future internet

Article

A Context-Aware Edge Computing Framework for Smart Internet of Things

Abdelkarim Ben Sada [1], Abdenacer Naouri [1], Amar Khelloufi [1], Sahraoui Dhelim [2,*] and Huansheng Ning [1]

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
[2] School of Computer Science, University College Dublin, D04 N2E5 Dublin, Ireland
* Correspondence: sahraoui.dhelim@ucd.ie

Abstract: The data explosion caused by the rapid and widespread use of IoT devices is placing tremendous pressure on current communication, computing and storage resources. In an ambient ubiquitous computing environment, taking advantage of the context of the application scenario can significantly improve the system performance of IoT networks. Therefore, in this paper, we propose CONTESS, a context-aware edge computing framework with selective sensing that leverages the context information of the sensed environment to improve its applicability to smart IoT systems. CONTESS is composed of two parts: context management, where context acquisition, modeling and reasoning happens; and selective sensing, where data selection happens. We demonstrate the capabilities of CONTESS in the scenario of a parking management system for a smart city environment. We implement CONTESS using linked data and semantic web technologies. We start by designing an OWL-based ontology and then simulating the proposed scenario using the OMNET++ network simulator along with the Veins framework and SUMO traffic simulator. The results show an improvement compared to traditional sensing methods in both communication overhead and the application results.

Keywords: context-aware computing; edge computing; selective sensing; iot; smart internet of things

check for updates

Citation: Ben Sada, A.; Naouri, A.; Khelloufi, A.; Dhelim, S.; Ning, H. A Context-Aware Edge Computing Framework for Smart Internet of Things. *Future Internet* **2023**, *15*, 154. https://doi.org/10.3390/fi15050154

Academic Editor: Paolo Bellavista

Received: 23 March 2023
Revised: 17 April 2023
Accepted: 20 April 2023
Published: 22 April 2023

1. Introduction

The recent and rapid growth in IoT adoption in various fields such as agriculture, healthcare, smart city and industry is undeniably challenging current communication and computing infrastructures. Huge amounts of sensed raw data require timely transportation, processing and storage, which existing networks are incapable of handling. IoT sensing systems that rely on real-time data processing are becoming an essential part of IoT environments which are the most affected by the witnessed data explosion [1–3].

Typical sensing systems allow applications to access sensors either directly or through middleware. Direct access uses dedicated communication channels between processing centers and sensors. These are normally seen in early sensing systems at the small scale or in applications that require high bandwidth and time-critical data processing capabilities, such as surveillance systems using high definition cameras [4]. Despite the benefits that this type of access offers, such as efficiency and a high level of control over sensors, it requires a high technical level and effort to deploy, in addition to being costly and difficult to scale. Most IoT sensing systems, on the other hand, tend to use middleware frameworks deployed in edge devices to bring processing closer to the sensors for faster responses and serve various purposes such as data and device management [5]. This provides faster response times and helped building heterogeneous, scalable and reliable sensing systems [6]. Many middleware frameworks offer some basic data reduction techniques as an initial step before the main objective of the framework, which has helped alleviate the data explosion issue but to a limited extent. A great amount of middleware and frameworks with varying purposes have been surveyed in [7]. The main take is that most works offer very limited real-time analytics capabilities and very few of them attempted to utilize these analytics results to overcome data explosion problems.

Systems that are capable of acquiring, understanding and recognizing their surrounding environment, in addition to performing actions according to its changes, are considered to be context-aware. With that ability, a context-aware sensing system decides what information and which service should be presented to the user. According to Dey et al. [8], the context is defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application". Such systems are a one step closer to reducing the effects of limited sensing resources. However, their main goal is focused on understanding the context and the reactions to situational changes rather than reducing the amounts of sensed data. Perera et al. [9] and Sezer et al. [10] have conducted comprehensive surveys on context-aware computing in IoT.

Sensing in any environment unavoidably involves capturing extra noise through the sensors, which is irrelevant to the purpose of the application. The raw data can contain temporal and spatial redundancies in addition to the undesired attributes of the real world [11]. Processing all the raw sensed data is impractical and unfeasible, especially with the ever-increasing IoT growth. Preprocessing the raw data with compression and filtering is of little help when it comes to complex environments such as smart cities and smart homes. Adaptive and smart sensing schemes that take advantage of the higher processing capacity that is becoming available at the edge of the network are a necessity [4].

Selective sensing is a sensing scheme in which only the desired and important parts of the generated data are considered for processing and dissemination [12]. This concept is introduced to combat the data explosion issue facing the limited sensing resources by leveraging the currently existing infrastructure. Selective sensing is an attractive concept that is receiving some traction in the research community. Among the numerous advantages it promises are real-time processing, scalable and resilient sensing systems, energy saving and more. However, most existing selective sensing implementations are designed to satisfy a specific set of requirements for certain applications. They fall into one of two major categories, namely passive, user request-based selection or single-goal application-tailored selection. Thus, the need for a general-purpose adaptive selective sensing scheme compatible with all IoT applications is the next significant and urgent step [13].

Context-aware systems offer a great deal of insights and understanding of data which can be exploited to develop a general-purpose selective sensing framework that is aware of its surroundings, giving it the ability to refine the selection criterion and adapt to every situation [10]. In this work, we propose a context-aware selective sensing framework (CONTESS) for IoT to tackle the data explosion issue and alleviate the impact of the huge sensed data on the limited sensing resources, while providing a real-time and scalable services.

The main contribution of this work can be summarized as follows.

- A novel general-purpose context-aware selective sensing framework is proposed. This framework takes advantage of the available context information to improve the selection of important sensed data to reduce the impact of data explosion.
- We demonstrate the applicability of our framework with a parking management system in a smart city using linked data and semantic web technologies.

The rest of this document is organized as follows. Related works are presented in Section 2. The proposed platform is explained in Section 3. A use case scenario is given in Section 4. Implementation and testing results are discussed in Sections 5 and 6, respectively. Finally, this paper is concluded in Section 7.

2. Related Works

In this section, we review the literature focusing on works that leverage context information for smart data selection and selective sensing in general. We organize the related works into three categories. (1) selective sensing; (2) context-based data/device selection; (3) context-aware parking management systems. Finally, we summarize the reviewed works in a table and identify the research gaps.

2.1. Selective Sensing

Inspired by the biological attention, Ning et al. [12] introduced AMiSS, a selective sensing framework. Using four steps consisting of attention building, maintaining, diversion and withdrawal, their work was tested with a video tracking system on a collected video dataset. It has shown that the tracking application has benefited from the attention mechanism which significantly reduced the streamed data size without affecting the accuracy of the system. Their positive results motivated us to employ context awareness for selective sensing and potentially achieve better results. Other forms of selective sensing have been observed in works focused on energy-saving and green computing. This has led to the proposal of many sorts of sensed data reduction methods.

