FlowCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN

Pan Wang*, Shuhang Li*, Feng Ye†, Zixuan Wang* and Moxuan Zhang‡
*School of Modern Posts, Nanjing University of Posts & Telecommunications, Nanjing, China
†Department of Electrical & Computer Engineering, University of Dayton, Dayton, OH, USA
‡Schools of International Education, Jinling Institute of Technology, Nanjing, China
Email: *wangpan@njupt.edu.cn, *lish@runtrend.com.cn, †fye001@udayton.edu, *wangzx@runtrend.com.cn
, ‡zhangmoxuan_7@126.com

***Abstract*—With more and more adoption of Deep Learning (DL) in the field of image proessing, computer vision and NLP, researchers have begun to apply DL to tackling with encrypted traffic classification problems. Although these methods can automatically extract traffic features to overcome the difficulty of traditional classification methods like DPI in terms of feature engineering, a large amount of data is needed to learn the characteristics of various types of traffic. Therefore, the performance of classification model always significantly depends on the quality of dataset. Nonetheless, the building of dataset is a time-consuming and costly task, especially encrypted traffic data. Apparently, it is often more difficult to collect a large amount of traffic samples of those unpopular encrypted applications than well-known ones, which leads to the problem of class imbalance between major and minor encrypted application in dataset. In this paper, we proposed traffic data augmentation method called FlowCGAN using Conditional GAN, one of a genre of Generative Adversarial Network (GAN) . As a generative model, FlowCGAN takes the advantage of GAN to generate new traffic samples by learning the characteristics of the traffic data and thereby balancing between major and minor classes of the data set. To verify the feasibility and evaluate the performance of this method, Convolutional Neural Networks(CNN) was designed to classify three datasets: the original unbalanced dataset, the dataset based on Random Over Sampling (ROS) method and the dataset based on FlowCGAN respectively. The experimental evaluation results show that the FlowCGAN method can achieve better performance than the other two in terms of encrypted traffic classification.**

***Index Terms*—encrypted traffic classification, deep augmentation, Conditional Generative Adversarial Network, traffic identification, class imbalance**

I. Introduction

As an important part of Smart Grid, IoT application mainly includes status monitoring of power transmission equipments and substation power environment, electric vehicle charging and discharging operational management, intelligent management of electric power supply, power asset management, electricity information collection, etc. In recent years, with the rapid development of Smart Grid and exponential growth of IoT nodes, e.g., smart meters, status monitoring sensors, traditional Cloud Data Center computing paradigm is unable to meet the requirements of IoT application in Smart Grid, including high bandwidth requirements, latency-sensitive and location-awareness. Fog Computing proposed by Cisco extends the Cloud Computing paradigm to run geodistributed applications throughout the network [1]. In contrast to the Cloud, the Fog not only performs latency-sensitive applications at the edge of network, but also performs latency-tolerant tasks efficiently at powerful computing nodes at the intermediate of network. At the upper layer of the Fog, Cloud Computing with data centers can be still used for deep analytics. Many related works have been carried out recently. Mohammad Aazaml et al. [2] proposed a Fog Computing and Smart Gateway Based Communication for Cloud of Things. Mohammad Abdullah Al Faruque et al. [3] brought forward an energy management platform based on Fog Computing architecture. Mohamed Saleem Haja Nazmudeen et al. [4] introduced a distributed processing framework for data aggregation based on fog computing architecture. Feyza Yildirim Okay et al. [5] presented an Smart Grid model based on fog computing. The model is divided into Smart Grid layer, fog layer and the cloud layer from bottom to top. Smart Grid layer mainly consists of smart meters, smart appliances and other intelligent equipments. The fog layer consists of multiple fog computing nodes. The cloud layer is mainly responsible for data storage, analysis and mining. Based on the concept of fog computing someone proposed a portable data storage and processing solution applying to Advanced Metering Infrastructure (AMI) [6]. In addition, many research works about programming model about IoT application have been carried out in recent years. I.Satoh [7] proposed a framework for data processing at the edges based on Mobile Agent and mapreduce. S.Cherrier et al. [8] introduced a distributed logic for IoT services based on OSGi to improve the modularizaion programming. K.Hong et al. [9] brought forward a programming model called Mobile Fog for large scale applications on IoT to try to develop IoT application by fog computing. However, existing schemes proposed before can not meet the new requirements of IoT application in smart grid, especially distributed coordination between fog computing nodes.

