PLA-MUX: Multiplexing Piece-wise Linear Approximations on Edge-assisted Sensor Networks

Xenophon Kitsios
Department of Informatics
Athens University of Economics and Business
Athens, Greece
xkitsios@aueb.gr

Panagiotis Liakos
Department of Informatics
Athens University of Economics and Business
Athens, Greece
panagiotisliakos@aueb.gr

Katia Papakonstantinopoulou
Department of Informatics
Athens University of Economics and Business
Athens, Greece
katia@aueb.gr

Yannis Kotidis
Department of Informatics
Athens University of Economics and Business
Athens, Greece
kotidis@aueb.gr

***Abstract*—Dense sensor networks produce high-frequency data while operating under tight resource constraints, making efficient transmission a central challenge. Piece-wise Linear Approximation (PLA) mitigates this by replacing raw samples with linear segments under a bounded error, but transmitting these segments as-is fails to exploit substantial redundancy within each sensor's stream and across sensors. In this work we propose PLA-MUX, a sensor–edge framework that compresses PLA segments within and across sensors to exploit shared structure. PLA-MUX quantizes starting values, groups segments with intersecting slopes, and encodes identifiers and timestamps using mixed-radix representation combined with delta encoding. These techniques exploit structural similarities, such as shared ranges, correlated trends, and bounded timestamp windows, to substantially reduce communication overhead while preserving PLA fidelity. The resulting representation is compact, streaming-friendly, and cloud-decodable.**

***Index Terms*—Sensor networks, Internet-of-Things, Piecewise-Linear-Approximation**

I. INTRODUCTION

Modern data-driven applications increasingly depend on dense sensor networks [1] to monitor and respond to physical environments in real time. Industrial IoT systems monitor vibration, temperature, and pressure to detect anomalies [2]. Smart cities collect environmental and mobility data to support utility services and infrastructure maintenance [3], [4]. Smart agriculture systems continuously track soil moisture and microclimate conditions to enable informed decisions by farmers [5]. These applications produce time series measurements that must be sent to remote cloud platforms for storage, analysis, and decision-making [6]. However, sending raw sensor readings is often not practical. Nodes often have limited resources, wireless bandwidth is restricted, and communication consumes a lot of energy [7]. As a result, modern edge computing architectures increasingly use intermediate edge nodes to preprocess, filter, and compress sensor data before sending it to the cloud.

A key observation is that precise sensor measurements are rarely necessary. Physical signals are inherently noisy, and many downstream tasks, like trend detection, threshold monitoring, or anomaly detection, perform well with controlled approximations. Previous studies have shown that approximate sensing and lossy compression can greatly reduce communication costs while maintaining application-level accuracy [8], [9]. This leads to the use of compact representations that summarize time series data with bounded error guarantees. Among the available techniques, Piece-wise Linear Approximation (PLA) [10] stands out as a particularly effective choice. It offers interpretable linear segments, supports strict error thresholds, and can be computed in a streaming manner with minimal memory and computation. This makes it ideal for embedded sensor platforms.

While PLA can be applied directly to the measurements of individual sensors, transmitting PLA segments independently from each node still incurs substantial overhead when many sensors operate concurrently. Our work identifies additional opportunities to consolidate and compress these PLA segments both at the sensor level, before they are sent to the edge, and at the edge devices, where segments from multiple sensors are accumulated and further compacted prior to cloud transmission. A typical PLA segment contains a measurement identifier, a starting timestamp, an intercept, and a slope. When sensors monitor related physical processes—such as temperature across nearby locations or humidity within a greenhouse—the resulting PLA segments often share structural regularities: starting values fall within comparable ranges, slopes reflect similar trends or seasonal patterns, and timestamps evolve predictably. Exploiting these shared patterns enables significantly greater compression than treating each sensor stream in isolation.

In this work, we introduce PLA-MUX, a Multiplexed Piece-wise Linear Approximation framework running on edge-assisted sensor networks that exploits shared regularities across

PLA segments from multiple sensor streams. PLA-MUX quantizes segment intercepts based on the application's error threshold, resulting in a small set of discrete starting values that many segments share and can therefore be represented jointly. It further groups segment slopes that fall within common trend ranges and encodes measurement identifiers and timestamps using *mixed radix representations* that take advantage of their natural correlations. Together, these techniques allow sensor and edge nodes to multiplex many PLA segments into a compact, unified representation that can be transmitted efficiently and decoded without loss in the cloud. The result is a lightweight, streaming-friendly compression method built for edge computing environments that significantly reduces communication overhead while preserving the fidelity guarantees of PLA.

II. PRELIMINARIES

A. Data Collection Infrastructure

We consider a sensor network in which individual sensor nodes collect numerical measurements over time. While sensors may also produce non-numerical observations—such as classification labels generated by local AI models—the processing of such data is orthogonal to the techniques studied in this work.

Each sensor may generate one or more types of measurements. We expect that measurements collected by nearby nodes are similar in nature; for example, multiple sensors may report values for the same physical quantities, such as temperature or humidity. Our objective is to reduce the overall data volume by identifying and exploiting correlations both within the measurements generated by a single node and across measurements produced by different nodes.

To uniquely identify each measurement originating from a particular node, we assume the existence of a unique measurement identifier. Each reported data point is therefore represented as a tuple (m_i, t_i, v_i), where m_i denotes the measurement identifier, t_i the timestamp (or epoch identifier) at which the measurement was collected, and v_i the associated numerical value. Thus, all values corresponding to measurement identifier m_i form a time series $TS_i = \langle (t_1, v_1), (t_2, v_2), \ldots \rangle$.