2.2. Context-Based Data/Device Selection

Liu et al. [14] proposed a context-aware sensing and collection (GCSC) scheme for wireless sensor networks to improve data collection accuracy and decrease the data sizes transmitted to the sink. They used a combination of data spatial and temporal correlation analysis along with an event data fusion mechanism during routing. Their techniques were adapted using the context information consisting of the residual energy of nodes.

Ardagna et al. [15] proposed a context-aware approach to support users in selecting optimal configurations for assessing data quality in big data according to a specified goal, such as time, cost minimization or confidence maximization. They tested their approach on data collected from the Curitiba smart city on three scenarios, each corresponding to one goal. Their model showed relatively high precision. Liu et al. [16] introduced a context-aware mobile crowd-sensing system. They collected context information from mobile phones to evaluate the quality of sensed data and estimated the reliability of the devices which are then used to select the suitable devices for the clients. A classification model was used to map the real-time context, which consists of the hardware information and user activity, to the quality of the sensed data.

To address the issue of limited energy resources in sensors used to detect landslides in India, Prabha et al. [17] proposed a context-aware sensing system in place of the existing continuous sensing scheme. Their work mainly consists of two different data collection modules; first, context-aware data management in which the sensors' sampling rate is adjusted according to the network context. The network context concerns events spanning large areas, it includes rainfall, moisture, pore pressure, movement and more information. Second, context-aware energy management in which sensors are activated and deactivated with different sampling rates based on small sensor area context. Compared to the continuous sensing scheme, the two modules extended the lifetime of the sensing system by six times and 20 times, respectively.

A cloud computing model for context-aware Internet of Things services was proposed by Lee et al. [18]. Their design consists of two layers, a cloud control layer which manages cloud resource allocation, service scheduling and service profiles. In addition to a user control layer which manages end-to-end service connections and service contexts. This design allows for service context management using machine learning.

Kavitha et al. [19] proposed a healthcare monitoring architecture to overcome the notification delays. The proposed architecture comprises four modules. First, an IoT module consisting of sensors and actuators. Second, a data preprocessing module which performs data collection and storage and insures redundancy. Third, a context-aware module is responsible for aggregating the appropriate data, extracting the context and sending these the cloud. This module is deployed in two layers, namely a fog layer and a cloud layer. Finally, a decision-making module in which feature extraction and classification are performed using a back-propagation neural network in addition to an adaptive grasshopper optimization algorithm.

Shapsough et al. [20] improved upon learning systems by leveraging the available context information in the learner's environment. They proposed a three-layer architecture: (1) front–end applications which allow the learners and teachers to interact with the system by viewing and posting content; (2) learning edge devices which allow the learners and teachers to interact and retrieve contextual information from their surroundings; and (3) all the content is published to back-end servers which are used to analyze and store the content. Using the constrained application protocol (CoAP) they showed better performance than other protocols.

SmartVent is a real-time context-aware indoor air quality measuring system developed by Lohani et al. [21]. It relies on measuring CO_2 levels, which is considered a tracer gas for air quality where its growth and decay is observed using an Arduino setup and a smartphone with a sensor drone. Their system relies on CO_2 data as context information instead of directly measuring air quality using multiple complex sensors to determine the indoor air quality in rooms and hallways. Their results indicate that CO_2 levels are directly related to air quality which greatly reduces the complexity of air quality-sensing systems and the size of sensed data.

The digital twin is used as context to improve healthcare by Elayan et al. [22]. They proposed an intelligent context-aware healthcare system using the digital twin framework. Using machine learning models trained on ECG data to diagnose heart disease and detect heart problems, their models successfully predicted a heart condition with the help of the digital twin context.

LISA is a context-aware IoT service architecture proposed by Gochhayat et al. [23] for efficient IoT services. LISA attempts to reduce the communication overhead between users and services by understanding the user context and acting autonomously on their behalf. This is performed using what are so-called agents that generate queries on behalf of the user, and select the most relevant and personalized services based on the received information. The selection happens locally in a distributed manner without reaching out to the cloud using locally constructed user models.

2.3. *Context-Aware Parking Management Systems*

Mahfooz et al. [24] developed a context-aware real-time parking management system. It starts with a user sending a request to reserve a parking space, their system finds the best place based on the available context information and user preference. It then tracks the duration of the parking and charges clients when leaving. The proposed algorithm takes into account the memory and computational time complexity constraints for efficiency reasons.

Biondi et al. [25] proposed a system for finding vacant parking spaces in which no deployed sensors are needed. It uses the sensors onboard the user's smartphone such as Bluetooth and GPS to extract the context information including the user's location and neighboring smartphones. Using this context information alongside a database of known parking areas, their proposed engine can find empty spots and report them to the users.

2.4. *Summary and Research Gaps*

From the previous section, we find that only the selective sensing work does not take advantage of any context information. On the other hand, in Section 2.2, the context-aware works are focused on specific scenarios and cannot be generalized to other applications. While the context-aware parking solutions mentioned previously do leverage context, they make no effort to reduce the impact of large data. A comparison of the works mentioned in Section 2 is shown in Table 1.

Table 1. Related works comparison.

	Data Selection	Leverages Context	General Purpose	Application
Ning et al. [12]	✓	×	✓	IoT
Liu et al. [14]	✓	✓	×	WSN
Ardagna et al. [15]	✓	✓	×	Smart City
Liu et al. [16]	✓	✓	×	Mobile
Prabha et al. [17]	✓	✓	×	Sensing
Lee et al. [18]	✓	✓	×	IoT
Kavitha et al. [19]	✓	✓	×	Healthcare
Shapsough et al. [20]	✓	✓	×	Education
Lohani et al. [21]	✓	✓	×	Sensing
Elayan et al. [22]	✓	✓	×	Healthcare
Gochhayat et al. [23]	✓	✓	×	IoT
Mahfooz et al. [24]	×	✓	×	Parking
Biondi et al. [25]	×	✓	×	Parking
CONTESS	✓	✓	✓	IoT

3. Framework Architecture

The network architecture of a sensing system is assumed to be organized into three layers (see Figure 1). The sensors layer consists of all the types of data sources spanning from physical sensors, such as temperature and humidity sensors, to virtual sensors such as calendar, social media and email, in addition to logical sensors such as weather and phone orientation. The following layer represents computing devices with various capacities at the edge of the network. The CONTESS framework is designed to be deployed in edge servers or cloudlets (e.g., smart home/building/district server), where it is located near the sensors for low latency and can interface with them through various connection mediums such as Wi-Fi, cellular networks or wired connections. The main role of the framework is to collect data and extract the context with which it is decided whether the data need to be sent to the next processing layer through the Internet. The final layer comprises processing centers in the cloud or applications.

