Open Access

Open Journal of Internet of Things (OJIOT)
Volume 8, Issue 1, 2022

http://www.ronpub.com/ojiot
ISSN 2364-7108

Development and Evaluation of a Publish/Subscribe IoT Data Sharing Model with LoRaWAN

Juan Leon, Yacoub Hanna, and Kemal Akkaya

Department of Electrical and Computer Engineering, Florida International University,
10555 W Flagler St, Miami FL 33174, USA, {jleon148,yhann002,kakkaya}@fiu.edu

Abstract

Publish/subscribe architectures are becoming very common for many IoT environments such as power grid, manufacturing and factory automation. In these architectures, many different communication standards and middleware can be supported to ensure interoperability. One of the widely used publish/subscribe protocol is MQTT where a broker acts among publishers and subscribers to relay data on certain topics. While MQTT can be easily setup on cloud environments to perform research experiments, its large-scale and quick deployment for IoT environments with a widely used wireless MAC layer protocol such as LoRaWAN has not been thoroughly tested. Therefore, in this paper we develop and present a simulation framework in NS-3 to offer MQTT-based on publish/subscribe architecture that can also support LoRaWAN communication standard. To this end, we utilize NS-3's LoRaWAN library and integrate it with a broker that connects to other types of publishers/subscribers. We enable unicast capability from the broker to LoRaWAN end-devices while supporting multiple topics at the broker. We tested several scenarios under this IoT architecture to demonstrate its feasibility while assessing the performance at scale.

Type of Paper and Keywords

Experiments and Analysis Paper: *Publish/Subscribe model, LoRaWAN standard, IoT devices, MQTT protocol, NS-3 Simulation, Performance evaluation*

1 Introduction

In the last two decades, there has been a lot of research on IoT and wireless sensor networks that enabled collection of data from tiny/small devices and managing that data in a cloud environment for decision making. In such research, the main focus was to develop communication protocols at all layers of the protocol stack that will fit the requirements of IoT devices that relate to their limited resources in terms of processing, communication and energy power. As such, many of these protocols utilized a client-server model, where the IoT devices act as clients and supply data that will travel to the remote servers. Examples of these protocols include WiFi, ZigBee [19] and LoRa [8].

However, the large-scale availability of IoT devices and sensors as well as the need to perform computing and processing at the edge enabled the proliferation of a new model based on publish/subscribe system [10]. In other words, this model was a shift from client-server

This paper is accepted at the *International Workshop on Very Large Internet of Things (VLIoT 2022)* in conjunction with the VLDB 2022 conference in Sydney, Australia. The proceedings of VLIoT@VLDB 2022 are published in the Open Journal of Internet of Things (OJIOT) as special issue.

to a peer-to-peer paradigm which offers each IoT device to be a supplier and consumer of produced data. In recent years, we witnessed the proliferation of these type of approaches such as Message Queuing Telemetry Transport (MQTT) [11] that have been deployed in many practical applications from smart grid to factory automation.

Publish/subscribe architectures also bring with them interoperability solutions and thus allow integration of different communication standards with each other. This means any widely used communication standards for IoT such as LoRaWAN can also be utilized with MQTT. While this provides an enrichment for practitioners, there is still a need to be able to test MQTT performance under LoRaWAN support to offer deployment hints to the practitioners before any deployment effort. There are many use-cases for this purpose. For instance, smart metering infrastructure may need to use wide area wireless standards to collect readings every 15 minutes [14]. Similarly, there has been a great interest in smart agriculture applications where sensing data is collected regularly from the remote farm fields and irigation system needs to be actuated based on data sharing among other wireless devices [2]. This means, there should be a supporting tool for researchers to observe any challenges arising in terms of utilizing MQTT with specific wireless standards.

Unfortunately, many of the existing simulators do not offer LoRaWAN support in this context. For instance, while the widely used NS-3 simulator supports LoRaWAN, it does not come with integrated with a publish/subscribe architecture such as MQTT. This severely limits the experimentation capabilities for researchers who are interested in testing the performance of publish/subscribe architectures at scale. A perfect example is Smart Grid research where hundreds of distributed energy resources (DERs) need to exchange communication through such architectures (i.e., east-west communication). Another Smart Grid use case is energy trading [3] where IoT nodes need to exchange information in real-time using MQTT like protocols. Scalability limitations could be addressed with the use of ns-3 based simulation for a greater understanding of the performance of their proposed approach when at scale as well as the propagation delays and propagation loss introduced when using real communication technologies. As such there is a need to extend NS-3 to support such an architecture that can incorporate LoRaWAN in addition to other options.

In this paper, we first present our design and implementation of such a framework where MQTT features are integrated to LoRaWAN Module to support topic-based sharing of data among the devices in NS-3. In addition to formation of a broker node, we offer LoRaWAN end-devices to subscribe and publish data to everyone which also includes broadcasting and multicasting capabilities through the broker. We enable use of Class-C LoRa devices to be able to receive data at anytime by allowing the tuning of the duty cycle parameter for the LoRa Gateway. Furthermore, we implement the topics based on the extension of NS-3 TAG class.

We then use this framework to analyze the behavior of LoRaWAN under MQTT. For this purpose, we implemented an IoT application scenario within Smart Grid environments that can mimic exchange of data among SCADA and distributed energy (DER) field devices. In particular, we followed Open Platform Communications United Architecture (OPC UA) architecture [7] that is very much dependent on such a solution. We show how our system behaves at scale when we increase the number of publishers/subscribers and topics. This offers the first implementation and evaluation of LoRaWAN when it is used under a publish/subscribe architecture such as MQTT for IoT applications.

This paper is organized as follows. In the next section, we summarize the related work that considered MQTT-based testing efforts. Section 3 provides some background and preliminaries on the topic. In Section 4, we detail our design and implementation in NS-3. Section 5 presents some performance results under an OPC UA use case. Finally, we conclude the paper in Section 6.

2 Related Work

With the growth of the IoT devices and lightweight resources, efficient communication protocols are being sought to satisfy the demands of having an interoperability environment [21] [7]. Several studies have been done using different communication protocols to this end. For instance, the authors in [24] [16], defined the common and widely accepted protocols such as MQTT, HTTP, CoAP, and AMQP as well as performing an evaluation in terms of resource requirements, message overhead, latency, and reliability. Thus, based on the suitability and the requirements of IoT systems, MQTT was the leading option in terms of various metrics compared to the other protocols. Consequently, in recent years, many of the emerging applications that relate to factory automation, transportation, smart grid started to utilize these MQTT-based solutions for the transmission of sensor data to apps running on standard network infrastructure. To tackle the growing needs within MQTT, in [9] [23], the authors introduced an MQTT-S which consists of MQTT Broker, MQTT-S gateways, and clients. Their approach is optimized for sensor devices with restricted processing, limited storage as well as battery limitations.

