Sensing at the Edge: Location-Aware Caching

Federico Trombetti
Computer Science Department
Sapienza University of Rome
Rome, Italy
trombetti@di.uniroma1.it

Novella Bartolini
Computer Science Department
Sapienza University of Rome
Rome, Italy
bartolini@di.uniroma1.it

Salvatore Pontarelli
Computer Science Department
Sapienza University of Rome
Rome, Italy
pontarelli@di.uniroma1.it

***Abstract*—Sensing has become a fundamental component of modern network infrastructures, powering applications from environmental monitoring to industrial automation, and bridging the gap between digital systems and the physical world. Given the large amount of data generated by these systems, it is important to find strategies that are able to intelligently manage the flow of data between all interconnected devices, in order to reduce the utilized bandwidth to a minimum.**

In this paper, we study the use of Wireless Edge Caching (WEC) techniques to reduce latency and optimize bandwidth and energy consumption of sensing applications. We study a specific sensing scenario where a network of wireless sensors spread over a vast territory continuously collects data, part of which must be transmitted to a central base station. To aid and enhance the performance of this application, we employ the use of WEC techniques. Since the frequency of data queries is related to the sensors' location, we introduce a novel caching algorithm, Closest In Farthest Out (CIFO), tailored to this scenario. CIFO is able to capture the characteristics of the sensing application and perform cache eviction decisions accordingly. We demonstrate the performance of our solution by implementing it in a simulated environment and comparing it to traditional caching strategies, showing how our solution is able to outperform the other strategies under multiple settings and different metrics.

I. Introduction

In recent years, sensing technologies have become the backbone of modern data-driven systems, enabling the continuous monitoring of the physical environment without the use of human personnel, through a wide variety of sensors. The application of these systems can span multiple types of applications, from smart cities, industrial automation, to environmental monitoring; collecting a large amount of real-time data, which can then be used to detect patterns and make meaningful predictions. As the demand for these systems increases, so does their complexity, along with the challenges to achieve efficient data transmission, storage, and processing. In particular, the amount of data that is collected by a network of sensors distributed in the field can be several TB/day when multiple cameras recording video are deployed for use cases such as smart city traffic monitoring, [7], [8], seismic or acoustic monitoring arrays [5], [6] or wildlife tracking [2].

This work is supported by the European Commission's Horizon Europe, Smart Networks and Service Joint Undertaking, research and innovation program under grant agreement #101139282, 6G-SENSES project and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001-program "RESTART").

In all the above-mentioned use cases, the data collection is often triggered by specific events. Event-triggered recording is a well-known strategy used in sensor networks to reduce data volume, power consumption, and bandwidth usage by recording or transmitting data only when specific conditions or events occur. In particular, when sensors are deployed over a large area, the data collection is localized around the location of the event that triggered the recording and transmission activity. Therefore, after a triggering event, a central base station starts querying data from specific sensors, i.e. the ones localized around the one that triggered the operation. Generally speaking, the events that triggered the transmission activity can be of different types, and in particular, are often moving around the sensor area. Therefore, a generic approach to provide an efficient sensing operation requires that the base station implement a specific algorithm to query interesting data from the sensor network. As a consequence, the single sensor is not aware of the importance of the data that is collected at a specific moment. Furthermore, due to the memory and processing restrictions of the single sensor, it is mandatory to deploy specific edge devices that collect information from the sensors and cache it to provide the base station with the collected information. These nodes act as Wireless Edge Caching (WEC) nodes that not only provide the usual features of latency and energy consumption reduction, but also collect data from multiple sensors and are able to select the most important data to be retained with respect to the requests of the base station. In fact, since the memory available in the edge device is also limited, it is important to understand which data must be retained from the edge device and which can be discarded with a minimum degradation of the information that the sensor network must collect. In this paper, we focus on the specific use case of monitoring and tracking of wildlife [2], even if we believe that most of the discussion presented for this use case can also be applied in different contexts, such as smart city traffic and seismic monitoring arrays. In this context, sensing systems are often deployed in remote and harsh environments in order to monitor animal movements and their behavior. The difficulty that these systems face is the need for a balance between the need for fine-grained data and the limitations of battery life, bandwidth, and intermittent connectivity.