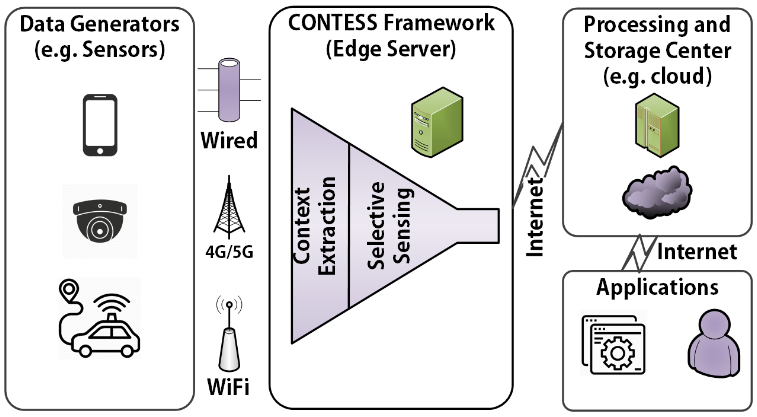
Data Generators (e.g. Sensors)
Wired
4G/5G
WiFi
CONTESS Framework (Edge Server)
Context Extraction
Selective Sensing
Internet
Processing and Storage Center (e.g. cloud)
Internet
Applications

Figure 1. Framework Network Architecture.

As shown in Figure 2, the CONTESS framework is composed of the two main components. Context management encapsulates the functions dealing with the context from its capture, creation to its destination. Selective sensing uses the constructed context in the selection process. These components are explained in the following sections.

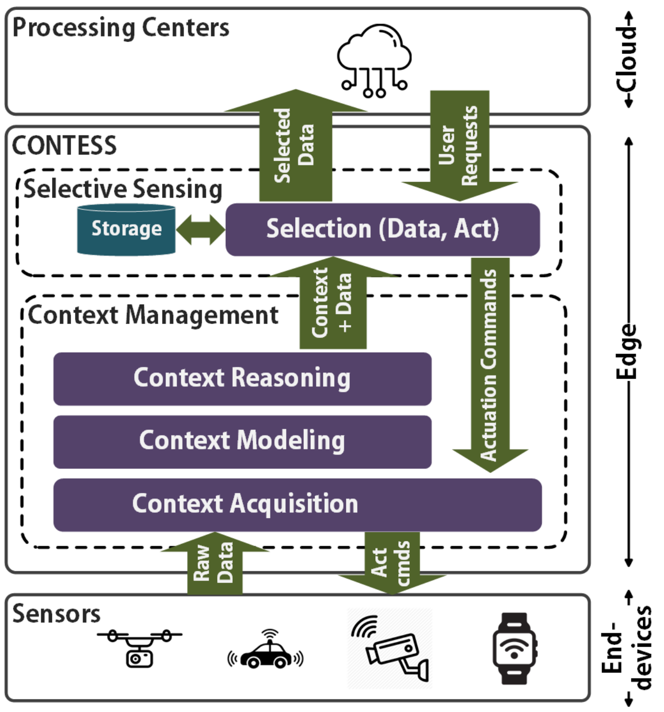

Figure 2. Proposed Framework Architecture.

3.1. Context Management

Context-aware applications are recognized to support three features which are context presentation, execution and tagging. These features are adapted to fit the needs of our framework. (1) Presentation is the ability to present appropriate information to the user depending on the context. This feature is represented in our framework by the selective sensing component. (2) Execution involves performing automatic actions without user intervention. For example, in the cases where an observed context scene is deemed to be missing some information, issuing actuation commands to a surveillance camera (e.g., tilt, rotate) results in obtaining a better understanding of the context. (3) Tagging, also known as annotation, is an important step before the interpretation of the collected context. Since, in most cases, single data sources can hardly contain enough information to construct the full context of a situation, the fusion of multi-source data is necessary. This only becomes possible if the data from every source are tagged with their corresponding context.

Depending on the extent of user involvement in the context-building process, context can be recognized at three levels; full user involvement, where the user's interest, preference and expectation dictates what, when and how the context is handled; partial user involvement, where changes in the environment invoke the context creation and are then presented to the user for decision making; and no user involvement, where the context is autonomously created and reacted to.

According to Perera et al. [9], a context goes through four steps in its life cycle. At the beginning, it is captured from different data sources. Then, it is modeled and expressed in a more meaningful form. These models are then combined and analyzed to extract higher-level context. Finally, results of previous steps are distributed to users. In designing our framework, we organized all the context management functions into a similar fashion (see Figure 2), however, the final step (i.e., dissemination) is handled by the selective sensing component. In the following sections, we describe each module in detail.

3.1.1. Context Acquisition

IoT is built on the heterogeneous nature of devices, therefore, one of the main goals of the context acquisition module is to provide an abstraction layer to the differences in physical communication mediums and the various data formats. This will allow CONTESS to isolate the complexities regarding devices' networking and protocols from the rest of the framework. Figure 3 shows the main functionalities of a context acquisition module.

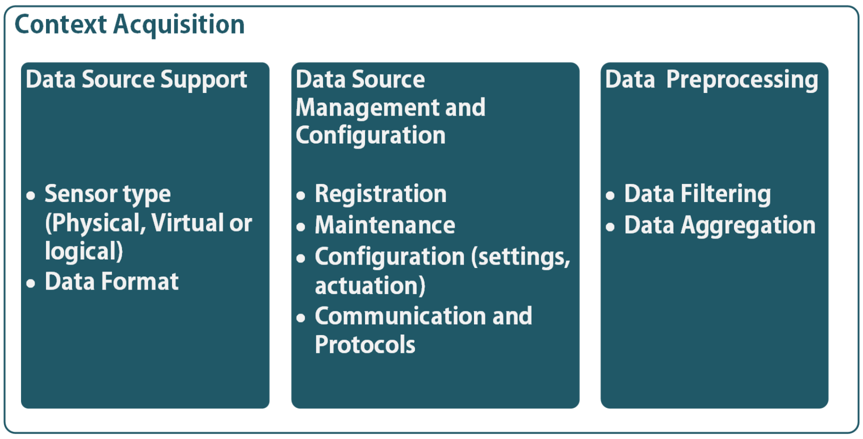
Context Acquisition
Data Source Support
• Sensor type (Physical, Virtual or logical)
• Data Format
Data Source Management and Configuration
• Registration
• Maintenance
• Configuration (settings, actuation)
• Communication and Protocols
Data Preprocessing
• Data Filtering
• Data Aggregation

Figure 3. Context Acquisition Module.

As the interface between the real world and the software, this module has to support all types of data retrieval methods such as pushing and pulling, instant fetching and interval-based methods. One of the issues that is plaguing IoT middleware is the huge amount and variety of protocols, data formats and standards used by different manufacturers. The context acquisition module has to deal this issue either by implementing support or providing format conversion methods to unify data for later analysis and fusion.

Context acquisition also involves the management of connected data sources by firstly registering newly connected devices, and keeping them maintained through failure reports and firmware updates. Secondly, devices require a setup, periodical and situational configuration changes including the delivery of actuation commands. This last part is often overlooked in most proposed frameworks [26]. Actuation is as important as sensing itself, and can be a change in a sensor configuration or a command to perform a specific task using a connected actuator. Such abilities are only possible if the data source management component can communicate using various protocols.

