Predictive Intra-Edge Packet-Source Mapping in Agricultural Internet of Things

Anandarup Mukherjee[1*], Nidhi Pathak[2], Sudip Misra[1], and Sushmita Mitra[3]
[1]Department of Computer Science and Engineering, [2]Advanced Technology Development Center
Indian Institute of Technology Kharagpur, India
[3] Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India

***Abstract*—Dense IoT implementations incur heavy data load on the implemented networks. In this paper, we propose and evaluate a low-latency method of increasing the packet throughput in agricultural IoT implementations. The proposed method envisions removal of node identifiers from packets before transmission and predictive packet-source mapping method within the edge layer of an agrarian Internet of Things (IoT) implementation. The edge layer following a master-slave architecture. Pre-trained lightweight machine learning models at the edge identify the origin of the incoming packets based on the long-term learned collective variations of the sensorial values from the slave node. This reduction in packets significantly frees up time-slots at the receiving master node, allowing for more simultaneous connections to it. This intra-edge packet origin mapping scheme is further compared with the approach of edge node identification at a remote server to adjudge the tradeoffs between accuracy and latency of transmission. The proposed method doubles the amount of sensor data transmitted between the slave to master nodes with significant energy savings over longer duration and increases the data throughput by approximately 1.5 times between the master node and the remote server for our implementation. The proposed method estimates energy savings in the order of 20 watts for a deployment setup of 100 nodes over a year. The energy savings over densely deployed IoT networks can be utilized to accommodate more nodes and increase the lifetime of the network.**

***Keywords*—IoT, Agriculture, Machine learning, Edge device processing.**

I. Introduction

The use of IoT in agriculture is rapidly gaining popular acceptance as is evident from various large multi-organization and multi-country initiatives to link agriculture and its associated domains of supply chain management, food processing, and storage to achieve regional and global food security [1]. Owing to its constant need with widespread presence and dependencies, technological and digital intervention in agriculture is a vast domain, which includes specializations such as crop, weed and pest monitoring, water management, remote sensing, nutrient management, yield forecasting, storage, and others. Our work is related to the domain of crop monitoring and water management in agricultural fields using continuous wireless sensing of field parameters such as soil moisture, soil temperature, and humidity. Considering a typical master-slave architecture, a slave node is a node which senses the data and simply forwards it to the master node. The master node processes the data and forwards it to the remote server. However, agricultural implementations of IoT require a huge number of sensor nodes, spread all over the agrarian area, which generates large volumes of data. This large data volume is not only a burden on the master node's processing resources but is limited by the network bandwidth. Considering an available network bandwidth of β units between the master and a set of n slave nodes $N_i : N_i = \{N_1, N_2, N_3, \cdots, N_n\}$. Each slave node consists of a set of k sensors $S_j : S_j = \{S_1, S_2, S_3, \cdots, S_k\}$ collectively transmitting $|N_i||S_j| \times n_{id}$ bytes (considering 1 byte per sensor reading) to the master node at each instant of time, where n_{id} is the data due to the node identifiers from each of the n slave nodes. Considering Shannon's channel capacity C [2] for a noiseless channel transmitting two symbols (0 and 1), $C = 2 \times B \times log_2 2 \geq 8|N_i||S_j| \times n_{id}$ bits per second. Accommodation of more nodes or sensors beyond what is permissible by C for the current setup is achieved by reducing n_{id}.

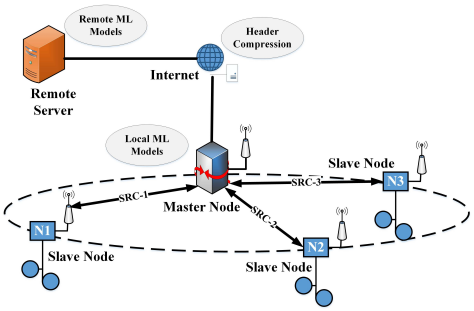


Figure 1: The communication and functional scheme of our implementation.

