Khulna University Studies

ORIGINAL ARTICLE

An Intelligent Edge Processing Framework using Situation Calculus for IoT based Smart City

SK Alamgir Hossain
Computer Science and Engineering Discipline,
Khulna University, Khulna, Bangladesh
Email: alamgir@cseku.ac.bd

Md. Anisur Rahman
Computer Science and Engineering Discipline,
Khulna University, Khulna, Bangladesh

M. Anwar Hossain
Department of Software Engineering,
CCIS, King Saud University, KSA

KUS: 21/01: 110121

Abstract

With advancements in IoT (Internet of Things), the number of applications and services for Smart Cities have increased in recent years, ranging from public services and safety to smart traffic and smart audio/video surveillance. As mobile and cloud computing have advanced, a large number of IoT devices have been connected to the Internet. This connected devices generates billions of bytes of data at the network edge. In order to fully realize the potential of this edge data there is a strong need to extend AI's frontiers to the network edge. Edge computing, a new paradigm that shifts computing workloads and services from the network core to the network edge, has been identified as a viable approach to satisfy that need. The concept of Edge Intelligence (EI) in Smart Cities is gaining a lot of attraction. However, EI research is still in its early stages, and research groups would welcome a dedicated platform for communicating recent breakthroughs in EI in Smart Cities. To this end, we introduced a new technique of intelligent edge processing framework using Situation Calculus for IoT based Smart Cities. In this paper, we provide an overview of our architecture, framework, and key techniques.

Keywords: Edge Computing, Smart City, Situation Calculus, Internet of Things

1 Introduction

We are living in an incredible time of Big Information explosion. It has lately seen a drastic transition in data source from massive cloud computing data centers to more common end devices, such as mobile/IoT devices. Usually, huge data was created and kept in large data centers. The trend is currently reversing, especially to the spread of mobile computing and IoT. Cisco anticipates that by 2021, all network edge users, devices, and objects will create over 850 ZB. In comparison, global data-center traffic is estimated to reach just 20.6 ZB by 2021. In smart cities the edge ecosystem will provide many new opportunities by processing huge amounts of IoT data. On the other hand transferring data to the edge ecosystem is challenging with rest to speed, cost, and privacy. The common assumption is that data from IoT devices should be transmitted to cloud datacenters for analysis [25]. But, when transferring huge data through the Internet, both cost and time may be excessively expensive, and privacy leakage may be a severe concern [9]. An alternative but very effective approach is to process the data through edge devices. Each device act as an AI agent that could process the data locally. Figure 1 demonstrating the edge computing paradigm.

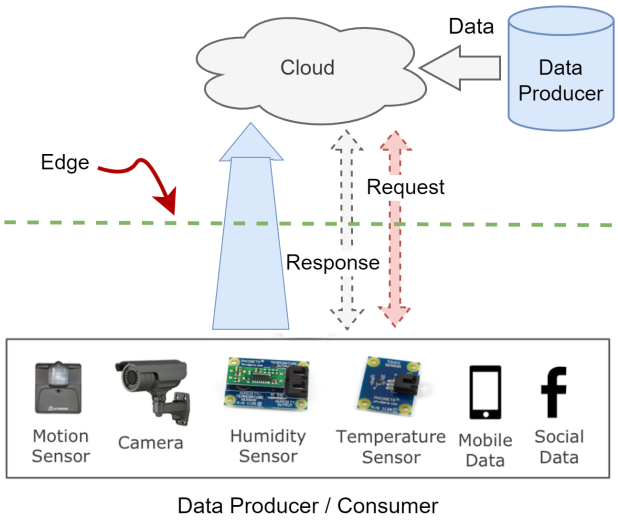

Figure 1: Edge computing paradigm.

In general, the physical proximity of information-generation sources provides several advantages over the traditional cloud based processing. Furthermore, the intersection of artificial intelligence (AI) with edge computing has created a new field known as EI (Edge Intelligence) [22]. Through EI we can easily deploy machine learning algorithms to the edge devices. So the edge devices will be more powerful in the context of information processing unit. So, in order to better understanding or analyzing a city's current situation it is a good way to deploy AI based agents in the edge devices [8]. Here in general a situation [10] of a system refers to the current state of the system. According to IoT paradigm a situation is a collection of conditions and events that represent the status of the system at that moment. So situation is the overall picture of a system that can be a useful observation or monitoring tools for the city officials. With the growth of

cities, many events are expected to occur, and any inefficient strategy to processing and handling such circumstances would impede prompt decision-making. Many initiatives [27, 1, 3, 15] that mix cloud and IoT technologies to overcome this difficulty.