B. Piece-wise Linear Approximation

Directly transmitting raw, atomic measurements places excessive strain on sensor nodes, consumes significant energy (a critical concern for battery-powered devices), and can easily saturate the available network bandwidth in dense deployments. To mitigate these issues, sensors typically perform local processing to filter out unnecessary transmissions. A simple approach is to report only those values that deviate significantly from the last transmitted measurement. However, such threshold-based filtering is of limited use when measurements follow a trend, for example, a monotonic increase or decrease, because each new value may still trigger a transmission. A more effective strategy is to exploit linear patterns in the data using a well-studied technique known as Piece-wise Linear Approximation (PLA).

Using PLA, a time series can be represented as a sequence of linear segments (v_i, a_i, t_i), where v_i is the starting value of the segment, t_i is its starting timestamp, and a_i is the slope. Measurements within this segment of the time series can be approximated via linear interpolation: $v(t) = v_i + a_i \cdot t$.

For applications that require adherence to a strict maximum error threshold ϵ, extensive research has addressed how to construct such segments while minimizing their total number. Fortunately, efficient streaming algorithms exist [11]–[14] that can solve the error-bounded version of PLA even under tight memory and processing constraints.

While PLA significantly reduces transmission costs compared to sending raw measurements, its true value emerges when it is used as a foundation to unlock additional optimizations. Specifically, the linear segments produced by PLA often exhibit strong similarities due to correlations present in the underlying measurements both within a single sensor's time series and across different sensors in the network. By identifying and exploiting these recurring segment patterns, additional compression opportunities become available.

III. OUR MULTIPLEXING PLA FRAMEWORK

A. Overview

The PLA-MUX data-collection pipeline consists of operators running on both sensor nodes and edge devices, each designed to reduce the volume of data transmitted across the network—from sensors to edge devices and from edge devices to the cloud. Each sensor node executes two functions: (i) a modified PLA procedure that consolidates consecutive measurements into linear segments, and (ii) a multiplexing operator MUX that exploits commonalities among these segments by merging their descriptions. A second instance of MUX runs at edge nodes, where segments originating from multiple sensors are further consolidated, yielding additional space savings and reducing upstream communication overhead.

B. Maximum Error and Latency Constraints

In our setting, we first process measurements *locally* at sensor nodes to convert them into linear segments. Each node enforces a maximum error threshold ϵ for this conversion on each individual measurement. This threshold may be globally defined by the application or may vary depending on factors such as the measurement type, the node's location, or the time period. In addition, we assume a maximum latency threshold L that must be respected when transmitting measurements. This constraint is essential for streaming applications that need to react quickly to changes in the data.

C. Adapting PLA to our setting

Each sensor generates groups of segments by executing a $PLA(\epsilon, L)$ subroutine that processes its local readings. Although the internal details of this procedure are not essential to our framework, certain properties are desirable. In particular, the $PLA(\epsilon, L)$ subroutine should operate in a

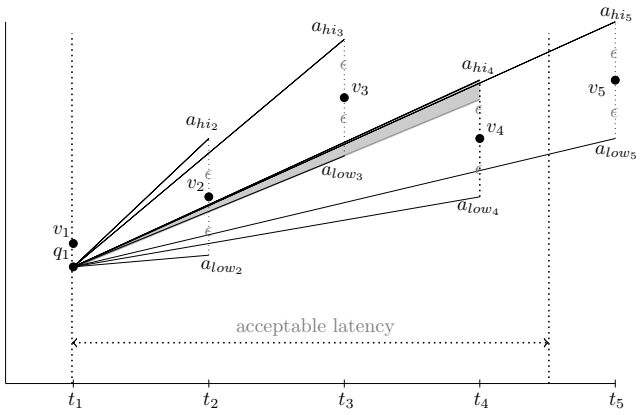

Fig. 1. $PLA(\epsilon, L)$ subroutine visualization.

streaming fashion and require minimal processing and memory resources. Among the available techniques in the literature, our implementation adopts the SWING [11] algorithm, as it has very low resource requirements and has been shown to perform well in practice. More precisely, the algorithm performs incremental calculations, meaning it only processes the new data point rather than recomputing over the entire dataset. To satisfy the latency requirement, we modify the PLA process to forcibly terminate a segment once the latency limit L is reached, regardless of whether the error threshold has not yet been exceeded. A second modification is that, whereas SWING closes a segment by selecting a final slope a_i from the permissible range of interpolating lines, we instead keep this choice open and retain the minimum and maximum allowable slopes, a_{low} and a_{hi}, deferring the final slope selection to the edge device running MUX. A third modification is that the starting value of each line segment is quantized. This idea was originally proposed in [15] to enable sharing of starting-point values across different PLA segments. The specific quantization function is orthogonal to the design of our algorithm. Following [15], we employ a simple linear quantization scheme for the starting values, defined as $q_i = \lfloor v_i/\epsilon \rfloor$.

A visual explanation is given in Figure 1. Point v_1 is first quantized to q_1. Then, the PLA process adds points v_2, v_3, and v_4 by adjusting the slopes a_{low} and a_{hi}. While Figure 1 shows that there exist lines that could also approximate v_5, the process is terminated before reaching this point, as the latency limit is exceeded prior to t_5.