The earliest form of simple selective sensing in the framework is performed at this level. The collected raw data potentially contain redundancies and unnecessary parts which are filtered out to reduce the later processing overhead. Data from similar purpose sensors can be aggregated together to form a more accurate and less error-prone context.

3.1.2. Context Modeling

The context modeling module is mainly responsible for three functions, as illustrated in Figure 4). A context model is a representation or a subset of the real context collected from sensors that can be used in executing a task. It consists of a set of attributes that describe the context, each of which has a type and a value; additionally, it could also have other properties to describe certain characteristics [9]. According to a predefined set of information to be collected (i.e., static context) or depending on the situation (i.e., dynamic), any newly collected data are represented in terms of attributes, characteristics and relationships to form a model. This process is sensitive to data inconsistencies, incompleteness and invalidity, therefore, it is necessary to clean the data before modeling. Several techniques can be used to eliminate anomalous values from the data such as hidden Markov models (HMMs), Bayesian networks and fuzzy logic. Missing values due to sensor imperfections can affect the modeling process, thus, using imputation or prediction methods to fill in the gaps is needed. Depending on the level of sensitivity, missing values can be approximated using more complex methods such as K-nearest neighbors (KNNs) and deep learning which are computationally expensive. Otherwise, simpler arithmetic methods are more suitable.

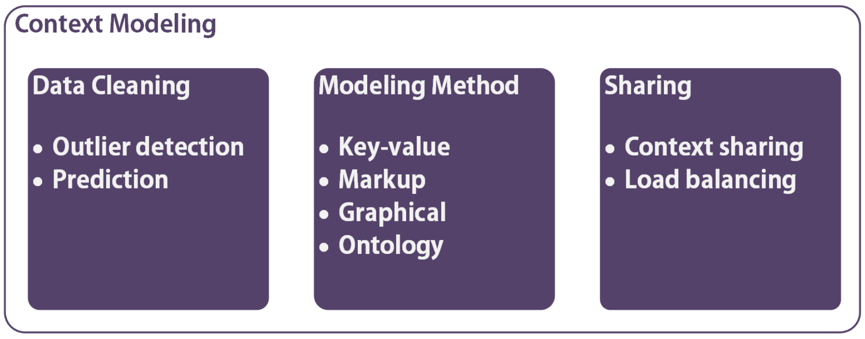

Figure 4. Context Modeling Module.

Numerous context representation (i.e., modeling) techniques exist, each with its advantages and disadvantages. A categorical review of the popular modeling methods is presented by Bettini et al. [27]; here, we only briefly mention the most notable ones. The choice depends on the developer to select the most appropriate technique for the application. The simplest context representation method is key-value pairs, most suitable for small contexts. Markup methods using tags improve on the previous method in terms of retrieval efficiency. To better capture relationships, graphical modeling is used, which can utilize databases with all their advantages to hold large amounts of data. Rules are used as a logic-based modeling method which helps in the reasoning process. The most widespread modeling method is the one based on ontologies: its advantages include the flexibility, extensibility and generality it offers to the widely available reasoning engines. It also uses standardized languages such as Resource Description Framework (RDF) and Web Ontology Language (OWL).

In a distributed sensing system, multiple edge servers could run instances of the CONTESS framework, each connected to a different set of sensors. A single context could potentially require data from the sensors connected to different edge servers. To gather and combine the context models from these servers, a sharing scheme must be implemented. Furthermore, a sensing system is not guaranteed to generate data evenly across the edge servers, therefore, the sharing scheme must also include a load balancing function to ease the computing load off overwhelmed servers.

3.1.3. Context Reasoning

The context reasoning module attempts to extract new knowledge out of the existing context. Its main functions are shown in Figure 5. IoT allows the deployment of millions of sensors which may serve different or similar goals. Combining data from multiple sensors, identical or different, gives information that is more accurate, complete and of higher certainty. Data fusion can occur in three scenarios: first, complementary fusion incorporates the different factors measured by different sensors to form a full and broader understanding; and second, redundancy fusion helps improve the measurement of a single factor through multiple sensors. Finally, cooperative fusion combines data from different factors to deduce new knowledge [26]. The next step after fusion is the inference which deduces a higher-level context from a low-level one (e.g., context models). These two steps are intertwined and can be distinguished in terms of the output context level.

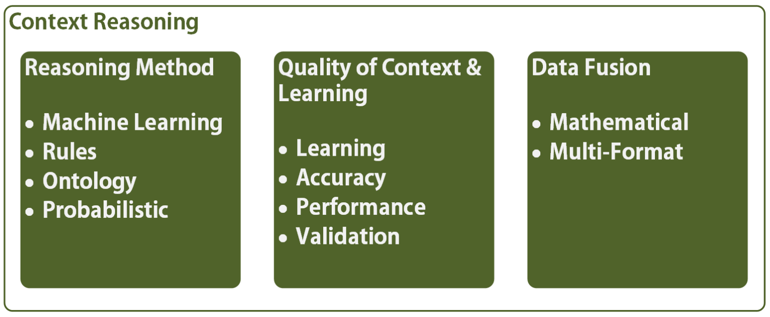

Figure 5. Context Reasoning Module.

Reasoning methods have been extensively surveyed in [9,10]; thus, in this section, we broadly mention just a few of them. Ontology-based reasoning uses description logic and integrates closely with ontology models and therefore has many advantages. It suffers from missing values although it can be mitigated using rules. It is mainly common with hybrid reasoning, activity recognition and event detection applications. Simple if–then–else rules can be used to extract higher levels of context. It is the most popular form of reasoning and it is usually used in combination with ontology reasoning. Supervised learning uses the collected data along with chosen (i.e., expected result) labels to train the models for reasoning. Decision trees, neural networks and support vector machines (SVM) are among the most commonly used models. Unsupervised learning, on the other hand, is used when labels are unavailable: it identifies structures on unlabeled data. K-Nearest Neighbor (KNN) is one of its most popular methods used to find clusters in data. Probabilistic reasoning employs probabilities linked to facts about an event or a task to make decisions. This approach can handle unseen and uncertain situations. Hidden Markov Models (HMMs) and Naive Bayes are known methods which were used in applications such as activity recognition [28].

A context model is assessed using a quality-of-context (QoC) measure that varies from implementation to another. However, it usually consists of three parameters, namely validity, precision and updatedness. A more comprehensive definition proposed by [29], which takes into account the quality of the sensor and the degree of fulfillment of the requirements, states that "QoC indicates the degree of conformity of the context collected by sensors to the prevailing situation in the environment and the requirements of a particular context consumer". QoC evaluators vary in complexity according to the needs of each system and its computational limits. Ref. [29] presents an overview of QoC models. The final step for a context is to be validated to guarantee the correctness of the data. Various conditions can be verified such as the data value, data type, consistency and data source quality.