Edge computing processes data locally and provides processed data to the server located in the Internet. This process reduces the volume of data that must be transported to a remote server. This technology has three key advantages: it can operate with enormous amounts of data, it concentrates on location awareness and it guarantees minimal latency. The concept of Edge Intelligence (EI) is gaining popularity in Smart Cities. However, EI research is still in its early stages, and research groups would benefit from having a dedicated space to share current EI in Smart Cities findings. To that end, we have introduced a situation calculus-based intelligent edge processing system for IoT.

The remainder of this paper is organized as: we look at many significant research publications on edge computing, situation calculus, and IoT-based smart cities in Section 2, the architecture of our proposed system is shown in Section 3, the implementation challenges, and development roadblocks that we discovered throughout our inquiry are discussed in Section 4. Finally, a conclusion and discussion of the paper's future work directions are presented in Section 5.

2 Related Work

The scientific community is familiar with the notions of edge computing and situation calculus. There has already been a lot of study completed in this domain. In this section we have discussed a few of their important works. Situation calculus has been investigated before in a variety of contexts, including cognitive modeling for reacting to IoT cyberattacks, distributed trust reasoning in cyber-physical-social smart systems, and logic-based emergency response systems. Levesque et. al. [16] was the first to coin the word. Although Levesque added several extensions to his initial concept, the coupling of Situation calculus and Edge computing or IoT remained absent. Researchers have recently taken this into account and published numerous papers in this area. Huang et al. [14] created one of the promising efforts among them. Trust is a significant issue, in cyber physical smart systems and their design. There are a variety of trust mechanisms available. Using one's web of trust is a popular way to make trust assessments about unknown entities. Computational trust models based on networks usually presume global knowledge of the network's trust relationships. This notion is faulty since each person only understands his or her own trust relationships.

Chouhan et al. [7] present a situation calculus based technique that could detect cyber attacks automatically. They construct a real world IoT situation to illustrate the usefulness of the technique for modeling and responding to threats. In an IoT-based smart environment, emergency response solutions are required. Mainak et. al. [4] was the first to use situation calculus to create formal logic-based emergency response systems. The study uses Situation Calculus to formalize and verify emergency response actions. Despite the fact that situation calculus is gaining popularity in a variety of smart fields, abstraction in situation calculus action theories was lacking prior to the study [5]. The situation calculus was used to construct a generic framework for agent abstraction in this paper. They presume that each agent has two types of specification: high level and low level. They characterize sound abstraction between such action theories as the existence of an appropriate dissimulation between their respective models. Instead of processing all data in a centralized server, data might be processed at the network's

edge. Researchers recently presented many IoT data processing approaches based on this economical methodology. Among these, a fog computing-based IoT infrastructure was presented in [6]. The authors distinguished two types of deployment: (1) collaborative and (2) integrated. Roman et.al. [23] published another article in which the authors used cellular network (because of availability data source in IoT) as an edge node to analyze data. All those papers mainly focus on edge computing or situation calculus separately, but no work is available that focuses on situation calculus in edge computing.

Other academics [17, 12] have also focused on improving mobile phones based IoT data processing paradigm. Mobile data is transmitted to fog servers, which they filter or process before transferring it to cloud servers in these works. In their work, Lv et al. [18] established a way to conduct calculations cooperatively to reduce the large computational demand. First and foremost, the strategy must address how to induce selfish gadgets to collaborate. Second, the approach addresses the issue of how to carry out collaborative computing when the device is willing to participate. e.g. how may computations be performed when machine learning jobs need extensibility and privacy? A moving edge server is chosen as the principal unit due to the aforementioned issues. After that the resources exists around the edge server are used to improve the computing performance. The problem is solved using the alternate direction multiplier method (ADMM). According to the findings of the study, an intelligent edge processing framework is required for IoT-based smart city development. So, inspired by [17, 28, 24, 2], we developed an intelligent edge processing framework for IoT-based smart cities that uses situation calculus to overcome the drawbacks of the systems outlined above.

3 Proposed Framework

This section contains our recommended solution of edge processing framework and their working process. In section 3.1 we present the data offloading mechanism. In section 3.2 we present the basic of situation calculus. Then in section 3.3 we present how the situation is represented in IoT. Finally in section 3.4 we present our intelligent edge processing framework in details.

3.1 Data Offloading

There are two types of data sending mechanism (familiar with the term offloading) from edge layer to cloud layer: vertical and horizontal offloading (see Figure 2). Horizontal offloading refers to the ability of the edge to transfer tasks to neighboring edges. Whereas vertical offloading refers to the ability of the edge to offload duties to the cloud. Which method is best depends on the system. If the system needs some pre-processing in the edge layer then horizontal offloading is better than vertical offloading as in the vertical offloading there is no central control for sending data to the cloud.

