart factories[7], etc.

Since the sensor nodes are constrained in terms of their computational capacity and energy budget, the processing of the collected data usually takes place at a nearby connected edge server. Edge servers can include devices such as Raspberry Pis, wireless routers, or low to mid-range servers installed close to the sensor nodes to reduce latency in data transmission. Low latency is important because the data may be used to trigger real-time responses, in addition to monitoring purposes, such as activating cooling procedures if the temperature of an industrial plant operation rises above a threshold.

Several wireless network technologies can be utilized for data transfer between the sensor nodes and the edge server. These include Bluetooth Low Energy (BLE), IEEE 802.11 power saving mode, IEEE 802.15.4/e, as well as long-range technologies such as LoRa and SIGFOX. Low energy consumption is a critical requirement for IoT sensor nodes since they are usually battery operated. In order to increase battery life, energy consumption of these sensor nodes have been optimized using techniques such as efficient routing[13], data compression[14], duty cycling[15], mobility[16], and approximate computing[17,18]. Energy harvesting[19,20] to power the sensor nodes is another approach that has been used to prolong the battery life in the field of wireless gas sensors for air quality monitoring.

For any application, the sensor nodes need to perform the following tasks: sense events, locally process the sensed data (if needed), and transmit the data to the server[21,22]. Each of these tasks contribute to the energy consumption at the sensor node and thereby influence the battery lifetime. The total energy consumption therefore consists of the circuit energy consumption (sensing and processing) and the transmission energy consumption. However, for a wide majority of sensor networks, the radio chip is the most power hungry component and hence most of the energy consumption occurs because of data transmissions[23,24]. For such networks, irrespective of the wireless network technology used, data transfer accounts for at least 85% of the total energy consumption[25]. A comparison of the energy consumption of various network technologies obtained from[26] has been provided in Table 1. The percentage is much higher for long-range technologies such as LoRa and SIGFOX, where data transmission can consume as high as 99% of the available energy. Therefore, reduction in data transmission from the sensor nodes to the edge nodes or to the cloud (as the case may be for the processing requirement) can

[1]Cisco Systems, San Jose, USA. [2]Beaconchain, Calgary, Canada. [3]LightBug, Bristol, England. [4]Purdue University, West Lafayette, IN, USA. ✉email: schaterji@purdue.edu

Power consumption comparison of wireless network technologies						
Wireless technology	Hardware platform	P_Tx (mW)	P_Rx (mW)	P_Idle (mW)	P_Sleep (μW)	Percentage power consumption for data transfer
802.11 PSM	G2M5477	699.6	170	66	13.2	92.9
BLE	nRF51822	37.2	42.3	13.2	7.8	85.8
802.15.4	SmartMeshIP	24.11	20.87	4.67	4.32	90.6
LoRa	GreenNet	158.4	44.06	–	4.32	99.9
SIGFOX	GreenNet	147	39	–	4.32	99.9

Table 1. Power consumption of various network technologies (assuming all the states are equally likely; for high data rate, the likelihood of sensor node being in Tx or Rx would be greater than the other states resulting in an even higher percentage power consumption for data transfer).

significantly reduce the energy consumption and increase battery lifetimes. It must be noted that there exist such wireless sensor networks in which the sensor itself is the most energy consuming and not the radio chip. For example, if we consider wireless gas sensor networks, the main consumer of energy would be the gas sensor[27,28]. Reducing data transmission for such sensor networks would still result in a lower energy consumption but the impact would not be as significant.

In this paper, we propose Ambrosia, an efficient technique for reducing the data transmission from the sensor nodes to the servers. In ancient Greek mythology, Ambrosia is the food and drink of the Greek gods, conferring longevity to whoever consumes it. In our work, we aim at increasing the lifetime of the IoT sensor nodes by reducing the bulk of data transfer. *The key intuition behind Ambrosia is that the data samples whose values can be accurately predicted at the edge server do not need to be transmitted from the sensor nodes to the backend servers.* Ambrosia utilizes a simple window-based prediction scheme to decide which data samples need to be transmitted from the sensor node to the server. Since the sensor nodes have limited resources, a computationally intensive technique for data reduction, no matter how accurate, would be sub-optimal. Executing a compute-intensive technique on the sensor node would overshadow the advantage accrued from data reduction. Further, a computationally intensive algorithm may not be able to keep pace with the rate at which streaming data is being generated resulting in missing timing deadlines. Thus, we design Ambrosia to be sufficiently lightweight to be able to easily execute on the constrained sensor nodes, as well as designed toward reduced data transmission to the edge servers while meeting the user-defined accuracy bounds.

The impact of errors in the transmitted data on the accuracy is different for different types of applications. Hence, the optimal value of the error threshold would depend upon the specific application under consideration. For example, a sensitive application such as operating a medical equipment based on the sensor readings would be susceptible to even a small error in the data. On the other hand, applications such as anomaly detection can tolerate a higher magnitude of error in the transmitted data. We utilized data collected from different production-grade sensor nodes in a livestock monitoring deployment on a large farm. We measured the effectiveness of Ambrosia in different application settings with different tolerances for errors. In particular, for the anomaly detection application, Ambrosia is hugely effective in reducing the amount of data transmitted from the sensor nodes to the edge servers.

This problem statement of reducing transmission in sensor networks has seen significant work (as pointed out with categorization above[13–16,29]). However, our approach is novel in that it creates a very lightweight approach, one that can run even on the most constrained of sensor nodes, like ear tags on livestock. Further, no prior work has looked at the interplay between short-range and long-range wireless technologies (BLE and LoRa respectively) and has not shown the effect of the error threshold on anomaly detection—we use a recent approach from the ML literature for *application-agnostic* anomaly detection[30].