3.2. Selective Sensing

Composed of three main modules and local storage (see Figure 6), the selective sensing component is considered the most important in the CONTESS framework. With the purpose of analyzing the received contexts accompanied by their corresponding data, with respect to user requests and requirements, they are assigned an importance metric which decides whether the data are worth distributing. The selection methods (SMs) range from simple rule-based decisions and context-matching with stored templates, to complex trained machine learning models. This depends on the type of managed contexts, and the computing capacity of the hardware running the framework, a selection method chosen by the developers to ensure real-time decision making and high selection confidence. The selection module accesses the local storage to store or retrieve SM-related data (e.g., rules, regression model). Similarly to how it receives rule changes and updates, it also receives feedback on its decisions to be used for training and improving the models.

Data distribution is performed via two mechanisms: (1) data consumers issuing requests (i.e., a queries) which are interpreted to find and send the requested data; and

(2) data consumers subscribe to specific contexts by describing their requirements, which allow the sensing system to send them data in time intervals or upon the occurrence of certain events. The latter is suitable for real-time applications.

Data consumers and application users are registered and identified by the user management module using their IDs. This allows the selective sensing component to authenticate and keep track of requests and subscriptions. In addition to user access control by managing privileges, user profiles also store the user request history for potential user interests mining which is useful for offering customized and automated services.

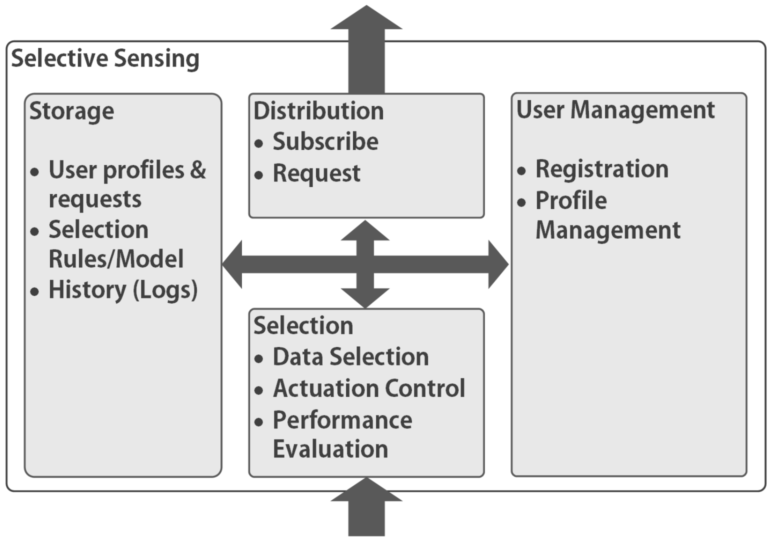

Figure 6. Selective Sensing Module Architecture.

4. Use Case Scenario: A Smart City Parking Space Detection

In this section, we provide a real-world use case scenario portrayed in a smart city to indicate the applicability of the CONTESS framework and describe its functionalities with a step-by-step process. In modern cities, finding parking spaces is a huge problem due to the higher population density in small areas. Drivers spend a significant amount of their time trying to find a parking space. A huge amount of money is wasted on parking fines and overpriced parking lots. Parking space management is an essential part of any smart city environment. It helps reduce congestion, pollution, conserve resources and provide convenience for residents. Existing parking management systems either use car counters positioned in park gates, or use physical sensors deployed under cars in parking spaces. Both methods require higher construction costs for big parking lots. Employing the already deployed video surveillance cameras, normally used for security, to cover huge areas offers a real-time and accurate service for very low costs [30].

Numerous applications based on image analysis for vacant parking space detection were proposed in the literature such as [30,31]. In a typical cloud-based video analytics systems, a middleware is introduced to manage the heterogeneous connected devices and offer some preprocessing capabilities. The video data still need to be fully processed in cloud servers. For a large city, this puts tremendous load on the communication infrastructure and consumes large computing resources, therefore, we propose to use the CONTESS framework to tackle this issue.

Assuming that the deployed surveillance cameras are connected to a nearby edge server, using wired or wireless data connections, the CONTESS framework is run (see Figure 7). When a driver approaches their destination, their navigation device issues a request to the cloud stating that they need a parking space. The cloud server combines their location with a set of requirements and send it to the nearest CONTESS server. These requirements can contain descriptions of events to look for, such as when a car is leaving an empty spot.

Assuming there are no available parking places, we recognize two possible scenarios. Firstly, the nearby parking lots are already equipped with surveillance cameras. In this situation, captured video data from multiple cameras are transmitted to the CONTESS framework and received at the level of the context acquisition module. The video data are first standardized into a specific predetermined data format. The modeling module receives preprocessed video data and extracts moving object coordinates according to the requirement descriptions provided by the cloud servers. The extracted low-level context is represented (i.e., modeled) in the form of a Markup model (e.g., XML) and sent to the reasoning module.

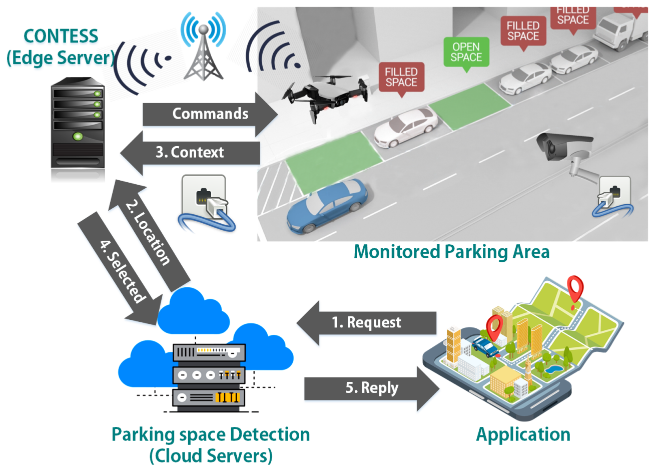

Figure 7. Parking space detection using CONTESS.

The reasoning module attempts to extrapolate the moving object coordinates into a full path (i.e., high-level context). It also tries to fuse contexts the provided from different angle cameras to produce a more complete context. The context is validated against a set of logical rules to filter-out unusual contexts such as those regarding objects moving in paths which are not normally associated with a vehicle. For example, a human or a bird could generate paths that are impossible for a vehicle. A high-level context is analyzed using a selection model (SM) in the selection module. The SM takes a feature vector consisting of the produced context markup model as input. It is trained in the cloud servers to classify feature vectors into a positive event (i.e., car leaving empty spot) or a negative event. Data related to positive events are selected and sent to the cloud servers implemented for scene analysis, which then measures the available spots using more complex methods. This approach enables the cloud applications to express their needs to the CONTESS framework which then selects interesting data according to those needs.

Secondly, in the case where no nearby surveillance cameras are available to monitor parking areas, the selection module, using the user's location, locates unmanned aerial vehicles (UAVs) in the proximity and issues actuation commands through the acquisition module to relocate and monitor destined areas.