3.2 Situation Calculus

The situation calculus is a logic language that is used to represent and reason about dynamical environments. John McCarthy [16] was the first to introduce it in 1963. *Situations*, *actions* and *fluents* are the fundamental ideas in the situation calculus. In a nutshell, as agents take activities, the dynamic environment changes from one condition to the next. The activity result is described by fluents which is a situation-specific functions. Fluents are classified into two types: i) relational (can only accept two values: true or

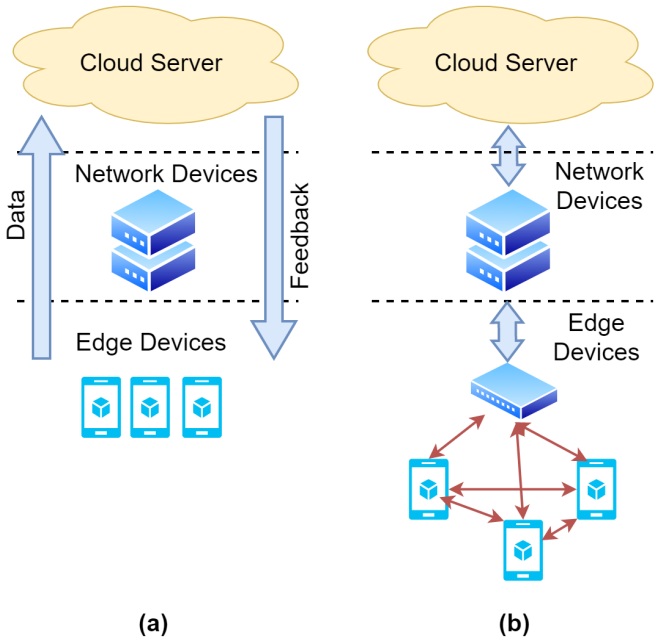

Figure 2: Off-loading patterns in IoT systems, (a) vertical and (b) horizontal.

false) and ii) functional (can accept an unlimited number of values) fluents. For example, handempty is a relational fluent that is true when the robot's hand is empty. Battery-level is a functional fluent with a value of an integer between 0 and 100 that indicates how much backup is remaining of a computer. For any calculation with the help of situation calculus it is the first step to identify all the actions available for the system and after that identify all the fluents to represent the actions. Consider the typical blocks universe, in which a set of equal-sized blocks can be arranged on a table to form towers. The capabilities of an imaginary agent define the collection of actions in this area. Consider that the agent is a one handed robot [19] and can grasp any block from the top of a tower and can either place it to another tower top place or can place it on the table to create a new tower. Few actions of this robot can be represented as follows:

- *stack(x, y)*: If x and y is clear then place x on top of y.
- *unstack(x, y)*: If the robot & x is free and x is on top of y then pick up block x from block y;
- *putdown(x)*: If robot is not empty and holding x in his hand, place block x on the table.
- *pickup(x)*: Pickup x from the table if the robot's hand id free.

The following relational fluents can be used to explain the results of these actions:

- *handempty*: Return true in the case of robot's hand is free.
- *holding(x)*: Return true if the robot's hand is holding block x;
- *on(x, y)*: Return true if block x is on block y;
- *ontable(x)*: Return true if block x is on the table;

- *clear(x)*: Return true if block x is clear and no one is on top of this block.

So, *clear(y)* and *holding(x)* should be true for action *stack(x, y)*. Also *handempty* and *on(x, y)* will be true after the execution of *stack(x, y)*. But *clear(y)* and *holding(x)* will not be true after the execution of *stack(x, y)*.

3.3 Situation Generation

In this section, we will explain how to use the situation calculus logic language in our proposed system. The fundamental goal of our proposed technique is to identify the present condition of a smart city, which represents the status of healthcare, transportation, security, and other elements. Efficient situation awareness enables city authorities to deliver services to residents and take appropriate decisions. In general, a situation is a collection of facts about an environment through time. The classical situation calculus can be used to model the circumstance. A situation, according to situation calculus, is a series of events that begins with a specific situation called initial situation S_0. S_0 denotes the beginning situation that develops throughout time. According to situation calculus when an action d_0 is initiated, a new situation is formed from the first situation S_0. If any action is triggered or surpasses a particular threshold value, T, the following circumstance will be formed. Preconditions may exist for some activities. Suppose we say, "If the place is free, open the camera." This job may be stated as $Poss(open(v), s) \leftrightarrow is\_free(v, s)$, where $Poss$ is a specific function for doing actions. Or "If garbage box is 80% full then send alert message." This job may be stated as $Poss(send\_message(v), s) \leftrightarrow is\_full(v, s, 80)$. Or more complex statement "If there is any traffic on the road at night, turn on the street lights.", $Poss(send\_message(v), s) \leftrightarrow is\_traffic(v, s, road1) \wedge is\_day(v, s)$. In this way we can list the set of actions and statement as a mathematical formulation and finally can generate a result with the help of situation calculus. Here in our proposed method the situation is generated through Algorithm 1.

