Poster: Building IoT *Resilient Edge* using LPWAN and WiFi

Vijay Kumar
University of Bristol, UK

Poonam Yadav
University of York, UK

Leandro Soares Indrusiak
University of York, UK

ABSTRACT

The resiliency and reliability are two fundamental requirements of Internet of Things (IoT) deployments. For different IoT applications, these requirements vary from non-critical (best delivery efforts) to safety-critical with time-bounded guarantees. The network connectivity of the edge devices in IoT deployments remains the central critical component that needs to meet the time-bounded Quality of Service (QoS) and fault-tolerance guarantees of the applications that are running on the IoT infrastructure. To meet these requirements, we need to investigate a fundamental question: how to meet IoT applications mixed-criticality QoS requirements using state-of-the-art communication technologies? Therefore, in this work, we systematically investigate how to utilize multi-communication networks to meet these requirements.

1 INTRODUCTION

Many safety-critical IoT applications such as self-health monitoring through wearable IoT devices connect to a mobile phone/local hub via Bluetooth, ZigBee or Wi-Fi and send the data to a cloud service or hospital central processing system through Internet. In the event of a network-failure, e.g., power outage or any other incidental connection failures, the Wi-Fi could be disconnected temporarily for minutes to hours, resulting in either data loss or delayed data communication. However, for safety-critical applications, it is essential to maintain resilient data connectivity at all time for the delivery of a time-critical message. The Low power wide area network (LPWAN) technologies such as NB-IoT, LoRa, Sigfox have explicitly been designed to meet IoT application requirements such as improved battery life, power efficiency and indoor and outdoor coverage area [3] at an affordable cost.

However, LPWAN technologies have limitations in terms of limited bandwidth; the number of messages allowed per day and payload size. Therefore, in our work we investigate how the IoT application traffic can be routed using multi-communication networks supported by LPWAN technologies by taking into accounts their limitations and the IoT application requirements (e.g., message criticality (such as high/low priority), privacy settings, message data length, message frequency). In case of a particular network unavailability or failure, the application can be informed of the network state and can decide on the suitability of the network and adapt accordingly. For instance, assuming the application is sending data over Wi-Fi and because of power failure Wi-Fi is disconnected, the application can choose to send data over NB-IoT, LoRa, Sigfox and adapt parameters such as payload size and frequency accordingly. Despite the recent popularity of LPWAN technologies, it is not fully understood if we can achieve network resiliency at the Edge using LPWAN and Wi-Fi for time-critical IoT applications.

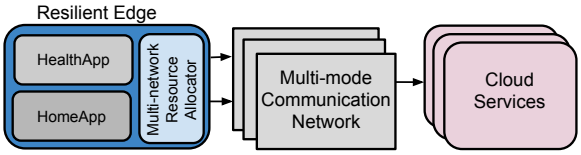

Figure 1: Resilient Edge End-to-end System.

Therefore in this work, we propose a hypothesis that using LPWAN technologies and Wi-Fi, we can achieve network resiliency at the edge IoT device by providing a capability to choose a suitable network m based on the application requirements.

2 SYSTEM MODEL AND A USE-CASE

We provide a usecase to better understand the network requirements of *Resilient Edge* applications. Figure 1 show an edge device running two sample applications to support assisted living facilities: one of the applications monitors the health of the resident (HealthApp), the other monitors their residential unit (HomeApp). To achieve continuous network connectivity needed by these applications, we make use of a multi-mode communication network. Our model allows system designers and administrators to decide how many levels of criticality $L = L_{max}$ to support, and then to allow the specification of the QoS requirements of each message flow at each of those levels. The *Resilient Edge*, as shown in Figure 1 is designed to support three levels of criticality, and the Table 1 shows the QoS required by each message flow at each level.

In normal operation (i.e. $L = 1$), message flows declare their most generous QoS requirements, with larger data volumes for home monitoring (e.g. including camera snapshots)

Conference'17, July 2017, Washington, DC, USA
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

	Criticality Level	1		2		3	
Applications	**Message Flow τ**	**C**	**T**	**C**	**T**	**C**	**T**
HealthApp	1 fall detection	1000	10	40	20	10	60
"	2 heart monitoring	1000	5	80	10	10	20
"	3 body temperature	30	30	10	120		
HomeApp	4 sensor bedroom	40000	10	10	30		
"	5 sensor bathroom	80	10	10	30		
"	6 sensor lounge/kitchen	40000	10	10	30		
"	7 sensor front door	40000	10	10	30		
"	8 energy usage	40	3600				

Table 1: Message flows on an edge device for assisted living facilities - C is maximum message size (bytes) and T is minimum interval between subsequent message (seconds). We assume that high criticality level messages are necessary to be delivered messages and are rare and have smaller size. In this example, for message flow 1, when criticality level 3 is requested and served, underlying network interface guarantees a message delivery service with message size of 10 bytes with 60 seconds subsequent message interval.

