Smart Transportation Delay and Resiliency Testbed based on Information Flow of Things Middleware

Jose Paolo Talusan
NAIST
Nara, Japan
talusan.jose_paolo.tg3@is.naist.jp

Francis Tiausas
NAIST
Nara, Japan
tiausas.francis_jerome.ta5@is.naist.jp

Keiichi Yasumoto
NAIST
Nara, Japan
yasumoto@is.naist.jp

Michael Wilbur
Vanderbilt University
Nashville, USA
michael.p.wilbur@vanderbilt.edu

Geoffrey Pettet
Vanderbilt University
Nashville, USA
geoffrey.a.pettet@vanderbilt.edu

Abhishek Dubey
Vanderbilt University
Nashville, USA
abhishek.dubey@vanderbilt.edu

Shameek Bhattacharjee
Western Michigan University
Michigan, USA
shameek.bhattacharjee@wmich.edu

***Abstract*—Edge and Fog computing paradigms are used to process big data generated by the increasing number of IoT devices. These paradigms have enabled cities to become smarter in various aspects via real-time data-driven applications. While these have addressed some flaws of cloud computing some challenges remain particularly in terms of privacy and security. We create a testbed based on a distributed processing platform called the Information flow of Things (IFoT) middleware. We briefly describe a decentralized traffic speed query and routing service implemented on this framework testbed. We configure the testbed to test counter measure systems that aim to address the security challenges faced by prior paradigms. Using this testbed, we investigate a novel decentralized anomaly detection approach for time-sensitive distributed smart transportation systems.**

***Index Terms*—Middleware, Distributed computing, Transportation**

I. Introduction

Internet of Things (IoT) are physically inter-connected devices embedded with software, sensors and network connectivity that enables them to both share and access information. In today's world, the number of IoT devices is expected to reach 50 billion by 2023 [1] and radically transform aspects of daily life such as health care and transportation systems. In order to manage IoT networks and the large amounts of data they produce, a combination of cloud, edge and fog computing paradigms are increasingly being studied and leveraged.

Quality-of-Service (QoS) and Security are two key challenges in large decentralized IoT networks. QoS deals with the ability to provide services within an acceptable time frame. Security deals with resilience to unwanted interference and monitoring. Recently, IoT systems have been targeted due to (1) the sensitive information they handle, and (2) their use in launching large-scale Distributed Denial of Service (DDoS) attacks [2]. Compromised devices affect the system's QoS as resources like network bandwidth are diverted towards unwanted tasks. Detecting these kinds of attacks is therefore important.

Traditional networks use Intrusion Detection Systems (IDS) to guard against such attacks. IDS's deploy special devices at key points in the network to collect and analyze network traffic which can then be used to detect attacks [3]. Similar approaches have already been proposed and implemented in IoT, aiming to detect different kinds of security threats.

IDS's provide network layer security to an IoT system, allowing middleware to focus on authentication and authorization at the application layer. Middleware are often flexible enough to provide additional data that can also help in intrusion detection. These metadata can be passed alongside regular information, at the cost of some network overhead. This cannot be quantified well without an actual network in place. Thus, an emulation testbed is necessary to study the effects of such security measures.

Such a testbed can potentially be used to study the effects of different attacks (e.g. DoS attacks, botnet infection, etc.) on the network and its QoS. We investigate a novel decentralized anomaly detection approach for time-sensitive distributed smart transportation systems with a focus on data-integrity attacks and implement it in a testbed. Our method is based on related work in power systems [4], in which we extend in two distinctive ways. First, we extend the work for time sensitive applications. Secondly, we implement this approach in a decentralized network architecture.

The testbed would also need to be built into an IoT middleware platform which is capable of handling potentially large amounts of data in near real-time. Said platform would also need to be edge-based to minimize data transfer time. Combining an edge-based platform with such a testbed would be a novel approach in this case.

This paper describes the Information Flow of Things (IFoT) [5] framework and develops a network delay emulation testbed based on the IFoT middleware. The end goal is to be able to test proposed security countermeasures for the issues detailed above. A smart traffic routing service is deployed on the platform to test its basic distributed processing capabilities, and then data is obtained from the integrated emulation testbed to quantify the overall delay introduced to the system.