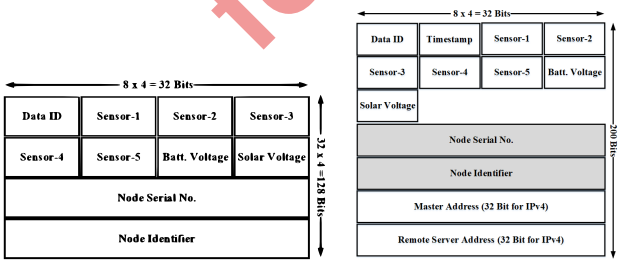
The constrained processing capabilities of the in-field nodes and the network throughput demands a solution which increases the efficiency of an existing network in terms of throughput and latency. In this work, we propose the complete elimination of n_{id} values logged from the slave nodes to the master node at each time instant. We deploy a pre-trained machine learning model at the master node, which is also the edge device of our communication architecture, to predict the origin of the incoming identifier-less slave sensor node values, and sort them accordingly for further transmission to the remote server via the Internet. This scheme of transmitting identifier-less packets from the slave nodes significantly increases the total effective sensor data transmitted $|N_i||S_j|$

without changing the network characteristics. As the edge device (master node) used in our implementation is a Raspberry Pi unit, which has constrained computational resources, we also evaluate the choice of the machine learning algorithm to be deployed on the edge. Section IV evaluates the performance of this identification scheme against the following algorithms – Decision Trees, Random Forests, Multilayer Perceptrons, and k-Nearest Neighbor. The proposed model is then compared against the traditional original data transmission in terms of throughput achieved for both slave-to-master and master-to-remote server connection and corresponding energy savings.

The positioning of a pre-trained model at the edge allows for lower network latencies between the master node and the remote server. The main crux of this work is to estimate whether it is possible to identify incoming packets based merely on the combination of sensors and the temporal variations of these readings, which also includes the erroneous values contained in these readings. The work can be further extended for other applications involving dense sensor networks

A. Communication Architecture

We implemented our model on one of our existing IoT-based wireless agricultural parameter monitoring system, which continuously senses agricultural field parameters at various locations and transmits them through the Internet to a remote server. The on-field implementation of our system, at the edge layer, follows a master-slave communication architecture as shown in Fig. 1. An intermediate master node is responsible for collecting sensed packets from variously deployed slave node using short-range communication (SRC) radios such as Zigbee (IEEE 802.15.4). The slave nodes N host a simple processor, which is powered using solar energy, and are only responsible for sensing and forwarding field parameters (soil moisture, soil temperature, humidity, and others) to the master nodes M. A typical packet of slave node data intended for the master node in our implementation has an 8-bit data identifier field (Data ID). This field is followed by 7 consecutive 8-bit fields for accommodating sensed analog-to-digital conversion data, and two 32-bit node-specific fields consisting each of node serial number and node identifiers, which totals to 128-bits as shown in Fig. 2(a). Provisions have been made at the



(a) Slave node packet (b) Master node packet

Figure 2: The network transmission packet structures for various agents of our implementation.

master node to enable it to communicate with a remote server for forwarding the data using long-range communication (LRC) using IPv4 over LTE, GSM, or Ethernet. The master nodes and the remote server derive power from a regular power distribution grid. Similar to many slave nodes communicating with a single master node, many geographically separated master nodes connect to a single remote server via the Internet.

Similar to the slave node packets, a typical packet from the master node (as shown in Fig. 2(b)) to the remote server has additional 8-bits for timestamp data, which gets added at the master at the instant of arrival of slave node data. In continuation, unlike slave node packets, the 64-bit node-specific fields are accentuated by two fields, each 32-bits long, which are intended for the IPv4 master node address and the address of the remote server respectively, totaling to 200 bits for each packet.

B. Slave Node Implementation

The slave nodes implemented in the agricultural fields are each equipped with soil moisture sensors, soil temperature sensor, a solar voltage sensor, and a battery voltage sensor. The soil moisture sensors are placed distally at four different vertical depths of $15cm$, $30cm$, $45cm$, and $60cm$ to capture the vertical changes in soil moisture at different root zone depths of the crops. The soil temperature sensor is placed laterally at a depth of $10cm$ below the soil surface. The battery and solar voltage sensors are responsible for updating the changes in battery and solar voltages respectively and are placed in the box housing the processor, as shown in Fig. 3B. The slave