5. Experiment and Evaluation

In this experiment, we will be focusing on extracting high-level context such as the duration and frequency of a parked vehicle. This will allow us to better demonstrate the benefits of the framework.

To evaluate the effectiveness of CONTESS, we tested its applicability for parking management in a smart city scenario. We implemented this use-case scenario using linked data and semantic web technologies. Firstly, we designed an OWL-based ontology that incorporates all the required components of CONTESS. We used Protege (www.protege.stanford.edu (accessed on 15 March 2023)) to implement the ontology shown in Figure 8. The parking management scenario is taking place in a smart city, where each street contains one or more roads and each road contains one or more parking spots. Each street has one or more buildings, where the drivers try to park their vehicles in the nearest parking spot. We used a part of the map of Haidian district, Beijing, China, taken from OpenStreetMaps with the roads map generated by SUMO for our experiments, as shown in Figure 9. As mentioned earlier, CONTESS focuses on context reasoning to improve the selective sensing process. In the context of parking management in a smart city, we focus on extracting the context of parking time and location associated with specific locations and vehicles. For instance, if a vehicle parks nearby a specific location from 9 am to leave after 5 pm, during working days, then it is highly likely that the pattern will continue next time. Such contextual information enables the system to predict future events, hence, better manage the parking assignments. We simulated the vehicle behaviors using the Veins framework (www.veins.car2x.org/ (accessed on 15 March 2023)) of OMNET++ network simulator (www.omnetpp.org/ (accessed on 15 March 2023)) and the traffic generation using SUMO traffic simulator (www.eclipse.org/sumo/ (accessed on 15 March 2023)). We used Apache Jena's triple store to store the inferred knowledge (www.jena.apache.org/ (accessed on 15 March 2023)). The traffic was divided into two classes: (1) randomly generated traffic, where vehicles park randomly and randomly leave the parking spots; and (2) context-related traffic, where the vehicles park regularly at the same location for the same time interval. For context acquisition, semantic reasoning and traffic assignment, CONTESS leverages the Semantic Web Rule Language (SWRL) (www.w3.org/Submission/SWRL/ (accessed on 15 March 2023)). Equation (1) shows an example of SWRL that adds a context property to the triplet store if a vehicle is parked in the same location for given time intervals.

$$VehicleParkAt(?c, ?L, TI1) \wedge VehicleLeavesAt(?c, ?L, TI2) \rightarrow AddContextProperty(?c, ?L, TI1, TI2); \quad (1)$$

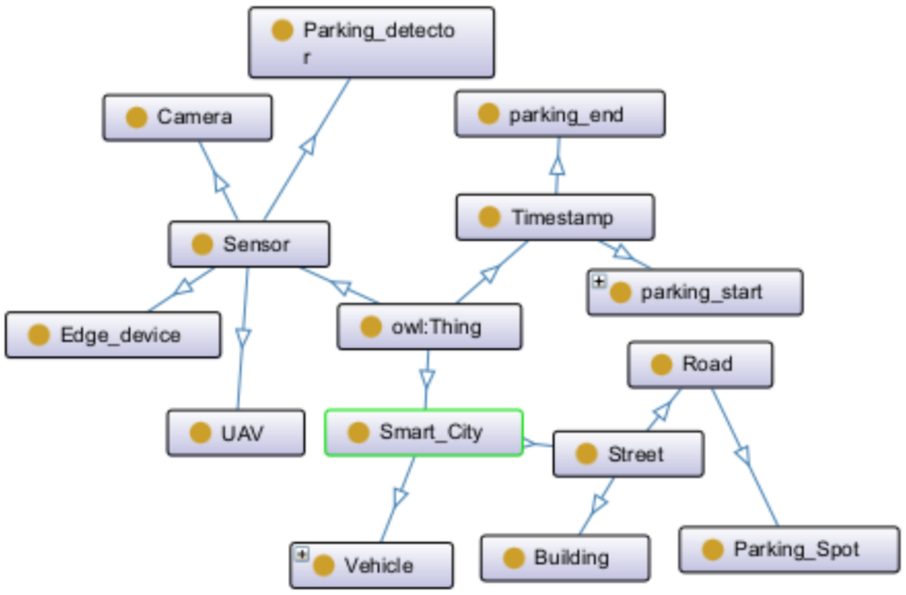

Figure 8. Parking management scenario ontology.

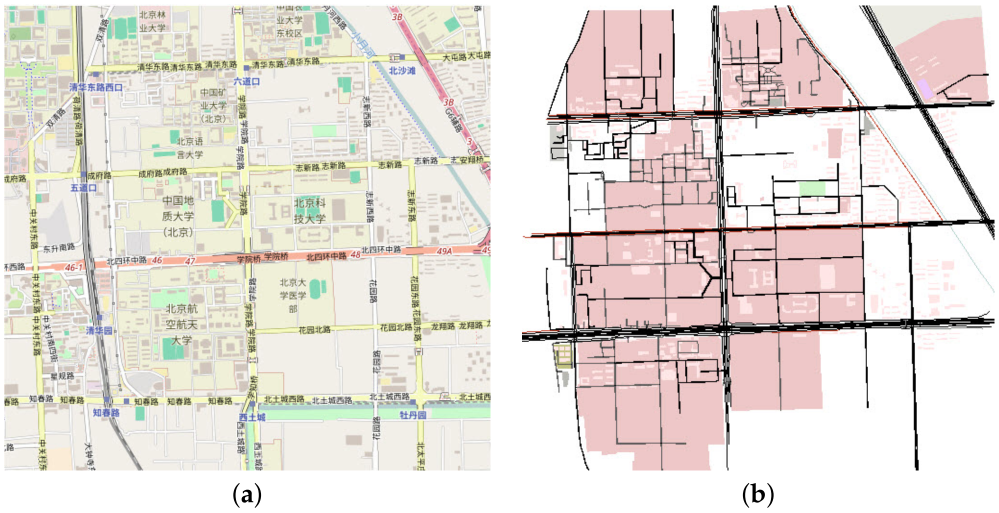

Figure 9. Map of the simulated scenario: (**a**) the real map of the simulated area; and (**b**) the road network of simulated area in the SUMO traffic simulator.

Once a vehicle enters the roads that contain the location of the destination, it sends a parking request message. The parking management system assigns the nearest parking location if available, or recommends a specific waiting time according to the parking context of nearby vehicles. The system performance is evaluated based on two metrics: Distance to location (DL)—how far is the assigned parking spot from the desired parking location; and time to park (TP)—the time required for a vehicle to find a parking location.