II. Related Work

A. Edge, Fog, Cloud Computing

Due to the large amount of data produced by billions of IoT devices, traditional centralized cloud servers will eventually face the problem of non-negligible delays when providing IoT-related services. Edge [6] and Fog [7] computing are approaches to mitigating the reduced quality and increased service costs of cloud computing.

Edge and fog computing are both "edge-heavy computing" paradigms where data processing is executed on components in or near the data source. However, the demerit of these approaches is the investment needed to replace such network constituents like Information-Centric Networks (ICNs). In this paper we propose a more practical solution which extends edge and fog computing by delegating the processing to the devices at the source in a distributed manner.

B. Smart Mobility

These computing paradigms have been leveraged to process and visualize data from sources such as road side units (RSU) and Vehicular Social Networks [8] to provide services such as Intelligent Transportation Systems (ITS) already present in Japan [9]. These are able to provide users with real-time wide-area traffic congestion information. One such application, SpeedPro [10], uses GPS location data fused with historical data to provide more reliable urban traffic speed estimates.

While Edge and Fog computing are promising for these applications, a number of challenges still exist that must be addressed. Eisele et al. [11] state that one challenge is to be able to provide a stable application environment despite the dynamism, heterogeneity, and increased failure potential of computing resources at the "edge" away from data centers.

Security and privacy [12] are also points of concern. Due to their reliance on spatio-temporal data, measures need to be taken in order to preserve data integrity and to detect anomalies within such systems [13]. A large focus of research in this field is on implementation of sensor systems for transportation, communication and infrastructure monitoring [14], [15], [16], [17], [11]. Traditional anomaly detection in this context is based on classification, statistical, state based, clustering or information theory [18]. Classification methods are usually based on Support Vector Machines (SVM), Bayesian Models, Gaussian Processes or Neural Networks [19].

C. Information Flow of Things

Information Flow of Things (IFoT) is a proposed framework aimed towards processing massive IoT data streams in real-time manner by edge servers and IoT devices [5]. It is designed to provide delay-aware services through mechanisms such as in-situ distributed computing and data aggregation. It aims to have a better cost-performance index than cloud-based and edge-based approaches [20]. The goal is to achieve an improved satisfaction level for delay-sensitive applications (such as smart city or smart mobility) while being able to aggregate user data in a secure and timely manner with a certain level of robustness against privacy and security threats.

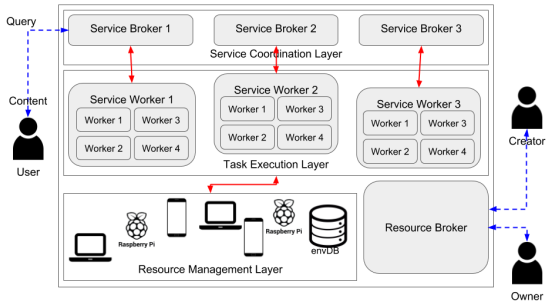

Fig. 1. IFoT Framework Architecture

III. In-Situ Distributed Processing Platform

To satisfy the requirements of the IFoT framework, we are developing a middleware platform [20] which allows services to be created by Service Creators. These services utilize the spatio-temporal data generated by sensors and processes it into useful information. The middleware system is comprised of a Resource Broker and Service Brokers described in [20]. Only Service Workers, which are new from the previous literature, which execute tasks are created and assigned to each Service Broker is discussed.

A. Service Worker (SW)

Service Workers are clusters of nodes that are able to perform operations on sensor data, and handle the computational tasks required to provide services to users. Each node executes tasks locally adhering to the shared-nothing architecture. To meet certain quality of service agreements, more nodes can be added to the cluster in order to scale up performance.

These are supported by three mechanisms: Environmental Database, Messaging Protocols and Task Graphs.

Environmental Database: store the data generated by sensors. These are time series DBs stored in the SWs.

Task Graph: are recipes that dictate how services are distributed and handled by the SWs. These contain instructions on how the SWs should collect, process and aggregate the sensor data for a particular service. These are generated by the SB taking into account service level agreements and QoS requirements to maximize the use of available nodes.