The main contributions of this paper are the following:

978-3-903176-75-1 ©2025 IFIP

1) We investigate a specific sensing use-case scenario related to wildlife tracking, and propose a WEC approach to improve the overall efficiency of the system.
2) We propose a novel caching algorithm: Closest In Farthest Out (CIFO). The proposed algorithm is directly tied to the sensing use case.
3) In a simulated environment, we implement our CIFO caching algorithm, along with other traditional caching strategies such as First In First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), and Random Replacement (RR). We compare their performances under different key performance metrics, based on HIT/MISS ratio, pertaining energy consumption, and end-to-end delay.

II. Related Work

Sensing has become a fundamental component of modern edge and IoT systems, enabling real-time data collection and rapid, responsive decision making in domains such as environmental monitoring, smart cities [7], and wildlife monitoring [1], [2], [4]. Wireless Edge Caching [3], [9] refers to a novel approach to distributed architectures, aimed at storing content at the edge of a wireless network, using devices such as base stations and user terminals. This method is designed to accommodate the recent surge in mobile devices and data-hungry applications. With recent infrastructure advancements, the deployment of WEC techniques has become an appealing solution. Efficiently deploying WEC solutions means tackling multiple challenges to meet specific application requirements. The way caching should behave in this context deeply differs from traditional caching paradigms which tend to retain data based only on past access; effective caching in this scenario must be context-aware and adapt to the characteristics of its application. In the context of sensing, WEC has had limited application. This is mainly due to the stringent requirements of the sensing tasks, where data are typically highly sensitive to time [10], making caching a less attractive solution. However, the amount of data produced by such sensing applications is often extremely high [7], [8]. An example of such requirements is provided by the work of Dyo et al. [2], in which sensing equipment was deployed to study the social co-location patterns of European badgers. In their system, data are organized into multiple tiers and evicted based on application-specific priorities, such as energy availability, data freshness, and communication likelihood. In this paper, we use localization information as input for the eviction policy, showing that this can provide a significant benefit in terms of cache efficiency with respect to the traditional cache algorithms. Differently from previous works, our location-based eviction policy represents a more general and adaptable approach, enabling support for a wider range of applications beyond the specific scenarios considered beforehand.

III. Sensing Use-Case Scenario

This work finds practical relevance in many IoT applications. A relevant example is moving object tracking, particularly in scenarios characterized by limited connectivity, energy constraints, and the need for delay-tolerant data handling. These conditions are often common across many animal tracking deployments, such as that described in [2], which studied the deployment of a wireless architecture to monitor the social behavior of badgers in a dense woodland. Similar challenges also arise in industrial monitoring settings, for example, in large-scale facilities where drones periodically collect sensor data from spatially distributed mobile equipment. In such cases, location-based caching can reduce redundant transmissions by allowing data to be temporarily stored at nodes closest to the source, improving efficiency in delay-tolerant collection scenarios.

Our approach addresses constraints similar to those introduced in [2], including intermittent connectivity, limited energy availability, and tolerance to delayed data delivery, by enabling location-aware adaptive caching at the edge. This improves the efficiency of sensor data handling and extends applicability beyond wildlife tracking to other scenarios that exhibit comparable operational challenges.

A. Problem Formulation

This section provides a formal definition of the sensing use case. The objective is the continuous monitoring of a set of targets, denoted by G. To achieve this, a set of sensing devices S is deployed. Each sensor $s \in S$ is capable of performing sensing operations and broadcasting its collected data within a predefined maximum range Δ_d. Additionally, a set of edge devices E is deployed to interact with the sensors and respond to requests issued by a central Base Station (BS). Sensing operations are conducted continuously and evaluated at discrete, fixed time intervals. At each predefined interval Δ_q, the BS initiates a sensing request targeting a selected entity $g \in G$. We denote as q_g^t the query made by the BS at time t for target g, which is translated into a request to the edge devices for the most recent, non expired data, of the closest sensor to $g \in G$, i.e. $\arg\min_{s \in S} \|\mathbf{s} - \mathbf{g}\|$, where $\mathbf{s}$ and $\mathbf{g}$ indicate the position vectors of sensor s and target g. The request can be fulfilled by any edge device $e \in E$ that has in its cache the requested, non-expired, data. Each sensing data entry is associated with a freshness value that quantifies its temporal validity. Data are considered expired and thus unusable once their freshness value drops below a specified threshold. We consider the problem of optimization in terms of HIT/MISS rate. We define a HIT event when a request from the BS for a target g at time t, q_g^t is satisfied by any caching device $e \in E$, and a MISS event otherwise.