6. Results and Discussion

Figures 10 and 11 show the distance to the location and time to park, respectively, with different percentages of the context-related traffic. In Figure 10, we can observe that using CONTESS can significantly decrease DL with the increase in context-related traffic, until reaching 60% where it stabilizes and no longer reduces DL, which is because, at this point, it already took advantage of the deduced context, and the newly arrived vehicle has to travel a long distance in order to reach the assigned parking. Similarly, Figure 11 shows the effectiveness of CONTESS in reducing the average TP. The latter decrease as the context-related traffic increases until finally stabilizing when context-related traffic is more than 50% of the total traffic.

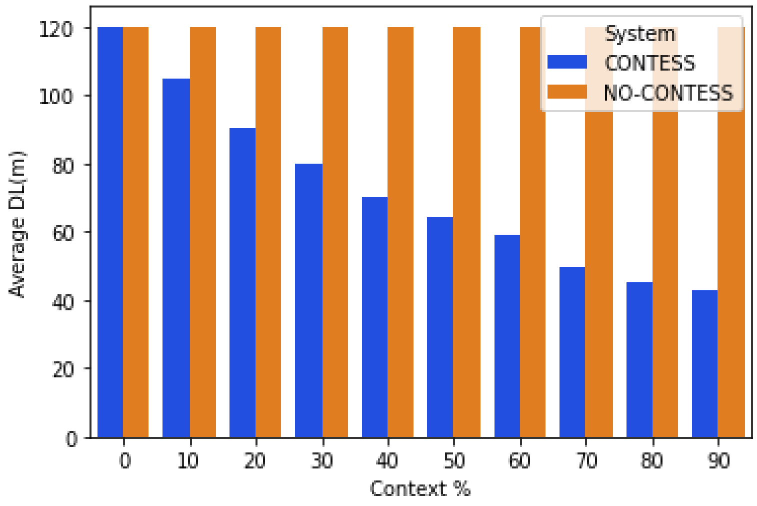

Figure 10. Average distance to location in different context-related traffic percentages.

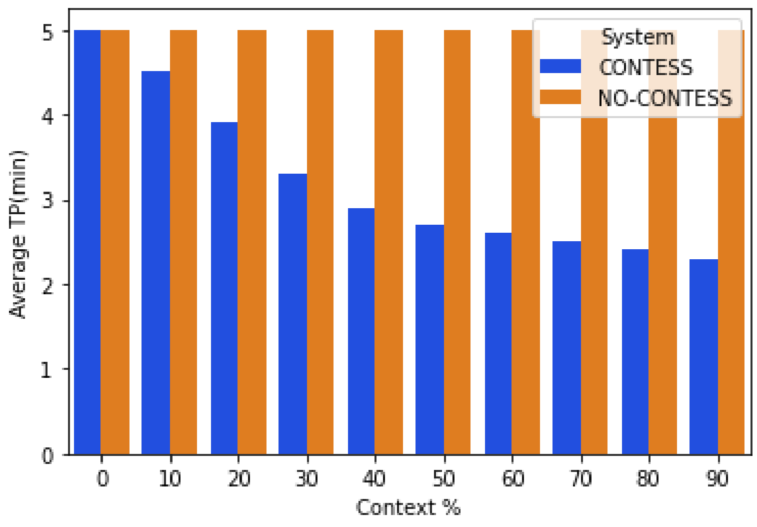

Figure 11. Average time to park in different context-related traffic percentages.

Each building is equipped with several edge devices and sensors that help in the computational and storage tasks. Each vehicle is associated with two timestamps, parking start time and parking end time. When a vehicle parks, a parking detection sensor sends a message to the nearest edge device that records the parking start event, and the number of the license plate of the identified vehicle. The computational cost of extracting the knowledge and populating the ontology is performed at the edge level and only the final linked data are stored in a remote cloud-based Triple store; therefore, the computational overhead is distributed across the smart city, which allows one to scale up the system and significantly improves its performance. Moreover, in CONTESS, the vehicles just need to update nearby edge devices, and there is no need to send parking requests to the remote server, which significantly reduces the communication overhead. Figure 12 shows the communication overhead measured by the total number of messages that the server receives from all vehicles. We can observe that, without CONTESS, the communication overhead exponentially increases with the increase in traffic density (number of vehicles per km), which is because each vehicle directly communicates with the remote server when requesting parking or leaving a parking spot. This is unlike when using CONTESS, where the edge devices handle the parking request messages.)

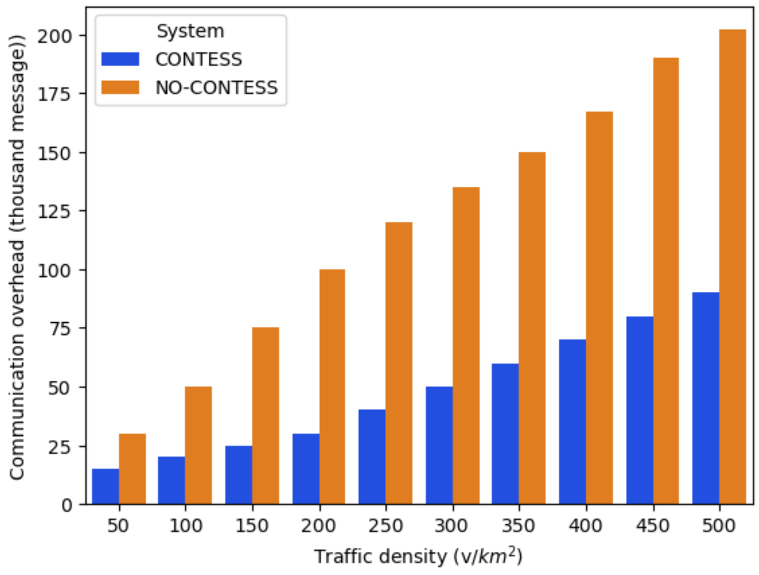

Figure 12. Communication overhead comparison.

7. Conclusions

In this work, we propose CONTESS—an edge-based framework that leverages context information in any sensing system to improve its efficiency by reducing the impact of large data and communication costs on the network. We presented a detailed view of the architecture of CONTESS which consists of two main components, namely context management and selective sensing. The context management component consists of four elements which are responsible for context acquisition, modeling and reasoning. We proposed a parking space detection system in a smart city as a use case scenario to demonstrate the capabilities of CONTESS. Using linked data and semantic web technologies, we implemented CONTESS in a parking management system where we focus on extracting the parking time and location context information which are used to improve the future parking space suggestions. The results show that using CONTESS reduced the parking time by up to 50% and the distance to location by up to 60%.