Publish-Subscribe-based Messaging Protocol (MQTT): used to facilitate communication between devices. Task graphs, heartbeat monitoring, and data for aggregation are sent via MQTT to the participating nodes (such as between SWs and SBs, or between SBs and RBs).

IV. Smart Mobility Service

In this section, we present a service that takes advantage of the IFoT middleware platform.

A. Smart Mobility Service

We assume that smart cities will feature roads and highways equipped with road-side units (RSUs). These RSUs receive information from vehicles such as speed. They are assumed to be devices with computational resources equal to those of

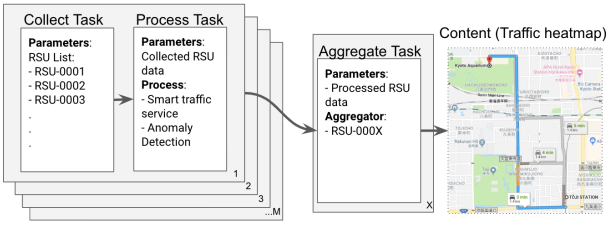

Fig. 2. Service Broker generated Task Graph for M clusters

Raspberry Pis or similar, and have their own wired network infrastructure allowing communication with each other.

RSUs connected to the IFoT middleware will host a smart mobility service that utilizes gathered data and publishes information about accidents, hazards, detours to its users. The service will also be able to respond to queries from users regarding the best routes for travel given the current situation.

Due to the spatio-temporal data being collected and processed by this service, it takes advantage of the properties of the IFoT framework for distributed computing. The middleware also allows security measures to be easily implemented within the service.

Other use cases While the service described above focuses on smart mobility, the infrastructure provided by the middleware can take advantage of any spatio-temporally distributed data. Distributed processing in this case decreases latency leading to improved QoS. The middleware also provides methods for introducing new services using various task graphs.

B. Details of the Task Graph

Task graphs dictate how services handle a user query, and are generated by the SB based on a particular service. Fig. 2 shows the task graph for the smart mobility service.

For this service, the task graph selects which RSUs will participate in the processing of the query. Selected RSUs would vary depending on the selected routes, desired QoS, delay requirements and the current load of RSUs. Afterwards, the tasks are distributed and executed.

1) Collecting Task: The collecting RSU will query vehicular traffic data from other RSUs specified in the task's parameters. Once done, collected data will be distributed to other RSUs for processing.

2) Processing Task: This includes all distributed processing that must be done on the data to produce the required result. In the case of smart mobility, traffic data will be checked for anomalies and then processed to generate route contexts (e.g. average speed information over a time-window, etc.) and other information for the user.

3) Aggregation Task: Once all RSUs have finished processing, their results will be aggregated by one RSU and returned to the SB for visualization.

C. Resiliency

In terms of resiliency, we are primarily concerned with falsified data from orchestrated data-integrity attacks and hardware faults at RSUs and sensors. We define such attacks as scenarios

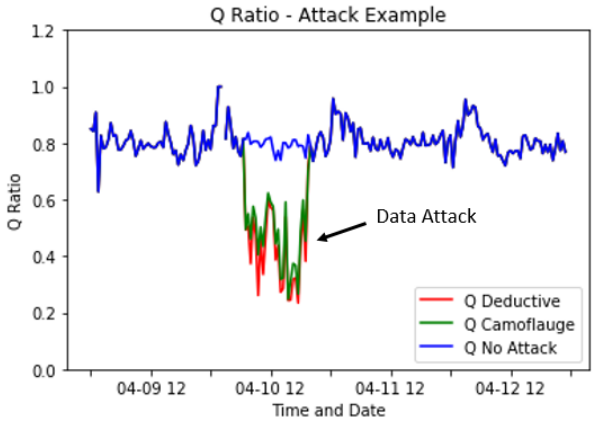

Fig. 3. Q Response to Attack

where an attacker can compromise a subset of sensors or RSUs by manipulating sensor readings. At each RSU, an anomaly detection check is run at a specified time window using a statistical means detection method based on Bhattacharjee et al.'s approach to data falsification in power grids [4]. This work extends their approach in two distinctive ways.

