Interoperable middleware for smartcities - Streetlighting on LoRaWAN as a case study

Rakshit Ramesh*, Srikrishna Acharya*, Vasanth Rajaraman*, Arun Babu*, Ashish Joglekar*, Abhay Sharma*, Bharadwaj Amrutur*, Prashant Namekar†
* Robert Bosch Center for Cyber Physical Systems, IISc Email: *rakshitr@iisc.ac.in
† Gaia Smart Cities Email: *prashant@gaia.in

***Abstract*—A smart city, is like every other organism a structured and organized entity. An organism comprises of a brain that performs its tasks via its appendages, just as smart cities should. However the brain needs a spine through which it communicates information to its appendages, becomes a point of confluence of different sensory and actuator pathways and facilitates activities. We herein put forth our work involving the building of such a spine for a smart city, and focus on the last mile communication aspects, especially using an LPWAN network and an implementation example with smart streetlights.**

***Index Terms*—IoT Test bed, smart city solutions, middleware, LoRa, Passive Optical Networks**

I. Introduction

A simple street is a prime example of a critical component of a smart city that can be smartened [1] by inclusion of sensors, and a switch that can be remotely monitored and controlled via intelligent applications. As an example, we provide in our streetpoles of our testbed, brightness controllable steetlights, each with a low cost compute box (the pole gateway) that also has a camera, and can do local data analytics which serves as a fog computer in some sense [2]. The gateway also includes a radio to facilitate communication with the middleware through Low-Bandwidth LoRa communication, or in some cases via wired Internet brought about through a passive optical fiber network.

The case of streetlighting provides both an interesting and complex demonstration of what a city would want to do as it smartens. A few of the applications are pedestrian density based lighting, or event based brightening, where for example an LED would be maximally bright only when there is an event to be captured on video. Thus the brightness of the LED need not just be controlled by the ambient luminescence that the pole can detect, but also by various extraneous sensory stimuli such as cameras from another pole, or an app that a pedestrian under distress could use. Enabling such an application would require a sophisticated level of interoperability and means for a stimuli to discover nearby entities (sensors or actuators) to go about accomplishing the task and is facilitated by our middleware [3].

In this poster abstract, we discuss our experiences in the practical and scalable deployment of a smart streetlighting solution, talking over a mix of Low and High Bandwidth communication modalities, and abstracting out the underlying complexities in such a design through our middleware making the access and control of large scale streetlighting feasible.

Robert Bosch Foundation, IISc, Gaia Smart Cities

Our testbed comprises of 6 smart poles. Three of these poles are basically just the streetlight LED and a LoRa Radio. One of these is a hybrid with PON as the primary networking interface and LoRa as a redundancy(for testing), the other two are purely PON networked streetlights.

II. Hardware aspects of the smart pole

The hardware components of the LED can be divided into the actuatory units(LED's), the Radio (LoRa) and the sensors(LUX, Temperature, Power).

A. *The LED of a smart pole*

A standalone LED unit Fig.1 comprises of 10 LED strings of 3 Watt LEDs giving us a total of 30 Watts of power. The LED unit also has an 8 bit microcontroller (ATMega 8) that controls these strings and also retrieves information from the sensors like a photodiode that tells how bright the LED has become, a thermistor for temperature, and voltage and current sensors. The mode of communication between the radio and the LED controller is through UART.

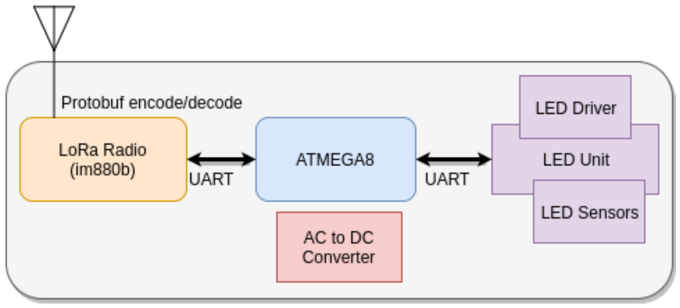

Fig. 1. A standalone LED unit

There are two state machines running in the unit, one inside the LoRa radio, that monitors the status of the slave ATMega8 device and the other inside the AtMega8 itself, which in the event that a communication link to the radio is lost, reverts to a fallback mode where the light is automatically actuated based on ambient lighting conditions. In such an event, the middleware infers that the radio link is non-functional and alerts for maintenance. In an event that the ATMEGA is not responding with a heartbeat, the radio infers that the actuator is non-functional and alerts for maintenance.