The main contributions of this paper are:

1. We propose Ambrosia, a lightweight protocol that can reduce the number of data samples that need to be transmitted from the IoT sensor node to the edge server in a wireless sensor network, thereby reducing energy consumption. We show that Ambrosia can reduce the data transmission by at least 60% thereby increasing lifetime by at least 1.7 × for low traffic intensities and 2 × for high traffic.
2. We take into consideration the variation in the error tolerance of different applications and design our technique in a way that ensures that the accuracy of the applications is not compromised because of reduction in the data transmission. For an error-sensitive application like displacement computation, we obtain ≥ 35% reduction in data transfer whereas for an error-tolerant application like anomaly detection, the reduction in data transfer is more significant [≥ 68%].
3. We perform evaluation with multiple sensors (such as MEMS vibration sensors, temperature and humidity sensors) and applications, including a real-world deployment in a production livestock farm, to demonstrate that Ambrosia is applicable to a wide range of sensors and applications.

The rest of the paper is organized as follows: Section "Background and related work" presents the background on wireless sensor networks and their energy consumption. Section "System model" provides the details of our

system model and the proposed protocol. Section "Experimental setup" presents the experimental setup and section "Evaluation" the evaluation results. Finally, section "Conclusion" concludes the paper.

Background and related work

Power consumption of different wireless technologies. Different wireless communication technologies can be used for data transmission in an IoT sensor node depending upon the required data rate, data size, and range. We present a comparison of the power consumption of the IoT sensor node using various wireless technologies in Table 1 obtained from[26].

From Table 1, it is evident that majority of the power is consumed for data transmission and reception for all the technologies. Moreover, for long-range technologies such as LoRa and SIGFOX, the amount of power consumed for data transmission is higher than that for low-range technologies such as BLE ((99.9% versus 85.8%). Thus, Ambrosia can be more beneficial in energy savings when deployed for long-range technologies.

Time-series forecasting. State space models (SSMs) provide a principled framework for modeling and learning time series patterns, with prominent examples being ARIMA models and exponential smoothing. SSMs predict the future values of data streams indexed by time, based on previously observed values. In a forecasting method without online tuning (offline mode), the predicted values of previous samples are used for future predictions. In contrast, with online tuning, future samples are always predicted using the true values of the previous samples. Online means the model is re-calibrated based on the actual values observed *at runtime*. In the online mode, it is assumed that the true values of the samples would be available after prediction. For our data-reduction protocol, we propose a window-based forecasting technique ("Window-based forecasting") that utilizes a combination of the online and offline modalities. A popular and widely used technique for time-series forecasting is the **AutoRegressive Integrated Moving Average (ARIMA)** model. Time-series forecasting plays an important role in our data-reduction protocol. We provide a comparison of our window-based forecasting protocol vis-à-vis ARIMA in section "Comparison of window-based and ARIMA forecasting". There are other neural network-based models, e.g.[31,32] that can extract higher-order features and complex patterns within and across time series, but these models are too compute-intensive for our purposes.

Anomaly detection using Robust Random Cut Forest Algorithm (RRCF). RRCF[30] is a scheme that utilizes an ensemble, robust random-cut data structure, for detecting anomalies from IoT sensor data streams. RRCF does not have a preconceived notion of anomaly. It defines anomalies from the viewpoint of model complexity and determines a point as anomalous if the complexity of the model increases substantially with the inclusion of that point in the data stream. The anomaly score of a data point is obtained using its Collusive Displacement (CoDISP) with the outliers resulting in large CoDISP values. A point is labeled an anomaly if its anomaly score exceeds a threshold.

Software stack optimization for edge or cloud analytics. The software stack, such as of the database itself, or of cloud-hosted database instances can be reconfigured in the face of rapidly changing IoT workloads, as done for on-premise NoSQL DBs[33] or for cloud-hosted or serverless infrastructure in the face of changing workloads[34,35]. This reconfiguration can result in performance benefits whether in terms of the more conventional throughput-based metrics (e.g., for genomics workloads) or for latency-based metrics (e.g., p95 or p99 latency). These systems are often designed to help achieve cost and performance efficiency for cloud-hosted databases, rightsizing resources to benefit both the cloud vendors who do not have to aggressively over-provision their cloud-hosted servers for fail-safe operations and to the clients because the data center savings can be passed on them. Such optimized software stacks can improve the end-to-end performance of IoT pipelines.

System model

Network architecture. The network architecture consists of multiple IoT sensor nodes connected wirelessly to an edge server using BLE or LoRa network as shown in Fig. 1. The edge server is in turn connected to the cloud. Data collected by the IoT sensor nodes is sent to the edge server for analysis. From the edge server, the data can be sent to the cloud for storage and further analysis as needed. Henceforth, when we refer to a *"server"*, we mean an *edge server*, rather than a cloud server (unless explicitly mentioned otherwise).

Ambrosia 's rationale and features. The main goal of our protocol Ambrosia is to reduce the amount of data that is sent from the sensor node to the edge server thereby conserving power and increasing battery life. *The animating intuition is that the sensor node does not need to send the data samples whose values can be predicted accurately at the server*—accurately implies within the bounds of a specified error threshold. If the sensor node can track the sample values that will be predicted successfully at the edge server even if the data sample is not sent, then the sensor node can make an informed decision as to whether or not to send those data samples to the server. This decision would be based on the error between the true sample and the predicted sample value.

Figure 2 shows the system model for our protocol Ambrosia, showing the steps performed at the sensor node and the edge server. The first w (window size) true samples collected by the sensor node are sent to the server. For every sample after that, the sensor node predicts its value and compares the predicted value with the collected true value. The true sample is sent to the server only if the absolute difference between the true and the predicted samples is greater than a user-specified (equivalently, application-specific) error threshold δ. A simple window-based forecasting scheme (described in section "Window-based forecasting") is used for prediction. The