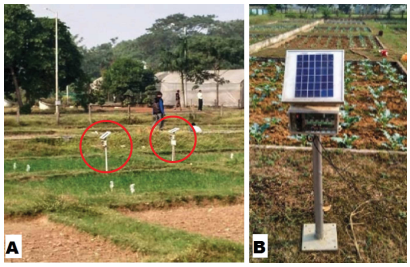
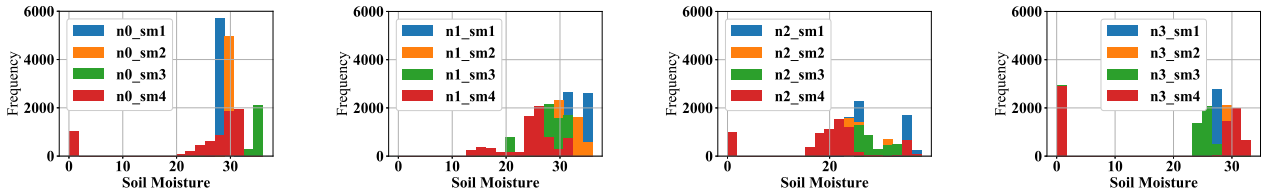


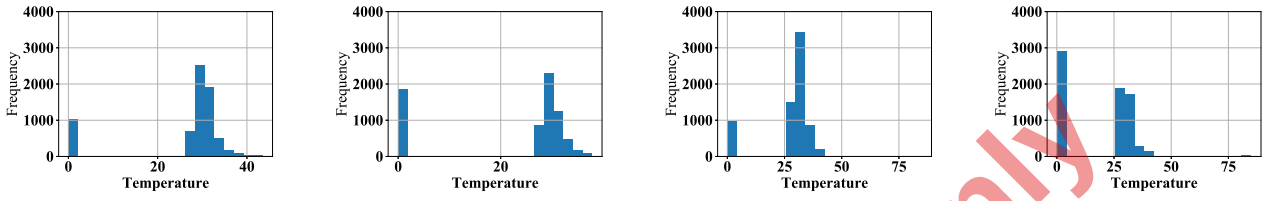
Figure 3: The implemented system in the agrarian field – (A) multiple slave nodes deployed in the field, (B) a single slave node.

nodes communicate to the master node using a Zigbee S1 radio module (IEEE 802.15.4) attached to the processor. The processor logs sensor data after every 30 minute and transmits the packetized data to the master node by appending a data index (Data ID) to it as shown in Fig. 2(a). Originally, the slave nodes followed the packet format shown in Fig. 2(a), however using our proposed scheme, we discarded the two 32-bit slave-specific fields (shaded part in Fig. 2(a)) to enable the transmission of a reduced packet to the master node. The implementation was done on a pre-existing network of 4 slave nodes communicating to a single master node, without making any physical changes or addition of new nodes to the network.



(a) Node-1 (b) Node-2 (c) Node-3 (d) Node-4

Figure 4: The soil moisture histogram for 4 months, polled every 30 minute at 4 different levels beneath the soil for various slave nodes of our implementation.



(a) Node-1 (b) Node-2 (c) Node-3 (d) Node-4

Figure 5: The soil temperature histogram for 4 months, polled every 30 minute beneath the soil for various slave nodes of our implementation.

C. Soil and Sensor Heterogeneity as a Unique Identifier

The special nature of agricultural data sensing of soil moisture and temperature involves a phenomenon called soil heterogeneity, which insinuates variability in the physical, chemical, and biological properties of soil from place to place [3]. This heterogeneity implies a minute, yet unique spatial footprint of the soil. In continuation, inherent manufacturing errors in the sensors [4] such as offset and sensitivity errors provide additional character to the measurement location of the soil profile as shown in Theorem 1. The histograms of the soil moisture sensors from 2 of the deployed slave nodes as shown in Fig. 4 indicate unique distribution patterns from the nodes during the same season, placed $30m$ apart. Nodes 2 and 3 (Figs. 4(b)) show a wider spreading as compared to nodes 1 and 4 (Figs. 4(a)) for the sensors with similar specifications and deployment depths, at similar instants of time. We attribute this behavior to the variations in irrigation applied to these locations resulting in the reported spreading. In contrast, we see from Fig. 5 that nodes 1 and 2 follow a soil temperature trend which is quite similar to the ones followed by nodes 3 and 4. However, we see a few erroneous readings from nodes 3 and 4, extending to 80, which is not possible. We attribute this behavior to the wiring and sensor errors.