There are two main contributions in this paper. Firstly, we propose a new distributed Fog Computing architecture for IoT application in smart grid. To improve the application latency, we integrate distributed coordination capacity called FCC(Fog Computing Coordinator) in our architecture, which gather information of FC nodes in the same area periodically. In addition, FCC also devotes itself to assigning jobs to FC nodes so that all nodes can fulfill some complex tasks collaboratively.

Secondly, a programming model is proposed to realize the architecture.

The remaining of the paper is organized as follows. The new Fog Computing architecture is presented in Section 2. In Section 3, the programming model corresponding to the architecture is disscussed. Section 4 presents the evaluation of our architecture and programming model. Finally, Section 5 draws conclusion of this paper.

II. Fog Computing Based Architecture for IoT application in Smart Grid

A. The new requirements for IoT application in Smart Grid

In this section we will identify several requirements that need to be taken into consideration to effectively deploy IoT application in Smart Grid.

1) Latency Sensitivity: Many IoT applications in Smart Grid depend on instant decisions and even second level latencies are not tolerable [10]. For example, electric substations in smart grid systems are equipped with various sensors to monitor status of power transmission. In this scenario, any latency may lead to serious accident like power failure.

2) Distributed Coordination: There are always a lot of sensors distributed in geographic area, e.g., charging piles for electric vehicles. It is very important that coordinating multiple sensors or nodes distributed in several areas to provide electric vehicles charging services of IoT application in Smart Grid.

3) Locations awareness and Mobility Support: In many IoT applications end devices of Smart Grid are mobile and geographic distribution such as electric vehicles. As the main goal of the Fog Computing is to move computing power close to where the data is generated, it is necessary to be able to aggregate data at the closest network element while the end devices are moving.

B. Fog Computing Based Architecture for IoT application in Smart Grid

As shown in Fig. 1, our fog computing based architecture for IoT application in smart grid is still divided into terminal layer, fog layer and cloud layer from bottom to top.

1) Terminal nodes layer: This is the bottom layer which is made up of smart devices, which are responsible for sending raw sensed data and event logs to upper layer.

2) Fog layer: The middle layer consists of fog nodes deployed at the edge of network to extend the processing ability of cloud center. Compared with the traditional Fog Computing model, our fog layer is divided into fog nodes(FN) sub-layer and fog nodes coordination(FNC) sub-layer. With the ability of computing and storage, these fog nodes of FN sub-layer provide a mechanism for migrating processing logic to the edge of the network. The FN sub-layer also has the aggregation capability for the sensed data from terminal nodes layer. After gathering and analyzing raw sensed data, part of them directly feed back to the active nodes in terminal nodes layer to complete the real-time response and process to the emergency event, the other part is transmitted to FNC sub-layer. FNC sub-layer consists of multiple coordinators located in the geographical areas. Fog nodes are divided into several clusters, where there are a few equipments with computing and storage selected by some principle. We call this equipments FCN(Fog Computing Coordinators), which focus on coordinating the fog nodes to deal with some complex tasks due to the problem of distributed collaboration during the service, for example, query a suitable charging station for moving electric vehicles. In addition to undertaking data analysis and processing of the sub-region, the coordinator is responsible for coordinating all fog computing nodes in the region to further improve application performance by using the parallel computing capabilities.

3) The cloud layer: This layer is the upper layer in this architecture. It is composed of servers, such as Data Centers which are responsible for analyzing massive historical data.