The final output of the $PLA(\epsilon, L)$ subroutine is a sequence of segments s_i, each represented as a tuple $s_i = (q_i, a_{low_i}, a_{hi_i}, t_i)$, where these fields uniquely identify the quantized starting value of the segment, the range of permissible slopes for the interpolation line and its starting timestamp.

D. Batching Segments on Sensors

The multiplexing operator MUX running on each sensor, collects the local segments generated by $PLA(\epsilon, L)$ into a working set of segments $S = \{s_i\}$, ordered by their starting value timestamp, t_i. The operator maintains the minimum timestamp:

$$t_{\min} = \min_{s_i \in S} t_i$$

to enforce the latency L. This timestamp defines the start of the current batching interval. As new segments arrive, they are appended to S without reordering. When the condition:

$$\text{now} - t_{\min} \geq L$$

is satisfied, all segments currently accumulated in S are multiplexed (as described next) before being transmitted to the edge device. After emission, the working set S is cleared and the batching process restarts with the next arriving segment.

TABLE I
NOTATION USED IN THE MULTIPLEXING OF SEGMENTS.

Symbol	Meaning
S	Set of segments to be encoded
q_i	Quantized starting value of segment s_i
Q	Set of distinct quantized values in S
S_q	Subset of segments with quantized value q
a_{low_i}, a_{hi_i}	Lower/upper bounds of the feasible slope interval of s_i
$\mathcal{G}_q = \{G_{q,1}, \ldots, G_{q,K_q}\}$	Slope-based groups within S_q
$G_{q,j}$	Group of segments with mutually intersecting slope intervals
$\tilde{a}_{q,j}$	Representative slope chosen for group $G_{q,j}$
m_i, t_i	Measurement ID and timestamp of segment s_i
Min_m, Max_m	Minimum/maximum measurement ID in S
B	Radix for multiplexing: $B = Max_m - Min_m + 1$
$mux_{m,t}(s_i)$	Multiplexed integer encoding of (m_i, t_i)
$M_{q,j}$	Sorted (delta-encoded) sequence of multiplexed values for $G_{q,j}$
$\mathcal{E}_q$	Encoded representation of all groups in S_q
$\mathcal{E}_{mux}$	Final multiplexed representation of S

E. The Multiplexing Operator

The multiplexing operator MUX merges the descriptions of several segments into a single, compact representation. It operates both on sensor nodes—where it condenses the data before transmission to edge devices—and on edge nodes, which apply an additional layer of compression before forwarding the data to the cloud. Figure 2 provides a high-level overview of how the segments in set S (that is the input to the operator) are processed. Table I collects the notation used throughout this section.

The encoding process transforms the working set of segments S through three successive stages: 1) partitioning by quantized starting values, 2) grouping by intersecting slope intervals, and 3) multiplexing timestamp–identifier pairs into compact integer sequences. The resulting structure is a nested collection of encoded groups, each carrying only the minimal information required for reconstruction. The key objective of the multiplexing task is to reduce the memory footprint required to represent the segments in S using the $\mathcal{E}_{mux}$ encoding.

$$S \longrightarrow \underbrace{\{S_q\}_{q\in Q}}_{\text{quantized partitions}} \longrightarrow \underbrace{\{G_{q,j}\}}_{\text{slope-based groups}} \longrightarrow \underbrace{\{M_{q,j}\}}_{\text{multiplexed \& delta-encoded}} \longrightarrow \underbrace{\mathcal{E}_{mux}}_{\text{final multiplexed representation}}$$

Fig. 2. Processing pipeline for multiplexing PLA segments.

1) Partitioning on quantized starting values: Quantizing the starting PLA values q_i restricts them to a finite set of discrete levels, increasing the likelihood that multiple segments share identical values. Thus, a first step is to partition the segments in S according to their corresponding q_i values. Let Q be the set of distinct quantized values in S:

$$Q = \{\, q_i \mid s_i \in S \,\}$$

For each $q \in Q$, we define the subset $S_q = \{\, s_i \in S \mid s_i[0] = q \,\}$. Then $\{S_q\}_{q\in Q}$ forms a partition of S. This partitioning scheme enables us to transmit a single starting value q_i for each partition, reducing the communication overhead.

2) Grouping by intersecting slope intervals: Each segment s_i in S_q is characterized by an interval $[a_{low_i}, a_{hi_i}]$ of permissible slopes. Instead of selecting a distinct slope value a_i for each segment we seek the minimum number of distinct slope values that collectively cover all segments in S_q. This is equivalent to partitioning an *interval graph*, whose edges denote overlapping intervals, into the minimum number of *cliques* and is solved in $O(n \log n)$ time [16] as follows. We first sort the segments in S_q in increasing order of a_{low_i} values and then perform a forward scan to form a list of groups of mutually intersecting segments. Formally, let the sorted sequence be $s_{(1)}, s_{(2)}, \ldots, s_{(n_q)}$, such that $a_{\text{low}_1} \leq a_{\text{low}_2} \leq \cdots \leq a_{\text{low}_{\text{n}_\text{q}}}$. We construct a sequence of groups $\mathcal{G}_q = (G_1, G_2, \ldots, G_K)$, where each $G_k \subseteq S_q$ is a maximal set of segments whose slope intervals have non-empty common intersection. This forward-pass processing of the segments in S_q results in the minimum number of groups, K [16]. Each group G_j is associated with an interval range $[a_{g_{low_j}} \ldots a_{g_{hi_j}}]$ where $a_{g_{low_j}} = max_{s_i \in G_j}(a_{low_i})$ and $a_{g_{hi_j}} = min_{s_i \in G_j}(a_{hi_i})$. A single representative slope for all segments in a group G_j is chosen as the midpoint of the group's common feasible interval $\frac{a_{g_{low_j}} + a_{g_{hi_j}}}{2}$, when the operator is running on an edge node.