3.4 Intelligent Edge Processing

In order to get the full potential of IoT, it is necessary to process data intelligently in the edge layer. Edge based processing is economical, faster, and easily integratable. The proposed solution is depicted in Figure 3. It offers services for collecting sensor data, extracting and integrating information, processing it locally, and then pushing it to the cloud. Thus, structured (e.g. sensor data)/semi-structured (e.g. inspection reports)/unstructured (e.g. photos) data must be evaluated by pushing or pulling data to or from the edge servers. The intelligent edge connects and manages field equipment to increase availability and support. It can be installed on a computer or a gateway, depending on the complexity and resource consumption of the services to be used. According to Figure 3, components are grouped in layers, beginning with data sources at the bottom and progressing to communication services at the top. Intelligent edge processing is divided into the following layers:

3.4.1 Plug & Play Management

This layer offers plug-and-play connectivity on demand. New sensors may be simply added, and some can even be deactivated depending on the application. Generally, in IoT plug & play means any IoT device can be easily connectable with an existing system. This does not mean that there should not be any authentication. A device will be

Algorithm 1: *Situation generation*

```
/* Constructs a situation based on the data received from IoT
   devices.                                                    */
while true do
    m^e ← fetchSensorData(m, K); /* K is the authentication key to
       access the sensor                                       */
    e* ← partitionStream(m^e);
    /* Consider the server will process the task in First In First
       Out (FIFO) pattern                                      */
    assignToEdgeServer(e*, FIFO);
    while data available in edge server do
        /* Send the edge server data after unified representation */
        η = AggregateCloudData();
        initialize(last_Situation);
        if last_Situation = null then
            last_Situation ← convertToInitialSituation(η[0]); /* render the
               last_Situation to display devices               */
            continue
        foreach D ∈ η do
            /* D is the data set in 2d array format            */
            if (value(D) > Threshold, T_i) then
                /* means there is a change                     */
                last_Situation ← convertToSituation(D, last_Situation) /* render the
                   last_Situation to display devices           */
```

Algorithm 2: $convertToSituation(D, last_{Situation})$

```
/* This algorithm generates a situation based on the input data and
   last situation.                                             */
```

Input: Data set: D and $last_{Situation}$

Output: A new situation

```
/* Relational Actions and Fluents                             */
```

$f_1 \leftarrow traffic(D, MAX)$;
$f_2 \leftarrow empty\_parking\_lots(D)$;
$f_3 \leftarrow sunny\_areas(D)$;
.....
$f_i \leftarrow pollution\_areas(D, LOWEST)$; `/* need to update with domain specific fluents */`
$last_{Situation} \leftarrow calculate(f_1, f_2, ..., f_i)$
return $last_{Situation}$;

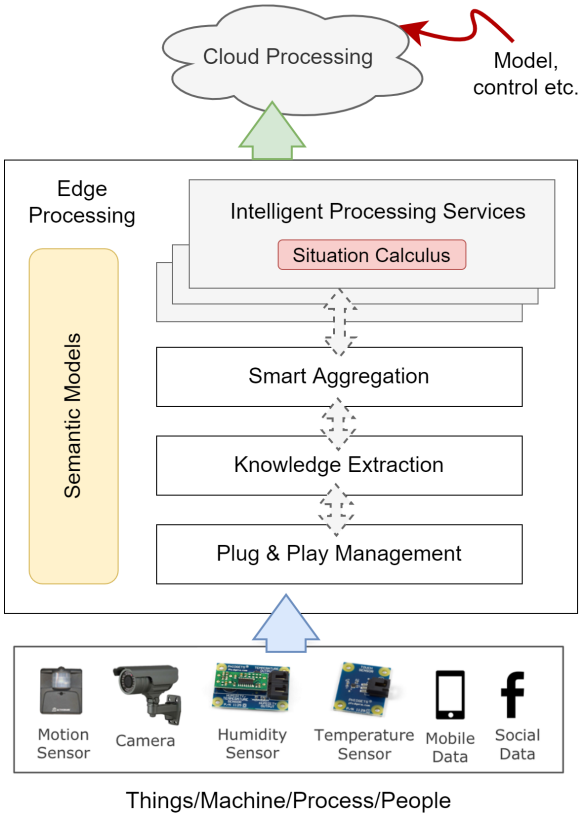

Figure 3: Intelligent Edge Processing.

connected with an existing infrastructure if it is able to complete proper authentication. Plug & Play features reduce user involvement or changes to the remaining system's design or implementation.