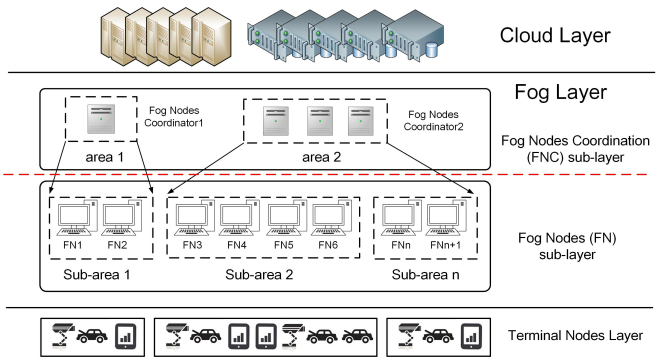

Fig. 1: Fog Computing Based Architecture for IoT application in Smart Grid

C. System Model

As shown in Fig. 2, the system model consists of sensors, action devices, communication nodes, fog computing nodes, cloud computing servers, FNC, service orchestration and scheduling servers called OSS servers. It is different with the system model proposed in papers, we introduce FNC sub-layer composed by FNC from each area. FNC accept requests from TN and decompose the services data flow according to the resource usage and service flow capacity of each FN and cloud computing servers, and then dispatch the jobs execution to related Fog nodes. Finally, FNC collect all the execution results to make the final decisions and instructions to the action devices. FN within the same layer have to achieve a complete user request under the coordination of FNC because there are no direct interaction between them. The interaction between FN and FNC can be realized by the way like relay communication, or Flow Table in SDN controller [11].

Besides, we introduce OSS servers in Cloud Computing Center, which are able to decompose services data flow based on resource usage and capacity collecting from computing nodes. OSS servers dispatch job execution images to computing nodes by the way like traditional virtual machine with better security isolation or Docker container [12] with less start latency. OSS servers mainly aim at initialization deployment of new applications provided by service providers

and setup service related execution logic on computing nodes. In contrast, FNC provide application services that meet the functionality and QoS requirements for end users by resource allocation and service logic coordination deployed in various computing nodes after receiving service requests from end users.

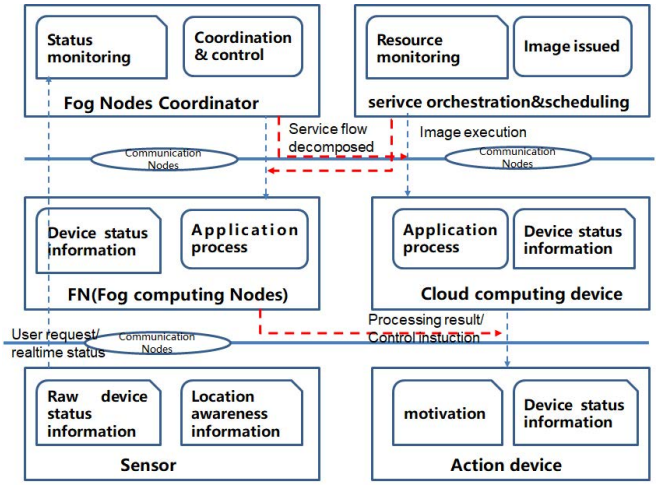

Fig. 2: system model

III. Programming Model

A. Programming Model Introduction

The IoT application in Smart Grid can be considered as a distributed application system essentially. However, the traditional programming model of distributed system was based on the model of "request-response" message interaction, which cannot meet the requirements of realtime processing large amount of data generated by devices. Therefore, we need to design a programming model based on Data Flow programming [13]. The typical programming framework is WoTKit processor based on WoTKit platform [14] and NR of IBM [15]. WoTKit is developed on JAVA Spring framework. Developer can run data flow program by creating wire between modules. However, WoTKit is designed for deployment of server level, it is better that using NR framework in IoT application, which was designed for application development in single computing unit, including nodes of input, output, processing and visual developing environment based on Web.In this paper [16], author added some feature about distribution to the traditional NR framework so that a data flow can be deployed in several computing nodes in the manner of multiple data flow slices.

B. Distributed Coordination Dataflow Programming model

Distributed Coordination Dataflow Programming model is shown in Fig. 3. Under the control of FNC, cloud servers and FN distributed in different geographic area together accomplish data analysis and processing of application services. There are two types of computing nodes, one is FN with rich computing resource,which choose Node-Red as distributed data flow computing framework. The other is FN with limited resource, which choose uFlow [17] as flow processing framework.

There are resident processes in every distributed computing nodes, which are responsible for collecting information like resource and capacity and then reporting to upper layer so that FNC will make better decision and instruction. After receiving substream and data ready to process, FN translate these data flow into instructions that can be identified and executed in terminal nodes. For example of parking service of electric vehicles, when a vehicle under the control of a sub-area FN apply a vehicle parking service, FNC of this sub-area will dispatch the request to all FN of this sub-area. After distributed coordinative processing of all FN, FNC can provide a best parking information for vehicles. As for online monitoring service all the time, we should consider the collaboration of Cloud Layer.After processing of FN in local area, FNC should report all the statistical data to cloud servers for future analysis.