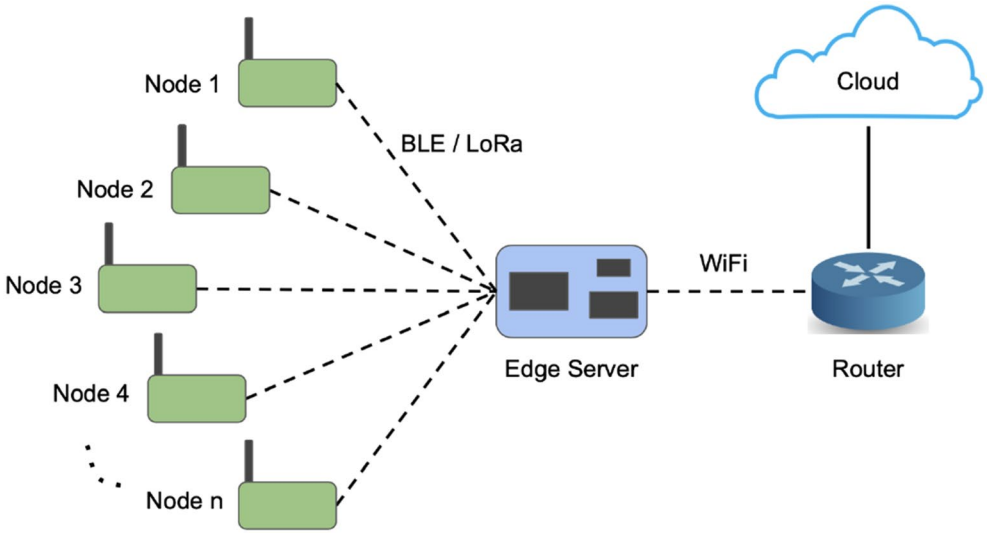


Figure 1. Network architecture.

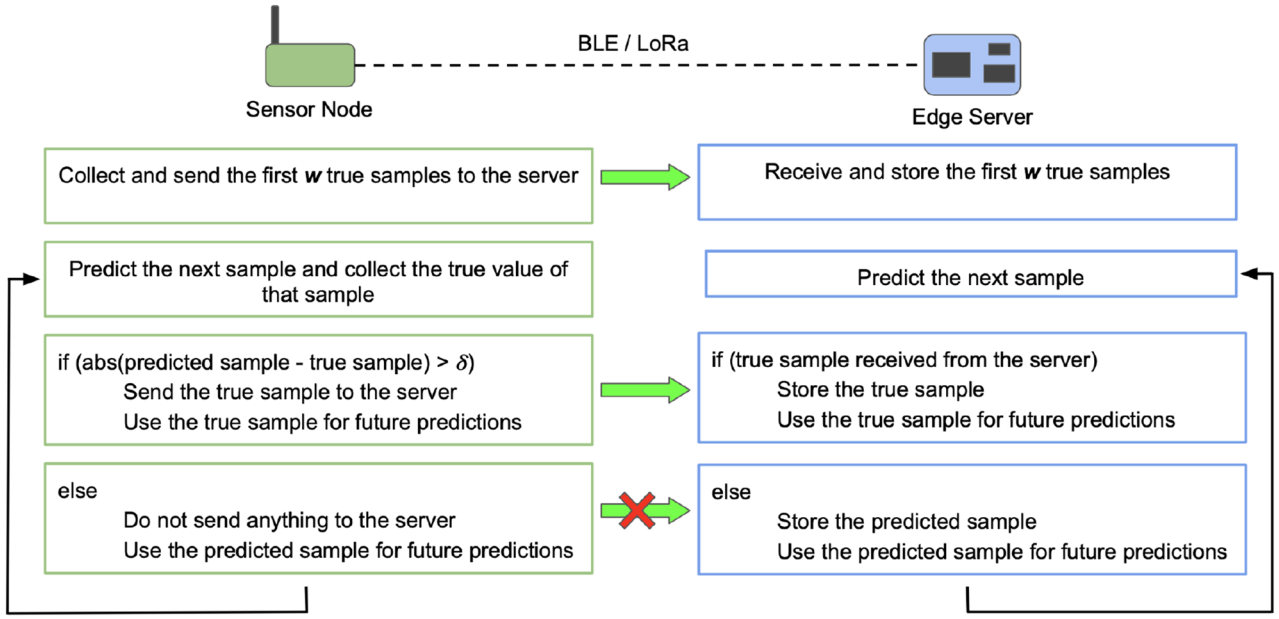


Figure 2. Proposed protocol—Ambrosia: w is the window size and δ is the error threshold.

same prediction scheme is used at the server. This assures that if the true sample is not sent, the value predicted by the server would be within the value generated by the sensor node ± the error threshold.

There is another important construct of the protocol needed to assure that there is no mismatch between predicted values at the sensor node and those at the server. When the sensor node decides not to send the true value of a particular sample to the server, it uses the predicted value of that sample for future predictions and *not* the true value although it has access to the true value. This is to replicate the settings at the server, namely, the server does not have the true value of that sample and can only use the predicted value for future predictions.

Hence, the proposed protocol—Ambrosia—has two important features:

1. True samples are sent from the sensor to the server only when required, i.e., when the difference between the true and predicted sample values crosses the error threshold.
2. The samples for which the true value is not sent from the sensor to the server, the predicted value at the server is the same as the value that was predicted at the sensor (using which it was decided whether to send that sample or not). Hence, when the true sample is not sent, the predicted sample at the server never crosses the error threshold.

Window-based forecasting. The forecasting technique should be simple enough to run on the energy- and memory-constrained sensor nodes without incurring high costs. It may not be feasible to run a computationally intensive algorithm on the sensor nodes for a slightly improved prediction. This rules out using algorithms like Recurrent Neural Networks (RNN), which have been used for time-series prediction but have also been found to be rather resource intensive[32]. Further, for certain applications, an approximate prediction may be sufficient. We propose a window-based forecasting technique for predicting the next sample using the past samples. The next sample is predicted by adding the average of the difference between w adjacent previous samples to the current sample. The parameter w is the *window size*. In section "Comparison of window-based and ARIMA forecasting", we compare the performance of this scheme with ARIMA, a traditional approach for time-series forecasting and one which is heavier weight than our calculation. Our window-based forecasting method simplifies as follows:

$$t[n+1] = t[n] + \frac{1}{w}\sum_{k=1}^{w} t[n-(k-1)] - t[n-k] \tag{1}$$

$$t[n+1] = t[n] + \frac{1}{w}(t[n] - t[n-w]) = t[n] + d[n] \tag{2}$$

The next sample is predicted using the current sample and wth past sample. A lower value of w implies that a more recent sample is used for the prediction of the next sample. Also, the normalization by w guarantees that if a recent past sample is used, its contribution to the prediction is higher. The optimal value of w would depend upon the IoT data collected and the application.