3.4.2 Knowledge Extraction

This layer facilitates intelligent multimodal sensing. Conventional data analytics uses structured data. Structured data is well formatted and easy to apply any algorithm, but in the case of semi-structured or unstructured data (e.g. text, photos, videos, and so on), it is more difficult. For semi-structured or unstructured data analysis, prior preparation is necessary to get it in the correct format. In general, it is not a good idea to send all raw data to the server, instead of this a set of information extraction methodologies can be applied to convert the data to a well-defined format. Here in our proposed method we used the knowledge extraction work [13].

3.4.3 Smart Aggregation

As in an IoT environment high volume of data from different sensors are coming into a common gateway so in-order to process the data on time it is important to use smart aggregation method. In our proposed method we are focusing real-time data fusion near the data source. So data aggregation in the edge layer is a good location for smart aggregation.

3.4.4 Intelligent Processing

This layer mainly responsible to provide different intelligent processing services. There are two type processing techniques one is the top-down (model-driven) and bottom-up (data-driven) approaches [26]. Top-down approach work from top to bottom for example it extract data directly from the sensors to recognize any patters and execute action if there will be any. On the other hand the bottom-up approach work from the predictive models at the edge. Based on previous data, this assures that edge analytics improve over time.

3.4.5 Semantic Models

The semantic data model is a manner of organizing data such that it may be represented logically. In our proposed model, this step is further divided into some sub steps as described as follows:

- *Sensors*: Information is gathered through the use of different software, hardware, and sensors. Establishing the link between raw data and the intended concept is one of the most important challenges in the IoT. We do semi-automatic translations between raw sensor data and relevant ontologies. Sensory data generally consists of a time stamp identifying the moment of measurement, a device ID, and the values observed by the sensor. The advantage of applying semantic technologies to sensor data is that the raw data is conceptualized and abstractly represented, making it machine interpretable and connectable to existing resources [11].

- *Domain Knowledge*: Without extra subject expertise, the vast bulk of real-time data is incomprehensible. As a result, all accessible sources (for example, background knowledge about corporate operations, strategy, vision, and so on) should be evaluated.

- *Services*: According to IR 4.0, the application layer should be designed in a manner so that it can hide all the internal complexity and be able to display the activities to the user in an easy way. We followed this methodology in our edge processing model where semantic web services results are used to develop conceptual and semantic representations for expressing services that will be implemented on the intelligent edge. The semantic definition of services enables not only dynamically adding a new service or replacing an existing one, but also building and validating processing pipelines composed of many services. Protégé [1], a well-known open-source ontology building application, was used to generate the semantic models. Several parsers have been created to "import" existing formal and semi-formal models into a single, common, syntactically and semantically coherent model.

Figure 4 depicts the lifecycle of a dataflow task in the edge network. According to the sequence diagram, all the communication will start by initiating a discovery message to all the available nodes. Consider a special node called the coordinator node at first send the offloading request to the respective node (Node1, Node2 ... Node n). After receiving each offloading request, the respective node will return an acknowledgment to the coordinator node. After the discovery step, the next task is to select and configure

[1]Protégé, https://protege.stanford.edu/

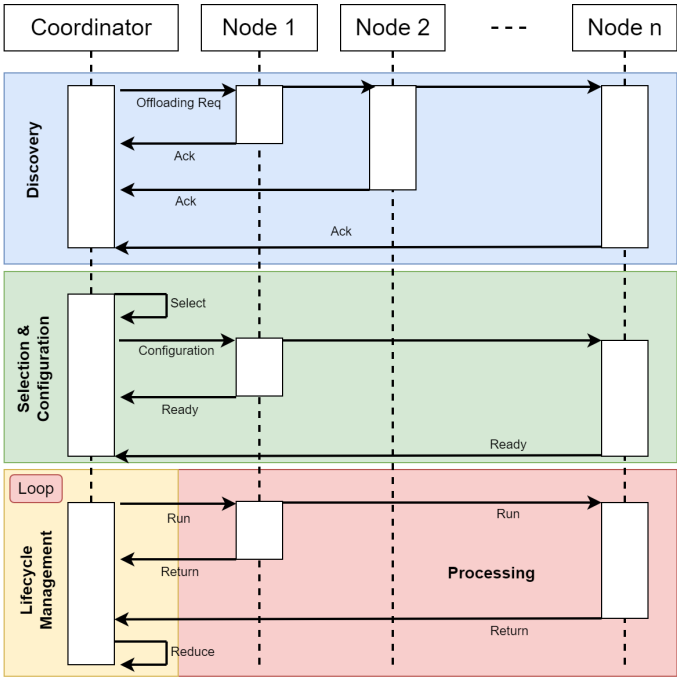

Figure 4: Sequence diagram of job processing in the edge cluster.

the appropriate node for the data transfer. According to the figure, the coordinator can request a select message to another node and send the configuration message and finally if the appropriate node is ready, a ready acknowledgment will be returned to the coordinator. The final step is life cycle management and orchestration. At first run any command to the node and the node return the result and this process iterates until the full job done in the edge cluster.