On the data or MAC layer side, there are many standards which are offering a low-cost and low-power solutions such as LoRa, ZigBee, and Bluetooth Low Energy (BLE) [8] [4]. Among these for the wide area communication options, LoRa can be considered as one of the commonly used low power wide area network (LPWAN) technologies (LoRaWAN). IoT systems provide several features such as scalability, coverage, and robustness. However, it also brings some new challenges in terms of bandwidth, latency, and throughput [5]. Therefore, there is a pressing need to use various evaluation tools that will realistically assess the performance of these offerings. While there are prior works that have evaluated LoRa physical and LoRaWAN performance [27] [20], this has not been in the context of a publish/subscribe model.

In this paper, we implement and simulate a publish/subscribe model supporting LoRaWAN architecture that will offer a wide range of experimentation options to researchers in terms of latency, bandwidth, and scalability as well as its co-existence with other standards. To the best of our knowledge, this is the first simulation framework model that will support publish/subscribe model in ns-3 which already covers many of the underlying communication protocols for IoT. We would like to note that we do not implement any new solution but instead we are the first to assess the performance of a popular Publish/Subscribe architecture -MQTT, under LoRaWAN so that the practitioners can weigh the suitable configurations to be used based on their application needs (i.e., in terms of delay, bandwidth, scalability etc.).

3 Background and Preliminaries

In this section, we provide a brief overview of three key concepts and technologies that we used in our work.

3.1 NS-3 Network Simulator

NS-3 is a discrete event network simulator built using C++ and Python that allows the user scripting functionality [17]. NS-3 is the third iteration of network simulators and is the successor to NS-2. NS-3 was developed primarily for research and educational use. It has been used as the primary tool for researchers to implement and test a variety of network protocols, models and architectures. It has an event based simulation model and offers realistic implementations of the physical layer for many protocols. We choose NS-3 over other network simulators such as Mininet, Matlab or GNS3 due to its customization capabilities and extensive documentation.

In order to implement our proposed framework we require the use of the LoRaWAN NS-3 module. The LoRaWAN module allows for realistic simulations of LoRa communications. It also allows for customization of the physical parameters and offers LoRa-specific abstractions such as Gateways and Network Servers which is commonly used in the literature [6]. In conjunction to the LoRaWAN model, we also implement an MQTT model within NS-3 to allow for a Publish/Subscribe architecture.

3.2 Publish/Subscribe Model and Applications

While there are many options to implement publish/subscribe model, we will present only three of the applications one of which will be used in our ns-3 implementation framework. However, as shown below, this MQTT-based implementation can be easily modified to support the other standards since we will be offering the infrastructure in NS-3 to adjust the topics and the types of messages for subscription.

3.2.1 MQTT

The MQTT protocol [11] is one of the well-known and commonly used communication protocols for machine-to-machine communication (M2M). It is a lightweight protocol for establishing a connection between remote devices, particularly those with low-bandwidth connections. Instead of a direct connection between transmitter and recipient, a broker retains the data and acts as a mediator, filtering messages based on subject subscriptions and sending them to the right recipient. Neither the sender nor the receiver must know each other and they do not need to run at the same moment. As shown in Fig. 1 an MQTT transmitter can serve both as a publisher and subscriber. It enables bi-directional communication by allowing a device to subscribe to one or more topics and receive messages from them. It also offers different levels of service quality depending on the application. Security offerings such as transport layer security (TLS) can also be used on top to secure communication, and the broker can authenticate the users. MQTT does not directly support multicasting and thus depends on unicasts. To enable multicast (or broadcast), a sensor network gateway can be integrated next to the broker forming a new version called MQTT-SN. In this architecture, the gateway will communicate with the broker using UDP-based multicasts. Therefore, in our framework, we utilize the MQTT protocol with a LoRaWAN gateway to enable

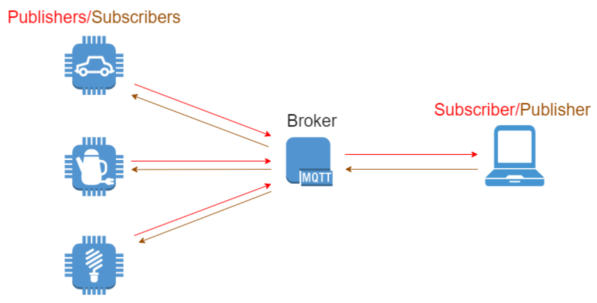


Figure 1: Sample MQTT Environment in a Smart Grid Use Case

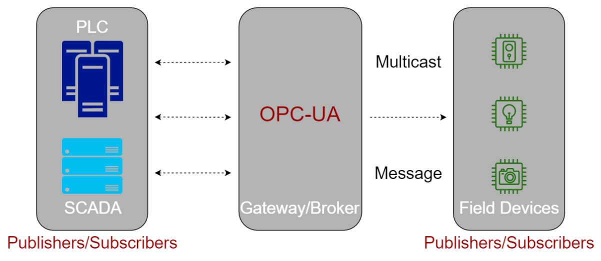


Figure 2: OPC-UA Architecture [7]

this feature. Any device within the MQTT architecture can be a subscriber and publisher in which a device can publish or receive data based on a topic. For instance, a control center could act as the remote server to collect data and make decisions in case of a smart grid environment [22].

3.2.2 OPC UA

Open Platform Communications United Architecture (OPC UA) is a standard platform that allows many types of devices and systems to interact by exchanging information messages by using request/response, or by using the publish/subscribe mechanism similar to MQTT for IoT environments. Moreover, (OPC UA) is commonly used as publish/subscribe model for Smart Grid as well as industrial control systems (ICS) [13]. As illustrated in Fig. 2, any device may communicate with the other through the broker. That removes the need of servers and enables polling the clients on regular basis. Furthermore, such a paradigm is ideal for utilizing broadcast abilities to reach several devices at once while avoiding the additional bandwidth generated by the unicast connections. Every network devices, data structure, and procedures may be seen using an (OPC UA) application in which a publisher/subscriber that allows to call certain methods, read, write, and so on.

3.2.3 DDS

Data Distribution Services (DDS) is a decentralized communication standard that uses a publish/subscribe model as well as data-central platform to address the data sharing demands of highly scalable and distributed real-time systems [18]. The DDS's architecture consists of different topics in a DDS domain which are a collection of information items in the domain, each of which is identifiable by a key. Publishers which are the Data Writers will decide the state of publishing events to a specific topic as well as assigning the type which is writing the event and then send the data operations. On the other hand, the subscribers such as Data Readers will announce the intent to subscribe to a topic and provide the type read then will receive data operations.