Experimental setup

The hardware used in this experiment consisted of 2 parts: Tag nodes attached to the animals, and gateways connected to a PC. The tag nodes were standard Lightbug LoRa GPS trackers modified to run custom firmware and encased in 3D printed plastics with ear tag attachment. The electronic component of these devices consisted of a LoRa modem (Semtech SX1726 chipset), pig tail helical antenna, atmel 8-bit microcontroller and MPU6050 6 axis accelerometer.

Once configured, the devices could be set to an "active" state. In this mode, they streamed sampled inertial and GPS data to the gateway over LoRa. Quaternions were computed on the device using instantaneous unfiltered data and only sent at 1 Hz to reduce data transmission size. Data was buffered locally and then transmitted in batches to the gateway over LoRa at a data rate of approximately 1 kB/s, in chunks of 60 bytes (due to module/band licensing restrictions).

Once the animals were tagged with devices, they were released into a small enclosure to ensure they could be observed and stayed within transmission range. Data was then recorded for activities like walking, eating, or running. In the following section, we present our evaluation results on the acceleration dataset. Similar results were also obtained for other datasets such as temperature, humidity, and vibration [not shown due to space constraints].

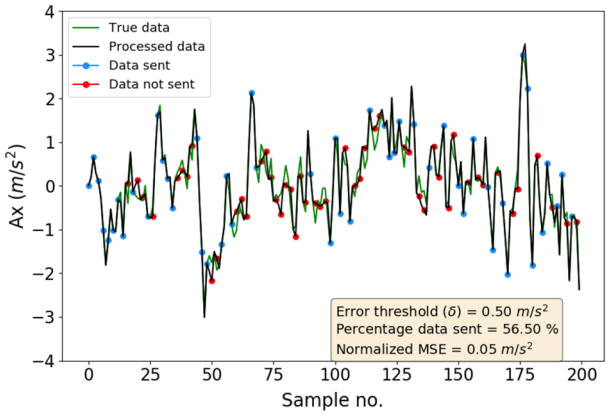
Evaluation

Reduction in data transfer. We now present the effectiveness of our approach in reducing the amount of data that needs to be sent from the sensor node to the server. Figure 3 shows the transmission reduction for first 200 samples of the acceleration data collected by the sensor node. We evaluated the impact of changing δ on the percentage of true samples that need to be sent to the server. In Fig. 3a,b, 'Processed data' are the samples stored at the server. It includes the true samples (marked in blue) received from the sensor node and the samples predicted (marked in red) by the server when the true sample is not received. For clarity, marking in the figures is done for every other sample.

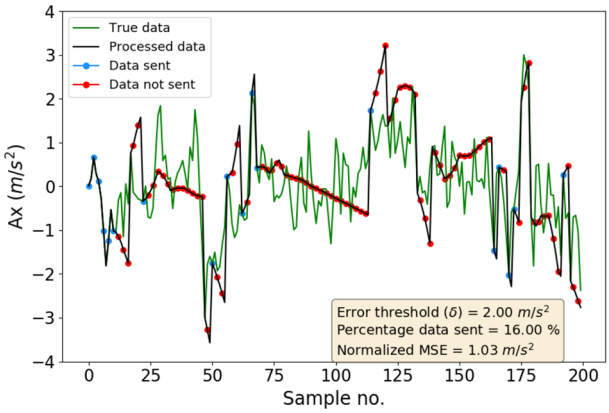
From Fig. 3a, we can see that even a small $\delta = 0.5$ can reduce the percentage of samples sent to 56.50%, thereby providing a 43.50% reduction in transmission energy consumption. Notably, the processed data is remarkably close to the true data with a normalized Mean Squared Error (MSE) between them = 0.05. For δ as high as 2.0, only 16% of the data needs to be sent. However, the normalized MSE in this case is very high (= 1.03) and a substantial distinction can be spotted between the true and processed data (in Fig. 3b). This high an error may not be suitable for most applications and hence it is important to choose an optimal value of the error threshold.

The optimal value of δ would depend upon a lot of factors such as how sensitive is the desired application to the errors in data samples, how much reduction in the power consumption is required (based on an energy budget) or what is the maximum normalized MSE that can be tolerated. Applications such as anomaly detection are more immune to the errors in data samples and can endure a high δ value. On the other hand, applications such as sensor tracking of health devices require every individual data sample to be very accurate. Such applications would require δ to be low.

Impact on application: anomaly detection. We have evaluated the impact of reducing data transmission on the accuracy of RRCF anomaly detection application. The aim is to determine if a reduction in the percentage of samples sent negatively impacts the application's performance. Figure 4 shows a comparison between



(a) ***Data sent for error threshold = 0.50***



(b) ***Data sent for error threshold = 2.00***

Figure 3. Data sent for various error thresholds. Expectedly, as the threshold becomes higher, fewer data points are sent and the deviation between the true data and the processed data also increases.

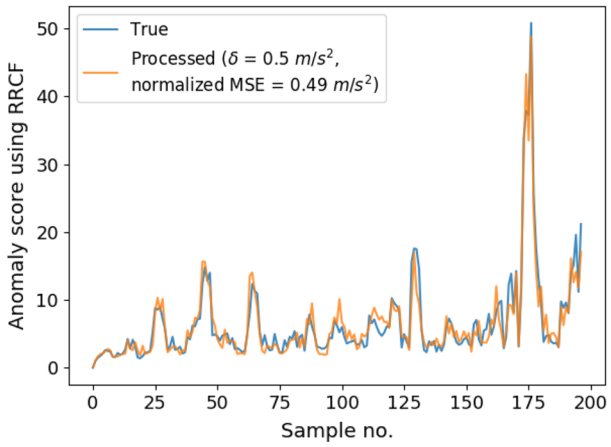
the anomaly scores of the true data (when all the data samples are sent) and the processed data for different error thresholds ($\delta = 0.5$ for Fig. 4a; $\delta = 2.0$ for Fig. 4b).