4 Implementation and Result

This part describes the process of implementing our prototype, which illustrates the method provided in Section 3. In order to justify our technique, we performed several experiments. Our experiments focuses mainly on two aspects: (1) an evaluation of the proposed framework's performance with various targets using some off-line data sources; and (2) test the method in a real-time situation.

4.1 Experimental Setup

As the primary data sources, the City Pulse [21] and City of Chicago [20] smart city data sources were used. Although different IoT datasets are available now but we selected this two data set because in our experiment we need csv formatted data as well as API hook so that our process can call in realtime manner. Table 1 demonstrates the CityPlus and City of Chicago data source details.

We also used the Amazon EC2 service (3.2 GHz Core i7, 16GB memory, Windows Server OS). We ran our tests in two modes, one in offline mode and the other in real-time mode. We created a specific component in our system to accept data from the dataset and transmit it to our model in the offline mode. We utilized our test bed site for the real case (details of our test bed may be found in our previous work [13]). The size of the region is around $0.45km^2$. A map view of the deployment is shown in Figure 5. Distinct

Table 1: CityPlus and and City of Chicago data source details.

Dataset ID	Name	Instances
D1	Traffic Data 1	7306
D2	Traffic Data 2	8372l
D3	Traffic Data 3	14000
D4	Traffic Data 4	9397
D5	City Pollution Data	11569
D6	Weather Data	16369
D7	Parking Lot Data 1	55285
D8	Parking Lot Data 2	53267

pins on the map indicate different nodes in this map view. We put nodes to measure light levels, ambient noise levels, ambient temperature, and traffic presence, particularly automobile presence. We placed the sensor nodes in a manner so that the nodes could get power all day and night. We attached a solar panel so that the node's backup batteries can charge during the night. We utilized smart phones as the edge processing nodes in our experiment since they are simple to construct AI-based agent programs for and link to the cloud server through a 4G network.

Figure 5: Web application interface prototype for real-time situation monitoring.

Table 2: List of job for the experiment.

Job ID	Name	Action
HT	Highest traffic area	$traffic(D, MAX)$
EP	List of empty parking lot	$empty\_parking\_lots(D)$
SD	List of sunny day area	$sunny\_areas(D)$
LP	Lowest pollution cities	$pollution\_areas(D, LOWEST)$

4.2 Edge Processing Performance

In order to test the intelligent edge based processing performance, we split our system into three approaches: i) without edge based processing; ii) with vertical offload based edge

processing; iii) horizontal offload based edge processing. In the next step, we execute the list of jobs that are listed in Table 2 on the data sets as listed in Table 1. Finally, Figure 6 depicts the comparison of execution times for cloud and edge-only processing. It should be noted here that the experiment was performed both in a low and high bandwidth scenario. The bandwidth were: average bandwidth of 1.4 MB/s (11.4 Mbits/s) with a standard deviation of 15.3. The minimum bandwidth was $B_{low} = 0.3MB/s$ (2.33 Mbits/s) and the maximum bandwidth was $B_{high} = 11.6MB/s$ (92.8 Mbits/s). Edge-based processing shows better results because less data must be transferred to the cloud server.