B. Energy Consumption Optimization

HIT and MISS events are directly tied to the energy consumption of the sensing devices. In a context where sensing data is actively requested by the base station, a MISS event corresponds to the sensing devices performing an additional sensing task, thus increasing energy consumption of the system and, for battery-constrained tasks, reducing the overall duration of the sensing application. We want to provide a

caching strategy that manages to reduce sensors' activation to a minimum while maintaining optimal performance.

C. End-to-End Delay Optimization

The increase of MISS events is also perceived in additional delay between the requests from the BS and their completion, as fulfilling a sensing request is bound to take significantly longer time than simply accessing a cached entry, often by orders of magnitude. This added delay negatively impacts application performance, resulting in increased end-to-end latency, and significantly degrading the responsiveness and efficiency of sensing applications, particularly those that rely on timely data for real-time decision-making or control.

IV. Location-Aware Caching Strategy

In this work, we address the problem of content caching in the specific context of wireless sensing applications used to track and monitor moving objects. Unlike general IoT scenarios, sensing tasks often involve geographically distributed sensors whose data relevance depends heavily on spatial and temporal proximity. To reflect this, we propose a novel caching algorithm, Closest In, Farthest Out (CIFO), designed to exploit the spatial locality inherent in sensing data. CIFO makes eviction and caching decisions based on the physical distance between sensors and the locations of recent queries, prioritizing data that are most likely to be reused.

Algorithm 1: CIFO Caching Algorithm

```
1  C_e ← cache table of device e;
2  r^t ← latest request from BS;
3  S ← set of sensors on the field;
4  Event: Data received e_s^t
5      if s ∈ C_e then
6          update entry for sensor s;
7      else
8          if C_e is full then
9              s_farthest ← arg max_{s∈C_e} ‖s − r^t‖;
10             C_e = C_e \ s_farthest;
11         C_e = C_e ∪ e_s^t;
12 Event: Request received r^{t_last}
13     r^t ← r^{t_last};
14     s_closest ← arg min_{s∈S} ‖s − g‖;
15     if s_closest ∈ C_e then
16         return s_closest;
17     else
18         request sensing task to s_closest;
```

We present the CIFO strategy in Algorithm 1. The algorithm is run by a caching device $e \in E$, which keeps in memory its caching table C_e, the latest request from the BS r^t, and a view of the sensors on the field S. The execution of the policy is driven mainly by two events. The first event is triggered when a sensor broadcasts new data (line 4) . This broadcast could either be triggered by a passive sensing activity by the sensor or actively by an edge device previously requesting the sensing data. When the data is received, the edge device checks if it already has old sensing data for that sensor (line 5), and updates it accordingly. If the data is not present, it means that a new entry for it has to be generated. If the cache is full, the device first finds the farthest sensing data from the latest queried position r^t (line 9) and evicts it. The second event is triggered when the edge device receives a query request from the BS (line 12). When this happens, the edge device first updates its last queried position, and then verifies which sensor data is needed to satisfy the BS's request, which is given by the closest sensor (line 14). If the data for that sensor is present in its memory C_e, then it is promptly returned; otherwise, a sensing task is requested to that sensor. The proposed caching solution is more computationally intensive than traditional methods, mainly due to the linear-time calculation of $s_{farthest}$ and $s_{closest}$ with respect to the number of sensors. Optimizing these computations could improve performance, but we leave this for future work.

A. Expanding CIFO Algorithm for Multiple Targets

The CIFO algorithm presented is useful for tracking the position of a single target accurately. When multiple targets need to be tracked, requests for distant targets are likely to result in cache misses, unless some portion of the caching storage is reserved for them. Partitioning the cache is one possible strategy to ensure such coverage. We hereby define the data handling strategy used to support multiple-target scenarios. Note that this introduces some parametrization which is application-specific, and needs to be tuned on a defined use-case scenario, depending on how the main BS is interested in querying the different targets. We partition the storage of an edge device into multiple sections, up to $|G|$, where each section behaves as a single CIFO cache table on its own. The way memory is allocated among these partitions is given by a focus factor $\lambda \in [0, 1]$, which determines the normalized amount of space allocated for the most recent requested target, with the remaining space equally shared between other targets. A λ value of 1 means that all the space is reserved for the latest queried target, and a value of 0 means that the space is evenly distributed among the G targets. Formally, the fraction for the allocated space of the latest target l_s is computed as follows:

$$l_s = \lambda + \frac{1 - \lambda}{|G|}$$

The maximum allocated space for each targets expands and shrinks, following the queries from the BS. This maximum allocated space for each target is considered a "soft" bound, meaning that eviction from other targets happens only when the latest target partition does not have enough space to store the newly received data.

