Advances in Internet of Things and Edge Computing; Research 1 (1) 1-19 Received 18-04-2024; Accepted 21-05-2024; Published 28-05-2024

Efficient Mist Computing in Cyber-Physical Systems

Rolando Herrero r.herrero@northeastern.edu
College of Engineering
Northeastern University
Boston, MA, USA 02115-5000

Corresponding Author: Rolando Herrero

Abstract

There is a growing trend to move some of the analytics that are typically performed on cloud applications to the access side of the network of cyber-physical systems in order to reduce latency, improve response times and extend the infrastructure lifetime. However, this approach, known as mist computing, introduces problems associated with the broad sharing of sensor readouts. Specifically, smart devices are deployed in constrained networks and have limited power, making them constrained devices. Because these devices are unsupervised and rely on batteries for power, they are inherently power-limited. Moreover, to preserve battery life, they typically transmit data at very low rates. This situation leads to a number of issues in the context of smart device analytics. This paper introduces these problems and provides a solution that enables the efficient support of analytics in a network of constrained smart devices in the context of mist computing.

Keywords: Mist Computing, TinyML, LoRa, CoAP, 6Lo, SD-DNS

1. INTRODUCTION

Cyber-Physical Systems are key to supporting a number of initiatives ranging from Industrial *Internet of Things* (IoT) to smart grids and home automation [1]. In these systems, human intervention is non-existent. Interaction with the physical environment occurs through sensing devices, which convert data into information. This information is then used by applications that convert the information into knowledge. The applications use this knowledge to trigger actuation back at the devices. The critical factor in this interaction is the latency between sensing and actuation. This latency, in turn, depends on the location of the devices and the applications, as well as the technologies and the interconnectivity used to enable their communication. Another crucial factor is network lifetime. Since devices rely on batteries, it’s essential to preserve their power to ensure the infrastructure remains active for as long as possible [2].

The applications that transform information into knowledge using *Artificial Intelligence* (AI) techniques have traditionally been deployed on cloud infrastructure. However, due to latency constraints, this has led to a growing trend of moving applications closer to the devices. Fog computing at edge border gateways is one initial approach. While fog computing is a viable approach, for many scenarios mist computing offers the best solution. In mist computing, one of the devices on the access side of the network performs the computing that enables the conversion from information to knowledge.

Refer to FIGURE 1, for an example of a *Low Power Wide Area Network* (LPWAN) where devices take turns processing information from other devices and forwarding the processed information to the edge device and other devices over multiple rounds. In summary, for any given round, a device receives the aggregate sensor information from all other devices, converts the information into actionable knowledge, and generates actuation commands that are propagated back to the relevant devices. Depending on the scenario, the processed information is sent to the edge device, which then broadcasts it to the broader Internet.

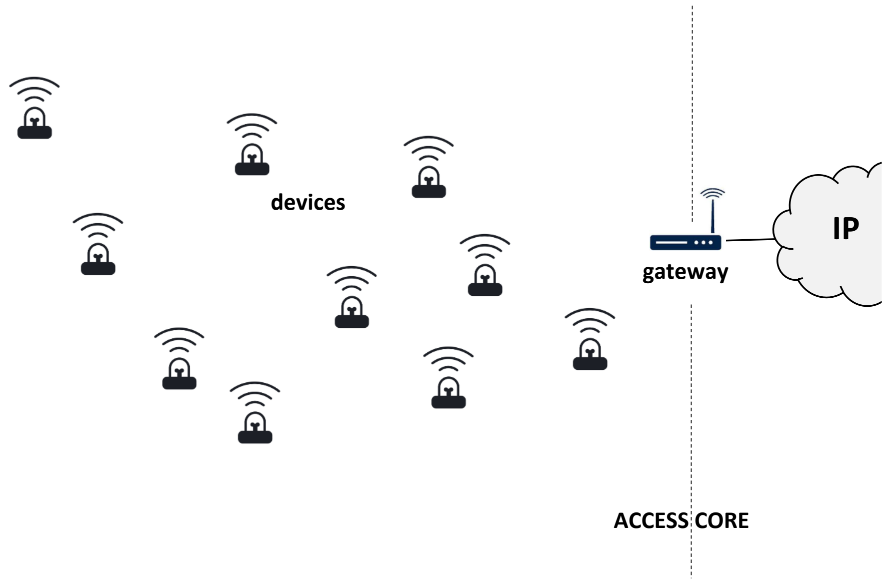


Figure 1: Multi-Hop LPWAN Topology

This paper explores a well-known legacy mechanism known as *Power-Efficient GAthering in Sensor Information Systems* (PEGASIS) that was developed to support routing in the context of *Wireless Sensor Networks* (WSNs) [3]. PEGASIS operates within the context of the physical and link layers of the *Internet Engineering Task Force* (IETF) Layered Architecture and provides an efficient way to enable sensors to forward data to a *Cluster Head* (CH) that forwards the aggregated traffic to a sink. Since CHs consume more energy, the sensors take turns becoming one, thus distributing the energy consumption in order to maximize the overall lifetime of the WSN. This very same principle can be adapted to support Mist based AI [4].

The goal is to recycle some of the principles introduced by PEGASIS by relying on standard physical, link, network and transport layer protocols and moving its functionality to the application layer. Specifically, the work presented in this paper relies on devices that rely on physical and link layers that are *Long Range* (LoRa) based. LoRa is an umbrella term for a full protocol stack that provides LPWAN capabilities in devices that run on a single battery for more than ten years [5–7]. Through network layer adaptation, LoRa can be used with Internet suite protocols to support IoT architectures. This paper introduces an algorithm that adapts PEGASIS to these IoT procotols to dynamically support information aggregation and knowledge extraction to enable both low-latency sensing and actuation [8].

The remainder of the paper is organized as follows: Section 2 presents a detailed discussion of the related work. Section 3 details the fundamental technologies used to support the proposed algorithm. Section 4 presents an algorithm that supports low-latency analytics and extends the lifetime of the infrastructure on a Mist topology. Section 5 provides a description of the evaluation framework as well as comparative results. Conclusions and future work are introduced in Section 6.

2. RELATED WORK

Mist computing on smart devices has been discussed in [9] and [10]. Mist computing for real-time processing is presented in [11]. In [12], the authors introduce a mist computing scheme that relies on LoRa. Similarly, in [13], the authors develop a mist computing application that supports gesture recognition. Mist computing to support occupancy detection is detailed in [14]. Mist computing to support face identification is introduced in [15]. In [15], the same authors introduce *Convolutional Neural Networks* (CNNs) to support face identification. Finally, in [16] the system is upgraded to enable face recognition on a system that relies on solar panels for energy harvesting. With the advent of battery-less devices, the use of mist computing applications has become crucial [10]. To this end, schemes of intermittent computing where multiple applications process sensor readouts are discussed in [17] and [18]. The corresponding scheduling mechanisms are detailed in [19]. In [20], the authors present an energy-aware scheduler. Similarly, in [21], a novel scheduling mechanism that applies to energy-harvesting devices is presented. Devices that rely on super-capacitors to enable scheduling are presented in [21]. Note that none of these works addresses the technologies and their interaction to support the distributed mist computing algorithm presented in this paper.