and resident monitoring (e.g. detail accelerometer data for fall detection, full (electrocardiogram) ECG data for heartbeat monitoring). The next criticality level (i.e. $L = 2$) allows the declaration of degraded QoS levels, which in this example is provided for all message flows except for the one monitoring energy usage (which will not be forwarded by the edge device in case of degraded service). Notice that the QoS requirements declared for $L = 2$ and show that monitoring will be performed less often and less data will be provided (e.g. simple movement detectors for home monitoring, average temperature and heartbeat for health monitoring). Finally, only two message flows declare QoS requirements at the highest level of criticality (i.e. $L = 3$), representing the alarms for fall or severe arrhythmia/cardiac arrest. In the case of degraded service, all available resources should be used to provide those two flows with their declared QoS.

Multi-Network Resource Management: We can formulate a criticality-aware QoS allocation problem for each message flow of each application, of its allowed criticality level of service and its allocated network interface. We propose the use of simple bin-packing algorithms and devise criticality-aware best fit (CABF) and its variant $CABF_{inv}$.

3 IMPLEMENTATION

The *Resilient Edge* prototype setup is shown in Figures 2. A Raspberry Pi model 4 [1] (RPi) is interfaced with Pycom FiPy [2]. FiPy provides connectivity to five different networks Wi-Fi, Bluetooth, LoRa, Sigfox and LTE (CAT-M1/NB-IoT). To enable the data transfer between RPi and FiPy, a message payload from the applications is written to the RPi UART and read by FiPy continuously. On FiPy, a python script checks the messages received from the RPi, the network interface assigned to the message flow, its criticality level and attempt to send it via that network interface.

Implementation details of RPi components: We use TCP/IP serial bridge to create a socket listening on port 8080 connecting to the UART (/dev/ttyAMA0) to send and receive data to and from the UART. Further, we use *python select lib* to monitors sockets for incoming data to be read and send outgoing data when there is room in the buffer and use message queues to store the outgoing messages. To simulate the message flows running on the RPi, we use threads to write a message payload on the socket. The thread sleeps for the period specified by the message flow before sending the next message. The FiPy runs multi-network resource allocator and sends an allocation message back to the RPi stating which message flows have been assigned with which criticality level.

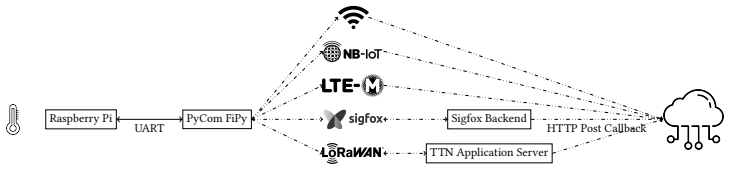

Figure 2: Block diagram of current experimental setup.

Implementation details of FiPy components: On FiPy, before assigning any network to a message flow, we need to create a network `"bin"` of the available networks (Wi-Fi, LTE (CAT-M1/NB-IoT), LoRa and Sigfox) and add the corresponding network interfaces to the network `"bins"`. As part of the Multi-network resource allocator - we implement variant $CABF_{inv}$ and set the initial parameters and perform the allocations of the message flows to the network interface.

Receiving messages on Cloud: To store the messages sent by the FiPy, we use Tornado to run an `HTTP` server on a machine hosted on a cloud. The `HTTP` server accepts `HTTP POST` messages and receives them directly from the FiPy via Wi-Fi, The Things Network (TTN) application server via LoRa, Sigfox backend via Sigfox and Pybytes via NB-IoT. The `HTTP` server checks for the `URI` and fetch the data from the post data and stores it in a `influxdb` database.

The resiliency and reliability requirements of IoT applications vary from non-critical (best delivery efforts) to safety-critical with time-bounded guarantees. In this work, we systematically investigated how to meet these applications mixed-criticality QoS requirements in multi-communication networks. Our work will help build reliable applications on IoT Edge and provide solutions from the perspective of communication networks to improve service quality and fault tolerance on resource-constrained edge devices.

REFERENCES

[1] Raspberry Pi. 2020. *Raspberry Pi 4 Model.* https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ Accessed:2020-06-30.

[2] Pycom. 2020. Fipy Experiment Board. https://pycom.io/product/fipy/. (2020). Accessed: 2020-06-30.

[3] Jesus Rubio-Aparicio, Fernando Cerdan-Cartagena, Juan Suardiaz-Muro, and Javier Ybarra-Moreno. 2019. Design and implementation of a mixed IoT LPWAN network architecture. *Sensors (Switzerland)* 19, 3 (2019). https://doi.org/10.3390/s19030675