V. Performance Evaluation

We test our new CIFO algorithm starting with a simulation scenario tailored to wildlife tracking use case described in [2]. We initially deploy a single target on the field, which moves in the simulation by progressively choosing a new random destination in the covered area. To assess the effectiveness of the proposed CIFO caching algorithm, we conduct a series of simulations across different scenarios. Our evaluation focuses on key performance metrics such as energy consumption and end-to-end delay. To highlight the advantages of our solution in handling location-sensitive data, we compare CIFO

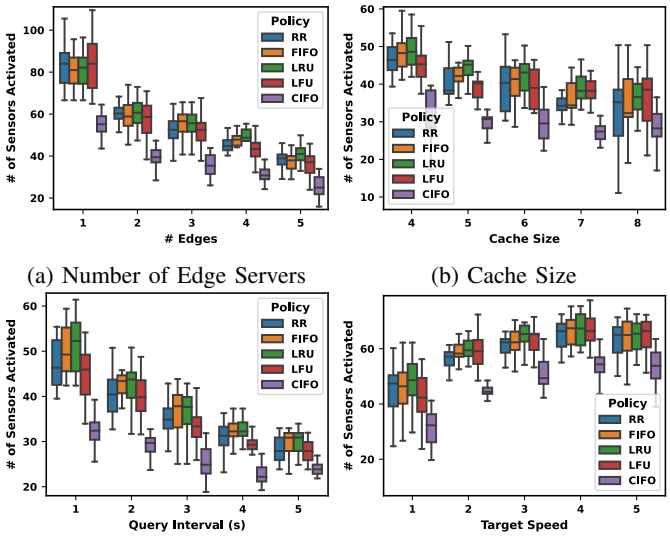
Parameter	Default Value	Varying Range
Sim. Time	240s	N/A
Edge Servers	4	[1,5]
Sensors	40	N/A
Cache Size	4	[4,8]
Query Interval	1s	[1,5]
Target Speed	1 m/s	[1,5]
Freshness Thd.	10s	N/A

TABLE I: Simulation Parameters

against other widely used standard caching strategies: Least Recently Used (LRU), Least Frequently Used (LFU), First In First Out (FIFO), and Random Replacement (RR). Each simulation instance is designed to reflect the dynamics of wireless sensing applications, where sensor-generated data is both continuously produced and queried by a central BS via intermediate edge devices. We vary different factors through our testing instances: the number of edge devices and their cache size, the rate of the BS's queries, and also the speed of the tracked targets. All tests are evaluated on a run of a duration of 4 minutes, starting from the initial deployment of the edge devices with empty cache tables. We perform 4 different set of tests, with each one varying a single parameter. The parameters of the different configurations are reported in Table I. Each bar of the box-plot refers to the distribution of 15 different results.

A. Energy Consumption

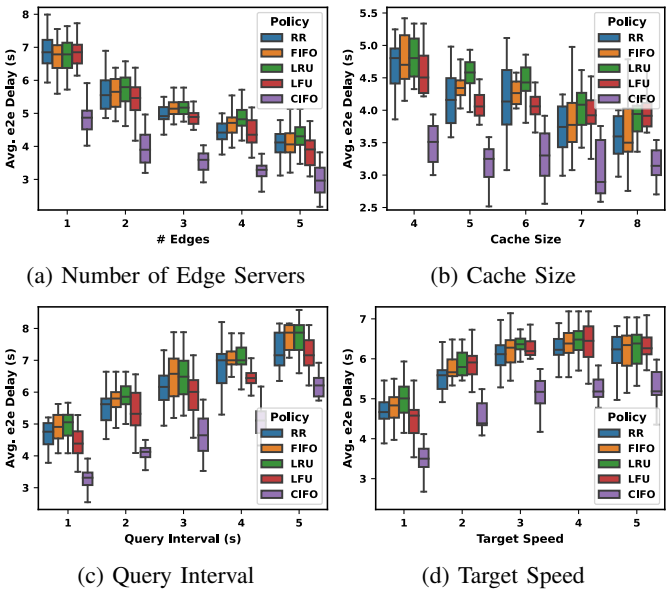
We analyze the performance of all the strategies by looking at the energy consumption of the application. We consider the energy consumption based on the number of times the sensing devices were activated, either by passive sensing, or by an active request triggered by the MISS of a query from the BS. The final results of this test can be seen in Fig. 1, where the CIFO caching strategy consistently achieves lower energy consumption across all evaluation settings compared to the other policies, with a reduction of sensor activation of roughly 30% across all cases.