We are interested in finding out whether the peaks are preserved or not as they are used to detect anomalies. In short, there should not be any false positives (introduction of a false peak) or missed detections (failure to detect a peak) because of the limited data samples sent. We observed that for low values of error threshold, the anomaly score curve for the processed data is comparable to the curve for the true data. For instance, all the anomaly peaks are preserved for $\delta = 0.5$, as shown in Fig. 4a. We observed similar behaviour as δ increases up to 1.2, beyond which the correlation between the two curves reduces. Figure 4b shows the introduction of false positives and missed detection when $\delta = 2.0$.

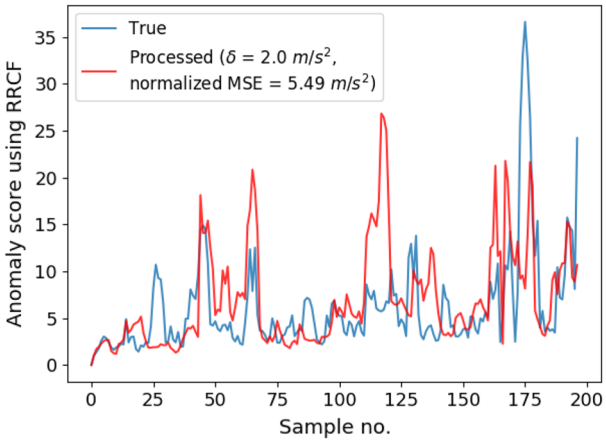
It is desirable to have the value of δ as high as possible (since it reduces the percentage of samples sent), until it starts (negatively) impacting the application's performance. We saw that for anomaly detection, this value is $\approx$ 1.2, which requires only 32.00% of the data to be sent from the sensor node to the server. It must be noted that if the application's performance is more sensitive to the accuracy of individual data samples, then the desirable δ for that application would be lower. One such application is displacement computation.

Comparison of window-based and ARIMA forecasting. As mentioned in section "Window-based forecasting", we use a simple window-based forecasting method for prediction at the sensor nodes and the server. In this section, we compare the performance of our window-based method with the well known ARIMA forecasting technique. In Table 2, the two forecasting methods are compared based on the percentage of data sent and the normalized MSE for different values of δ. It is evident from the table that the performance of both the methods is similar in terms of the amount of data sent and normalized MSE. This means that both the methods result in a similar increase in the lifetime of the sensor nodes. However, window-based forecasting is computationally much simpler vis-à-vis ARIMA, making it more feasible to be implemented on the resource-constrained sensor nodes.

In Fig. 5, we compare the execution time of the two methods for different number of data samples. From the figure, we can observe that the execution time for ARIMA is significantly higher than that of the window-based method. Also, the slope of the ARIMA plot is $\approx 100\times$ higher than that of Ambrosia 's window-based forecasting method. The longer execution time of ARIMA indicates that it cannot sustain a high data rate of incoming samples from the sensor node. Hence, it is preferable to use window-based forecasting method as it provides matched performance with much lower execution time than ARIMA.



(a) ***Anomaly Detection for error threshold = 0.5*** m/s^2



(b) ***Anomaly Detection for error threshold = 2.0*** m/s^2

Figure 4. Anomaly detection for various error thresholds. For threshold up to 1.2, the accuracy of the anomaly detection application is not affected.

	Data sent %		Normalized MSE	
Error threshold (δ)	ARIMA forecasting	Window forecasting	ARIMA forecasting	Window forecasting
0	100.00	100.00	0	0
0.40	61.00	66.00	0.028	0.019
0.80	41.00	41.00	0.153	0.161
1.20	26.00	32.00	0.328	0.334

Table 2. Data reduction and normalized error comparison between window forecasting and ARIMA (p = 3, d = 1, q = 0) forecasting; number of samples = 200.

Reason for using ARIMA as baseline. Our goal is to use lightweight technique so that it can run on an embedded microcontroller. Thus, we do not want to use more complex models like neural network models. We choose ARIMA as this is a superset of many time series models and specific time series models can be derived from it by setting appropriate values for its three parameters. For the baseline evaluation, we optimize the values of the parameters of ARIMA.

Further, ARIMA despite being an old technique, continues to be successfully applied to time series prediction and is found to be especially suitable for lightweight computation. Publications highlighting this aspect of ARIMA continue to appear till now with regularity[36–38].

Increase in sensor node battery lifetime. In this section, we utilize the technique illustrated in[26] to correlate the reduction in the amount of data transmitted by the sensor node to the increase in the battery lifetime of the node. We performed the evaluations for LoRa and BLE wireless network technologies. For each

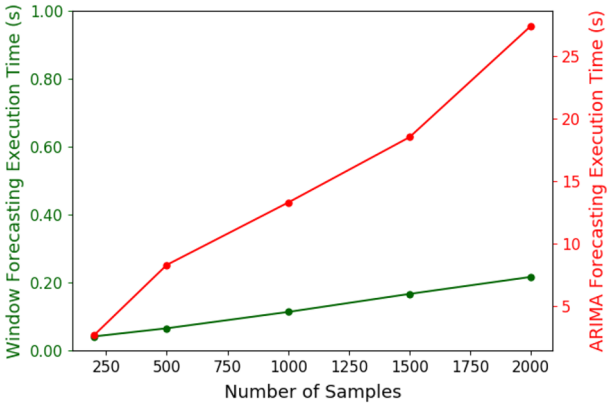


Figure 5. Comparison between the execution time of Ambrosia 's window-based and ARIMA forecasting. The slope of the ARIMA plot is $\approx 100\times$ higher than that of ours.