B. The Pole Gateway

As mentioned earlier, few of the smart poles in our testbed come equipped with a pole gateway box Fig.2, which is essentially a Raspberry Pi3 grade computer that performs edge analytics, for example, simple video analytics. We do this to reduce the load on the cloud platform, and to bring about a federated and distributed architecture of fog computing. In cases where a pole computer is present, the computer controls the LEDs of the streetlight.

The communication could be brought about either through LoRa, where only derived information is transmitted, and/or through wired PON, where data streaming (video) is also possible. Sensors on RS485/232, BLE, etc can connect to the pole gateway. In our case, we have CO2, PM2.5 and PM10 air quality sensors as auxiliary sensors on a pole gateway.

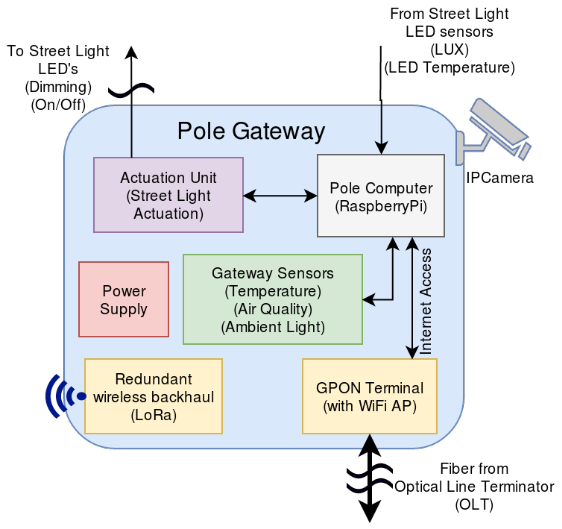
To Street Light LED's (Dimming) (On/Off)
From Street Light LED sensors (LUX) (LED Temperature)
Pole Gateway
IPCamera
Actuation Unit (Street Light Actuation)
Pole Computer (RaspberryPi)
Power Supply
Gateway Sensors (Temperature) (Air Quality) (Ambient Light)
Internet Access
Redundant wireless backhaul (LoRa)
GPON Terminal (with WiFi AP)
Fiber from Optical Line Terminator (OLT)

Fig. 2. The pole Gateway

III. Communication and Networking aspects

A. Details about the LoRa Network

LoRa stands for Long Range, and as the name suggests, is a sub gigahertz (in the 865Mhz Band) radio. It boasts of features such as long range, low power consumption and cheap pricing [4]. Every standalone streetlight module comes equipped with a LoRa Radio [5] (im880b based on Semtechs SX1272 LoRa Module). Every LoRa Radio (End Node) can talk to a LoRa gateway among one of three 125 kHz uplink channels in the 865 to 867 Mhz bands. Over this, there are 6 orthogonal spreading factors or data rates that a radio could pseudo-randomly choose to transmit at.

A LoRa gateway has the ability to listen to multiple channels and spread factors simultaneously and push the payload onto the Network Server [6] which resides in the cloud, which essentially routes the payload into the radios specific database and allows to serve this payload to any applications demanding it.

In our deployment, we provide a simple means to connect and on-board a radio to our Network Server. Weve deployed 3 gateways(IMST, Kerlink, RisingHF) in our campus and one specifically to serve our test bed.