Theorem 1. *The soil parameter measuring sensor combinations, internal sensor errors, and soil heterogeneity provide a unique identifier for each agrarian sensor node of our implementation.*

Proof. We consider the four soil moisture sensors of our implementation, first. The soil water content θ, capillary head ψ, time instant t, and vertical coordinate z which increases with depth, the 1-D Richard's equation [5] for an empirically calculated hydraulic conditioning function $k(\theta)$ is given as $\frac{\partial\theta}{\partial t} = \frac{\partial}{\partial z}\left[k(\theta)\left(\frac{\partial\psi(\theta)}{\partial z} - 1\right)\right]$. This equation highlights the dependence of θ on t and z, implying that the soil moisture content measured at each time instant is bound to change, and the moisture content recorded by our 4 sensors placed at different depths may have varying readings at the same time instant. Again, considering the remaining 3 sensors of our implementation – soil temperature, solar voltage and battery voltage. Let, each node N with k sensors be represented as $N \rightarrow \bigcup_1^k \{S_k + \epsilon_k\}$, where ϵ is the minute, yet unique inherent error due to manufacturing or calibration of the sensors. As $\epsilon_1 \neq \epsilon_2 \neq \cdots \neq \epsilon_k$, the logged values from different sensors at the same time instant t for the same location will be different. Considering $N_i(t) = \bigcup_1^k \{S_k + \epsilon_k\}$ representing the i^{th} sensor node at the time instant t, $N_i(t) - N_i(t-1) = \delta_1$, and $N_i(t) - N_{i-1}(t) = \delta_2$. The remnants δ_1 and δ_2 are very small and occur due to the inequality between the ϵ values.

Combining the heterogeneities incurred due to the soil moisture sensors and the other sensors in our implementation, we conclude that the combination of these values in a packet provides unique identifying features for the nodes over a period. □

D. Contributions

This work envisages the discarding of node identifiers from packet transmission to increase the network throughput without the change in the network characteristics or the characteristics of the implemented system. The following distinct contributions have been made in this work:

- An agricultural IoT implementation has been established, which records agricultural field parameters and transfers them to a remote server via a master node at the edge.

- A machine learning-based identifier-less packet origin mapping scheme at the edge layer is hosted on a master node of a master-slave implementation.
- Performance of various machine learning algorithms is evaluated on the edge as well as the remote server to choose the algorithm with the minimum trade-off between accuracy of classification and execution time on a resource-constrained processor.

II. Related Work

The advent of IoT has been a boon for sustainable technologies such as precision agriculture and farm management. Precision agriculture, which aims to usher-in food security [1] by making judicious use and balancing the application of water, fertilizer, and pesticides for increasing crop yield. Present day agricultural systems are making use of clouds, and Internet gateways to acquire agronomic data from the fields [6], smartly monitor and control irrigation systems, predict and warn farmers about yields and weather using advanced analytics [7]. Agricultural IoT deployments are characterized by dense implementations, which generate voluminous data per unit time. This surge in transmitted data has necessitated the use of efficient data transmission techniques [8]. Instead of forwarding data all the way to a cloud, low-computational analytics are performed locally in fog nodes to achieve faster results and save energy [9].

Edge computing, commonly used interchangeably with Fog computing is a promising alternative to cloud-based data offloading and analytics. These devices connect at the edge of the network, offering computation and processing power to the devices at the edge of the network, in between the end device and the cloud servers [10]. Fog computing is reported to be more efficient concerning power consumption, service-latency, and costs involved [11]. Additionally, the benefits of security, the privacy of data, and selective data processing based on quality and criticality of the data are also assured. Edge-based schemes such as inter-edge data offload [12], and content caching [13] has been shown to reduce network latency further and is proving beneficial towards managing IoT network traffic.

Synthesis: Most of the digital agricultural pursuits are standalone solutions, which often rely on remote servers and clouds for data storage and analysis of the data, where network and processing latencies, network throughput and energy efficiency are seldom addressed. In contrast, works on the edge-based decrease in network latency, energy usage optimization, cost optimization and improving user experience in case of online web services and e-commerce is being widely pursued. In this work, we show the benefits of edge-based processing of data and machine learning towards the reduction of latencies and increase in network throughput for agricultural IoT, which and can be extended to other domains having dense IoT deployments.

III. Predictive Identification of Packets

The predictive identification of identifier-less packets is performed by making use of a pre-trained machine learning model placed at the receiving end of the network. We choose two locations for testing this approach – 1) the edge (or, the master node), and 2) the remote server. The machine learning model at the edge node is responsible for identifying the incoming packets from the slave nodes, whereas the remote server model is responsible for determining the originator slave nodes from the master nodes packets. The identification mechanism is outlined in Algorithm 1. The models are trained by data collected from the same set of sensor nodes over a period of 4 months, polled at an interval of 30minutes during a full cropping season. The machine learning model trained using this dataset is used for identifying the incoming data from the next cropping cycle.