3.3 LoRa Protocol

In this subsection, we provide a brief overview of an important concept of LoRaWAN Module which is the key concept and technologies that we apply in our work. As mentioned before that LoRaWAN [12] is a Low-Power communication protocol that is ideal for devices located in a wide area and apart from each other. LoRaWAN is optimized for having a long range transmission range, low energy consumption as well as limited bandwidth capacity. LoRaWAN network utilizes a star-of-stars network topology where the gateway receives data from the end-devices and then dispatches these data to a remote server through an IP-based wired network. The LoRaWAN protocol stacks consists of four layers which are LoRa Physical, Regional Parameters, Link Layer, and Application Layer. There are separate regional parameters which will vary based on the location since there are several frequencies that regions have to follow. Note that there are three types of nodes supported in LoRaWAN standard. Class A, B and C. While Class A and B are restricted in the way they can receive messages to save energy (i.e., very low duty cycle), Class C devices do not have such restrictions. As shown in Fig. 3 LoRa also provides an application server for an ease of integration and deployment such as The Things Network [25].

4 LoRaWAN Model for Publish/Subscribe Architecture

4.1 Motivation and Overview

The LoRaWAN Module we will utilize allows for an accurate simulation of the physical layers of LoRa communications on NS-3. It allows for customization of the physical parameters and offers LoRa-specific abstractions such as Gateways and Network Servers.

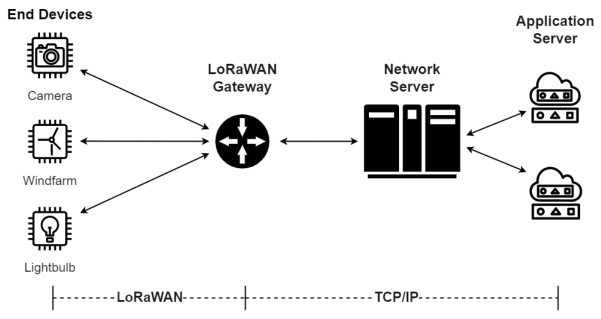


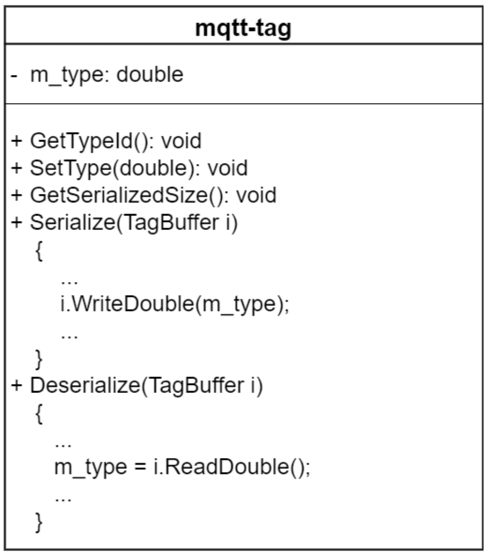
Figure 3: Sample LoRaWAN Architecture

Due to this reason it is very commonly used in the literature [6]. Yet, one major limitation of LoRaWAN technology is its sensitivity to the presence of downlink traffic, small duty cycles or high Spreading Factors (SF) [15],[1] since it will deteriorate the performance of both uplink and downlink traffic. According to [1], the maximum data rate available for LoRaWAN is of 27 kbps (when using SF of 7 and Bandwidth of 500kHz). In other words, LoRaWAN suffers from a restricted data rate. Nevertheless, this makes LoRaWAN very attractive for many of the IoT applications where sensor/actuators generate and share data among themselves. In particular, there is a growing trend in using publish/subscribe model in several applications such as smart grid, transportation, manufacturing, etc. as opposed to utilizing a main server to collect all the data and make decisions based on that. Among many examples, MQTT is the most common one that is widely used to support heterogeneous devices and standards as long as there is a broker in between the publishers and subscribers.

However, MQTT is mostly used with popular communication standards such as WiFi and LTE and thus its testing for large-scale IoT applications is not feasible in NS-3 even though NS-3 supports all of these communication standards. For this reason, there is a need to integrate the widely used LoRaWAN Module in NS-3 with a publish/subscribe architecture. This will also enable easy setup for scalability experiments within NS-3. It is worth mentioning that the NetworkServer abstraction in LoRaWAN very closely resembles the behaviour of a broker in an MQTT network giving further reason for their integration. Given the aforementioned reasons the main goal of this paper is to provide an insight regarding the performance of LoRaWAN under a widely used wide area network wireless standard, namely MQTT.

4.2 Implementing Publish/Subscribe Tag

ns-3 possesses a virtual class definition of a `TAG` . Users who want to use a custom tag must declare their own Tag class definition as shown in Fig. 4. Certain methods



```
mqtt-tag
- m_type: double
+ GetTypeId(): void
+ SetType(double): void
+ GetSerializedSize(): void
+ Serialize(TagBuffer i)
   {
      ...
      i.WriteDouble(m_type);
      ...
   }
+ Deserialize(TagBuffer i)
   {
      ...
     m_type = i.ReadDouble();
      ...
   }
```

Figure 4: UML Diagram for Pub/Sub Tag

such as `GetSerialized Size()`, `Serialize()` and `Deserialize()` are called automatically by ns-3 when using the `AddPacketTag()` and `Remove PacketTag()` methods. Consequently, great care needs to be taken to ensure that the implementation of this method is correct since any modification to their declaration will result in errors when compiling. To implement the publish/subscribe architecture, we need to define the type of control messages that our broker will recognize. The type of message will be sent within the tag itself. For this implementation, we only define two types of messages, *Subscribe* and *Publish*. It is worth mentioning that MQTT protocol as of specification 3.1.1 has defined 13 types of control messages but not all are required and most are only used if the application is dependent on them. Implementing new methods in the `TAG` class is only required for accessing the data that is to be written by the previously mentioned methods. If the Custom Tag is meant to have only hard coded values, no further modification is needed. Since we will be allowing clients to define the type of message in the Tag, we need to setup the *setters* and *getters* for it. To this end, we created 2 new methods called `SetType()` and `GetType()`. They will modify the value of our control message type so it can be serially written on the packet to later be deserialized and extracted by the broker upon reception.