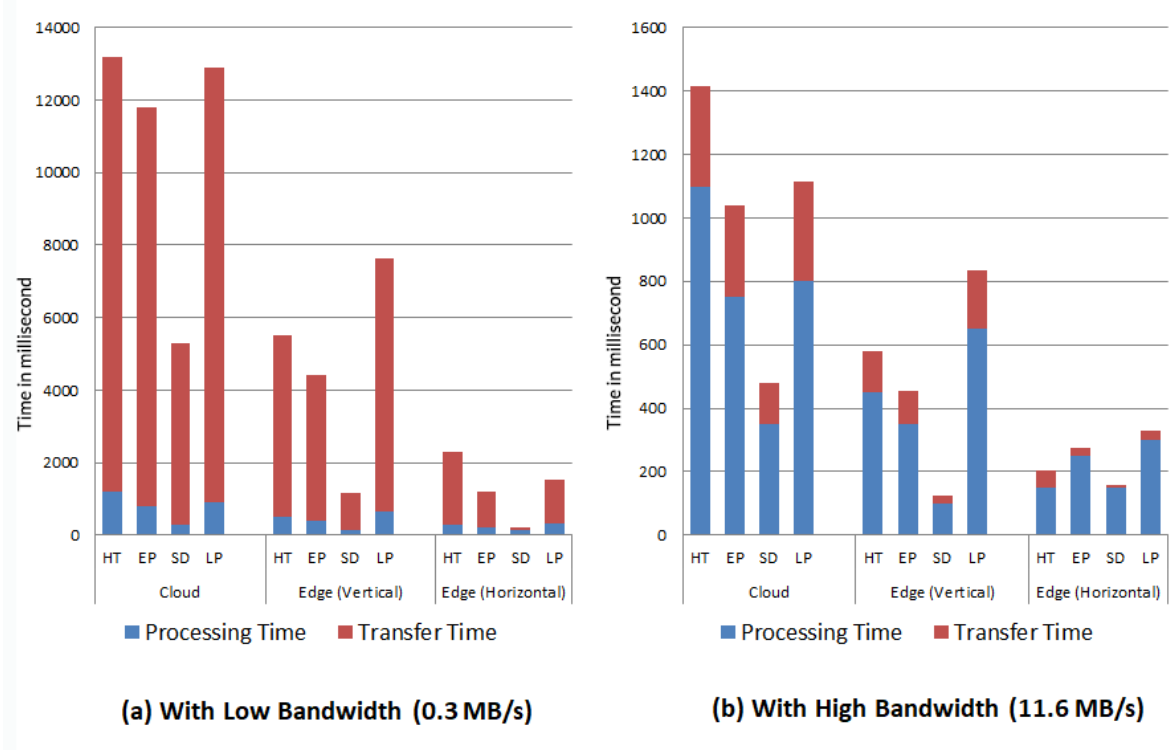

Figure 6: Comparison of execution times for cloud and edge-only processing.

In Figure 6 the horizontal axis indicates different jobs (as listed in Table 2). And the vertical axis indicates the execution time in milliseconds. In Figure 6(a) the red color bar indicates the data transfer time and the blue color bar indicates the data processing time in the case of low bandwidth. Similarly, Figure 6(b) the red color bar indicates the data transfer time and the blue color bar indicates the data processing time in the case of high bandwidth. If we see the figures carefully, we see that in the case of low bandwidth in our cloud only test, the data transfer is more than the data processing time. Although for job SD (List of sunny day areas), the processing and transfer time are less than the other three cases (HT, EP, and LP). This is because we have only one weather data set (D6, which has 16369 instances) compared to traffic data sets, which have four data sets (D1-D4). If we see sub figure (a) we can see that cloud-based processing time is more than the other two methods. If we compare the performance based on offloading technique, then horizontal offloading based edge processing takes less time, as in this case the edge layer is using a cluster and only refined data is sent to the coordinator node (see Figure 4) to the cloud. As a result, both processing and transfer times are reduced. If we go to the high bandwidth case, we see that transfer time is less as for bandwidth speed, the data transfer saves time. In this experiment, it is clear that our edge-based processing is

showing good performance in comparison to the traditional cloud-based processing.

4.3 Situation Detection Performance

To assess the performance of the situation detection approach, we monitored the environment for a few weeks and computed the number of events that happened vs the events detected by our system. Our prototype was able to recognize 176 occurrences out of about 181 distinct events such as high temperature, low humidity, and a scorching sunny day. Due to connection delays and server load, our algorithm may provide deceptive information at times.

5 Conclusion

The variety of Smart City applications and services has increased in recent years, ranging from public services and safety to smart traffic and smart surveillance. As mobile and cloud computing have advanced, a huge number of IoT devices have been connected to the Internet, creating billions of bytes of data at the network edge. As a result of this growth, there is an urgent need to extend AI's frontiers to the network edge in order to fully capitalize on the value of edge data. Edge computing, a new paradigm for shifting computing workloads and services from the network core to the network edge, has been suggested as a viable option to meet this requirement. In Smart Cities, the notion of Edge Intelligence (EI) is gaining popularity. However, EI research is still in its infancy and research groups would appreciate a dedicated venue for discussing recent advances in EI in Smart Cities. To that aim, we presented a novel intelligent edge processing system based on Situation Calculus for IoT-based Smart Cities. With our prototype, we conducted a number of quantitative and qualitative studies. Our research indicates that processing raw IoT data at the edge devices reduces latency and gives decision makers situational awareness. In the future, we hope to further examine the system in order to assess its effectiveness in more realistic scenarios. We think that the suggested edge computing architecture in an IoT-based smart city environment will steer IoT research in a new direction.

References

[1] Nurul Hidayah Ab Rahman, Niken Dwi Wahyu Cahyani, and Kim-Kwang Raymond Choo. Cloud incident handling and forensic-by-design: cloud storage as a case study. *Concurrency and Computation: Practice and Experience*, 29(14), 2017.