(a) Number of Edge Servers (b) Cache Size

(c) Query Interval (d) Target Speed

Fig. 1: Performance Evaluation - Energy Consumption



(a) Number of Edge Servers (b) Cache Size

(c) Query Interval (d) Target Speed

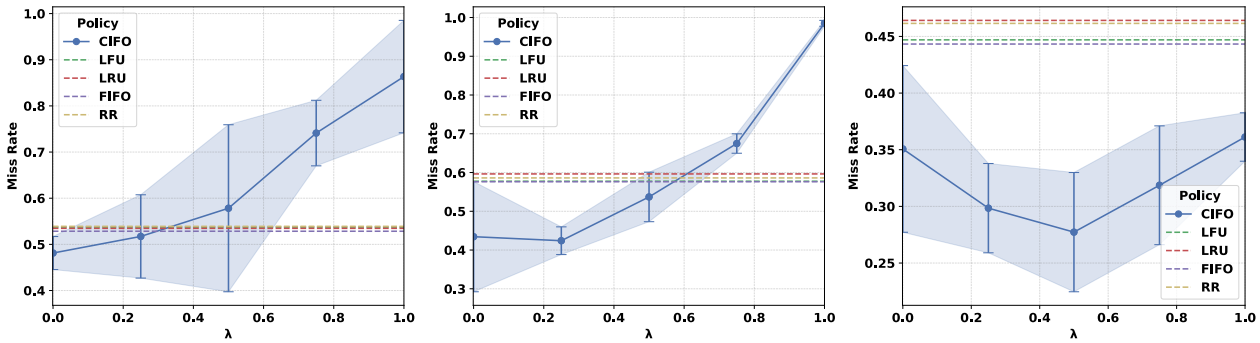
Fig. 2: Performance Evaluation - End-to-End Delay

B. End-to-End Delay

The results for average end-to-end delay are presented in Fig. 2. In this evaluation, cache HIT events are assigned a single second delay, while MISS events incur a delay of 10 seconds, reflecting their significantly higher latency. In line with the energy consumption analysis, CIFO consistently achieves the lowest end-to-end delay across all configurations.

C. Multiple Targets

In this section, we evaluate the performance of the CIFO caching algorithm when monitoring multiple targets, applying the memory partitioning described in Section IV-A. For this test, we deploy 40 sensors and a single edge caching device with a memory size of 16. With these parameters, we study how the variation of the focus value λ influences the caching results, over three different query orders from the base station: i) Random Query: where the BS selects, at each step, a random target, ii) Round Robin Query: where the BS selects targets in a predefined order and in sequence, iii) Focused Query: where the BS selects a target in a round robin fashion and keeps monitoring that target for a duration of 10 queries. We evaluate the performance of these tests based on the raw MISS rate. Due to integer rounding, the size of allocated memory for each target changes only at fixed λ values, which are 0, 0.25, 0.5, 0.75, and 1.0. The results of our tests are shown in Fig. 3. The results of the other baseline caching strategies are displayed with horizontal dashed lines, which denote their average value across all the experiments. Their results are fixed through the execution, as they are not influenced by the focus value λ. For the case of random query (Fig. 3a), we can see that giving equal space to each target yields the best results, as a λ value of 0 means that every target has a reserved space of 4 in the cache of 16 spaces. A similar trend can be observed for Round Robin Query (Fig. 3b), where higher λ values drastically increase the MISS rate. Different results can be seen, as expected, for the



(a) Random Query (b) Round Robin Query (c) Focused Query

Fig. 3: CIFO Performance Evaluation - Varying Focus Value λ

case of Focused Query (Fig. 3c), where the best performance can be seen for larger values of λ; although too large values, 0,75 and 1.0, tend to degrade the effectiveness of the CIFO caching strategy, which however manages to perform better than all the other strategies, even in the worst case. It has to be noted that these results are additionally influenced by the density of the sensors in the covered area, the query rate, and all the configuration variations discussed in the past sections. We omit the results of such tests due to space constraints, highlighting, however, how the choice of an appropriate λ value is important to achieve the optimal performance. As stated before, the choice of the best λ value is application-specific, and therefore should be tuned only at the deployment phase. Applications oriented towards a coverage model are more likely to query different targets during their execution, meaning that a λ value closer to 0 is bound to return better performances; likewise, applications with a more "focused" approach, such as those tracking and monitoring more in-depth single targets at once, are likely to benefit from greater values of λ.