3. TECHNOLOGICAL BACKGROUND

This Section reviews the main technologies used to support the proposed mechansim of mist computing in multi-hop LPWAN cyber-physical systems. This includes details of PEGASIS and IoT networking standards like LoRa, *Internet Protocol version 6 over Networks of Resource-constrained Nodes* (6Lo) and *Constrained Application Protocol* (CoAP) [22–24].

3.1 PEGASIS

PEGASIS is a routing protocol that reduces energy consumption by sharing the responsibility of CH duties among all devices within a network section. Additionally, PEGASIS lowers network delay by performing data aggregation on multiple devices concurrently. These mechanisms combined contribute to extending the overall lifespan of the network [1].

In the PEGASIS mechanism, each device is aware of the geographical location of its neighboring devices. This allows the device to adjust its transmission power to only reach its closest neighbor.

This power control is typically achieved using spectrum modulation techniques. PEGASIS utilizes a chain structure formed through direct communication between devices and their immediate neighbors. FIGURE 2 illustrates an example. Device *A* begins by selecting its closest neighbor, device *B*, as the first link in the chain. Device *B* then selects its closest neighbor, device *C*, as the second link. This process continues with all other devices selecting their closest neighbors to establish the chain's links. It's important to note that chain formation starts with the farthest device and progresses by sequentially adding devices to the chain. The selection process is greedy, meaning devices choose their closest neighbor based on the strength of the received signal, which can be used to estimate distance.

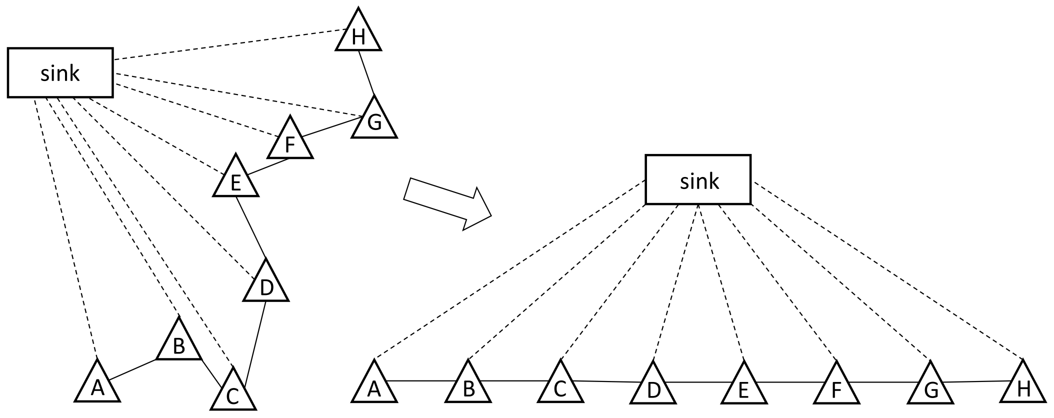


Figure 2: PEGASIS Structure

With PEGASIS, data aggregation is not solely performed by the CH. During a PEGASIS round, approximately half of the devices participate in some form of data aggregation. FIGURE 3 illustrates a PEGASIS round with eight devices, labeled *A* through *H*, positioned 0 through 7 within the chain. In this example, device *D* (initially in position 3) is the CH. The first stage involves devices in even positions (0, 2, 4, 6) transmitting their data to their nearest neighbors in the odd positions (1, 3, 5, 7). Following this stage, device *B* possesses data from device *A*, device *D* possesses data from device *C*, device *F* possesses data from device *E* and device *H* possesses data from device *G*. Subsequently, devices *B*, *D*, *F*, and *H* move to positions 0, 1, 2, and 3, respectively. The second stage involves devices in the remaining even positions (0 and 2) transmitting their data to their neighbors in the odd positions (1 and 3). After this stage, device *D* possesses data from device *B*, device *H* possesses data from device *F*. In the final stage, only the CH (device *D*) and device *H* remain. Since the CH always aggregates data regardless of its position, device *H* forwards all its data to the CH. Once the CH has accumulated data from all other devices, it transmits it to the sink. The mechanism is designed to ensure fair distribution of energy consumption across the network. This is achieved by rotating the role of the CH and the data transmitting devices based on their location. For any given round with *N* devices, the number of stages required is approximately the ceiling of the base-2 logarithm of N ($\lceil \log_2 (N) \rceil$). Additionally, energy is conserved by limiting the transmission power of each device such that it only reaches its designated uplink neighbor. This approach leads to a higher *Signal-to-Noise Ratio* (SNR), which enables parallel transmissions. This parallelism maximizes the number of simultaneous transmissions and minimizes the number of stages required.

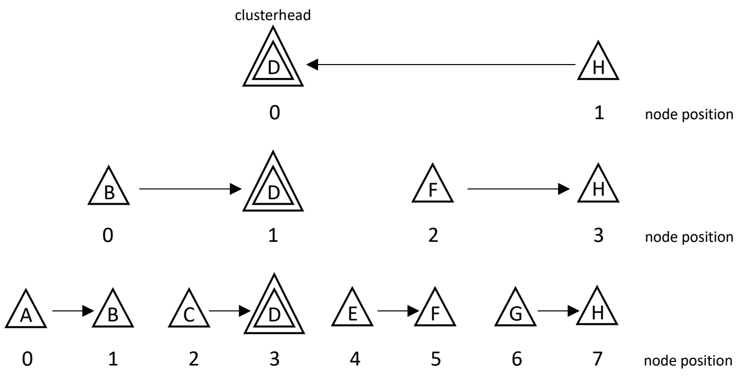


Figure 3: PEGASIS aggregation

3.2 IoT

IoT technologies provide the connectivity between devices and applications as well as between devices themselves. A critical factor in qualifying a Cyber-Physical System for IoT compliance is the use of end-to-end Internet suite protocol connectivity. Specifically, LPWAN connectivity is provided through LoRa for the physical and link layers, 6Lo for the network and transport layers, and CoAP for the application layer. *Service Discovery Domain Name Service* (SD-DNS) provides a way for devices to identify themselves without the need of any human intervention [25, 26].

3.2.1 LoRa

LoRa is a detailed set of communication protocols designed for LPWAN topologies. It allows devices to function for over ten years on a single battery charge. As shown in FIGURE 4, the LoRa architecture consists of three layers: physical, link, and application. Network and transport layers are not included because LoRa focuses on direct communication between LoRa devices and the absence of built-in IP support. Each layer is further divided into sublayers with specific functionalities. LoRa's modulation techniques prioritize power efficiency and enable exceptional range. It can exceed 10 kilometers at low data rates (the exact coverage depends on physical barriers). A single gateway can potentially cover hundreds of square kilometers. It's important to note that the number of supported devices and transmission speed are inversely proportional to range. In other words, longer range comes at the cost of fewer devices and slower speeds.