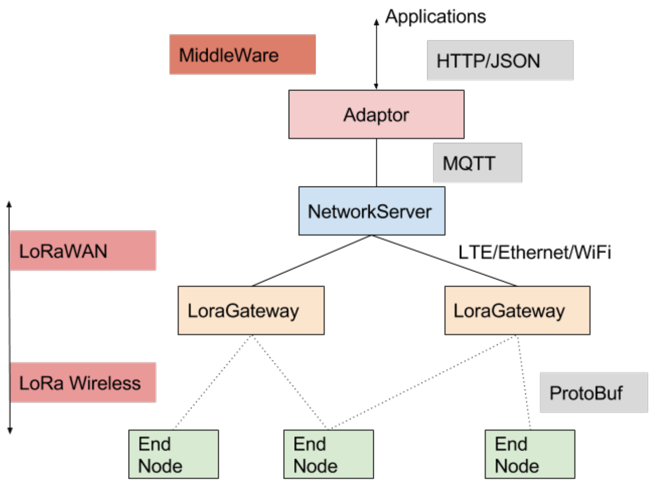

Fig. 3. LoRa Network Setup

Our LoRa infrastructure serves applications belonging to streetlighting, water management (water level, flow) and pollution analysis (PM2.5, PM10, CO2) at the moment. A brief overview of our network topology for LoRa devices is as shown in Fig.3.

B. Details about the LoRa Network Infrastructure management

In a network deployment, it is critical that the LoRa Network Service Provider have continuous access to their LoRa gateway infrastructure at all times, for monitoring and debugging purposes. Most often, such a LoRa gateway is connected to a 4G/LTE network to stream out the data it obtains from the LoRa end nodes to any middleware. A problem with 4G networks is that most incoming ports are blocked, hence an SSH connection request cannot be made to the LoRa gateway.
To handle such cases, we have developed a network management service whereby the administrator is still able to SSH to these seamlessly. We achieve this as shown in Fig.4 by making the gateway establish a reverse SSH tunnel to a Management Server whereupon a client web browser, through secure means, is able to open up a terminal that is supported by a Web Service running inside the manager server.

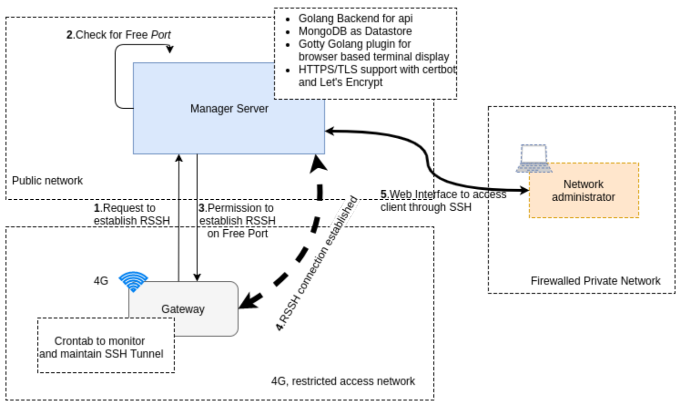
Golang Backend for api
MongoDB as Datastore
Gotty Golang plugin for browser based terminal display
HTTPS/TLS support with certbot and Let's Encrypt
2.Check for Free Port
Manager Server
Public network
1.Request to establish RSSH
3.Permission to establish RSSH on Free Port
4.RSSH connection established
5.Web Interface to access client through SSH
Network administrator
Firewalled Private Network
4G
Gateway
Crontab to monitor and maintain SSH Tunnel
4G, restricted access network

Fig. 4. LoRa Infrastructure Management Life Cycle

Along with this, we provide a LoRa signal testing device, which is a simple LoRa radio that can be connected to an Android phone over bluetooth. The device can seamlessly talk to a gateway/server to determine the quality of signals on the downlink and the uplink through parameters like packet error rate (PER), RSSI and SNR

C. Details about the on-air message encoding

An important aspect in LoRa communication is the payload size. In an earlier work [7], we proposed a schema for communication with the smart light, which is based on JSON owing to it's properties of self description and sheer readability. However due to LoRaWAN packet size limitations, we see that sending a JSON data representation for the sensor values, and for actuation commands on the downlink is bandwidth inefficient, due to unnecessary usage of characters like "{" "," and so on. Also, describing the field names is an additional overhead. We therefore use Googles Protobuf encoding and decoding for embedded platforms [8], as our payload representation format. This involves defining a protofile, which describes a field name corresponding to byte flags. Therefore, the data is described according to the presence of this flag and the memory location of a field according to it's datatype. We can clearly see the savings achieved in space owing to discarding unnecessary characters and field names to describe the payload. On the receiving end, the same proto file is used to reference the memory location of a field in the message body. What was once around 51 bytes of sensor data with JSON was stripped down to around 13 bytes with Protobuf.