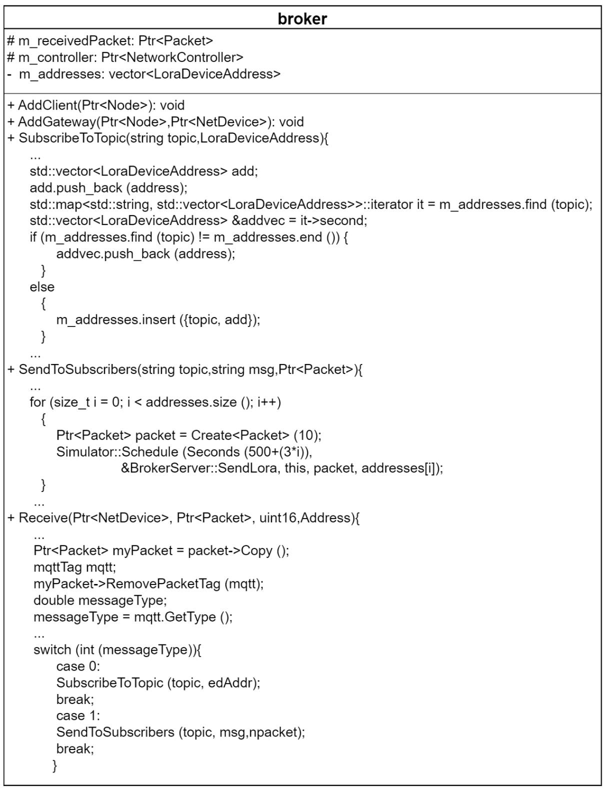
4.3 Integration of LoRa with publish/subscribe architecture

The `LoRaWAN` module possesses a class definition of a *NetworkServer*. This definition includes all variables and methods required to handle LoRaWAN communications and Gateways. In order to integrate the LoRaWAN Module with a publish/subscribe architecture, we need to modify the NetworkServer definition to include a data structure capable of handling the name of the topics and the addresses associated with them. To this end, we utilize the `std::map` C++ structure. We choose this data structure for a variety of reasons as listed below:

- Unique Keys: Allows for only one topic of the same name to exist since no two keys can have the same value.
- Allocator-aware: Uses an allocator object to handle the storage needs dynamically, allowing for any amount of Topics and any amount of addresses associated to them.
- Time Complexity: Look ups are proportional to $\log_{10}$(N). However, it can be up to $\log_{2}$(N) as well as insertions are proportional to $\log_{2}$(N).

Our map structure will use the topics as *Keys* and associated to the keys will be a list of addresses of the devices that are subscribed to the topic. Once the data structure is built within the NetworkServer, we can begin our modification of it to include the logic required to recognize the type of message and route the packages to the correct destination. Hereafter, the NetworkServer will be referred to as a *Broker* as shown in Fig. 5. To implement this logic, we need to define three additional methods. The first is `SubscribeToTopic()`. This method is called when the message received in the Broker is a Subscribe control message. We extract from the received packet the topic alongside with the address of the device that sent it. We then pass to this method the Topic and address extracted. The topic gets added as one of the keys of our `std::map structure` and the address appended to the list of addresses associated to the topic.

The next method to be defined is `PublishToTopic()`. This method is called when the message received in the Broker is a Publish control message. We extract the topic alongside with the payload of the message from the received packet. The topic and payload are then passed on as arguments to this method. Next, the existence of the topic is verified. If the topic exists, then a message can be scheduled to be sent to the subscriber of the topic. To do this, one final method is defined, the `Send()` method. This method will do a callback to the `Send()` method of the underlying NetDevice associated to the Broker. If a Broker with different communication interface is required, different NetDevices may be installed on its node and then accessed by this method.



```
broker
# m_receivedPacket: Ptr<Packet>
# m_controller: Ptr<NetworkController>
-  m_addresses: vector<LoraDeviceAddress>

+ AddClient(Ptr<Node>): void
+ AddGateway(Ptr<Node>,Ptr<NetDevice>): void
+ SubscribeToTopic(string topic,LoraDeviceAddress){
     ...
     std::vector<LoraDeviceAddress> add;
     add.push_back (address);
     std::map<std::string, std::vector<LoraDeviceAddress>>::iterator it = m_addresses.find (topic);
     std::vector<LoraDeviceAddress> &addvec = it->second;
     if (m_addresses.find (topic) != m_addresses.end ()) {
          addvec.push_back (address);
       }
     else
       {
          m_addresses.insert ({topic, add});
       }
     ...
+ SendToSubscribers(string topic,string msg,Ptr<Packet>){
     ...
     for (size_t i = 0; i < addresses.size (); i++)
       {
          Ptr<Packet> packet = Create<Packet> (10);
          Simulator::Schedule (Seconds (500+(3*i)),
                         &BrokerServer::SendLora, this, packet, addresses[i]);
       }
      ...
+ Receive(Ptr<NetDevice>, Ptr<Packet>, uint16,Address){
      ...
      Ptr<Packet> myPacket = packet->Copy ();
      mqttTag mqtt;
      myPacket->RemovePacketTag (mqtt);
      double messageType;
      messageType = mqtt.GetType ();
      ...
      switch (int (messageType)){
          case 0:
          SubscribeToTopic (topic, edAddr);
          break;
          case 1:
          SendToSubscribers (topic, msg,npacket);
          break;
         }
```

Figure 5: Broker's UML Diagram

4.4 Simulation Setup and Assumptions

In this section, we provide a brief overview of the simulation setup and the smart grid system applications that could be used using our framework.

4.4.1 Simulation Setup

For our simulation setup, we assume we will be using LoRaWAN interfaces to enable the publish/subscribe architecture in an OPC UA smart grid use case. To this end, a gateway must be used in order to relay the messages coming from the end-devices to the broker. We assume one gateway and one broker for all simulations. The end-devices will be placed randomly around a disc that surrounds the gateway. For all experiments we assume a maximum distance from gateway to end-device of 1000 meters. To allow the gateway to send messages to the broker we use a point to point link. The end-devices are assumed to have a constant position as well as the gateway and broker. Since the original regional parameters of the `LoRaWAN` module are set to comply with the EU specifications, we had to increase the default duty cycle from 1% to 50% for the majority of the experiments. This was done since a 1% duty cycle is too restricted to enable the gateway to transmit to the end-devices. In addition, we were able to use a Class-C

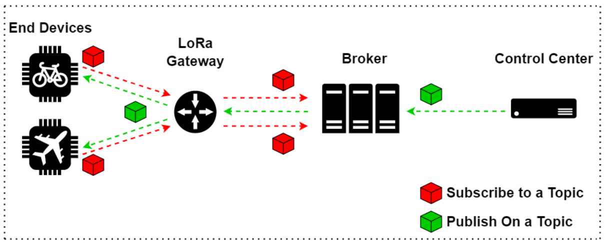


Figure 6: LoRa Publish/Subscribe Architecture

LoRa device that can always listen to the channel after its transmission. A sample setup is shown in Fig. 6. We also make the code available through GitHub. [1]

4.4.2 Assumptions

In the current smart grid systems, utilities are deploying many outside devices including distributed energy resources. LoRaWAN can be considered as a good candidate to deploy here due to its range and near-zero costs compared to cellular options. Therefore, our implementation would provide an idea about the capacity of pub/sub approach such as MQTT under LoRa. We are assuming that many of the smart grid applications can be simulated within this framework. That is because combining a lightweight protocol such as MQTT that provide a real-time interaction and does not demand a high bandwidth with LoRa, will provide a reliable, efficient, and low cost services that will accelerate the change of the traditional power grid to the smart grid [26]. That includes Advanced Metering Infrastructure (AMI), Substation Automation System and Demand Response (DR). Implementing MQTT with LoRa using ns-3 will allow us to scale the system in terms of deploying more sensors (end devices) within a large field.