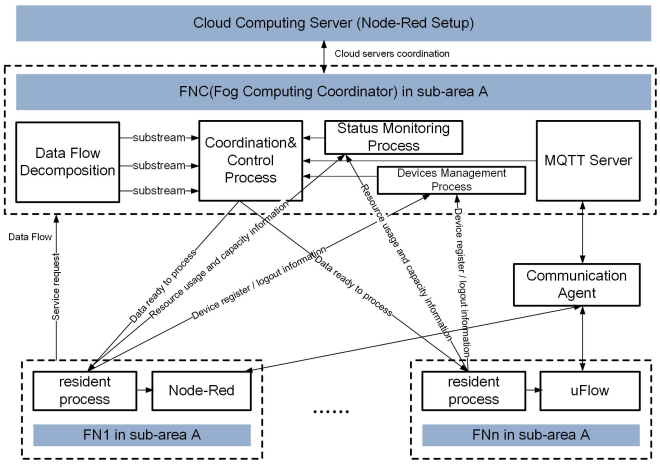

Fig. 3: programming model

Because of mobility of Terminal Nodes, the final data flow made by FNC may not suitable to current network condition, FN need coordinate themselves to adapt this situation at that time. That is called Fog Computing Nodes Migration. According to the initiator of migration, we can devide the migration into two types: one is initiated by FN; the other is Terminal Nodes. Two API used during the process of migration are following:

1)State on_migration_start(nodeID): Invoked by migration device before migration process start. It returns a state object for flow information waiting for migration.

2)Void on_migration_end(state_s): Invoked by migration target device after receiving request message from migration initiator.

The following is pseudocode describing the migration process for example of FN initiator.

1 procedure migration_source(nodeID)
2: obtain the actual latency T to node whose ID is nodeID
3: compare T with threshold Tupper
4: if T less than Tupper

5: return;
6: else
7: obtain the best candidate Vb from candidate group V
8: V = V - Vb
9: send a Start Migration message to node Vb
10: wait for response Resp
11: if Resp is ACCEPT then
12: S = on_migration_start(nodeID)
13: send Object State message to node Vb
14: release local resource of dataflow relate to nodeID
15: return
16: else
17: if V is empty
18: warn the message(can not migrate)
19: return
20: endif
21: endif
22: goto 7
23: endif
24: end procedure

Compared with QoS of Terminal Nodes and FN's default threshold during process of substream computing, FN will decide whether migrate or not, see line 2-5.The detail are following: 1) choose the best nodes from node group with proper capacity, resource and QoS requirements, see line 7-8; 2) Send a migration start request message to alternative nodes and wait for response, see line 9-10; 3) if alternative nodes accept, then call on_migration_start to get all status information of current nodes and send to alternative nodes. Meanwhile, free all related resources of data flow ready to migrate, see line 11-15; 4) if alternative nodes do not accept, send the warning message,like can not migrate, and quit. Otherwise, choose the next alternative node from node group and repeat above-mentioned. target nodes will call on_migration_end to take over following work after receiving migration status message from migration initiator.

IV. Evaluation

A. Simulation Setup

The electric vehicle intelligent service system is composed of electric vehicle, charging pile, regional coordinator, regional application server, cloud service center, communication proxy server and basic communication network. As sensing devices of network at the edge of network, electric vehicles report relevant realtime information and request for services, as well as to provide services to users. The charging pile device is located at the edge of the network as a fog computing device. It processes the data according to the established application logic, and transfers the processed data through communication proxy server to application server of the region or romote cloud service center according to control of area coordinator. The regional application server joined with cloud service center provides related services to users. The difference is that the regional application server is more emphasis on providing some geographically related or strict requirements of delay services (such as navigation, intelligent parking, etc.), and the cloud center server provides some long-term analysis and forecasting services. According to density of charging pile and the size of traffic flow, we set a certain service area, and set up a regional coordinator in each area. The coordinator completes the data flow chart and transmits datas to relevant devices according to the service requests from users and the resources, capabilities, location information reported by the equipments.It coordinates these devices to complete the related service logic. For services that require cross-domain provisioning, coordination is done by regional application servers in different regions.