As described earlier, the edge device is computationally constrained, compared to the remote server. The machine learning models hosted at the remote server may not perform well on the edge device considering execution time as the factor. We modify the architectures of the models at the remote server (which we consider as *heavy models*) to be more lightweight to incur lower execution latencies on the edge device, and refer to these models as *light models*.

Algorithm 1 Identification of identifier-less packets

Require:
INPUT:
$D_t = \{S_1, S_2, S_3, \cdots S_m\}$ ▷ D : Data incoming from slave nodes ▷ t : Time instant ▷ S : Sensors
OUTPUT:
$max\ P(N_i) \ni N_i = \{N_1, N_2, N_3, \cdots N_k\}$ ▷ N : Set of nodes deployed
Initialize:
Trained Model(M) ▷ M : Trained ML model at the master node

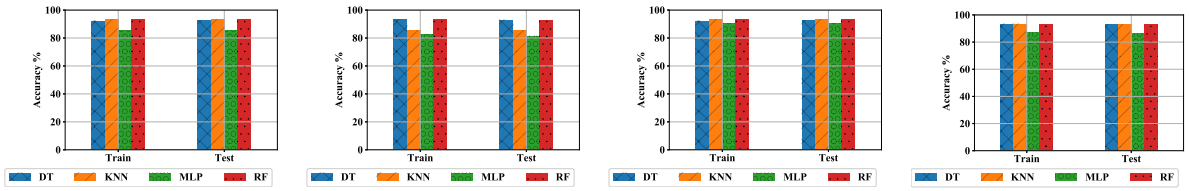
```
while (D_t) do
    Predict P(N_i)
    Return max P(N_i)
end while
```

A. Machine Learning Preliminaries

This section provides an overview of the four implemented machine learning algorithms.

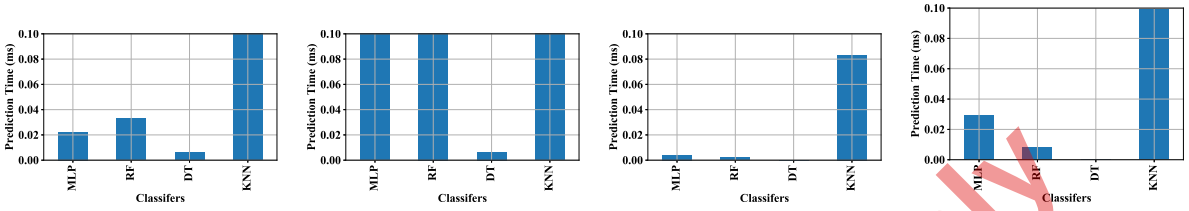
1) Decision Trees (DT): A decision tree uses a tree-like structure to arrive at a decision or goal. Decision tree in machine learning (ML) has been used for classification and regression modeling. It uses recursive binary splitting to divide the set of data into different subsets using a cost function. The attribute selection can be broadly based on calculating the information gain and Gini index. Entropy is used to find the randomness in the information contained in the attributes. In this work, the heavy model of DT has a maximum depth of 10, whereas the light model has a maximum depth of 5.

2) Random Forests (RF): Random forest method is an ensemble method for classification and regression, introduced by Leo Breiman in [14]. It uses an ensemble of decision trees



(a) Light model on edge (b) Heavy model on edge (c) Light model on remote server (d) Heavy model on remote server

Figure 6: The accuracies of the chosen classifiers on both edge node as well as remote server with both – a) light models, and b) heavy models.



(a) Light model on edge (b) Heavy model on edge (c) Light model on remote server (d) Heavy model on remote server

Figure 7: The execution time of the chosen classifiers on both edge node as well as remote server with both – a) light models, and b) heavy models.

during training to make the final ranking. The idea is to utilize multiple weak decision trees to reach a stronger decision. This method helps overcome the over-fitting problem in decision trees due to the noises present in the training dataset. In this work, the light RF model has 5 estimators, whereas the heavy model has 20 estimators.