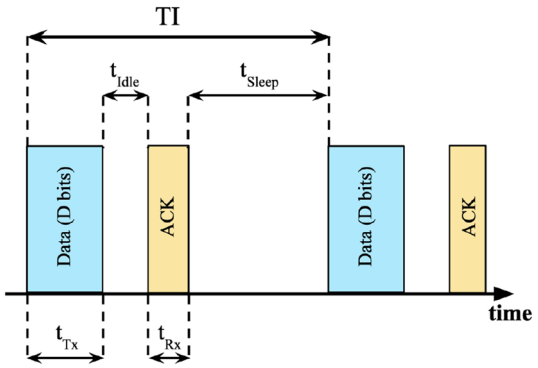
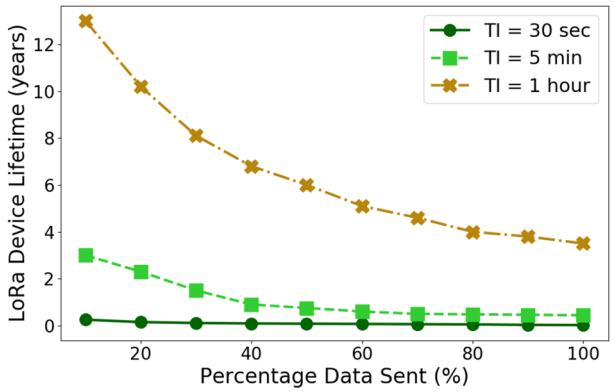


Figure 6. States of the sensor node during a Transmission Interval (TI).

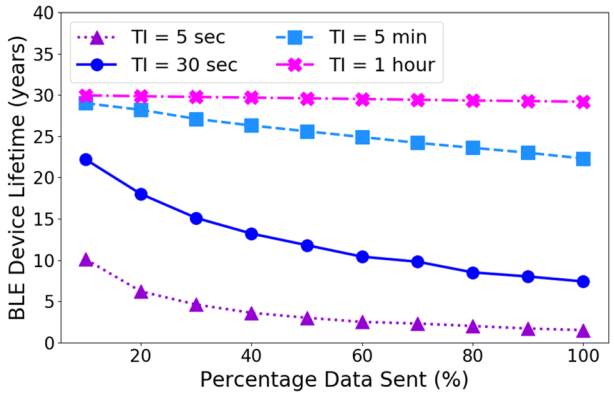
wireless technology, we consider varying traffic intensities controlled by the transmission interval (TI) which is the interval between two instances of data transmission. A high value of TI indicates that data is transmitted less frequently thereby corresponding to a low traffic intensity. TI is divided into four states. The first state is the transmission state (t_{Tx}) during which data is transmitted from the sensor node to the server. Let D be the data size in bits (b) sent every TI and R be the data transmission rate in bits/s (b/s). Then, $t_{Tx} = (D/R)$. After that, the node waits to receive the *ACK*. This state between the data transmission and *ACK* reception is the idle state (t_{Idle}). Next is the reception state (t_{Rx}) during which the *ACK* is received. Finally, the node goes to the sleep state (t_{Sleep}) which continues until the start of the next transmission interval. Figure 6 shows the different states of the sensor node during a transmission interval.

To compute the battery lifetime of the node, we start with an initial energy E_0 corresponding to two AAA batteries (1.5 V, 1250 mAh each). Therefore, $E_0 = 2 \times (1.5\text{ V}) \times (1.25\text{ Ah}) \times (3600\text{ s}) = 13{,}500\text{ J} = 13.5\text{ kJ}$. The energy consumed during a TI is computed using the energy consumption model in Eq. (3) derived from[26]. Lifetime (L) of the node is computed using the total number of TIs possible until the initial energy (E_0) is exhausted as shown in Algorithm 1.

$$E(TI) = \sum_{S} P_S * t_S, S \in \{Tx, Rx, Idle, Sleep\};\ TI = \sum_{S} t_S \tag{3}$$



(a) *Increase in lifetime for LoRa devices*



(b) *Increase in lifetime for BLE devices*

Figure 7. Increase in IoT sensor node lifetime due to reduction in data transmission; TI is the transmission interval, thus we have low traffic intensity (TI = 1 h) to high traffic intensity (5 s in BLE, 30 s in LoRa). The gain of Ambrosia is significantly higher for higher traffic intensity and is available for both wireless technologies LoRa and BLE.

Algorithm 1: Battery lifetime computation for sensor node

function $EnergyConsumed(TI)$
$E_{total} = \sum_{S} P_S * t_S, \quad S \in \{Tx, Rx, Idle, Sleep\}; \quad TI = \sum_{S} t_S$
return E_{total}

function $ComputeLifetime()$
$E = E_0$
$L = 0$
while $E > E_0$ **do**
  $E = E - EnergyConsumed(TI)$
  $L = L + TI$
end
return L

We computed the lifetime of the sensor node for LoRa and BLE wireless technologies for varying data sizes (D) which depends upon the percentage of data transmitted. We aim to demonstrate the effectiveness of transmitting a lower percentage of data on the increase in the lifetime of the node. Figure 7 shows the lifetime of the sensor node battery for different values of TI as the percentage of data sent varies. For our experiments, 100% data transmission corresponds to 8 kb of data. The data transmission rate (R) considered for LoRa and BLE is 11 kb/s and 2 Mb/s respectively.

For LoRa (Fig. 7a), if data sent is reduced to 50%, the lifetime increases by 71.43% (from 3.5 to 6 years) for low traffic intensity (TI = 1 h) and by 300% (from 0.02 to 0.08 years) for high traffic intensity (TI = 30 s). LoRa is an expensive technology as long range transmissions consume more energy. If the traffic intensity is high,

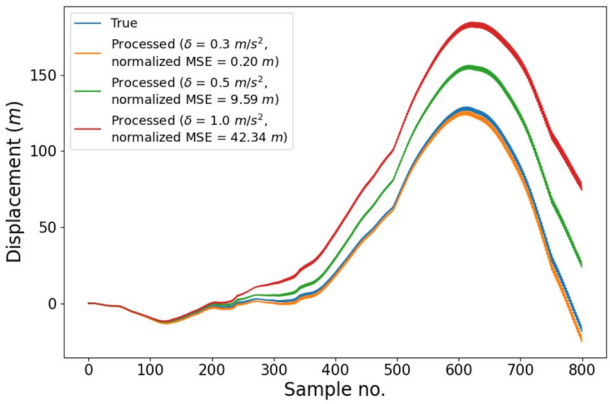


Figure 8. Displacement computation for various error thresholds. For threshold up to 0.3 m/s^2, the accuracy of displacement computation is not affected.

transmitting 100% of the data would drain the battery very quickly. Therefore, we obtain a higher percentage increase in the lifetime by reducing the data transmission for high traffic intensity. For BLE (Fig. 7b), if data sent is reduced to 50%, the lifetime increases by 1.44% (from 29.18 to 29.60 years) for low traffic intensity (TI = 1 h) and by 100% (from 1.5 to 3 years) for high traffic intensity (TI = 5 s). BLE does not consume as much power as LoRa, especially if the traffic intensity is low. For low traffic, even with 100% data transmission, the device has a very high battery lifetime of 29.18 years. Hence, for low traffic, reducing the amount of data sent does not result in a significant percentage increase in the lifetime. However, in many scenarios, BLE is used for streaming data and with a reasonable rate of data, the total data volume can be quite larger and in those cases Ambrosia has significant benefit.