5 Performance Evaluation

In this section, we present some experiment results based on an NS-3 simulation of a publish/subscribe network architecture using LoRaWAN communications.

5.1 Performance Metrics

We used the following metrics to measure the performance of our approach.

- *Average Publish Delay:* This metric measures the average delay it takes to attempt to send a Published message to all the subscribers for a particular topic. In other words, this metric will allow us to quantify the performance of the network when we increase the number of possible subscribers to a topic. Note that if it is not possible to publish the message to all subscribers in a given duty cycle (DC) period, then the average will be calculated by taking into account only the successfully transmitted messages.
- *Packet Delivery Ratio:* This metric is used to assess the amount of packets transmitted (or expected to be transmitted) with respect to the packets received. In other words, we can define it as the number of packets received divided by the number of packets sent.

5.2 Simulation Results

In this section, we provide the results from our ns-3 simulations that were repeated 30 times. We note that the graphs offer error bars to indicate the statistical significance of the results. The results are within (mean ± 1.96 * standard error) with a 95% probability.

5.2.1 Node Count Effects on Publish Delay

For this experiment, we varied the number of nodes that will act as clients (publisher and subscriber) for our broker. We assume that there are a total of 50 topics. To ensure that all topics have at least one subscriber and to ensure that each end device subscribes to at least one topic we manage the topic subscription by having different behaviours. If the number of end devices is greater or equal to the number of topics, then each topic will have multiple subscribers and each end device will be subscribed to a single topic. If instead the number of topics is greater than the number of end devices, then each end device will be subscribed to multiple topics and each topic will have one subscriber. We place our nodes randomly inside a circle of radius equal to 1000m. The gateway is located at the center of the circle.

From Fig. 7, we can see that as we increase the number of available nodes, the delay to publish the message to all subscribers increases. This is because the topics are gaining more subscribers as we increase the node count. In order for the broker to transmit the Published message to the subscribers, it needs to address each subscriber directly through the gateway in individual unicast messages while respecting the Gateway's duty cycle. Therefore, these results are taking into consideration the limitation of LoRaWAN bandwidth for certain applications. For instance, if there is a real-time requirement for data sharing and actions, increasing the number of LoRa end-devices will significantly impact the results. In particular, after 20 devices, the delay becomes really a bottleneck (e.g., more than 5 secs to send a packet). For smart grid use cases such as Smart Meters can be considered as one of

[1] https://github.com/JVoltagic/NS3-ADWISE.git

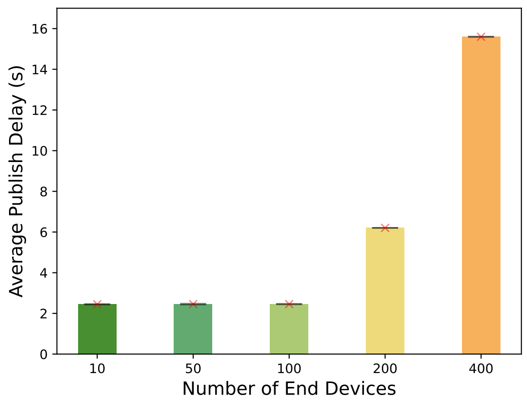


Figure 7: Average Publish Delay - Node Count

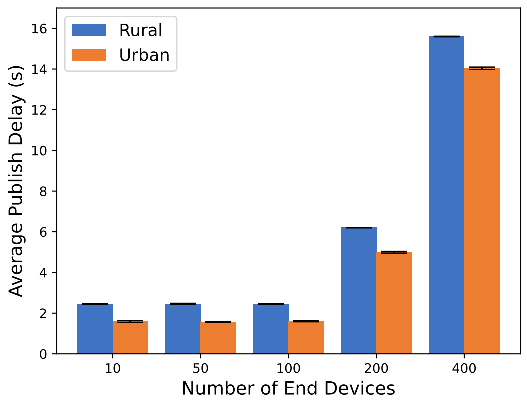


Figure 8: Average Publish Delay - Node Count Rurals Vs Urban

the applications that is required to transmit data every 15 minutes at least in the US [14]. Moreover, other Smart Grid application that requires a certain time to transmit data is a smart agriculture monitoring in which several sensors could be deployed on the field which are required to send data such as temperature and humidity every 1-2 hours. This may not be an issue but for applications which require data collection every minute, the performance will suffer. The practitioners might need to consider using multiple gateways to cluster the data sources.

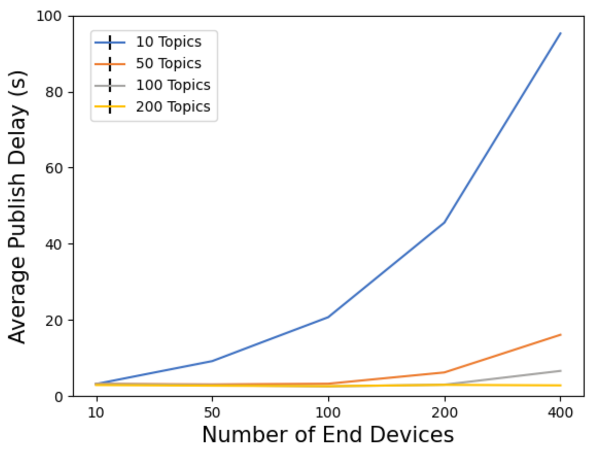


Figure 9: Average Publish Delay

5.2.2 Node Count Effects: Rural Vs Urban Environment

For this experiment, we repeated the same procedure as in Node Count Effects experiment but this time we included buildings in the simulations scenario. By introducing buildings placed within a grid, we can simulate an urban environment. The grid center lines up with the center of the circle in which we place the end devices. Each building size is 130 m x 60 m with each 4 floors and the size of the streets are in average 25 m wide. We use a set the radius of the circle to 1000m and a topic count of 50.