3) Multilayer Perceptrons (MLP): A Multilayer Perceptron is a regression model with a deep artificial neural network where each perceptron classifies the input by separating it into 2 outputs. It consists of an input layer, an output layer and hidden layers between the input and output layers. The hidden layers perform the regression and classification process using backpropagation. Each layer has multiple nodes or neurons that are connected to each node with some weight in the next layer resulting in a high degree of connectivity We use Regularized Linear Units (ReLU) [15] as the non-linearity in the hidden layers in conjunction with a Limited Memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) [16] solver for our MLP model. The heavy model in this work consists of 4 hidden layers, each with 64, 128, 5, and 5 neurons respectively, whereas the light model uses 2 hidden layers with 5 neurons each.

4) K-Nearest Neighbours (kNN): K-nearest neighbor or kNN algorithm is a non-parametric method for classification and regression models.It uses the k nearest object classes in the feature space for training. As part of the supervised learning, the unknown classes are classified as per the known dataset with maximum voting in the defined k value. As kNN classification works at runtime, its performance degrades with larger datasets. The kNN heavy model in this work makes use of 5 neighbors, whereas the light model used 3 neighbors for classification of the nodes with a leaf size of 30.

IV. Performance Evaluation

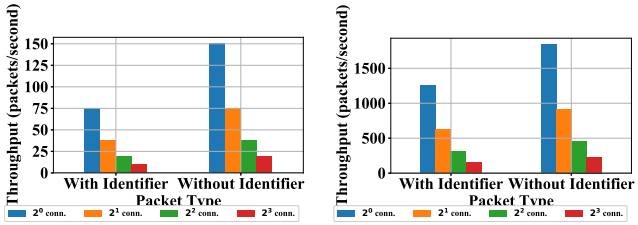
This section is divided into three parts – 1) performance of the machine learning algorithms, 2) network throughput, and 3) energy savings.

A. Performance of Machine Learning Algorithms

Fig. 6 shows the accuracy of node classification of heavy and lightweight models running on both the edge nodes and the remote server. The identification of the slave node is carried out on the packetized sensor values received by the master node using the four selected algorithms. While an overall average performance is maintained in both the heavy and lightweight models at the edge node, there is a significant difference in the performance of kNN and MLP classifiers for their corresponding heavy and lightweight models in both the edge node as well as the remote server.

We attribute this difference to the low computational power of the edge device and the runtime complexities of the algorithms. RF and DT give a good and consistent performance in both models, which makes them stand out from the other classifiers concerning training and testing accuracies. The execution time taken by kNN is the highest, which is attributed to the runtime processing nature of the algorithm and is expected to increase with the increasing size of the dataset. Considering the edge node, the execution time of DT outperforms RF classification model in both its lightweight and heavyweight forms as is seen in Fig.7(a) and 7(b). The same trend for node classification performance is observed at the remote server. Collectively, it is inferred from Fig. 6 and 7 that DT has better performance for both edge nodes and the remote server in comparison to the other classifiers. The performance of DT is closely followed by RF. MLP, although has lesser accuracies for node classification than kNN but

performs significantly better than kNN concerning execution times.



(a) Slave to master link. (b) Master to remote server link.

Figure 8: The network throughput for original and identifier-less packet types during transmission between – a) slave to master at 9600 bps, and b) master to the remote server at 250 kbps.

B. Throughput

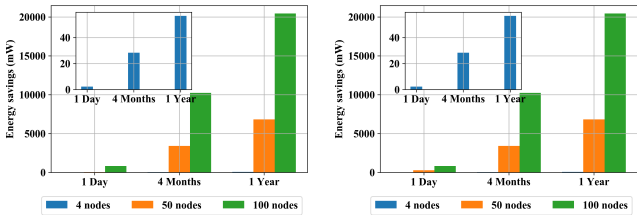
The network throughput, represented in terms of data packets transmitted per second, is compared for the two connections – 1) slave to the edge, and 2) edge to the remote server. Fig.8(a) shows the comparison of the network throughput at the link between the edge and slave nodes, obtained with the identifier information intact (original) and without the identifier in the transmitted packet. The identifier-less packet data sustain twice the number of connections as compared to the original data packet. A similar trend is seen for the edge to the remote server link with a Wi-Fi rate of 250kbps as summarized in Fig.8(b), where the identifier-less packets enable an increase in throughput by 1.5 times in comparison to the original packet type. It is prominent from the results that the compression of data enables more substantial data transmission and efficient utilization of the available bandwidth by removing the redundant data.