LoRaWAN [27], the core technology of LoRa networks, goes beyond just connecting devices. It establishes the entire network architecture and defines communication protocols. Unlike traditional *Wireless Personal Area Networks* (WPANs) that rely on complex mesh networking for extended range, LoRaWAN utilizes a simpler approach: direct communication between devices and gateways. This eliminates the need for data aggregation through intermediary nodes, which helps conserve battery life and network capacity for devices with limited resources. LoRaWAN

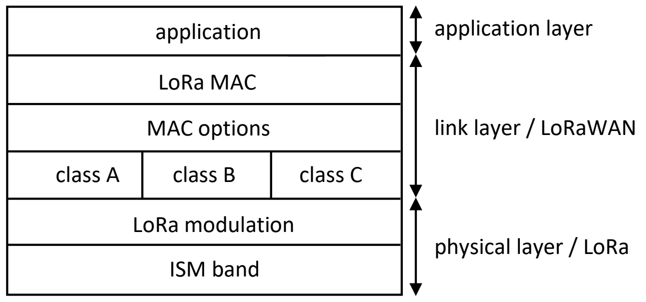


Figure 4: LoRa Stack

devices fall into three distinct classes (A, B, and C). These classes influence the device's power consumption, how it interacts with gateways, and the balance between factors like battery life, latency, and throughput. The most suitable class depends on the specific application's needs. For instances where battery life is the top priority and latency is less critical, Class A might be the best choice. On the other hand, applications requiring real-time communication might benefit more from Class C, even though it consumes more power. Class B serves as a compromise, ideal for scenarios where both power efficiency and timely downlink communication are important.

Internet Protocol version 6 (IPv6) adaptation for LoRa utilizes the protocol stacks depicted in FIGURE 5. LPWAN technologies share a common layered structure. The key difference, as expected, lies in the physical and link layers. In the context of LoRa, *IPv6 over Low-Power Wireless Personal Area Networks* (6LoWPAN), a variant of 6Lo technology, acts as the adaptation mechanism. Specifically, 6LoWPAN enables encapsulation of IPv6 and *User Datagram Protocol* (UDP) to facilitate the transport of CoAP packets.

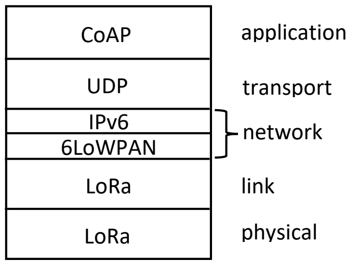


Figure 5: CoAP over LoRa

3.2.2 6Lo

IPv6 over LoRa is not a standardized protocol, but 6LoWPAN, a related technology, can be used for encapsulation. To support IPv6 datagram encapsulation, 6LoWPAN employs header compression techniques for both IPv6 and UDP headers, along with specific fragmentation rules. 6LoWPAN achieves header compression through two mechanisms. *Stateful IP Header Compression* (IPHC) leverages redundancy between datagrams to minimize header size. Similarly, *Next Header Compression* (NHC) compresses the UDP header specifically. Both IPHC and NHC are small headers containing essential fields for processing network and transport layer information. These headers are identified by their initial bit sequences, known as dispatch values. When an IPv6 datagram needs fragmentation, 6LoWPAN uses two different headers prepended to the datagram fragments. Initial Fragments include a dispatch value, the entire datagram size, and a datagram tag for identifying the fragment group. Subsequent Fragments include an additional field specifying the fragment offset within the datagram, measured in 8-byte units. To minimize processing complexity, every fragment carries the full datagram size. This information allows for immediate buffer allocation upon receiving the first fragment, facilitating efficient reassembly [28].

3.2.3 CoAP

CoAP is a standardized, session-layer protocol designed specifically for efficient data exchange in Cyber-Physical Systems. Introduced in 2014 by the IETF through RFC 7252 *The Constrained Application Protocol (CoAP)* [22], CoAP is anticipated to become the dominant protocol for accessing and managing billions of devices within the IoT landscape. Applications of CoAP span various domains, including smart energy grids, building automation, intelligent lighting control, industrial control systems, asset tracking, and environmental monitoring.

CoAP differs from other protocols in its use of the UDP as its transport layer. Because UDP is connectionless, it functions well in various scenarios, including unicast, multicast, and broadcast communication, which are frequently encountered in IoT applications. Furthermore, UDP contributes to lower latency by omitting a built-in mechanism for retransmitting lost packets. While this eliminates guaranteed delivery, it avoids potential delays that retransmissions could cause [29].

CoAP is designed to provide some of the functionalities offered by the *HyperText Transfer Protocol* (HTTP) but caters to the IoT environment. In an IoT network, CoAP is typically used for accessing data at the edge of the network, while HTTP is used for communication within the core. A gateway acts as an intermediary, translating messages between the two protocols. This is illustrated in FIGURE 6.

An application, often performing analytics, initiates data retrieval by sending an HTTP GET request to obtain sensor readings. This message travels across the network using both the *Transmission Control Protocol* (TCP) and IPv4, encapsulated within IEEE 802.11 frames. The gateway intercepts the HTTP request and converts it into a CoAP GET message. This CoAP message is then transmitted using UDP over IPv6. It's important to note that IPv6 is adapted for low-power networks using 6LoWPAN.

Upon receiving the CoAP request, the sensor transmits a response containing the readout as a CoAP *2.05 Content* message. The gateway translates this CoAP message into an HTTP *200 OK* response for the application. Since both CoAP and HTTP are stateless protocols, the message translation at the gateway also operates in a stateless manner.

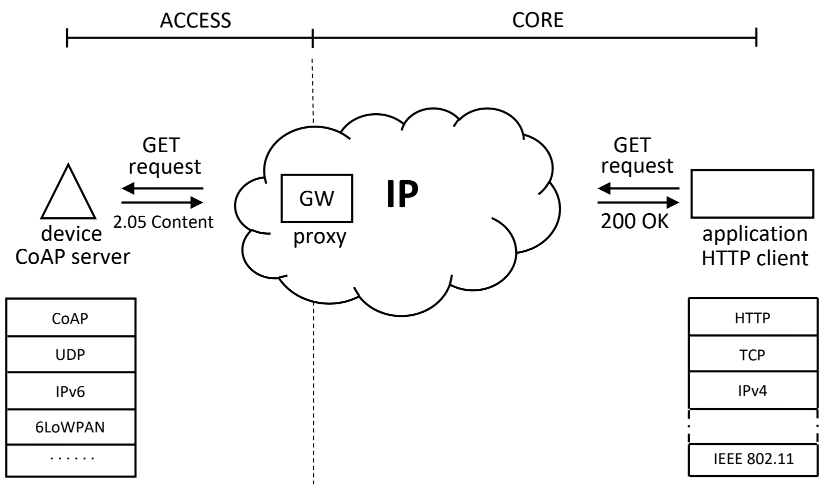


Figure 6: CoAP vs HTTP

3.2.4 SD-DNS

Service discovery in the context of IoT, is standardized as an extension to traditional *Domain Name Service* (DNS) operations supported by the Internet protocol suite [30]. The DNS infrastructure consists of a network of DNS servers that can be globally reached by any client. In IoT architectures, the deployment of this infrastructure is not always possible due to a number of technical issues [1]. Because of this, service discovery is carried out by means of *Multicast DNS* (mDNS).