Second application: displacement computation for livestock. We now present the impact of reduction in data transmission on a different application, namely, displacement computation, to track the movement of the animals with ear tags attached. In this application, we used the acceleration data to compute the displacement by double integration. Figure 8 shows a comparison between the displacement computation for the true data (when all the samples are sent) and for the processed data with various error thresholds (δ = 0.3, 0.5, and 1.0 m/s^2). Unlike the anomaly detection application, this application is more sensitive to the accuracy of individual samples. Therefore, we expect the desirable value of δ to be lower for this application. The intuition behind this is that the accuracy of this application relies highly upon the numerical values of the samples being accurate compared to the anomaly detection application. In the anomaly detection application, the accuracy depends more on the aberration patterns in the sample values and not much on the numerical values of the samples.

We measured the performance of displacement computation for different δ values by comparing their displacement curves with the true displacement curve. We quantified the error in terms of normalized MSE. From Fig. 8, we can see that the displacement curve for δ = 0.3 m/s^2 is comparable to the true displacement curve and the normalized MSE is very low. For higher values of δ, there is a visible disparity between the curves and the normalized MSE is considerably high. For instance, for δ = 1.0 m/s^2, the normalized MSE is 42.34 m. Therefore, the desired value of δ for this application is around 0.3 m/s^2, which requires 64.62% of the data to be sent from the sensor node to the server, i.e., 35.38% lesser data transmission than baseline. Recall that for the anomaly detection application, we could obtain a higher reduction (68%) in data transmission. This goes on to show that the optimal value of δ (and hence, the reduction in data transmission obtained) depends highly upon the application under consideration and how sensitive the application quality is to the accuracy of the individual samples.

Conclusion

Here we presented Ambrosia, a lightweight protocol for reducing the amount of data transmissions from IoT sensors to the server resulting in an increase in the battery lifetime of the sensor nodes. We introduced a window-based time series forecasting mechanism that forms a key element of our data reduction protocol and can be easily implemented on even the most resource constrained sensor nodes, such as ear tags put on livestock. Compared to the state-of-the-art ARIMA forecasting, Ambrosia is considerably faster (99% lower execution time) while providing similar data reduction. We evaluated Ambrosia on different wireless network technologies such as LoRa and BLE and obtained more than 60% reduction in data transmission (almost 2 × increase in sensor node battery lifetime). We identified the correlation between the data reduction and error tolerance of different applications and provide a configurable error threshold to ensure that the accuracy of the end applications is not impacted by the reduction in data transmissions.

Received: 10 June 2021; Accepted: 21 October 2021
Published online: 17 November 2021