C. Energy Savings

The energy savings for our proposed scheme is calculated and projected for a data transmission interval of every 30 minute for varying duration of 1 day, 4 months (a cropping cycle), and a year against varying deployment sizes. The energy saving for the link between the master and slave nodes, and the link between the master node and remote server are shown in in Fig. 9. The resulting energy savings also ensure that our scheme optimizes the implementation, without changing the actual hardware of the implementation. Our scheme is able to save additional 20 watts of energy for a 100 node deployment over a period of a year, simply by discarding the identifier fields in the transmitted data. The proposed scheme has the capacity to usher-in more energy savings for larger and longer-duration deployments.

V. Conclusion

The performance of our proposed method on the implemented hierarchical master-slave IoT architecture for agriculture enables the support of almost twice the number of



(a) Slave to master (b) Master to remote server

Figure 9: The energy saved for original and identifier-less packet types during transmission between – a) slave to master, and b) master to remote server.

slave connections at the master node, in comparison to the transmission with original packet types, within the same time interval. The proposed method also enables the support of almost 1.5 times more data within the same range of time as compared to original packet type. Upon evaluation of the performance of the chosen classifiers for our scheme, we see that the DT classifier outperforms the other classifiers concerning classification accuracy and execution time for both locations – remote servers as well as edge nodes for the nature of the data generated by our implementation. Concerning classification accuracies, DT is followed by RF, kNN, and MLP. However, regarding execution times, DT is followed by RF, MLP, and finally kNN in increasing order. Additionally, our scheme allows for energy savings in the order of 20 watts for larger deployments over long duration of time, which is significant considering the nodes and communication medium used are extremely low-power ones. This signifies the allowance of more number of nodes or increased lifetime of the existing network.

The proposed method although capable of increasing the throughput within the network bandwidth along with notable energy savings, without the change in infrastructure, however, is highly location and sensor specific. In the future, we plan to implement online learning schemes with the edge devices to remove the dependencies on location and sensor-specific datasets.

References

[1] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "IoT in Agriculture: Designing a Europe-Wide Large-Scale Pilot," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 26–33, 2017.

[2] C. Shannon, "The zero error capacity of a noisy channel," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 8–19, 1956.

[3] T. Elkateb, R. Chalaturnyk, and P. K. Robertson, "An overview of soil heterogeneity: quantification and implications on geotechnical field problems," *Canadian Geotechnical Journal*, vol. 40, no. 1, pp. 1–15, 2003.

[4] R. C. Jaeger, J. C. Suhling, and R. Ramani, "Errors associated with the design, calibration and application of piezoresistive stress sensors in (100) silicon," *IEEE Transactions on Components, Packaging, and Manufacturing Technology: Part B*, vol. 17, no. 1, pp. 97–107, 1994.

[5] S. E. Serrano, "Modeling infiltration with approximate solutions to Richard's equation," *Journal of Hydrologic Engineering*, vol. 9, no. 5, pp. 421–432, 2004.

[6] N. Pavón-Pulido, J. López-Riquelme, R. Torres, R. Morais, and J. Pastor, "New trends in precision agriculture: a novel cloud-based system for

enabling data storage and agricultural task planning and automation," *Precision Agriculture*, vol. 18, no. 6, pp. 1038–1068, 2017.

[7] S. Rajeswari, K. Suthendran, and K. Rajakumar, "A smart agricultural model by integrating IoT, mobile and cloud-based big data analytics," in *International Conference on Intelligent Computing and Control (I2C2)*. IEEE, 2017, pp. 1–5.

[8] P. Lerdsuwan and P. Phunchongharn, "An energy-efficient transmission framework for iot monitoring systems in precision agriculture," in *International Conference on Information Science and Applications*. Springer, 2017, pp. 714–721.

[9] M. Roopaei, P. Rad, and K.-K. R. Choo, "Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 10–15, 2017.

[10] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management," *ACM Comput. Surv.*, vol. 51, no. 2, pp. 39:1–39:34, Apr. 2018.

[11] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," *IEEE Transactions on Cloud Computing*, 2015.

[12] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-Aware Video Analytics on Edge Computing Platform," in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2573–2574.

[13] E. Ahmed and M. H. Rehmani, "Mobile edge computing: Opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59 – 63, 2017.

[14] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[16] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems," *EECS Department, Northwestern University, Evanston, IL, Technical Report No. NAM-11*, 1994.