First, we extend their work to time-sensitive applications. In the case of transportation, such an attack can have cascading effects on traffic behavior throughout the system. Since these effects are rapid, the time between the start of the attack and detection is critical. We address this by using anomaly detection time windows ranging from 15 to 30 minutes. This method uses historical data to estimate the ratio of the harmonic mean to the arithmetic mean, which we refer to as the Q ratio. It was found on average that this process takes between one to two time windows to detect an attack.

Secondly, their work focuses on large-scale data. We extend this analysis to distributed transportation networks in which anomaly detection is run independently at each RSU and applied in the IFoT middleware. Using the testbed, we are able to quickly simulate and investigate the effects of various attacks on smart transportation networks, as well as the effects of the anomaly detection on network performance.

Hardware faults at sensors result in missing data readings at each time window, effectively simulating a large deductive attack. Thus, this model inherently extends to system failures in addition to data-integrity attacks.

As shown in Fig. 3, this metric proves to be stable over time, and responds quickly to simulated data-integrity attacks. This stability allows for easy integration with efficient sequential on-line anomaly detection methods as well as historical threshold methods. As anomaly detection done in this way does not depend on the other RSUs in the network, the detection process can be distributed throughout the network.

This approach provides numerous benefits over traditional anomaly detection, including simplified deployment over decentralized IoT networks. Additionally, the statistical means approach is computationally efficient compared to traditional anomaly detection methods such as Support Vector Machines

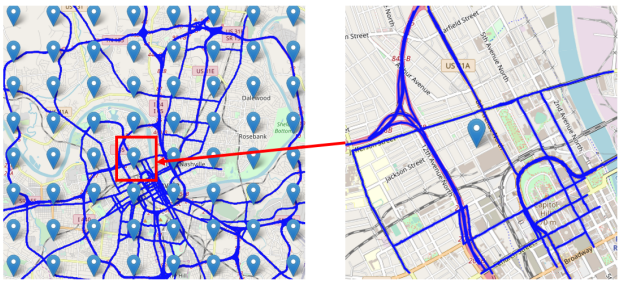

Fig. 4. RSU Locations - Grid Layout

(SVM), Bayesian Models, Gaussian Processes and Neural networks which require large scale, accurate models of system behavior and significant processing power. Statistical methods such as this have shown to be a more computationally efficient alternative [4]. This is particularly important for deployment on resource-constrained IFoT devices.

D. RSU Location Considerations

An important component of designing a smart city IFoT framework is in the placement of RSUs throughout the transportation network. In this sense, the number of RSUs available is a resource constraint in designing smart transportation grids. The major challenge is therefore determining the optimal spatial layout of these devices.

Optimal can be defined in any number of ways depending on device constraints and system goals. By designing a distributed testbed, optimal parameters such as delay and security can be compared between various system configurations through simulation. This reduces network design time and provides analytic feedback regarding expected system performance.

In the context of the transportation example, we focus on RSU layout in terms of network transmission delay and data security. We consider the case where each RSU is responsible for a subset of sensors streaming speed data into it. Thus, the RSU location problem is how to efficiently map these sensors to RSUs. The network layout is constrained by the number of RSUs available and the processing power of each device, corresponding to the number of sensors it can feasibly handle.

As a baseline RSU layout configuration we divide the city into 8×8 grids, resulting in 64 RSUs as shown in Fig. 4. A detailed investigation of delay performance for this configuration is provided in a later section.

Optimal RSU configurations can also be framed in terms of data-integrity resiliency. The effectiveness of the anomaly detection discussed in Section IV-C increases for RSU zones consisting of sensors with traffic patterns similar and dependent on each other. The grid layout provides a good proxy for grouping dependent sensors together. We look to improve this by providing a constrained hierarchical clustering approach in which sensors are grouped together by historical traffic pattern. To maintain network performance for data transmission between sensors and RSUs, we constrain the clustering procedure geo-spatially by restricting the maximum distance sensors can be from an RSU. Additionally we set a maximum number of sensors allowed per RSU in proportion to RSU processing capability.