In essence, mDNS leverages the existing structure and syntax of DNS messages, including *Resource Record* (RR) types, operation codes, and response codes. However, mDNS defines how devices cooperate to send and receive multicast requests and responses effectively. DNS/mDNS messages travel over UDP on port 53. A DNS/mDNS message consists of a fixed-size header and a variable number of questions and RRs categorized as answer records, authority records, and additional records. Questions encompass the requested record name, type, and class, while RRs include only the record name, type, class, but also the time-to-live and the corresponding data associated with the record. The name identifies the resource within the device, while the type specifies the resource's nature. Common RR types include A and AAAA for specifying IPv4 and IPv6 addresses, PTR for reverse lookups, SRV for service information, and TXT for configuration information. The class is invariably set to IN for IP resource types. The TTL field indicates the number of seconds a resource record remains valid.

mDNS establishes the foundation for exchanging RRs between devices on the same network segment, as detailed in RFC 6763 [26]. However, mDNS itself doesn't define a process for discovering newly added devices. This is where SD-DNS comes in. SD-DNS leverages the mDNS infrastructure to enable automatic configuration. While SD-DNS dictates how RRs are named for service discovery, it doesn't modify the structure of DNS messages, operation and response codes, nor most DNS protocol values. Using SD-DNS, a client can issue DNS queries specifying the service type and domain to search for. In response, it receives a list of available devices offering that service.

4. THE ALGORITHM

This Section introduces an algorithm that combines mainstream IoT standards like LoRa, 6LoWPAN, CoAP and SD-DNS along with PEGASIS to maximize the network lifetime and minimize overall latency.

4.1 Prediction Service Clustering

In IoT architectures, machine to machine communications are key. The idea is for devices to be deployed in remote locations, and have the devices discover and associate themselves with AI prediction services. To this end, SD-DNS is crucial.

FIGURE 7 shows the interaction between devices using SD-DNS. A device will transmit a PTR mDNS request to find out the identity and configuration of all other devices involved with the prediction service. They are transmitted as multicast packets that can be observed and processed by all devices. Information related to addresses, session management, as well as geographical coordinates, are supplied by these messages. Note that this functionality leads to the formation of a prediction service cluster.

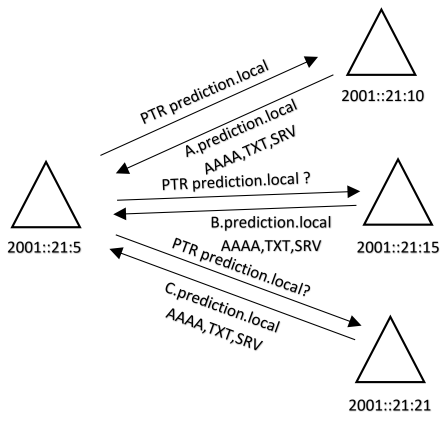


Figure 7: Prediction Service Discovery

Because devices know the location of the other cluster members, a chain can be formed by starting at the device that is farthest away from the center of the cluster (device A) as indicated in FIGURE 8. Subsequent devices in the chain are chosen based on proximity, with device B being closer to A, device C being closer to B, and so on, until the chain is fully formed. The algorithm is shown in FIGURE 9. Note that after the execution of these steps, the vector **S** will carry the sequence all devices in the chain.

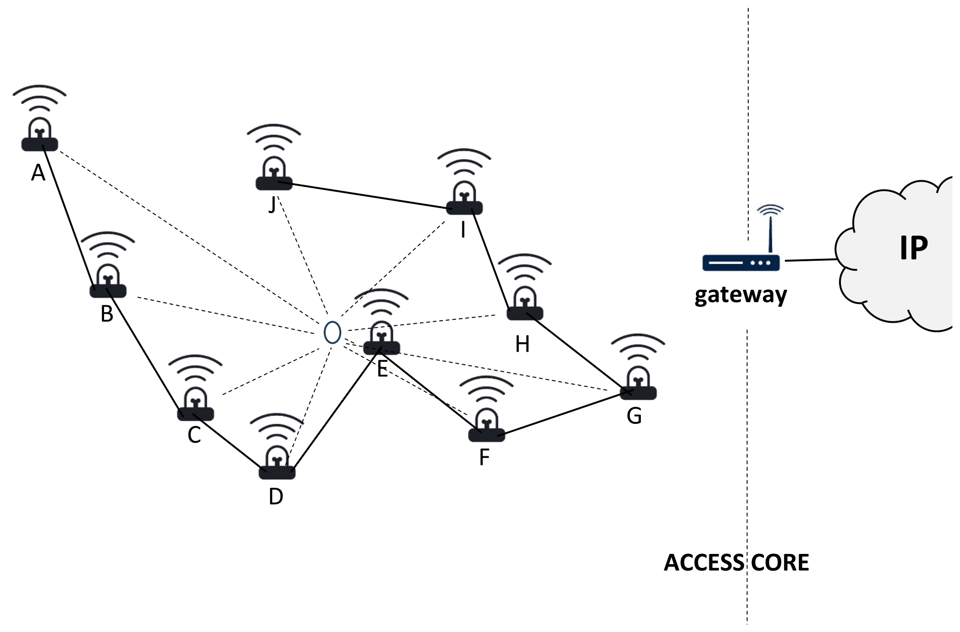


Figure 8: Prediction Service Clustering

4.2 Predition Information Collection

The interaction between devices is carried out in rounds of limited time duration. Devices take turns to perform analytics such that if device A is the predictor in one round, device B will be the predictor the following round.

FIGURE 10 shows an example of a round where device E is the predictor. All devices receive an index based on their location in the device chain. In the first stage, even-position located devices request readouts (by means of CoAP) to their odd-position located neighbors. This enables aggregation because devices in even-position locations have their own readouts as well as the readouts of their neighbors (which are removed from the chain). In the next stage, again, all remaining devices receive an index based on their location in the device chain. Then even-position located devices request readouts (by means of CoAP) to the odd-position located neighbors. This scheme continues until only the predictor and no more than two devices are left. At this point, the predictor requests the readouts from the remaining devices and performs analytics. If needed, the predictor forwards the knowledge to the gateway for wider distribution. There are two exceptions to this mechanism: (1) if a device doesn't have a neighbor and (2) if the predictor ends up in one of stages in an odd-position location. If a device doesn't have a neighbor, it doesn't perform any aggregation and skips to the next stage. If the predictor is located in an odd position, it must perform aggregation.

FIGURE 11 shows the algorithm. After the execution of these steps, the predictor can run analytics on the aggregate information to extract knowledge and perform actuation and forward the knowledge to the gateway. Actuation involves the predictor transmitting an actuation command to a designated actuator.