The effect of this multiplexing scheme, which partitions the segments of S_q into groups G_j, is that we transmit to the cloud only one slope per group, instead of transmitting n_q distinct slope values, i.e., one slope for each segment in S_q. Since the number of groups satisfies $K \leq n_q$, our approach results in a potentially substantial reduction in the number of slope values that must be communicated.

3) Multiplexing timestamps and measurement ids: When MUX runs on an edge node, it receives updates from a small group of nearby sensors. Due to the latency constraint, the reported timestamps are expected to fall within a small time window restricted by L. A naive representation of the measurement ids m_i and the timestamps t_i would require transmitting two separate numbers for each segment. However, given their bounded ranges, we can instead employ a mixed-radix representation inspired by [17].

Let Min_m (respectively Max_m) denote the minimum (respectively maximum) value of m_i within S and define:

$$B = (Max_m - Min_m + 1)$$

Then each pair (m_i, t_i) can be compactly encoded as a single integer:

$$mux_{m,t} = B \times t_i + m_i$$

In the cloud the original values can be recovered by applying

$$m_i = Min_m + (mux_{m,t} \bmod B) \quad \text{and} \quad t_i = \left\lfloor \frac{mux_{m,t}}{B} \right\rfloor$$

PLA-MUX sorts segments within each group G_j according to their $mux_{m,t}$ values. The resulting sorted sequence is then delta-encoded.

F. Final Multiplex Representation

We represent the processed segments in S as:

$$\mathcal{E}_{mux} = \{\, \mathcal{E}_q \mid q \in Q \,\},$$

where each $\mathcal{E}_q$ is the collection of encoded groups for partition S_q:

$$\mathcal{E}_q = \{\, E_{q,j} \mid j = 1, \ldots, K_q \,\}.$$

Each encoded group $E_{q,j}$ is a triple:

$$E_{q,j} = \big(q,\ \tilde{a}_{q,j},\ M_{q,j}\big),$$

where $\tilde{a}_{q,j}$ is the representative slope of group $G_{q,j}$ and

$$M_{q,j} = \big(\, \mu_{q,j}^{(1)}, \mu_{q,j}^{(2)}, \ldots, \mu_{q,j}^{(n_{q,j})} \,\big)$$

is the delta-encoded sequence of multiplexed m_i and t_i identifiers associated with the segments in $G_{q,j}$.

IV. EXPERIMENTAL EVALUATION

In this section, we first describe the dataset used in our experiments. We then evaluate the performance of our approach and examine how effectively PLA-MUX compresses data compared to existing methods under different latency and maximum error threshold ϵ constraints.

A. Experimental Setup and Data

We evaluate the performance of PLA-MUX using real-world measurements from the "IR Biological Temperature" dataset [18], that has also been used in earlier studies [9], [19]. Biological, i.e., surface, temperature is measured using infrared (IR) sensors deployed within the soil array and at multiple heights along the tower infrastructure. The sensors report measurements with a precision of two decimal places from 47 sites with a total of 192 sensors, illustrated in Figure 3. Assuming that each site is equipped with a single

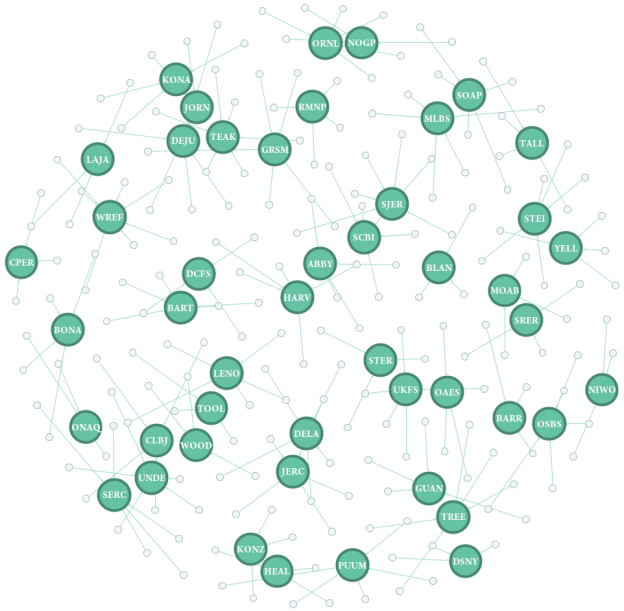

Fig. 3. The topology of our dataset [18].

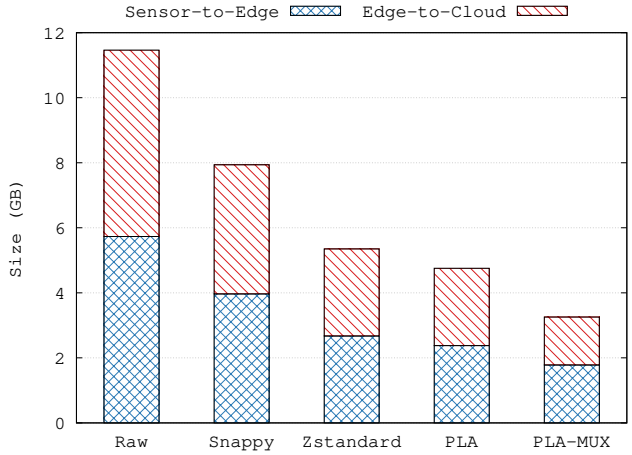

Fig. 4. Data transmission volume (Sensor → Edge → Cloud). PLA-MUX reduces the amount of data transmitted at both the sensor nodes and the edge devices, achieving substantially lower communication overhead compared to alternative compression methods.

edge device, the average number of sensors per edge node is approximately four. For stress-testing purposes, we set a uniform error tolerance of $\epsilon = 0.01°C$ across all sensors in our experiments, unless specified otherwise.