V. Implementation

In this section we discuss how the platform is implemented on a testbed and how a service is deployed. To realize a testbed based on this architecture, it must meet the following requirements: (1) should be easy to deploy on heterogeneous IoT devices and should able to deal with heterogeneous data streams, (2) should have an area-by-area aggregation mechanism for spatio-temporal data streams, and (3) should be able to provide results in a timely manner, taking into account communication and processing delay between devices.

A. Testbed Implementation

As the platform should be easy to deploy on heterogeneous devices, it was initially implemented on Raspberry Pis with Debian using Docker for ease-of-deployment.

We developed a testbed to implement and test the middleware using various configurations. The platform could be deployed on the testbed to mimic a large number of nodes, simulated on a single 2019 Mac mini with 6-core 3.0 GHz i5 processor and 64GB of RAM. Each *SB* and *SW* is virtualized as a Docker service. To simulate constrained computation resources like Raspberry Pis, each service is assigned a limited amount of CPU and memory via Docker.

B. Service Simulation

Each RSU is assigned to a node and given unique parameters to simulate real world deployment scenarios. Each one is given location (latitude and longitude) information as well as a unique ID for communication between RSUs. Each one also has an envDB that contains data received from the vehicles travelling along the road.

In our simulation, we divide a 80km^2 map of Nashville, TN into 8×8 grids, where each vertex corresponds to an RSU collecting data from vehicles in specific sections of the road network seen in Fig. 4. To evaluate the system, each RSU is set to behave as either a *SW* or normal Worker. RSUs are grouped into clusters with a single *SW* and one or more Worker nodes. Data for the simulation uses the 2014 Nashville city road records [21] which collected speed data from vehicles travelling the roads. These roads contain sensors placed at specific traffic message channel (TMC) points which make up a segment of a road. A combination of speed data, TMC points and optimal RSU locations are used to determine which RSU will store which road's data in their envDB.

Connections between RSUs are assumed to use wired Ethernet connection. In order to simulate the real world work flow of this service, users are able to query the platform through the Service Broker's web interface.

It is assumed that the SB determines the route the system will select in response to the query of the user. The system represents the variations of these routes as the variations of the number of clusters and workers within the cluster. Once the user has successfully sent a query to the SB, the execution

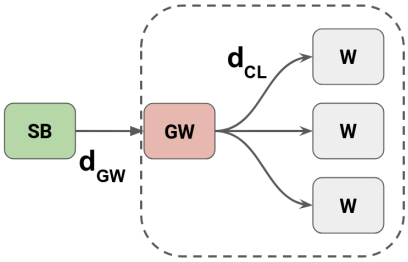

Fig. 5. Injection Points for Delay Components in Simplified System

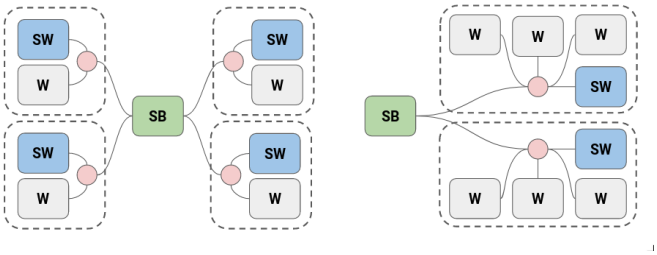

Fig. 6. Different network configurations for the system

timer starts and the task graph for the service is sent to these clusters.