VI. Conclusions and Future Work

In this work, we addressed the problem of efficient data caching in wireless sensing applications, introducing CIFO, a location-aware caching strategy tailored to the spatial characteristics of sensing data. Unlike traditional caching methods, CIFO leverages location awareness to retain data most likely to be reused, improving performance in scenarios where query relevance depends on geographic proximity. Through extensive simulations, we demonstrated that CIFO consistently outperforms standard strategies such as LRU, LFU, and FIFO under multiple evaluation metrics. These results highlight the importance of domain-specific caching strategies in edge-assisted IoT systems, paving the way for more adaptive and efficient architectures in data-intensive sensing environments.

In future work, we aim to investigate the potential of the CIFO algorithm as an effective solution for caching services in mobile environments. Beyond the already mentioned use cases of smart city, traffic monitoring and seismic monitoring that can be easily adapted to the algorithm presented here, we plan to explore use cases where requests originate directly from edge devices, rather than exclusively from a central BS or a main controller. This would involve examining how the CIFO algorithm can be leveraged to optimize caching strategies in decentralized networks, where edge devices such as smartphones, IoT devices, or other mobile nodes dynamically generate and request data. We will investigate use cases where not only targets, but also sensors and edge devices may move, leading to more complex scenarios where an edge device may cover different sensors at any given time, and also cover scenarios where targets may move out of reach of sensors, creating instances where caching would be the only way to retrieve sensing data, and the only solution for a resilient, non-interrupted application.

References

[1] Michael L. Casazza et al. Aims for wildlife: Developing an automated interactive monitoring system to integrate real-time movement and environmental data for true adaptive management. *Journal of Environmental Management*, 345:118636, 2023.

[2] Vladimir Dyo, Stephen A. Ellwood, David W. Macdonald, Andrew Markham, Niki Trigoni, Ricklef Wohlers, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, and Kharsim Yousef. Wildsensing: Design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Trans. Sen. Netw.*, 8(4), September 2012.

[3] Yaru Fu, Yue Zhang, Qi Zhu, Hong-Ning Dai, Mingmei Li, and Tony Q. S. Quek. A new vision of wireless edge caching networks (wecns): Issues, technologies, and open research trends. *IEEE Network*, 38(1):247–253, 2024.

[4] Jarrod C. Hodgson, Shane M. Baylis, Rowan Mott, Ashley Herrod, and Rohan H. Clarke. Precision wildlife monitoring using unmanned aerial vehicles. *Scientific Reports*, 6(1):22574, 2016.

[5] Julio Antonio Jornet-Monteverde, Juan José Galiana-Merino, and Juan Luis Soler-Llorens. Design and implementation of a wireless recorder system for seismic noise array measurements. *Sensors*, 22(21):8103, 2022.

[6] Aliyu Makama, Koojana Kuladinithi, and Andreas Timm-Giel. Wireless geophone networks for land seismic data acquisition: A survey, tutorial and performance evaluation. *Sensors*, 21(15):5171, 2021.

[7] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Sensing as a service model for smart cities supported by internet of things. *European Transactions on Telecommunications*, 01 2014.

[8] Shikhar Suryavansh, Abu Benna, Chris Guest, and Somali Chaterji. A data-driven approach to increasing the lifetime of iot sensor nodes. *Scientific Reports*, 11(1):22459, 2021.

[9] T. X. Vu, E. Bas¸tu˘g, S. Chatzinotas, and T. Q. S. Quek. *Wireless Edge Caching: Modeling, Analysis, and Optimization*. Cambridge University Press, 2021.

[10] Roy D. Yates, Yin Sun, D. Richard Brown, Sanjit K. Kaul, Eytan Modiano, and Sennur Ulukus. Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications*, 39(5):1183–1210, 2021.