References

1. Deljoo, A. & keshtgary, M. An efficient wireless sensor network for precision agriculture. *Can. J. Multimed. Wirel. Netw.* **3** (2012).
2. Ullo, S. *et al.* Application of wireless sensor networks to environmental monitoring for sustainable mobility. In *2018 IEEE International Conference on Environmental Engineering (EE)*, 1–7. https://doi.org/10.1109/EE1.2018.8385263 (2018).
3. Jiang, X. *et al.* Hybrid low-power wide-area mesh network for IoT applications. *IEEE Internet Things J.* **8**, 901–915 (2020).
4. Aponte-Luis, J. *et al.* An efficient wireless sensor network for industrial monitoring and control. *Sensors* **18**. https://doi.org/10.3390/s18010182 (2018).
5. GombÃl', B. *et al.* A saw wireless sensor network platform for industrial predictive maintenance. *J. Intell. Manuf.* **30**. https://doi.org/10.1007/s10845-017-1344-0 (2019).
6. Li, H. & Savkin, A. V. Wireless sensor network based navigation of micro flying robots in the industrial internet of things. *IEEE Trans. Ind. Inform.* **14**, 3524–3533. https://doi.org/10.1109/TII.2018.2825225 (2018).
7. Thomas, T. E., Koo, J., Chaterji, S. & Bagchi, S. Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations. In *2018 10th International Conference on Communication Systems and Networks (IEEE-COMSNETS)*, 129–136 (2018).
8. Lloret, J., Garcia, M., Bri, D. & Sendra, S. A wireless sensor network deployment for rural and forest fire detection and verification. *Sensors (Basel, Switzerland)* **9**, 8722–8747 (2009).
9. Upton, D. W. *et al.* Wireless sensor network for radiometric detection and assessment of partial discharge in high-voltage equipment. *Radio Sci.* **53**, 357–364. https://doi.org/10.1002/2017RS006507 (2018).
10. Shankar, K., Wang, P., Xu, R., Mahgoub, A. & Chaterji, S. Janus: Benchmarking commercial and open-source cloud and edge platforms for object and anomaly detection workloads. In *2020 IEEE 13th International Conference on Cloud Computing (IEEE-CLOUD)*, 590–599 (2020).
11. Chaterji, S. *et al.* Resilient cyberphysical systems and their application drivers: A technology roadmap. *arXiv preprint* arXiv:2001.00090 1–36 (2019).
12. Chaterji, S. *et al.* Lattice: A vision for machine learning, data engineering, and policy considerations for digital agriculture at scale. *IEEE Open Journal of the Computer Society (IEEE-OJCS)*, Vol. 2, 227–240 (2021).
13. Nurchis, M., Bruno, R., Conti, M. & Lenzini, L. A self-adaptive routing paradigm for wireless mesh networks based on reinforcement learning. In *MASCOTS*, 197–204 (ACM, 2011).
14. Dias, G. M., Bellalta, B. & Oechsner, S. A survey about prediction-based data reduction in wireless sensor networks. *ACM Comput. Surv. (CSUR)* **49**, 1–35 (2016).
15. Ganesan, D. *et al.* Networking issues in wireless sensor networks. *JPDC*, 799 – 814. https://doi.org/10.1016/j.jpdc.2004.03.016 (2004).
16. Tirta, Y., Li, Z., Lu, Y.-H. & Bagchi, S. Efficient collection of sensor data in remote fields using mobile collectors. in *International Conference on Computer Communications and Networks*, 515–519 (IEEE, 2004).
17. Xu, R. *et al.* ApproxDet: content and contention-aware approximate object detection for mobiles. in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (ACM-SenSys)*, 449–462 (2020).
18. Lee, J. *et al.* Benchmarking video object detection systems on embedded devices under resource contention. in *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning (ACM-EMDL)*, 19–24 (2021). https://doi.org/10.1145/3469116.3470010
19. Baranov, A., Akbari, S., Spirjakin, D., Bragar, A. & Karelin, A. Feasibility of rf energy harvesting for wireless gas sensor nodes. *Sens. Actuators A Phys.* **275**. https://doi.org/10.1016/j.sna.2018.03.026 (2018).
20. Adu-Manu, K. S., Adam, N., Tapparello, C., Ayatollahi, H. & Heinzelman, W. Energy-harvesting wireless sensor networks (EH-WSNs): A review. *ACM Trans. Sens. Netw.* **14**. https://doi.org/10.1145/3183338 (2018).
21. Culler, D., Estrin, D. & Srivastava, M. Guest editors' introduction: Overview of sensor networks. *Computer* **37**, 41–49. https://doi.org/10.1109/MC.2004.93 (2004).
22. DupÃl', V., Terrasson, G., EstÃl'vez, I. & Briand, R. Autonomy constraint in microsensor design: From decision making to energy optimization. In *2012 IEEE International Conference on Green Computing and Communications*, 647–650. https://doi.org/10.1109/GreenCom.2012.102 (2012).
23. Srbinovska, M., Dimcev, V. & Gavrovski, C. Energy consumption estimation of wireless sensor networks in greenhouse crop production. In *IEEE EUROCON 2017—17th International Conference on Smart Technologies*, 870–875. https://doi.org/10.1109/EUROCON.2017.8011235 (2017).
24. Bouguera, T., Diouris, J.-F., Chaillout, J.-J., Jaouadi, R. & Andrieux, G. Energy consumption model for sensor nodes based on lora and lorawan. *Sensors* **18**. https://doi.org/10.3390/s18072104 (2018).
25. Zhao, S. *et al.* Understanding energy efficiency in IoT app executions. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 742–755 (IEEE, 2019).
26. Morin, E., Maman, M., Guizzetti, R. & Duda, A. Comparison of the device lifetime in wireless networks for the internet of things. *IEEE Access* **5**, 7097–7114 (2017).
27. Baranov, A., Spirjakin, D., Akbari, S. & Somov, A. Optimization of power consumption for gas sensor nodes: A survey. *Sens. Actuators A Phys.* **233**, 279–289. https://doi.org/10.1016/j.sna.2015.07.016 (2015).
28. Spirjakin, D., Baranov, A., Somov, A. & Sleptsov, V. Investigation of heating profiles and optimization of power consumption of gas sensors for wireless sensor networks. *Sens. Actuators A Phys.* **247**. https://doi.org/10.1016/j.sna.2016.05.049 (2016).
29. Tan, L. & Wu, M. Data reduction in wireless sensor networks: A hierarchical LMS prediction approach. *IEEE Sens. J.* 1708–1715 (2015).
30. Guha, S., Mishra, N., Roy, G. & Schrijvers, O. Robust random cut forest based anomaly detection on streams. In *Proceedings of The 33rd International Conference on Machine Learning, PMLR* **48**, 2712–2721 (2016).
31. Salinas, D., Flunkert, V., Gasthaus, J. & Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **36**, 1181–1191 (2020).
32. Rangapuram, S. S. *et al.* Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, 7785–7794 (2018).
33. Mahgoub, A. *et al.* { SOPHIA } : Online reconfiguration of clustered NoSQL databases for time-varying workloads. in *2019 USENIX Annual Technical Conference (USENIX ATC)*, 223–240 (2019).
34. Mahgoub, A. *et al.*{OPTIMUSCLOUD}: Heterogeneous configuration optimization for distributed databases in the cloud. In *2020 USENIX Annual Technical Conference (USENIX ATC)*, 189–203 (2020).
35. Mahgoub, A. *et al.SONIC*: Application-aware data passing for chained serverless applications. In *2021 USENIX Annual Technical Conference (USENIX ATC)*, 973–988 (2021).
36. Cook, A. A., Mısırlı, G. & Fan, Z. Anomaly detection for IoT time-series data: A survey. *IEEE Internet Things J.* **7**, 6481–6494 (2019).
37. Wang, Y., Wang, C., Shi, C. & Xiao, B. Short-term cloud coverage prediction using the ARIMA time series model. *Remote Sens. Lett.* **9**, 274–283 (2018).
38. Bhandari, S., Bergmann, N., Jurdak, R. & Kusy, B. Time series data analysis of wireless sensor network measurements of temperature. *Sensors* **17**, 1221 (2017).

Acknowledgements
This work has been sponsored by the Army Research Laboratory (ARL) under grant W911NF-20-2-0026 and National Science Foundation, Cyber-Physical Systems (NSF-CPS) program proposal number CNS-2038986/2038566.

Author contributions
S.S. and S.C. conceptualized the idea and the experiments. S.S. did the main design, implementation, and experiments. S.C. added corrections and modifications to the experiments and analyses and edited the manuscript. S.C. provided funding for the project. A.B. and C.G. provided some of the datasets, the methods for data collection, and helped interpret the results. All authors reviewed the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to S.C.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.