C. Delay Emulation

Since the RSUs are simulated, data transfer delays are obtained via synthetic calculations and injected wherever such communications happen. For instance, after the collection task, the RSU handling it passes off results to other RSUs for processing. A short delay is injected here through a sleep function right before data is sent to each processing RSU. Since the simulated RSUs are also doing real processing on the data, delays due to processing will be left as is.

$$d_{SB \to SW} = d_{GW} + d_{CL} \tag{1}$$

Execution time measurement starts after the user query arrives at the *SB*. The *SB* publishes a message to the cluster gateway which then routes that message to the *SW*. This introduces a delay given by Eq. 1, where d_{GW} is the delay between the *SB* and the cluster gateway and d_{CL} is the delay between the cluster gateway and any one of the cluster's RSUs.

$$d_{W \to W} = 2d_{CL} \tag{2}$$

The *SW* then performs the task designated for it on the received task graph. Usually, it is given the collection task and the result - along with the task graph - is passed on to other RSUs. These then proceed to perform their designated tasks, using the results passed from the previous step. Passing data between RSUs introduces a delay as shown in Eq. 2.

$$d_{W \to SB} = d_{GW} + d_{CL} \tag{3}$$

Once all RSUs have finished processing, resulting data must be aggregated back at the *SB*. The delay for this is given by Eq. 3. Fig. 5 summarizes where these delay components are injected in a simplified system.

$$d_{comp} = d_{trans} + d_{prop} \tag{4}$$

Each delay component is broken down further into relevant parameters as shown in Eq. 4, where they are defined as:

$$\begin{aligned} d_{trans} &= \textit{data length}/\textit{bit rate} \\ d_{prop} &= \textit{link length}/\textit{propagation rate} \end{aligned} \tag{5}$$

The *link length*, *bit rate*, and *propagation rate* can be configured for the simulation, while *data length* is based on the actual quantity of data sent during run-time. For this simulation, the values used are shown in Eqs. 6 and 7.

$$d_{GW}: \begin{array}{l} \textit{bit rate} = 2Mbps \\ \textit{link length} = 17.6km \\ \textit{propagation rate} = 299.792 * 10^6 m/sec \end{array} \tag{6}$$

$$d_{CL}: \begin{array}{l} \textit{bit rate} = 2Mbps \\ \textit{link length} = 8.0km \\ \textit{propagation rate} = 299.792 * 10^6 m/sec \end{array} \tag{7}$$

VI. Evaluation

In order to evaluate the system's utility, we use it to measure the overall execution time of a service implemented over IFoT. Overall execution times of tasks processed on a distributed network can vary depending on the network architecture. In this simulation, the network can be configured in multiple ways as shown in Fig. 6. For example, with 8 total RSUs, Fig. 6a shows how we can split RSUs into 4 clusters, each having 1 RSU as the *SW* and 1 dedicated Worker. Fig. 6b shows how the same network can be configured as 2 clusters, each with 1 *SW*, and 3 dedicated Workers.

For example, the testbed can be used to identify which configuration will lead to the least overall execution time. Using a Command Line Interface (CLI), this experiment was repeated with varying combinations of clusters and workers within a cluster. Fig. 7 shows the total time for the different combinations of clusters and workers. X axis represents the different combinations with the following naming convention: *case_AAxBB_nodes* where *AA* is the number of workers in a cluster and *BB* is the number of clusters in the system.

As these results show, the testbed allows us to draw preliminary conclusions about how changing the configuration of the simulated network affects execution speed.

VII. Conclusion

Edge and Fog computing paradigms and IoT provide new opportunities for distributed processing and data analytics. However these paradigms still face challenges especially with respect to security and privacy.

In this paper we designed and developed a middleware platform that meet IFoT framework requirements. Compared to the current mechanisms, this middleware utilizes the computational capabilities of each node directly. This allows for