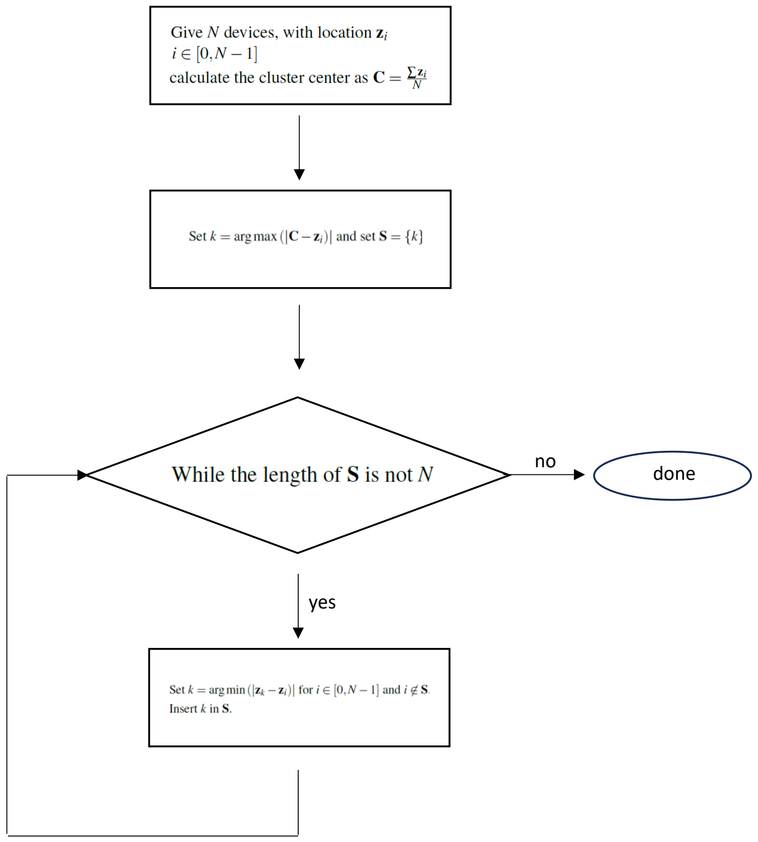

Give N devices, with location $\mathbf{z}_i$
$i \in [0, N-1]$
calculate the cluster center as $\mathbf{C} = \frac{\sum \mathbf{z}_i}{N}$
Set $k = \arg\max(|\mathbf{C} - \mathbf{z}_i)|$ and set $\mathbf{S} = \{k\}$
While the length of $\mathbf{S}$ is not N
no
done
yes
Set $k = \arg\min(|\mathbf{z}_k - \mathbf{z}_i)|$ for $i \in [0, N-1]$ and $i \notin \mathbf{S}$.
Insert k in $\mathbf{S}$.

Figure 9: Prediction Service Clustering Algorithm

5. EVALUATION RESULTS AND DISCUSSION

This Section evaluates the algorithm presented in Section 4 by taking into account two main factors: (1) latency and (2) infrastructure lifetime. The goal is to measure the improvements of these two parameters when compared to a scenario where mist computing is carried out with a single device acting as predictor. In this latter case, the fixed predictor requests by means of CoAP transactions the readouts from devices in the grid.

To compare both architectures, an experimental framework that includes a grid of $N = 16$ devices is deployed. FIGURE 12a shows the legacy topology based on a fixed predictor while FIGURE 12b shows the proposed dynamic mist computing mechanism. For both scenarios, the topology involves a single device that acts as both actuator and sensor, including the fixed predictor that enables the evaluation of the performance of mist computing without the use of the proposed mechanism. To support the corresponding IoT protocols relevant to this architecture, Netualizer is used. Netualizer is a *Protocol Stack Virtualization* (PSV) framework that supports the creation of networking scenarios by enabling the emulation of a myriad of IoT protocol stacks [31, 32].

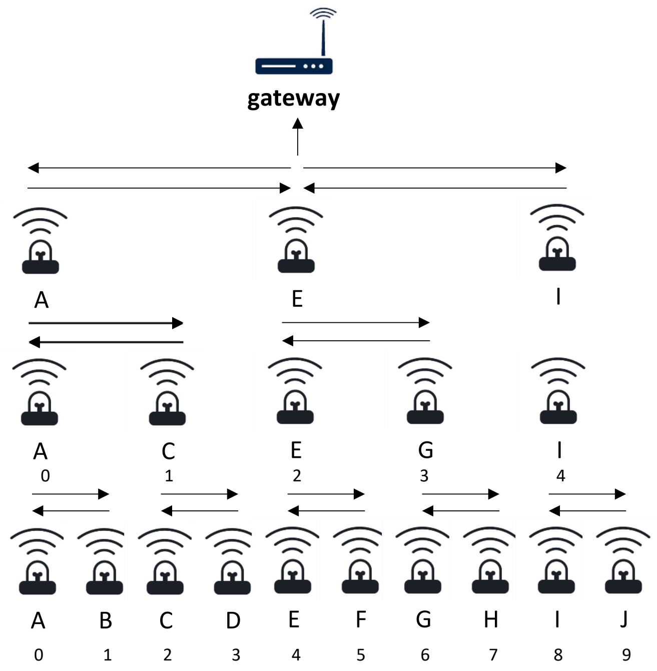


Figure 10: Prediction Information Collection

To summarize, the following two test cases are consider:

- Legacy: A fixed predictor device requests 2-byte readouts from 16 devices every second. The predictions trigger the transmission of a single 1-byte actuation command to the actuator device.
- Proposed: In 5-minute rounds, devices take turns becoming predictors and dynamically request 2-byte readouts from 16 devices every second. The retrieval of the readouts is done in stages intending to preserve battery life and lower overall latency, as described in Section 3. The prediction triggers a single actuation 1-byte command that is transmitted to a device acting as actuator.

To guarantee 32 rounds, the testing run time is set to 160 minutes. The metrics to be measured are: (1) the average latency between the transmission of sensing requests and the actual actuation, and (2) the average device energy consumption during the test. Although the prediction itself is not relevant to these test cases, for the purpose of the experimental framework, the predictor is a 33-node 577-coefficient 2-layer *Artificial Neural Network* that takes 16 emulated sensor temperature readouts as input to produce a single binary output. The hidden layer has 32 nodes. The training is based on 3200 records.

FIGURE 13 shows the actual implementation of the topology in FIGURE 12a in Netualizer *Integrated Development Environment* (IDE). A simple script is used to implement the two test cases above, including the algorithm described in Section 3. Note that in order to support IoT connectivity,