Experiments were conducted using a custom-built Java-based network simulator designed to model configurable sensor-node and edge-device topologies. We implemented PLA by adapting the SWING algorithm to honor the latency constraint, and PLA-MUX, which integrates the $PLA(\epsilon, L)$ segmentation procedure introduced in this work with the MUX operator. To improve compression efficiency of timestamp representations, we adopt integer-based encoding schemes for both PLA and PLA-MUX. For PLA, timestamps are compressed using delta encoding followed by Variable-Byte encoding, reducing redundancy in consecutive temporal values while maintaining lightweight decoding overhead.

The PLA-MUX encoding scheme builds upon the approach presented in [20], which we further enhance by integrating FastPFOR128 [21] for block-based integer compression. FastPFOR128 operates on fixed-size blocks of 128 integers and therefore requires sufficiently large batches of timestamps to be effective. Such batch sizes are naturally available in PLA-MUX, where multiplexing aggregates time series streams from multiple sensors. In contrast, PLA processes sensors independently and does not multiplex streams, resulting in smaller timestamp batches that are insufficient to efficiently leverage FastPFOR128.

We also report results using two general-purpose compression algorithms, Snappy [22] and Zstandard [23]. Although they are less effective at reducing transmission volume, they serve as useful reference points that highlight the additional savings achieved by our PLA-MUX framework.

B. Volume transmitted across network tiers

Figure 4 illustrates the reduction in data volume achieved by PLA-MUX across the different network tiers, specifically from the sensors to the edge and from the edge to the cloud. Unlike the baseline methods, which transmit an equal amount of data across both network tiers, PLA-MUX performs additional data reduction at the edge. The raw data generated by the sensors is 5.7 GB per hop, resulting in a total transmission of 11.5 GB by the time it reaches the cloud. Applying lossless compression reduces this total volume to 7.9 GB for Snappy and 5.3 GB for Zstandard. By introducing a small error tolerance ($\epsilon = 0.01°C$), the PLA algorithm manages to increase space savings by an additional 0.6 GB compared to the best lossless algorithm, reaching a total of 4.8 GB.

In contrast, PLA-MUX requires only 1.8 GB for the sensor-to-edge tier, a 69% reduction over raw data at the first hop. Furthermore, due to the additional MUX operation at the edge devices, this volume is further reduced to 1.5 GB for the edge-to-cloud transmission. This results in a total of 3.3 GB, which is 34% better than PLA and 39% better than Zstandard.

C. Compression efficiency vs latency

Figure 5 illustrates how compression efficiency evolves under different latency constraints. The compression gains of all algorithms increase as the allowable latency grows, because additional latency enables batching and the processing of larger groups of measurements. PLA-MUX achieves space savings ranging from 51% to 72% compared to raw data as the latency threshold increases.

PLA-MUX consistently attains higher compression ratios, yielding a substantially smaller data footprint. This advantage grows with increasing latency, as larger latency budgets allow more PLA segments to accumulate in the MUX pipeline of Figure 4. For very small latency budgets, the compression

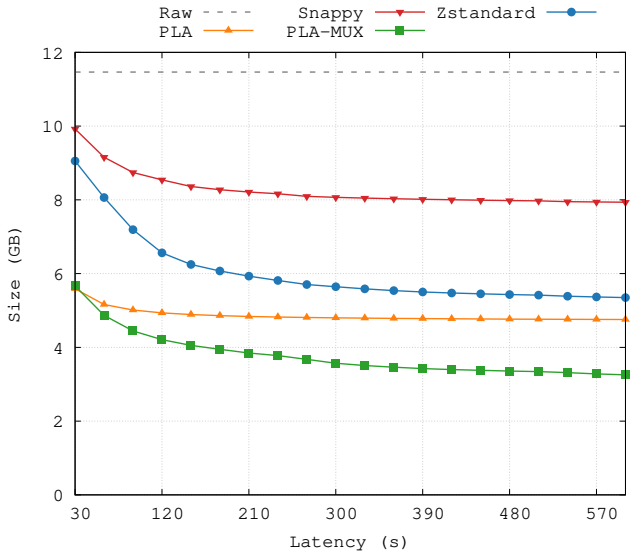

Fig. 5. Comparison of total data transmission (Sensor → Edge → Cloud) versus latency. The graph shows that although both PLA and PLA-MUX outperform standard compression schemes, PLA-MUX consistently delivers greater data reduction across all evaluated latency settings.