From Fig. 8, we can observe that the publish delay for devices in the rural scenario had a higher Average Publish Delay. This is due to having a higher Packet Delivery Ratio which causes the time it takes to receive all the packets to be increased since more packets are being received. In the case of the urban scenario the packets are being dropped, due to obstruction of line of sight.

5.2.3 Topic Count Effects on Publish Delay

For this experiment, we varied the number of total topics available when having different number of end-devices. To assess the performance of the network, we varied the number of topics from 10 to 200. All end-devices are randomly placed withing a circle of radius 1000m. The gateway is located at the center of the circle.

From Fig. 9, we can observe that the average publish delay is higher when using 10 topics in relation to 200 topics for every amount of end-devices. This is attributed to the subscription count. This means that with the same amount of nodes but a lower topic count there will be topics with more total subscribers. This increases the average delay since more unicast messages are required to address each individual node subscribed to that topic. We observe that the publish delay increases with the node count which is expected based on our previous findings. However, these experiments showed us that

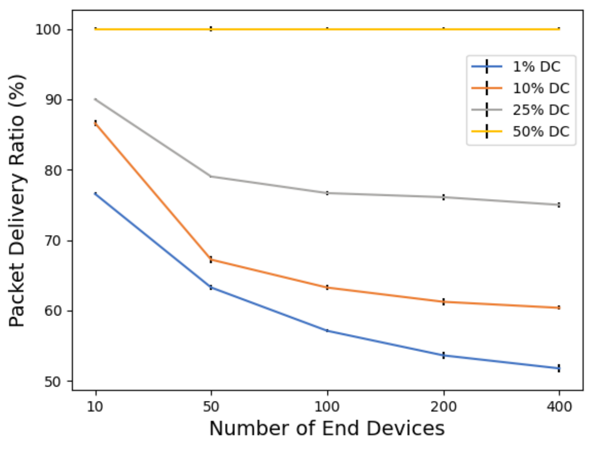


Figure 10: Packet Delivery Ratio

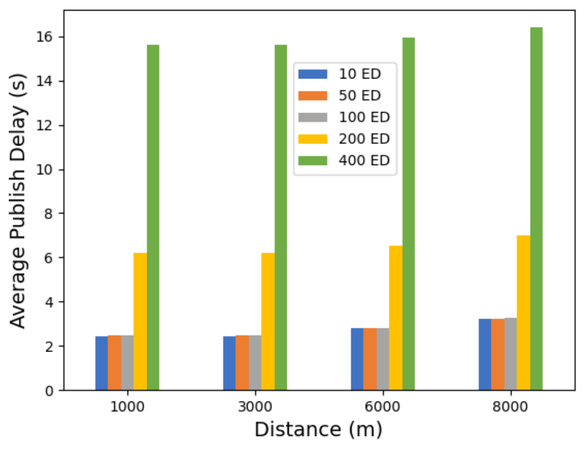


Figure 11: Average Publish Delay

topic count is only significant when there is large number of nodes subscribed to a topic. Otherwise, it may not impact the scalability of the performance.

5.2.4 Duty Cycle Effects on Packet Delivery Ratio

For this experiment, we varied the duty cycle (DC) of our gateway while maintaining all other parameters the same. We experimented with 10 topics and different amount of nodes. The DC values we used are 1%, 10%, 25%, 50%. All nodes are randomly placed within a circle of radius 1000m. The gateway is placed at the center of the circle.

From Fig. 10, we observe that as we increase the available DC of the gateway, we increase the Packet Delivery Ratio. This is because the DC is related to the time it takes to transmit. Having a lower percentage DC implies a higher suspend period before the device can transmit again. The results indicate that DC has a major impact on the performance; after 10 nodes and beyond, the Packet Delivery Ratio drops significantly. We need to offer 50% duty cycle to achieve full Packet Delivery Ratio results, which is not possible with the LoRa policy constraints in Europe or the U.S.

5.2.5 Distance Effects on Publish Delay

For this experiment, we varied the area of the circle in which the end-devices are randomly placed. We choose a radius of 1000m, 3000m, 6000m and 8000m due to them being common ranges for LoRaWAN communications. We used a total of 50 topics and a DC of 50%. This was done to ensure all messages are transmitted in order to observe the performance of the network.

From Fig. 11 we can observe that by increasing the range there is a small increase in the average publish delay. This increase in delay is due to the use of different SF values for each unicast message. LoRaWAN calculates the correct SF based on the distance from the end-device to the gateway. An SF of 7 is used for devices that are close to the gateway while an SF of 12 will be used for devices further away from the gateway. Using higher SF values result in longer Time-On-Air (TOA) values which in turn result in longer publish delays. As seen from previous experiments, increasing the node count will result in an increase in the average publish delay.

5.2.6 Distance Effects: Urban Vs Rural Environment

In this experiment, we repeated the distance effects on Average Publish Delay experiment in a an urban environment. In order to simulate an urban environment, we placed the end devices within a grid. The lines of the grid represent the streets and the buildings are placed in between them. We use 50 topics and a DC of 50%.

In Fig.12, we see the average publish delay as we increase node count for different distances. We observe that when comparing the scenario without buildings against the scenario with buildings, the average publish delay is higher for the latter. This is due to having a lower packet delivery rate. In other words, since the average publish delays measures the average time it takes to publish to all subscribers, it implies that when messages are lost then the total publish delay will decrease. The lost occurs because LoRaWAN is extremely sensitive to obstacles. The maximum range of LoRaWAN can only be achieved by having direct line of sight from the gateway to the receiver. By comparing Fig.12(A) with Fig.12(D), we can observe that overall Average Publish Delay for both scenarios decreases as distance increases.