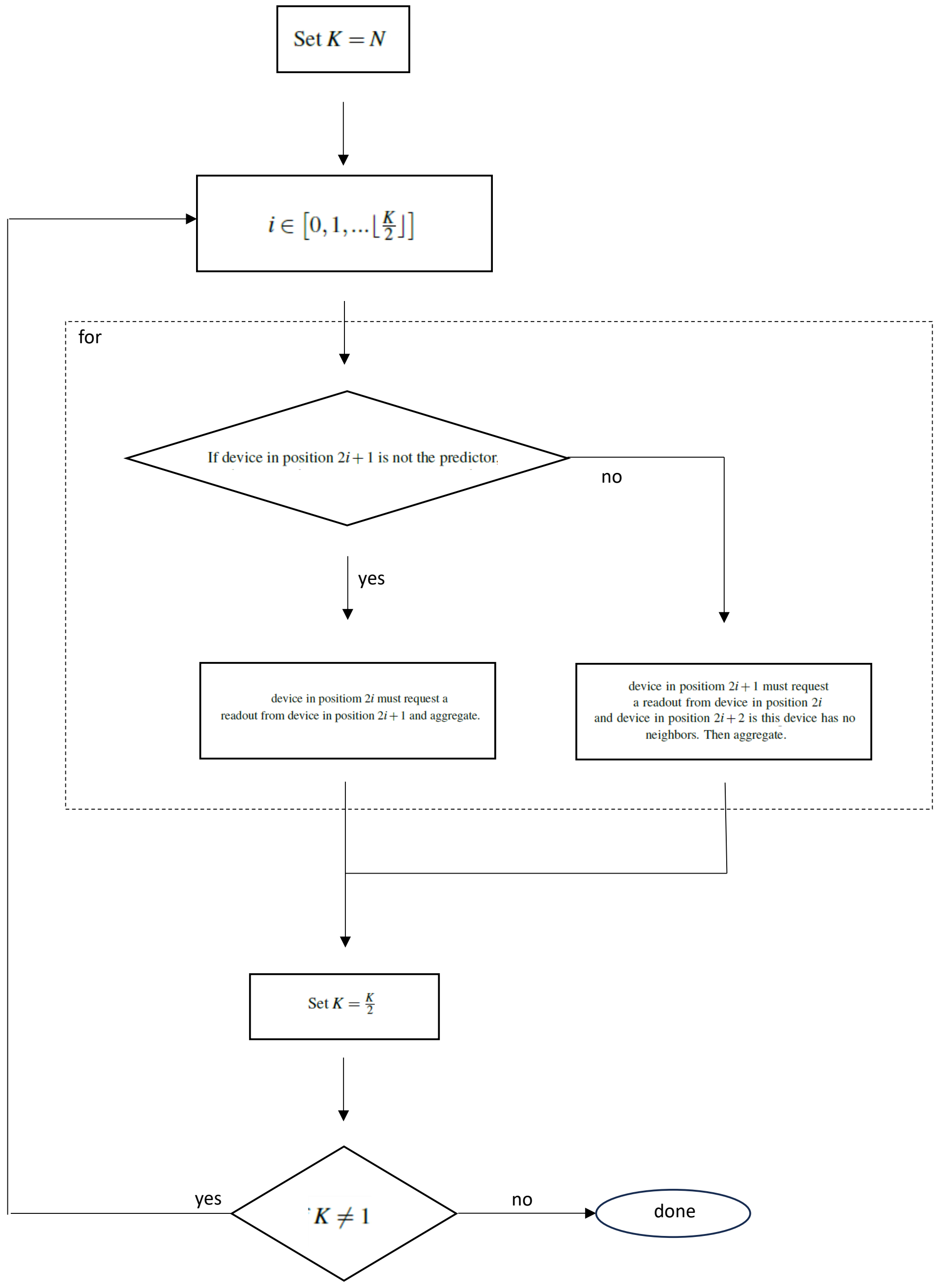
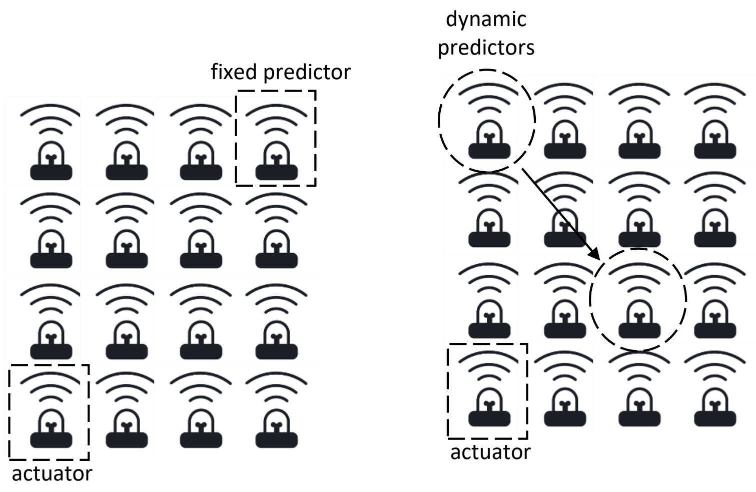

Set $K = N$
$i \in [0, 1, \ldots \lfloor \frac{K}{2} \rfloor]$
for
If device in position $2i+1$ is not the predictor
no
yes
device in positiom $2i$ must request a readout from device in position $2i+1$ and aggregate.
device in positiom $2i+1$ must request a readout from device in position $2i$ and device in position $2i+2$ is this device has no neighbors. Then aggregate.
Set $K = \frac{K}{2}$
yes
$K \neq 1$
no
done

Figure 11: Predition Information Collection Algorithm

Netualizer introduces a wireless channel that follows the Gilbert-Elliot model shown in FIGURE 14. The model, mimicking the behavior of wireless IoT topologies, assumes the existence of two states: (1) good and (2) bad respectively linked to low and high network loss. In the good state, the packet loss probability is e_G, while in the bad state, the packet loss probability is e_B. In addition, the model has two additional parameters: the channel good-to-bad transition probability p and the channel bad-



(a) Fixed Predictor (b) Dynamic Predictors

Figure 12: Experimental Framework

to-bad transition probability α. As extra simplification e_G and e_B such that the network packet loss and loss burstiness are controlled by the parameters p and α respectively.

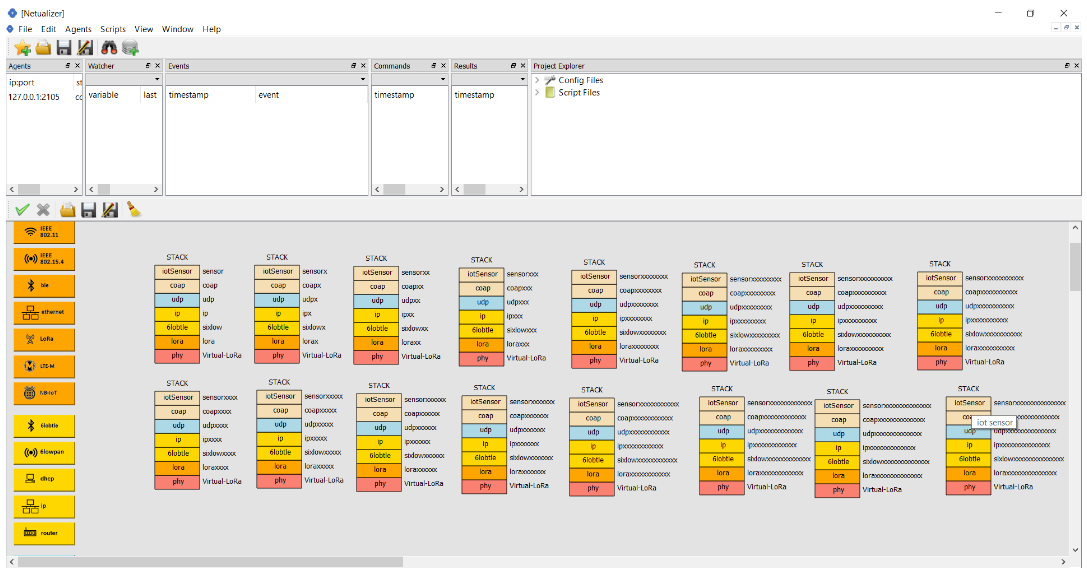


Figure 13: Topology Implementation

FIGURE 15 and FIGURE 16 show the peak energy consumption as a function of network packet loss, for cases of low and high packet loss burstiness. Similarly, FIGURE 17 and FIGURE 18 depict average sensing/actuation latency also as a function of network packet loss. All plots compare the performance of the legacy and proposed mechanism. The legacy mechanism involves a centrally located predictor that interacts with all devices and the actuator. The proposed mechanism dynam-