Author Contributions: Conceptualization, A.B.S. and S.D.; methodology, A.B.S., A.N. and A.K.; software, A.B.S. and A.N.; validation, A.B.S. and S.D.; formal analysis, A.B.S.; investigation, A.B.S.; writing—original draft preparation, A.B.S., S.D., A.N. and A.K.; writing—review and editing, A.B.S.; visualization, A.B.S.; supervision, H.N. and S.D.; project administration, H.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Anawar, M.R.; Wang, S.; Azam Zia, M.; Jadoon, A.K.; Akram, U.; Raza, S. Fog computing: An overview of big IoT data analytics. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 7157192. [CrossRef]
2. Arivazhagan, C.; Natarajan, V. A Survey on Fog computing paradigms, Challenges and Opportunities in IoT. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 28–30 July 2020; pp. 385–389.
3. Ning, H.; Farha, F.; Mohammad, Z.N.; Daneshmand, M. A survey and tutorial on "connection exploding meets efficient communication" in the Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 10733–10744. [CrossRef]
4. Pan, J.; McElhannon, J. Future edge cloud and edge computing for internet of things applications. *IEEE Internet Things J.* **2017**, *5*, 439–449. [CrossRef]
5. Saeik, F.; Avgeris, M.; Spatharakis, D.; Santi, N.; Dechouniotis, D.; Violos, J.; Leivadeas, A.; Athanasopoulos, N.; Mitton, N.; Papavassiliou, S. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Comput. Netw.* **2021**, *195*, 108177. [CrossRef]
6. Ngu, A.H.; Gutierrez, M.; Metsis, V.; Nepal, S.; Sheng, Q.Z. IoT middleware: A survey on issues and enabling technologies. *IEEE Internet Things J.* **2016**, *4*, 1–20. [CrossRef]
7. Razzaque, M.A.; Milojevic-Jevric, M.; Palade, A.; Cla, S. Middleware for internet of things: A survey. *IEEE Internet Things J.* **2016**, *3*, 70–95. [CrossRef]
8. Dey, A.K. Understanding and using context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7. [CrossRef]
9. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE Commun. Surv. Tutorials* **2014**, *16*, 414–454. [CrossRef]
10. Sezer, O.B.; Dogdu, E.; Ozbayoglu, A.M. Context-Aware Computing, Learning, and Big Data in Internet of Things: A Survey. *IEEE Internet Things J.* **2018**, *5*, 1–27. [CrossRef]
11. Cai, X.; Ning, H.; Dhelim, S.; Zhou, R.; Zhang, T.; Xu, Y.; Wan, Y. Robot and its living space: A roadmap for robot development based on the view of living space. *Digit. Commun. Netw.* **2021**, *7*, 505–517. [CrossRef]
12. Ning, H.; Ye, X.; Ben Sada, A.; Mao, L.; Daneshmand, M. An Attention Mechanism Inspired Selective Sensing Framework for Physical-Cyber Mapping in Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 9531–9544. [CrossRef]
13. Wang, W.; Ning, H.; Shi, F.; Dhelim, S.; Zhang, W.; Chen, L. A Survey of Hybrid Human-Artificial Intelligence for Social Computing. *IEEE Trans. Hum.-Mach. Syst.* **2022**, *52*, 468–480. [CrossRef]
14. Liu, Y.X.; Liu, A.; Guo, S.; Li, Z.; Choi, Y.J.; Sekiya, H. Context-aware collect data with energy efficient in Cyber–physical cloud systems. *Future Gener. Comput. Syst.* **2020**, *105*, 932–947. [CrossRef]
15. Ardagna, D.; Cappiello, C.; Samá, W.; Vitali, M. Context-aware data quality assessment for big data. *Future Gener. Comput. Syst.* **2018**, *89*, 548–562. [CrossRef]

16. Liu, S.; Zheng, Z.; Wu, F.; Tang, S.; Chen, G. Context-aware data quality estimation in mobile crowdsensing. In Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9. [CrossRef]
17. Prabha, R.; Ramesh, M.V.; Rangan, V.P.; Ushakumari, P.V.; Hemalatha, T. Energy Efficient Data Acquisition Techniques Using Context Aware Sensing for Landslide Monitoring Systems. *IEEE Sens. J.* **2017**, *17*, 6006–6018. [CrossRef]
18. Lee, T.D.; Lee, B.M.; Noh, W. Hierarchical cloud computing architecture for context-aware IoT services. *IEEE Trans. Consum. Electron.* **2018**, *64*, 222–230. [CrossRef]
19. Kavitha, D.; Ravikumar, S. IOT and context-aware learning-based optimal neural network model for real-time health monitoring. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4132.
20. Shapsough, S.Y.; Zualkernan, I.A. A generic IoT architecture for ubiquitous context-aware learning. *IEEE Trans. Learn. Technol.* **2020**, *13*, 449–464. [CrossRef]
21. Lohani, D.; Acharya, D. Smartvent: A context aware iot system to measure indoor air quality and ventilation rate. In Proceedings of the 2016 17th IEEE International Conference on Mobile Data Management (MDM), Porto, Portugal, 13–16 June 2016; Volume 2, pp. 64–69.
22. Elayan, H.; Aloqaily, M.; Guizani, M. Digital Twin for Intelligent Context-Aware IoT Healthcare Systems. *IEEE Internet Things J.* **2021**, *8*, 16749–16757. [CrossRef]
23. Gochhayat, S.P.; Kaliyar, P.; Conti, M.; Tiwari, P.; Prasath, V.; Gupta, D.; Khanna, A. LISA: Lightweight context-aware IoT service architecture. *J. Clean. Prod.* **2019**, *212*, 1345–1356. . [CrossRef]
24. Mahfooz Ul Haque, H.; Zulfiqar, H.; Ahmed, A.; Ali, Y. A context-aware framework for modelling and verification of smart parking systems in urban cities. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5401. [CrossRef]
25. Biondi, S.; Monteleone, S.; La Torre, G.; Catania, V. A context-aware smart parking system. In Proceedings of the 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Naples, Italy, 28 November–1 December 2016; pp. 450–454.
26. Wang, M.; Perera, C.; Jayaraman, P.P.; Zhang, M.; Strazdins, P.; Shyamsundar, R.K.; Ranjan, R. City data fusion: Sensor data fusion in the internet of things. In *The Internet of Things: Breakthroughs in Research and Practice*; Information Resources Management Association: Washington, DC, USA, 2017; pp. 398–422. [CrossRef]
27. Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A.; Riboni, D. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.* **2010**, *6*, 161–180. [CrossRef]
28. Maarala, A.I.; Su, X.; Riekki, J. Semantic Reasoning for Context-Aware Internet of Things Applications. *IEEE Internet Things J.* **2017**, *4*, 461–473. [CrossRef]
29. Manzoor, A.; Truong, H.L.; Dustdar, S. Quality of context: Models and applications for context-aware systems in pervasive environments. *Knowl. Eng. Rev.* **2014**, *29*, 154–170. [CrossRef]
30. Cai, B.Y.; Alvarez, R.; Sit, M.; Duarte, F.; Ratti, C. Deep Learning-Based Video System for Accurate and Real-Time Parking Measurement. *IEEE Internet Things J.* **2019**, *6*, 7693–7701. [CrossRef]
31. Màrmol, E.; Sevillano, X. QuickSpot: A video analytics solution for on-street vacant parking spot detection. *Multimed. Tools Appl.* **2016**, *75*, 17711–17743. [CrossRef]