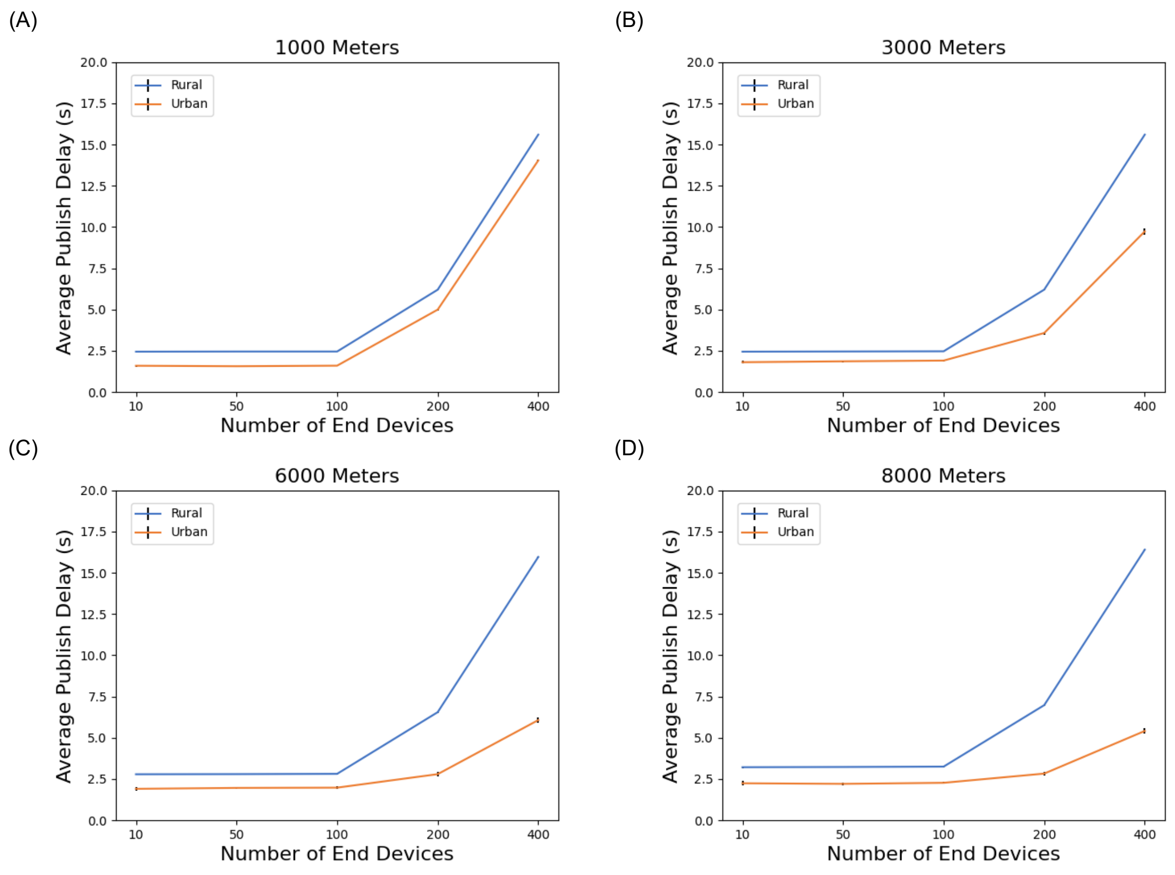


Figure 12: Average Publish Delay - Urban Vs Rural Environment

This is due to a higher amount of obstacles interrupting direct line of sight from gateway to end device.

In Fig. 13, we see how the Packet Delivery Ratio is a 100% when there are no buildings present (rural environment). This is true for all different amounts of end devices and all different distances. This is because when in direct line of sight, LoRaWAN can operate up to 15 km. This is not the case when we introduce buildings into the scenario. We can observe in Fig. 13(A) that the Packet Delivery Ratio for the rural scenario is around 97% but it quickly decreases as the distance increases. This is because the increase in distance also means an increase in the amount of possible buildings in between the gateway and end device.

6 Conclusion and Future Works

In this paper, we implemented and evaluated a publish/subscribe model that can work with LoRaWAN to be used in IoT applications. To this end, we introduced extensions to ns-3 `LoRaWAN` Module to support a new set of tags that will eventually build a publish/subscribe architecture framework for researchers. We also developed a series of experiments to evaluate the performance of the framework. Through these experiments we showed that our approach can support a publish/subscribe network architecture while using the underlying LoRaWAN physical devices. We also showed the performance of the architecture under different physical configurations, propagation models and transmission parameters. As LoRaWAN is limited with severe duty cycles, the design of the number of devices, topics, distance and spreading factors should be carefully selected. The results would offer insights regarding the performance of an emerging technology under another widely used wide area network wireless standard that will shed light to the feasibility of many applications.

Acknowledgements

Yacoub Hanna is supported by the USAID Organization under the grant No.1004242-S21-35022-00.

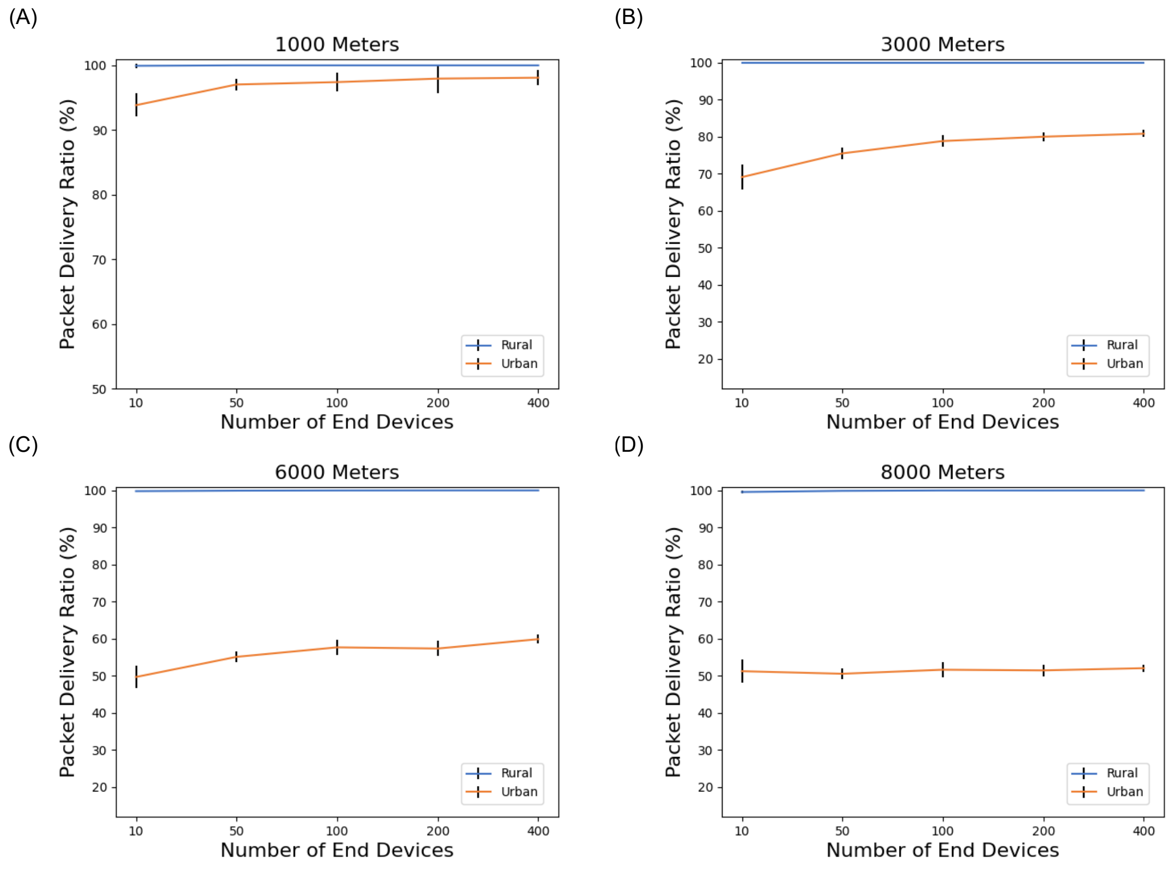


Figure 13: Packet Delivery Ratio-Urban Vs Rural Environment

References

[1] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[2] M. Ayaz, M. Ammad-Uddin, Z. Sharif, A. Mansour, and E.-H. M. Aggoune, "Internet-of-things (iot)-based smart agriculture: Toward making the fields talk," *IEEE access*, vol. 7, pp. 129 551–129 583, 2019.