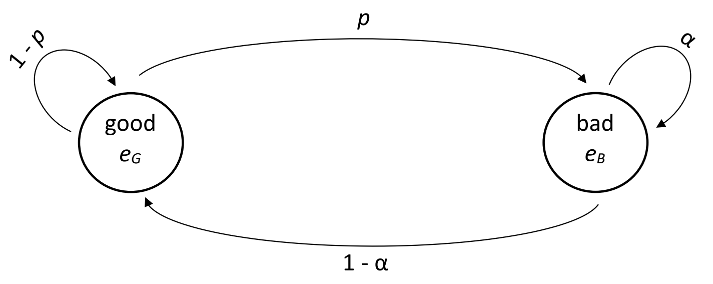


Figure 14: Gilbert-Elliot Channel Model

ically selects the predictor and distributes the energy consumption throughout the network. Still, the actuator is in the same location for both cases. In all cases, low and high burstiness respectively correspond to $\alpha = 0.3$ and $\alpha = 0.9$ in the Gilbert-Elliot model. Note that the network packet loss varies from $p = 0$ to $p = 0.1$.

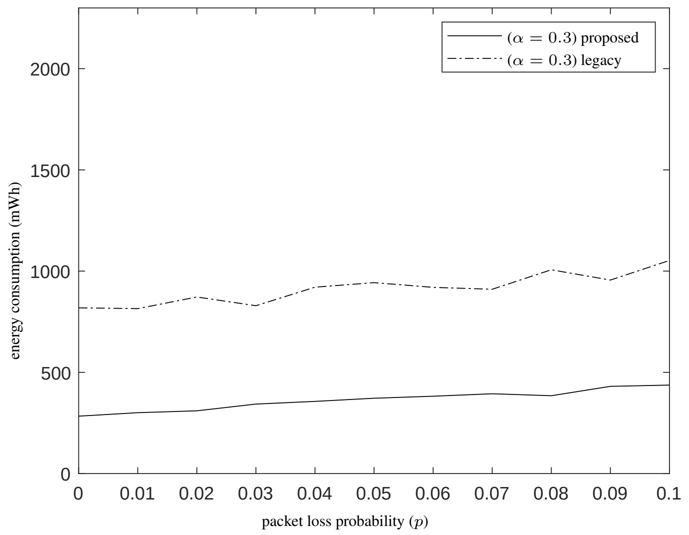


Figure 15: Peak Energy Consumption (Low Packet Loss Burstiness)

For low packet loss burstiness, the effect of network packet loss is minimal on peak energy consumption for both the proposed and the legacy mechanism. The proposed mechanism shows, on average, around a third the peak energy consumption of the legacy mechanism. This ratio is maintained even for high packet loss burstiness. Note that the energy consumption measured is that of the device that consumes the most energy. This device acts as a bottleneck and limits the lifetime of the network. Because the proposed mechanism dynamically changes the distance between sensors and actuators, the energy associated with CoAP level retransmissions is significantly lowered. Moreover, because the predictor is not fixed, energy consumption is not concentrated on a single device. Similarly,

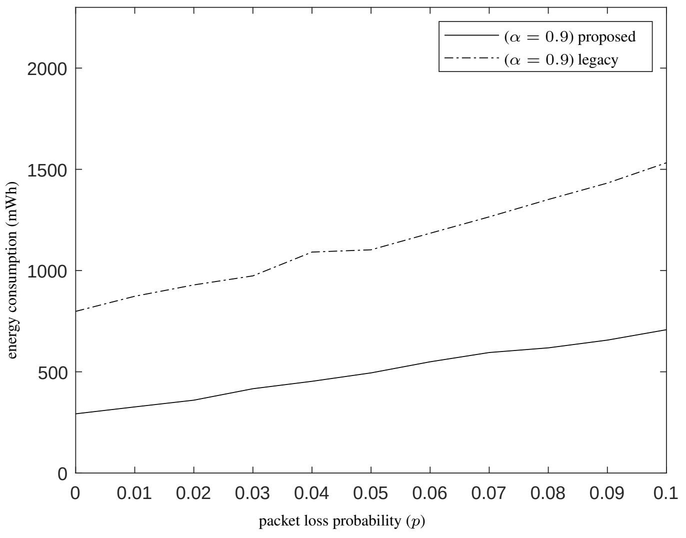


Figure 16: Peak Energy Consumption (High Packet Loss Burstiness)

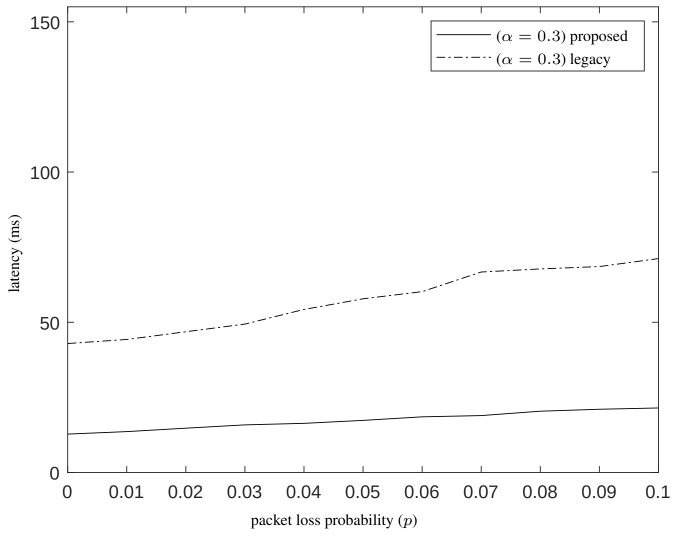


Figure 17: Sensing/Actuation Latency (Low Packet Loss Burstiness)

the proposed mechanism lowers the sensing/actuation latency considerably because, again, it dynamically changes the distance between the predictor and the devices, minimizing the effect of loss and limiting retransmissions that typically increase latency. This is true for both levels of packet loss burstiness, although for high packet loss burstiness the effect is much more considerable on the legacy mechanism. On average, the latency is lowered to a fourth when relying on the proposed mechanism.