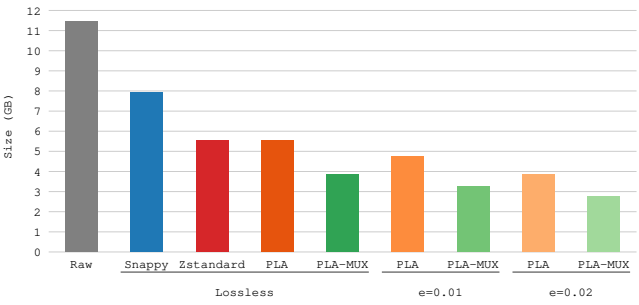

Fig. 6. Comparison of total data transmission (GB) under varying error thresholds. The advantage of PLA-MUX becomes increasingly pronounced as the maximum error threshold ϵ decreases, since more line segments can be jointly encoded. Notably, both PLA and PLA-MUX can operate as lossless compressors, achieving substantial space savings over general purpose compression algorithms. As the allowable error bound increases, the compression gains become even larger.

efficiency of PLA and PLA-MUX converges. Tight delay constraints permit only short segments, leaving little room to exploit long-term temporal structure or inter-sensor similarity. Consequently, the benefit of multiplexing becomes negligible in these extremely low-latency regimes.

D. Compression Efficiency vs. Error Threshold

Figure 6 presents the space requirements for different values of ϵ. As the dataset reports measurements with a precision of two decimal places, setting $\epsilon = 0.005$ guarantees lossless representation, since the maximum allowable deviation remains below the smallest measurable unit. Under this strict error tolerance, both PLA and PLA-MUX generate a large number of short segments. However, this setting particularly benefits PLA-MUX: the increased number of segments creates greater opportunities for multiplexing and joint encoding. As a result, PLA-MUX achieves its largest improvement over PLA in this setting, providing a compression ratio that is more than 30% better. For larger values of ϵ, the permitted approximation error increases, enabling longer segments and reducing the overall data volume for both approaches. Overall, PLA-MUX consistently outperforms PLA across all tested thresholds. The relative performance gain ranges from 28% to 32%, demonstrating that the advantages of multiplexing persist across both strict and relaxed error settings.

V. Related Work

Bounded-error Piece-wise Linear Approximation has been extensively studied in the literature [11]–[14]. Our techniques were motivated by recent algorithms [15], [20] that explore merging segment descriptions to reduce representation size, but these algorithms treat each time series independently and do not exploit cross-sensor regularities.

In-network aggregation techniques reduce communication by combining partial results along routing paths, enabling efficient computation of aggregates [24]–[32], and surveys have documented the trade-offs between accuracy, energy consumption, and resilience [33]–[35]. These approaches, however, are designed to compute global aggregates and intentionally discard per-sensor detail. PLA-MUX instead preserves the structure of each approximate time series and performs a structural multiplexing of PLA segments rather than a statistical aggregation, making it complementary to traditional in-network aggregation frameworks. Sampling can substantially reduce data transmission [36], but it does not provide the deterministic guarantees required in this work. Data sharing and neighborhood awareness are also prominent approaches for reducing transmission overhead [37]–[39].

Sensor and Edge computing research has emphasized pushing computation closer to data sources to reduce latency, bandwidth usage, energy consumption and cloud dependence [40]–[44]. Systems for IoT and mobile environments commonly incorporate filtering, compression, outlier-detection or lightweight analytics at intermediate nodes, and several studies evaluate compression techniques for edge devices under realistic constraints [45]–[50]. These works focus on collecting historical data and do not provide strict freshness guarantees like PLA-MUX.

VI. Conclusions

In this work, we presented PLA-MUX, a novel multiplexing-based compression framework for multi-sensor edge environments. By aggregating time series streams, PLA-MUX jointly encodes data exploiting spatiotemporal correlations that standard sensor-wise approaches cannot leverage. Our evaluation on real-world datasets demonstrates that PLA-MUX consistently achieves higher compression ratios across a range of latency and error thresholds, reducing both sensor-to-edge and edge-to-cloud traffic without sacrificing accuracy.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128601003024

[2] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis, "TACO: tunable approximate computation of outliers in wireless sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, A. K. Elmagarmid and D. Agrawal, Eds. ACM, 2010, pp. 279–290. [Online]. Available: https://doi.org/10.1145/1807167.1807199

[3] A. R. M. Forkan, Y. Kang, F. M. Carrillo, A. Banerjee, C. McCarthy, H. Ghaderi, B. G. S. Costa, A. Dawod, D. Georgakopoulos, and P. P. Jayaraman, "AIoT-CitySense: AI and IoT-driven city-scale sensing for roadside infrastructure maintenance," *Data Sci. Eng.*, vol. 9, no. 1, pp. 26–40, 2024. [Online]. Available: https://doi.org/10.1007/s41019-023-00236-5

[4] R. Hildebrant, R. A. Bhope, S. Mehrotra, C. Tull, and N. Venkatasubramanian, "DIM-SUM: dynamic imputation for smart utility management," *Proc. VLDB Endow.*, vol. 18, no. 11, pp. 4451–4464, 2025. [Online]. Available: https://www.vldb.org/pvldb/vol18/p4451-hildebrant.pdf

[5] L. Sciullo, A. Trotta, S. Bosi, L. Bononi, and M. D. Felice, "Digital shadow sensor framework for smart agriculture: Time series prediction through data segmentation and clustering," in *21st IEEE Consumer Communications & Networking Conference, CCNC 2024, Las Vegas, NV, USA, January 6-9, 2024*. IEEE, 2024, pp. 745–751. [Online]. Available: https://doi.org/10.1109/CCNC51664.2024.10454662

[6] K. Echihabi and T. Palpanas, "Scalable analytics on large sequence collections," in *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*, 2022, pp. 5–8.

[7] R. Wohlers, N. Trigoni, R. Zhang, and S. Ellwood, "TwinRoute: Energy-efficient data collection in fixed sensor networks with mobile sinks," in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, 2009, pp. 192–201.