[3] M. J. A. Baig, M. T. Iqbal, M. Jamil, and J. Khan, "Design and implementation of an open-source iot and blockchain-based peer-to-peer energy trading platform using esp32-s2, node-red and, mqtt protocol," *Energy Reports*, vol. 7, pp. 5733–5746, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352484721007939

[4] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the energy performance of lorawan," *Sensors*, vol. 17, no. 10, p. 2364, 2017.

[5] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, "Lorawan — a low power wan protocol for internet of things: A review and opportunities," in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, 2017, pp. 1–6.

[6] J. Finnegan, S. Brown, and R. Farrell, "Evaluating the scalability of lorawan gateways for class b communication in ns-3," in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, 10 2018, pp. 1–6.

[7] Y. Hanna, M. Cebe, S. Mercan, and K. Akkaya, "Efficient group-key management for low-bandwidth smart grid networks," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2021, pp. 188–193.

[8] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*,

vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3995

[9] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s — a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, 2008, pp. 791–798.

[10] S. Kul and A. Sayar, "A survey of publish/subscribe middleware systems for microservice communication," in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2021, pp. 781–785.

[11] R. A. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, 2017.

[12] LoRaWAN, "Lorawan web page to be referenced," https://lora-alliance.org/, Accessed 5th May 2022.

[13] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[14] M. Manbachi, "Impact of distributed energy resource penetrations on smart grid adaptive energy conservation and optimization solutions," in *Operation of Distributed Energy Resources in Smart Distribution Networks*. Elsevier, 2018, pp. 101–138.

[15] J. Marais, A. Abu-Mahfouz, and G. Hancke, "The impact of application-based downlink traffic in lorawans," in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021*, 11 2021, pp. 290–294.

[16] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *2017 IEEE International Systems Engineering Symposium (ISSE)*, 2017, pp. 1–7.

[17] NS-3, "Ns-3 web page to be referenced," https://www.nsnam.org/, Accessed 5th May 2022.

[18] G. Pardo-Castellote, "Omg data-distribution service: architectural overview," in *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, 2003, pp. 200–206.

[19] C. M. Ramya, M. Shanmugaraj, and R. Prabakaran, "Study on zigbee technology," in *2011 3rd International Conference on Electronics Computer Technology*, vol. 6, 04 2011, pp. 297–301.

[20] A. N. Rosli, R. Mohamad, Y. W. Mohamad Yusof, S. Shahbudin, and F. Y. Abdul Rahman, "Implementation of mqtt and lorawan system for real-time environmental monitoring application," in *2020 IEEE 10th Symposium on Computer Applications Industrial Electronics (ISCAIE)*, 2020, pp. 287–291.

[21] C. Sobin, "A survey on architecture, protocols and challenges in iot," *Wireless Personal Communications*, vol. 112, no. 3, pp. 1383–1429, 2020.

[22] D. Soni and A. Makwana, "A survey on mqtt: a protocol of internet of things (iot)," in *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*, vol. 20, 2017.

[23] A. Stanford-Clark and H. L. Truong, "Mqtt for sensor networks (mqtt-sn) protocol specification," *International business machines (IBM) Corporation version*, vol. 1, no. 2, pp. 1–28, 2013.

[24] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of mqtt and coap via a common middleware," in *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014, pp. 1–6.

[25] ThingsNetwork, "ThingsNetwork Web Page to be Referenced," https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/, Accessed 5th May 2022.

[26] L. Tightiz and H. Yang, "A comprehensive review on iot protocols' features in smart grid communication," *Energies*, vol. 13, no. 11, p. 2762, 2020.

[27] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan, "Evaluation of lora and lorawan for wireless sensor networks," in *2016 IEEE SENSORS*, 2016, pp. 1–3.

AUTHOR BIOGRAPHIES

Juan Leon graduated from Florida International University with a Bachelor of Science in Computer Engineering. He received the Excellence in Cyber Defense Education certificate. He is an international student from Venezuela. Currently, he is pursuing a master's degree in Computer Engineering. He performs research in the ADWISE lab in the areas of simulations, networking, vehicular communications, and cyber security. GitHub `https://github.com/JVoltagic/NS3-ADWISE`.

Yacoub Hanna graduated in 2020 with the honor of Magna Cum Laude from Florida International University with a Bachelor of Science degree in Computer Engineering and a Minor in Math Science. Currently, he is a second-year Ph.D. student and Graduate Research Assistant in the Department of Electrical and Computer Engineering at FIU. He is a member of the Golden Key International Honor Society. His research interests are Smart Grid Security, Internet-of-Things, and cyber-physical systems. He is currently working on research for LoRaWAN, NS3, and MQTT Protocol for IoT devices. More information about his research and lab can be obtained at `https://yhanna.com` and `https://adwise.fiu.edu/`.

Dr. Kemal Akkaya is a full professor in the Department of Electrical and Computer Engineering at Florida International University. He received his PhD in Computer Science from University of Maryland Baltimore County in 2005 and joined the department of Computer Science at Southern Illinois University (SIU) as an assistant professor. Dr. Akkaya was an associate professor at SIU from 2011 to 2014. He was also a visiting professor at The George Washington University in Fall 2013. Dr. Akkaya leads the Advanced Wireless and Security Lab (ADWISE) in the ECE Dept. His current research interests include security and privacy, internet-of-things, and cyber-physical systems. Dr. Akkaya is a senior member of IEEE. He is the area editor of Elsevier Ad Hoc Network Journal and serves on the editorial board of IEEE Communication Surveys and Tutorials. Dr. Akkaya was the General Chair of IEEE LCN 2018 and TPC Chair for IEEE ICC Smart Grid Communications. He has served as the guest editor for many journals and in the OC/TPC of many leading network/security conferences including IEEE ICC, Globecom, ICNP, INFOCOM, LCN and WCNC and ACM WiSec. He has published over 220 papers in peer reviewed journal and conferences. He has received "Top Cited" article award from Elsevier in 2010.