This section provides an electric vehicle intelligent service experiment system which is used to simulate and analyze the performance of the fog computing architecture presented in this paper. This paper chooses IBM's NR framework as an implementation tool for application development, and deployes a stream-based micro-runtime environment uFlow over resource-limited IoT devices. In the uFlow environment, we use Lua as the programming language, MQTT as a communication protocol. In this experiment, in order to supply charging service to the electric vehicle, the system selects the most suitable charging pile to the electric vehicle. There are two cases: one is based on the traditional fog computing architecture, in this case electric vehicles directly transmit the requests to all charging piles in a certain range, after the calculation, the charging pile met returns the confirmation to the electric vehicle. The other case is based on the fog computing coordinator architecture, in this case electric vehicles will directly send requests to the fog computing coordinator in the area. According to the information of the charging piles, the coordinator will forward the request to some charging piles with the possibility of providing the service and then the response is returned to the electric vehicle by the final eligible charging pile.

B. Simulation Results

The experimental system is mainly used to evaluate the contrast of application latency between the traditional fog computing architecture and our architecture based on fog computing coordinator. We used 20 software terminal nodes running on embeded system as electric vehicles, 10 software Fog nodes as charging piles and 2 FNC nodes. The range of all entity is located about 2000 meters. Fig. 4 shows the relationship between the application latency and the query distance of the two architectures. Clearly, it can be seen that when the query distance is short, the application latency is similar. However, when the query distance is gradually increased, our architecture has a lower application latency. In Fig. 5, we discuss the relationship between the application delay and the number of service requests from the electric vehicles. Obviously, as the numbers of service request increase, the proposed architecture delay is significantly smaller than traditional one. Fig. 6 describes the relationship between the number of fog computing coordinator and application latency. When the numbers of coordinators increase, the application latency decrease significantly. As shown above, the proposed

architecture effectively reduces the application latency of IoT application in smart grid.

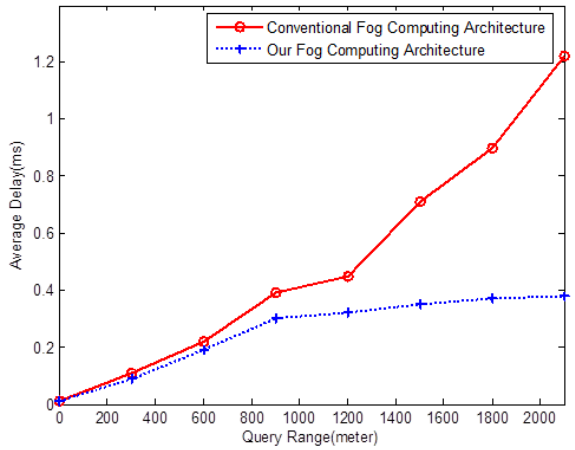

Fig. 4: application delay with query range

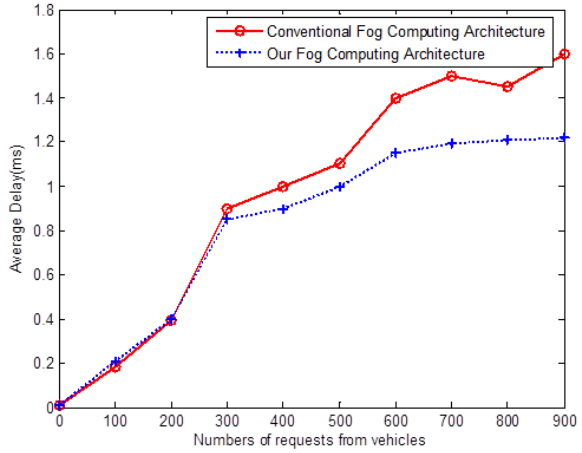

Fig. 5: application delay with service requests

V. Conclusion

The IoT applications in Smart Grid need higher requirements in terms of response time and location-awareness. It can be met well by distributed computing architecture based on fog computing (edge computing) as a supplement to the traditional "cloud - link - end" architecture. In this paper, the proposed distributed fog computing architecture and programming model for IoT applications in smart grid can effectively reduce service latency. The application development in this paper is mainly based on the Data flow mechanism to realize communication between devices. Next we will find a more appropriate communication protocol; and study the optimization of resource allocation algorithm in this architecture. At the same time we will carry out the studies about the handover between the mobile nodes in high-speed mobility.