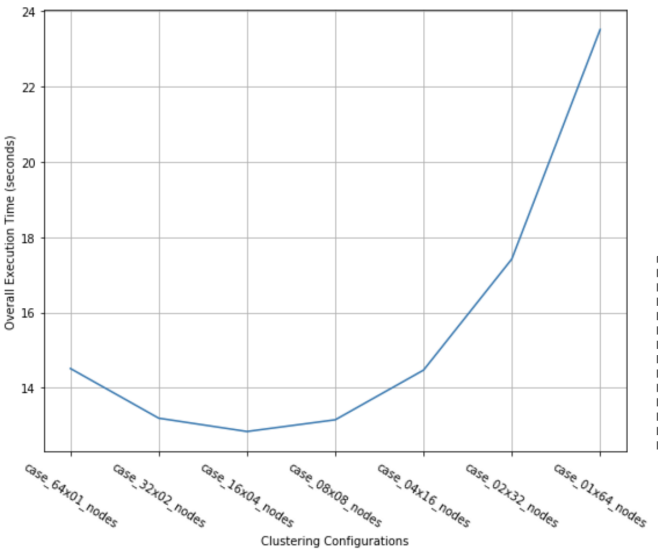

Fig. 7. Overall Execution Time vs Clustering Configurations

high availability and low latency communication. In addition, this middleware is suited for services that deal with spatio-temporal data. We also developed a testbed that is highly configurable and easy to deploy. A smart transportation service was developed and deployed on the testbed to demonstrate how the middleware deals with spatio-temporal data on multiple nodes. As an example, we show that the testbed can be used to analyze how the middleware can meet certain QoS level requirements by configuring its network architecture. Experimental security measures on top of the middleware could be evaluated in a similar way.

VIII. Acknowledgement

This work was in part supported by JSPS KAKENHI Grant Number 16H01721 and R&D for Trustworthy Networking for Smart and Connected Communities, Commissioned Research of National Institute of Information and Communications Technology (NICT).

References

[1] Iot: number of connected devices worldwide 2012-2025 — statista. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] D. Peraković, M. Periša, and I. Cvitić, "Analysis of the iot impact on volume of ddos attacks," in *33rd Symposium on New Technologies in Postal and Telecommunication Traffic (PosTel 2015)*, 2015, pp. 295–304.

[3] L. Santos, C. Rabadao, and R. Gonçalves, "Intrusion detection systems in internet of things: A literature review," in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2018, pp. 1–7.

[4] S. Bhattacharjee, A. Thakur, and S. K. Das, "Towards fast and semi-supervised identification of smart meters launching data falsification attacks," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 173–185.

[5] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of iot data streams," *Journal of Information Processing*, vol. 24, no. 2, pp. 195–202, 2016.

[6] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. Lin, "Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2018.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[8] Z. Ning, F. Xia, N. Ullah, X. Kong, and X. Hu, "Vehicular social networks: Enabling smart mobility," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 16–55, May 2017.

[9] Road bureau - mlit ministry of land, infrastructure, transport and tourism. [Online]. Available: http://www.mlit.go.jp/road/road_e/p1_its.html

[10] C. Samal, F. Sun, and A. Dubey, "Speedpro: A predictive multi-model approach for urban traffic speed estimation," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, May 2017, pp. 1–6.

[11] S. Eisele, I. Mardari, A. Dubey, and G. Karsai, "Riaps: Resilient information architecture platform for decentralized smart systems," in *2017 IEEE 20th International Symposium on Real-Time Distributed Computing (ISORC)*, May 2017, pp. 125–132.

[12] M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M. A. Ferrag, N. Choudhury, and V. Kumar, "Security and privacy in fog computing: Challenges," *IEEE Access*, vol. 5, pp. 19 293–19 304, 2017.

[13] S. Bhattacharjee, M. Salimitari, M. Chatterjee, K. Kwiat, and C. Kamhoua, "Preserving data integrity in iot networks under opportunistic data manipulation," in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, Nov 2017, pp. 446–453.

[14] G. P. Hancke, G. P. Hancke Jr *et al.*, "The role of advanced sensing in smart cities," *Sensors*, vol. 13, no. 1, pp. 393–425, 2012.

[15] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart cities of the future," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 481–518, 2012.

[16] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework," in *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE, 2012, pp. 2289–2297.

[17] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[18] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[19] F. Sun, A. Dubey, and J. White, "Dxnatdeep neural networks for explaining non-recurring traffic congestion," in *Big Data (Big Data), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2141–2150.

[20] Y. Nakamura, T. Mizumoto, H. Suwa, Y. Arakawa, H. Yamaguchi, and K. Yasumoto, "In-situ resource provisioning with adaptive scale-out for regional iot services," in *Proceedings of the Third ACM/IEEE Symposium on Edge Computing (SEC 2018)*, 2018, pp. 203–213.

[21] (2018) Here api. [Online]. Available: https://developer.here.com/