IV. Software and IoT platform aspects

Typical Low Bandwidth Radio networks present a siloed approach of an IoT system, wherein all applications are restricted to devices connected to LoRa radios only. It is not possible for a device on WiFi, for example, to be on the same network. It is also not possible also for a device to request information from another device on the network, or to query meta-information about its description. Having these problems in mind, weve tailored our middleware to break the silos of application and devices. A middleware must seamlessly offer features such as data queuing over a Pub-Sub broker, data storage, security and APIs to make application development simple. This can only be achieved provided the modalities of interactions, i.e the query language, semantics and keywords are well defined and regulated. A burgeoning problem in the world of IoT is platform interoperability, meaning, how easy is it to have applications running on one platform to consume sensor information from devices on another platform. We solve this issue through adapters shown in Fig.5.

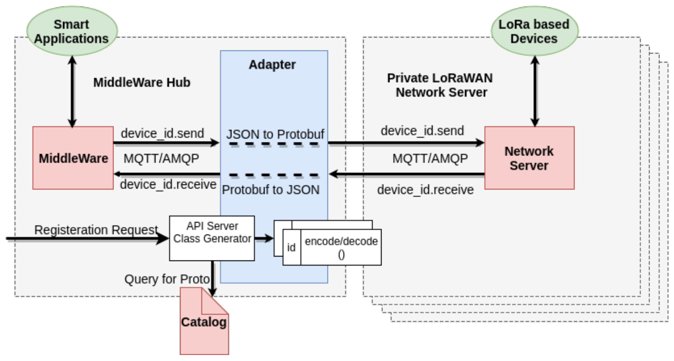

Fig. 5. Bridging middlewares through adapters and breaking silos

An adapter is essentially a software entity residing in the middleware that is a client to both the middleware(our platform) and the foreign platform(private LoRa Network Server for example). Its role is to consume sensor data from the foreign platform and publish it into our middleware on one side, and on the other, consume application data and to publish it to the devices on the foreign platform. It also has the ability to massage this data and make it useable by either platforms, in our case, it does the ProtoBuf to JSON conversion for messages originating from the streetlight on the foreign platform side and JSON to protobuf conversion for streetlight actuation commands from applications on our middleware side. Its able to download the encode/decode descriptions via entries made in a catalog on our middleware(from device registration) and generate the required classes to perform data conversion. This solves the interoperability problem.

V. Conclusions and Future Works

From the previous sections, weve detailed all the aspects that would typically be involved in deploying a streetlighting solution in general and the aspects of our testbed in specific. A practical network deployment with both wired and wireless communication facilities and management of networking gateways connected over otherwise unreachable 4G/LTE has been shown. An integrated smart-city solution can only be brought about by plugging in various middleware platforms and making possible fluent data exchange through well defined semantics and schemas, and weve described how we achieve this.

Future work would include a more holistic coverage of other networking aspects such as the integration of WiFi and Zigbee into the network management layers. A trending topic in the

IoT realm is to provide guaranteed quality of service (QoS) and it is something we intend to include in future, along with network uptime metrics and network diagnostics.

VI. Acknowledgment

We would like to thank the Robert Bosch Foundation and our institute, IISc, in offering us financial and technical support in undertaking this project. Also, we would also like to thank Mr.Bipin Kumar of Gaia Smart Cities for their support in setting up LoRa Gateways in our campus.

References

[1] *InteliLIGHT - Street Lighting Remote Management* https://intelilight.eu.
[2] Bonomi et. al. *Fog computing and its role in the internet of things*. MCC 2012.
[3] Bharadhwaj Amrutur, et.al *An Open Smart City IoT Test Bed: Street Light Poles as Smart City Spines: Poster Abstract*. ACM IoTDI 2017 (Poster).
[4] LoRa Alliance, *https://www.lora-alliance.org*.
[5] *im880b LoRa Radio and ic880a LoRa Gateway*, https://wireless-solutions.de/
[6] *LoRaServer*, https://docs.loraserver.io/overview/
[7] Abhay S. et. al. *Open Schemas*. ACM Buildsys Poster 2017.
[8] *Embedded Library for Protobuf, Nanopb*, https://jpa.kapsi.fi/nanopb/