[8] K. Hishida, C. Liu, J. Paparrizos, and A. J. Elmore, "Beyond compression: A comprehensive evaluation of lossless floating-point compression," *Proc. VLDB Endow.*, vol. 18, no. 11, pp. 4396–4409, 2025. [Online]. Available: https://www.vldb.org/pvldb/vol18/p4396-hishida.pdf

[9] P. Liakos, K. Papakonstantinopoulou, and Y. Kotidis, "Chimp: Efficient lossless floating point compression for time series databases," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 3058–3070, 2022.

[10] S. H. Cameron, "Piece-wise linear approximations," IIT Research Institute, Chicago, IL, Computer Sciences Division, Tech. Rep., 1966.

[11] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, and W. Zwaenepoel, "Online piece-wise linear approximation of numerical streams with precision guarantees," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 145–156, 2009. [Online]. Available: http://www.vldb.org/pvldb/vol2/vldb09-573.pdf

[12] J. O'Rourke, "An on-line algorithm for fitting straight lines between data ranges," *Commun. ACM*, vol. 24, no. 9, pp. 574–578, 1981. [Online]. Available: https://doi.org/10.1145/358746.358758

[13] Q. Xie, C. Pang, X. Zhou, X. Zhang, and K. Deng, "Maximum error-bounded piecewise linear representation for online stream approximation," *VLDB J.*, vol. 23, no. 6, pp. 915–937, 2014. [Online]. Available: https://doi.org/10.1007/s00778-014-0355-0

[14] R. Duvignau, V. Gulisano, M. Papatriantafilou, and V. Savic, "Streaming piecewise linear approximation for efficient data management in edge computing," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*. ACM, 2019, pp. 593–596.

[15] X. Kitsios, P. Liakos, K. Papakonstantinopoulou, and Y. Kotidis, "Sim-Piece: Highly accurate piecewise linear approximation through similar segment merging," *Proc. VLDB Endow.*, vol. 16, no. 8, pp. 1910–1922, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p1910-liakos.pdf

[16] U. I. Gupta, D. T. Lee, and J. Y. Leung, "Efficient algorithms for interval graphs and circular-arc graphs," *Networks*, vol. 12, no. 4, pp. 459–467, 1982. [Online]. Available: https://doi.org/10.1002/net.3230120410

[17] W. K. Ng and C. V. Ravishankar, "A tuple model for summary data management," in *Sixth International Conference on Management of Data, COMAD 1994, Windsor Manor Sheraton & Towers, Bangalore, India, December 19-21*, 1994.

[18] National Ecological Observatory Network (NEON), "IR biological temperature (DP1.00005.001)," 2026. [Online]. Available: https://data.neonscience.org/data-products/DP1.00005.001/RELEASE-2026

[19] P. Liakos, K. Papakonstantinopoulou, T. Bruineman, M. Raasveldt, and Y. Kotidis, "How to make your duck fly: Advanced floating point compression to the rescue," in *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*. OpenProceedings.org, 2024, pp. 826–829. [Online]. Available: https://doi.org/10.48786/edbt.2024.80

[20] X. Kitsios, P. Liakos, K. Papakonstantinopoulou, and Y. Kotidis, "Flexible grouping of linear segments for highly accurate lossy compression of time series data," *The VLDB Journal*, vol. 33, no. 5, p. 1569–1589, Jul. 2024. [Online]. Available: https://doi.org/10.1007/s00778-024-00862-z

[21] D. Lemire and L. Boytsov, "Decoding billions of integers per second through vectorization," *Softw. Pract. Exper.*, vol. 45, no. 1, p. 1–29, Jan. 2015. [Online]. Available: https://doi.org/10.1002/spe.2203

[22] Google, "Snappy," https://github.com/google/snappy, 2011, accessed: 2026-03-03.

[23] Y. Collet and Facebook, "Zstandard," https://github.com/facebook/zstd, 2016, version 1.5.x, Accessed: 2026-03-03.

[24] A. Deligiannakis, Y. Kotidis, V. Stoumpos, and A. Delis, "Collection trees for event-monitoring queries," *Inf. Syst.*, vol. 36, no. 2, pp. 386–405, 2011. [Online]. Available: https://doi.org/10.1016/j.is.2010.08.003

[25] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Processing approximate aggregate queries in wireless sensor networks," *Inf. Syst.*, vol. 31, no. 8, pp. 770–792, 2006. [Online]. Available: https://doi.org/10.1016/j.is.2005.02.001

[26] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, p. 131–146, Dec. 2003. [Online]. Available: https://doi.org/10.1145/844128.844142

[27] A. Deligiannakis, V. Stoumpos, Y. Kotidis, V. Vassalos, and A. Delis, "Outlier-aware data aggregation in sensor networks," in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds. IEEE Computer Society, 2008, pp. 1448–1450. [Online]. Available: https://doi.org/10.1109/ICDE.2008.4497585

[28] P. Andreou, D. Zeinalipour-Yazti, G. Samaras, and P. K. Chrysanthis, "A network-aware framework for energy-efficient data acquisition in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 46, pp. 227–240, 2014. [Online]. Available: https://doi.org/10.1016/j.jnca.2014.08.010

[29] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "TiNA: a scheme for temporal coherency-aware in-network aggregation," in *Proceedings of the Third ACM International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE 2003, September 19, 2003, San Diego, California, USA*. ACM, 2003, pp. 69–76. [Online]. Available: https://doi.org/10.1145/940923.940937

[30] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Bandwidth-constrained queries in sensor networks," *VLDB J.*, vol. 17, no. 3, pp. 443–467, 2008. [Online]. Available: https://doi.org/10.1007/s00778-006-0016-z

[31] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *SenSys*, 2004.