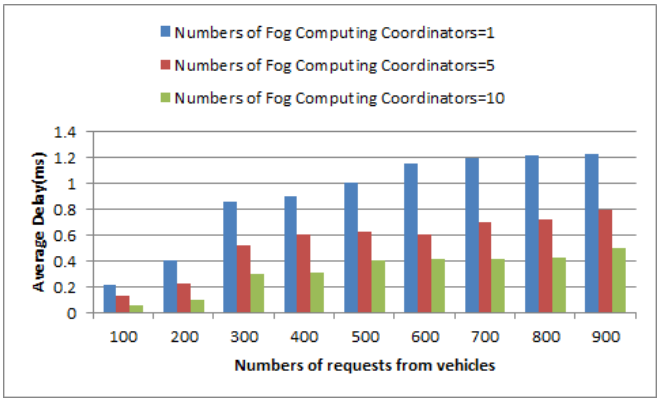

Fig. 6: application delay with FNC

Reference

[1] 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

[2] M. Aazam and E. N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *2014 International Conference on Future Internet of Things and Cloud*, Aug 2014, pp. 464–470.

[3] M. A. A. Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 161–169, April 2016.

[4] M. S. H. Nazmudeen, A. T. Wan, and S. M. Buhari, "Improved throughput for power line communication (plc) for smart meters using fog computing based data aggregation approach," in *2016 IEEE International Smart Cities Conference (ISC2)*, Sept 2016, pp. 1–4.

[5] F. Y. Okay and S. Ozdemir, "A fog computing based smart grid model," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, May 2016, pp. 1–6.

[6] Y. Yan and W. Su, "A fog computing solution for advanced metering infrastructure," in *2016 IEEE/PES Transmission and Distribution Conference and Exposition (T D)*, May 2016, pp. 1–4.

[7] I. Satoh, *A Framework for Data Processing at the Edges of Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 304–318. [Online]. Available: https://doi.org/10.1007/978-3-642-40173-2_25

[8] S. Cherrier, Y. M. Ghamri-Doudane, S. Lohier, and G. Roussel, "D-lite: Distributed logic for internet of things services," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, Oct 2011, pp. 16–24.

[9] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, ser. MCC '13. New York, NY, USA: ACM, 2013, pp. 15–20. [Online]. Available: http://doi.acm.org/10.1145/2491266.2491270

[10] Y. C. Chen, Y. C. Chang, C. H. Chen, Y. S. Lin, J. L. Chen, and Y. Y. Chang, "Cloud-fog computing for information-centric internet-of-things applications," in *2017 International Conference on Applied System Innovation (ICASI)*, May 2017, pp. 637–640.

[11] Y. Xu, V. Mahendran, and S. Radhakrishnan, "Towards sdn-based fog computing: Mqtt broker virtualization for effective and reliable delivery," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2016, pp. 1–6.

[12] H. J. Hong, P. H. Tsai, and C. H. Hsu, "Dynamic module deployment in a fog computing platform," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Oct 2016, pp. 1–6.

[13] W. M. Johnston, J. R. P. Hanna, and R. J. Millar, "Advances in dataflow programming languages," *ACM Comput. Surv.*, vol. 36, no. 1, pp. 1–34, Mar. 2004. [Online]. Available: http://doi.acm.org/10.1145/1013208.1013209

[14] M. Blackstock and R. Lea, "Iot mashups with the wotkit," in *2012 3rd IEEE International Conference on the Internet of Things*, Oct 2012, pp. 159–166.

[15] IBM Company, *Available from http://nodered.org/*.
[16] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, "Developing iot applications in the fog: A distributed dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, Oct 2015, pp. 155–162.
[17] T. Szydlo, R. Brzoza-Woch, J. Sendorek, M. Windak, and C. Gniady, "Flow-based programming for iot leveraging fog computing," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, June 2017, pp. 74–79.