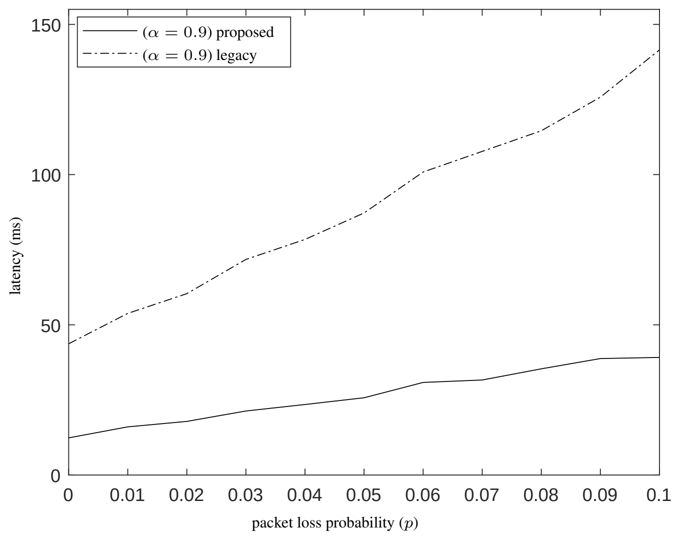


Figure 18: Sensing/Actuation Latency (High Packet Loss Burstiness)

6. CONCLUSION AND FUTURE WORK

Mist computing is key to supporting a myriad of IoT initiatives, including Industrial IoT and Connected Health. It is particularly important in the context of remote sensing scenarios where devices are scattered over a sensor field. IoT standards like LoRa and CoAP enable the connectivity that additionally supports actuation. When comparing the performance of traditional fixed-location prediction to prediction associated with the dynamic selection mechanism introduced in this paper, the latter exhibits comparably lower latency and lower peak energy consumption. This last factor in particular is crucial to ensuring the maximization of the network lifetime.

Going forward, the infrastructure presented in this paper can be further improved to dynamically reduce the activity of those devices with the lowest energy levels by taking into account their real-time battery charge. Moreover, the mechanism could reduce the activity and importance of the devices for which their sensing readouts have little weight on the prediction.

References

[1] Herrero R. Fundamentals of IoT Communication Technologies. Springer International Publishing. 2021.

[2] Herrero R. Practical Internet of Things Networking. Springer International Publishing. 2023.

[3] Thombre S, Ul Islam RU, Andersson K, Hossain MS. Performance Analysis of an IP Based Protocol Stack for WSNs. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2016:360-365.

[4] Xia F, Kong X, Xu Z. Cyber-Physical Control Over Wireless Sensor and Actuator Networks With Packet Loss. 2010. Arxiv Preprint: https://arxiv.org/pdf/1011.3115

[5] Lavric A, Popa V. Internet of Things and Lora™ Low-Power Wide-Area Networks: A Survey. In 2017 International Symposium on Signals, Circuits and Systems (ISSCS).IEEE. 2017:1-5.

[6] Shanmuga Sundaram JP, Du W, Zhao Z. A Survey on Lora Networking: Research Problems, Current Solutions, and Open Issues. IEEE Commun Surv Tutorials. 2020;22:371-388.

[7] Saari M, bin Baharudin AM, Sillberg P, Hyrynsalmi S, Yan W. LoRa - A Survey of Recent Research Trends. In: 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2018:872–877.

[8] Ayoub W, Mroue M, Nouvel F, Samhat AE. Prévotet, "Towards IP Over Lpwans Technologies: LoRaWAN, DASH7, NB-IoT. In: Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC). 2018;43-47.

[9] Zhao Y, Afzal SS, Akbar W, Rodriguez O, Mo F, et al. Towards Battery-Free Machine Learning and Inference in Underwater Environments. In: Proceedings of the 23rd annual international workshop on mobile computing systems and applications, Rt-PA'22. New York: Association for Computing Machinery. 2022:29-34.

[10] Lee S, Islam B, Luo Y, Nirjon S. Intermittent Learning: On-Device Machine Learning on Intermittently Powered System. Proceedings of the ACM InterAct. 2019;3:1-30.

[11] Xu K, Zhang H, Li Y, Zhang Y, Lai R. An Ultra-Low Power TinyML System for Real-Time Visual Processing at Edge. IEEE Trans Circuits Syst II. 2023;70:2640-2644.

[12] Benninger S, Magno M, Gomez A, Benini L. EdgeEye: A Long-Range Energy-Efficient Vision Node for Long-Term Edge Computing. In: Tenth International Green and Sustainable Computing Conference (IGSC); 2019:1-8.

[13] Viswanatha V, Ramachandra AC, Prasanna RR, Kakarla PC, Simha PV, et al. Implementation of Tiny Machine Learning Models on Arduino 33 BLE for Gesture and Speech Recognition. EasyChair Preprint: https://easychair.org/publications/preprint_open/4MBq

[14] Saffari A, Tan SY, Katanbaf M, Saha H, Smith JR, et al. Battery-Free Camera Occupancy Detection System. In: Proceedings of the 5th international workshop on embedded and mobile deep learning, EMDL'21. New York: Association for Computing Machinery. 2021:13-18.

[15] Giordano M, Mayer P, Magno M. A Battery-Free Long-Range Wireless Smart Camera for Face Detection. In: Proceedings of the 8th international workshop on energy harvesting and energy-neutral sensing systems, ENSsys'20. New York: Association for Computing Machinery. 2020:29-35.

[16] Jokic P, Emery S, Benini L. Battery-Less Face Recognition at the Extreme Edge. In: 19th IEEE International New Circuits and Systems Conference (NEWCAS). 2021:1-4.

[17] Maeng K, Colin A, Lucia B. Alpaca: Intermittent Execution Without Checkpoints. Proceedings of the ACM Program. Association for Computing Machinery. 2017;1:1-30.

[18] Hester J, Storer K, Sorber J. Timely Execution on Intermittently Powered Batteryless Sensors. In: Proceedings of the 15th ACM conference on embedded network sensor systems, SenSys'17. New York: Association for Computing Machinery; 2017,

[19] Yıldırım KS, Majid AY, Patoukas D, Schaper K, Pawelczak P, et al. Ink: Reactive Kernel for Tiny Batteryless Sensors. In: Proceedings of the 16th ACM conference on embedded networked sensor systems, SenSys'18. New York: Association for Computing Machinery; 2018:41-53.

[20] Yang F, Thangarajan AS, Ramachandran GS, Joosen W, Hughes D. Astar: Sustainable Energy Harvesting for the Internet of Things Through Adaptive Task Scheduling. ACM Trans Sens Netw. 2021;18:1-34.

[21] Islam B, Kevin J, Porter ED, Xiaofan J, Sridhar Duggirala P. Scheduling Tasks on Intermittently Powered Real-Time Systems. 2021. Available PDF URL: https://cdr.lib.unc.edu/downloads/rn301938b?locale=en

[22] https://www.rfc-editor.org/rfc/rfc7252.txt

[23] Shelby Z, Bormann C. 6LOWPAN: The Wireless Embedded Internet. Wiley Publishing. 2010.

[24] Herrero R. RTP Transport in IoT MQTT Topologies. Internet Things Cyber-Phys Syst. 2023;3:37-44.

[25] https://www.rfc-editor.org/info/rfc6762

[26] https://www.rfc-editor.org/info/rfc6763

[27] https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf

[28] https://www.rfc-editor.org/rfc/rfc4919.txt

[29] Herrero R. MQTT-SN, CoAP, and RTP in Wireless IoT Real-Time Communications. Multimedia Syst. 2020;26:643-654.

[30] Jara AJ, Martinez-Julia P, Skarmeta A. Light-Weight Multicast DNS and DNS-SD (lmDNS-SD): IPV6-Based Resource and Service Discovery for the Web of Things. In: Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing; 2012:731-738.

[31] Herrero R. Protocol Stack Virtualization Support in IoT. Trans Emerg Telecommun Technol. 2021;32:e4340.

[32] https://www.l7tr.com