[2] Ricardo S Alonso, Inés Sittón-Candanedo, Óscar García, Javier Prieto, and Sara Rodríguez-González. An intelligent edge-iot platform for monitoring livestock and crops in a dairy farming scenario. *Ad Hoc Networks*, 98:102047, 2020.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[4] Mainak Bandyopadhyay, Maharana Pratap Singh, and Varun Singh. Formalization and development of logic based emergency response systems using situation calculus. In *2013 International Conference on Machine Intelligence and Research Advancement*, pages 504–508. IEEE, 2013.

[5] Bita Banihashemi, Giuseppe De Giacomo, and Yves Lespérance. Abstraction in situation calculus action theories. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[6] Chii Chang, Satish Narayana Srirama, and Rajkumar Buyya. Indie fog: An efficient fog-computing infrastructure for the internet of things. *Computer*, 50(9):92–98, 2017.

[7] Pushpinder Kaur Chouhan, Liming Chen, Tazar Hussain, and Alfie Beard. A situation calculus based approach to cognitive modelling for responding to iot cyberattacks. In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pages 219–225. IEEE, 2021.

[8] Dana Cuff, Mark Hansen, and Jerry Kang. Urban sensing: out of the woods. *Communications of the ACM*, 51(3):24–33, 2008.

[9] Yucong Duan, Zhihui Lu, Zhangbing Zhou, Xiaobing Sun, and Jie Wu. Data privacy protection for edge computing of smart city in a dikw architecture. *Engineering Applications of Artificial Intelligence*, 81:323–335, 2019.

[10] Mica R Endsley. Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society annual meeting*, volume 32, pages 97–101. SAGE Publications Sage CA: Los Angeles, CA, 1988.

[11] Frieder Ganz, Payam Barnaghi, and Francois Carrez. Automated semantic knowledge acquisition from sensor data. *IEEE Systems Journal*, 10(3):1214–1225, 2014.

[12] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 9–16. IEEE, 2015.

[13] SK Alamgir Hossain, Md Anisur Rahman, and M Anwar Hossain. Edge computing framework for enabling situation awareness in iot based smart city. *Journal of Parallel and Distributed Computing*, 122:226–237, 2018.

[14] Jingwei Huang, Mark S Fox, and Michael Gruninger. Distributed trust reasoning in cyber-physical-social smart systems–a formalism in situation calculus.

[15] Andres Laya, Vlad-loan Bratu, and Jan Markendahl. Who is investing in machine-to-machine communications? In *24th European Regional ITS Conference, Florence 2013*, number 88475. International Telecommunications Society (ITS), 2013.

[16] Hector Levesque, Fiora Pirri, and Ray Reiter. Foundations for the situation calculus. 1998.

[17] Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

[18] Zhihan Lv, Dongliang Chen, Ranran Lou, and Qingjun Wang. Intelligent edge computing based on machine learning for smart city. *Future Generation Computer Systems*, 115:90–99, 2021.

[19] Nils J Nilsson. *Principles of artificial intelligence*. Springer Science & Business Media, 1982.

[20] City of Chicago. City of chicago open data. Technical report, https://data.cityofchicago.org/. Last accessed date: 26/05/2018.

[21] Dan Puiu, Payam Barnaghi, Ralf Tönjes, Daniel Kümper, Muhammad Intizar Ali, Alessandra Mileo, Josiane Xavier Parreira, Marten Fischer, Sefki Kolozali, Nazli Farajidavar, et al. Citypulse: Large scale data analytics framework for smart cities. *IEEE Access*, 4:1086–1108, 2016.

[22] Thomas Rausch, Waldemar Hummer, Vinod Muthusamy, Alexander Rashed, and Schahram Dustdar. Towards a serverless platform for edge {AI}. In *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[23] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[24] Smartsantander. Future internet research and experimentation. Technical report, http://www.smartsantander.eu/, 2016.

[25] Munsif Yousef Sokiyna, Musbah J Aqel, and Omar A Naqshbandi. Cloud computing technology algorithms capabilities in managing and processing big data in business organizations: Mapreduce, hadoop, parallel programming. *Journal of Information Technology Management*, pages 113–126, 2020.

[26] Ljiljana Stojanovic, Marko Dinic, Nenad Stojanovic, and Aleksandar Stojadinovic. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *2016 IEEE international conference on big data (big data)*, pages 1647–1652. IEEE, 2016.

[27] Brunno Vanelli, Madalena Pereira da Silva, Guilherme Manerichi, Alex Sandro Roschildt Pinto, Mario Antonio Ribeiro Dantas, Mauri Ferrandin, and Adao Boava. Internet of things data storage infrastructure in the cloud using nosql databases. *IEEE Latin America Transactions*, 15(4):737–743, 2017.

[28] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014.