[32] A. Deligiannakis and Y. Kotidis, "Data reduction techniques in sensor networks," *IEEE Data Eng. Bull.*, vol. 28, no. 1, pp. 19–25, 2005. [Online]. Available: http://sites.computer.org/debull/A05mar/kotidis.ps

[33] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, 2002.

[34] L. Wang, L. Chen, and D. Papadias, "Query processing in wireless sensor networks," in *Managing and Mining Sensor Data*, C. C. Aggarwal, Ed. Springer, 2013, pp. 51–76. [Online]. Available: https://doi.org/10.1007/978-1-4614-6309-2_3

[35] N. C. Hubig, A. Züfle, T. Emrich, M. A. Nascimento, M. Renz, and H. Kriegel, "Continuous probabilistic sum queries in wireless sensor networks with ranges," in *Scientific and Statistical Database Management - 24th International Conference, SSDBM 2012, Chania, Crete, Greece, June 25-27, 2012. Proceedings*, ser. Lecture Notes in Computer Science, A. Ailamaki and S. Bowers, Eds., vol. 7338. Springer, 2012, pp. 96–105. [Online]. Available: https://doi.org/10.1007/978-3-642-31235-9_6

[36] F. Safaridis, E. Katsarou, and S. Hadjiefthymiades, "Reducing IoT data for highly efficient cloud storage," in *The 16th International Conference on Ambient Systems, Networks and Technologies (ANT 2025)*, vol. 257. Elsevier, 2025, pp. 182–189. [Online]. Available: https://doi.org/10.1016/j.procs.2025.03.026

[37] A. Deligiannakis and Y. Kotidis, "Detecting proximity events in sensor networks," *Inf. Syst.*, vol. 36, no. 7, pp. 1044–1063, 2011. [Online]. Available: https://doi.org/10.1016/j.is.2011.03.004

[38] S. Rabinia, N. Didar, M. Brocanelli, and D. Grosu, "Algorithms for data sharing-aware task allocation in edge computing systems," *IEEE Trans. Parallel Distributed Syst.*, vol. 36, no. 1, pp. 15–28, 2025. [Online]. Available: https://doi.org/10.1109/TPDS.2024.3486184

[39] Y. Kotidis, "Snapshot queries: Towards data-centric sensor networks," in *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, K. Aberer, M. J. Franklin, and S. Nishio, Eds. IEEE Computer Society, 2005, pp. 131–142. [Online]. Available: https://doi.org/10.1109/ICDE.2005.134

[40] G. Chatzimilioudis, N. Mamoulis, and D. Gunopulos, "A distributed technique for dynamic operator placement in wireless sensor networks," in *Eleventh International Conference on Mobile Data Management, MDM 2010, Kanas City, Missouri, USA, 23-26 May 2010*. IEEE Computer Society, 2010, pp. 167–176. [Online]. Available: https://doi.org/10.1109/MDM.2010.16

[41] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, 2016.

[42] A. Michailidou, A. Gounaris, M. Symeonides, and D. Trihinas, "EQUALITY: quality-aware intensive analytics on the edge," *Inf. Syst.*, vol. 105, p. 101953, 2022. [Online]. Available: https://doi.org/10.1016/j.is.2021.101953

[43] M. Satyanarayanan, "The emergence of edge computing," *Computer*, 2017.

[44] D. Trihinas, M. Symeonides, J. Georgiou, G. Pallis, and M. D. Dikaiakos, "Energy-aware streaming analytics job scheduling for edge computing," in *IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2023, Naples, Italy, December 4-6, 2023*. IEEE, 2023, pp. 161–168. [Online]. Available: https://doi.org/10.1109/CloudCom59040.2023.00036

[45] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis, "In-network approximate computation of outliers with quality guarantees," *Inf. Syst.*, vol. 38, no. 8, pp. 1285–1308, 2013. [Online]. Available: https://doi.org/10.1016/j.is.2011.08.005

[46] X. Li, R. Chen, K. Patel, and A. Singh, "Evaluating compression techniques for IoT edge devices," in *IoTDI*, 2020.

[47] S. Lin, V. Kalogeraki, D. Gunopulos, and S. Lonardi, "Efficient information compression in sensor networks," *Int. J. Sens. Networks*, vol. 1, no. 3/4, pp. 229–240, 2006. [Online]. Available: https://doi.org/10.1504/IJSNET.2006.012038

[48] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Dissemination of compressed historical information in sensor networks," *VLDB J.*, vol. 16, no. 4, pp. 439–461, 2007. [Online]. Available: https://doi.org/10.1007/s00778-005-0173-5

[49] N. Giatrakos, A. Deligiannakis, M. N. Garofalakis, and Y. Kotidis, "Omnibus outlier detection in sensor networks using windowed locality sensitive hashing," *Future Gener. Comput. Syst.*, vol. 110, pp. 587–609, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2018.04.046

[50] T. Szalapski and S. Madria, "Toward energy efficient multistream collaborative compression in wireless sensor networks," in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2014, Miami, Florida, USA, October 22-25, 2014*. ICST / IEEE, 2014, pp. 124–133. [Online]. Available: https://doi.org/10.4108/icst.collaboratecom.2014.257289