UTLC

Rafik Zitouni[2], Jerimie ...[2], Aghiles Djoudi[1 2] and Laurent George[1]

[1]LIGM/ESIEE Paris, 5 boulevard Descartes, Champs-sur-Marne, France
[2]SIC/ECE Paris, 37 Quai de Grenelle, 75015 Paris, France
Email: {aghiles.djoudi, nawel.zangar, laurent.george}@esiee.fr, rafik.zitouni@ece.fr

***Abstract*—Most traffic light's control systems in smart cities are wired and have a semi-static behavior. They are time-based, with pre-configured pattern and expensive cameras. Although traffic lights can communicate wirelessly with incoming vehicles, they are less adapted to an urban environment. If we consider light signs as an Internet of Things (IoT) network, one issue is to model thoroughly the change of signs' states and the Quality of Service (QoS) of this network. In this paper, we propose a new architecture of Urban Traffic Light Control based on an IoT network (IoT-UTLC). The objective is to interconnect both roads' infrastructures and traffic lights through an IoT platform. We designed our IoT-UTLC by selecting motes and protocols of wireless sensor network (WSN). Message Queuing Telemetry Transport (MQTT) protocol has been integrated to manage QoS. It enables lights to adapt remotely to any situation and smoothly interrupt traffic light's classic cycles. Our experimental results show that the MQTT protocol is efficient when the packets rate exceeds 35% of traffic flow, it reduces traffic delay up to 0.05s at 90% of congestion. After verification and validation of our solution using a UPPAAL model checker, our system has been prototyped. Motes' functions have been implemented on Contiki OS and connected through a 6LoWPAN/IEEE 802.15.4 network. Time-stamping messages have been performed throughout the system to evaluate the MQTT protocol with different QoS levels. In our experiments, we measured the Round-trip delay time (RTT) of messages exchanged between the WSN and IoT Cloud. The results show that MQTT decreases the RTT when the Cumulative Distributed Function (CDF) of generated messages exceeds 35%.**

I. Introduction

According to the French agency of statistics on roads' accidents, 12% of them happen in intersections caused by the non-compliance to traffic rules. 23% of them led to hospitalization in which 14% are fatal **Routiere2015**. Urban Traffic Light Control (UTLC) are one possible solution to regulate vehicle flows at intersections. However, a static cycle of traffic lights (or lights signs) has a direct impact on traffic jams, particularly when an emergency vehicle must cross through as quickly as possible. A long period at red or green light might impact the fluidity of the city traffic.

The Internet of Things (IoT) and Everything (IoE) can be a solution to adapt traffic light control to traffic density. It allows heterogeneous connected objects, *e.g.* Zigbee, LoRa, SigFox, ITS-G5, to interact and exchange sensed data on roads, vehicles, pedestrians presence, time of leaving house, etc **Perera2014**. Therefore, connecting heterogeneous infrastructures following a Device-to-Device approach is possible through upper layers or an intermediate Cloud platform. Wireless Sensor Networks (WSNs) are the source of these data, and the Cloud is the remote entity that collects them. Fog and Edge computing have been proposed to address the low latency of IoT applications by efficient resource distribution and a local processing. Fog computing leverages Edge devices and remote and private Cloud resources with distributed data processing. It provides the advantage to process data closer to the source and thus mitigate latency issues and reduce network congestion. However, constructing a real IoT Fog Computing is costly for evaluation while the environment has to be controllable for experiments **Dastjerdi2016**. However, in our work we would implement our Edge computing with a remote Cloud data gathering and deal with latency through QoS protocol. Remote Cloud offers the possibility to integrate new services. For example, we can deploy sensors to measure noise or air pollution via traffic signs or roads.

Our objective is to model, prototype and evaluate the Quality of Service (QoS) of an IoT solution for a traffic light control system. Modeling of traffic light states control is essential to avoid incoherent situations. Number of models based on Petri Nets (PN) have been proposed in **difebbraro_trafficresponsive_2006** and [1]. Their main drawbacks are the limitation to the structural analysis of state transitions and the lack of verification. However, our design is based on UPPAAL (UPPsala and AALborg Universities) [2] timed automata for design and verification of coherent state of cross road's traffic light. It specifies a graph of states with clocks and data variables. To implement our Urban Traffic Light Control based on an IoT network architecture (IoT-UTLC), we setup a real IEEE 802.15.4 WSN with devices that can act as actuators and sensors. Small traffic light signs are driven by a Border Router (BR) device to a sink node which is a gateway to the Internet. This BR is connected to a host computer (or sink) also connected to an IoT Cloud platform. WSN devices forward their data to the IoT Cloud through this sink which defines required levels of QoS based on Message Queuing Telemetry Transport (MQTT) protocol **Al-Fuqaha2015**. The collected data can be transmitted to different devices such as sink, BR or wireless sensor/actuator devices. When WSN devices detect the arrival of high priority vehicles, sensed data are routed

to the IoT Cloud. Then, the sink node takes a decision to change the light's state and forward generated messages to actuator devices through BR with a high level of QoS. Thanks to IPv6 over Low power Wireless Personal Area Network (6LoWPAN) **chalappuram_development_2016**, our WSN is energy-efficient and IPv6 compatible.

This paper is organized as follows. In Section II, we review related work. Section **??** reports the design of our prototyping solution. We describe the use case defined with the design model. Section **??** defines our prototype (IoT Testbed), giving our specifications and discussing on our choices of technologies and protocols. Finally, Section VI presents the obtained results that show the usefulness and the best practice for MQTT.

II. Related work

Petri nets (PNs) are widely used for traffic light modelling and control [1]. In **difebbraro_trafficresponsive_2006**, deterministic-timed Petri Nets have been used to describe signalized intersections. Undesirable deadlock states might appear when the nets are tested for some use cases. The authors in [3] have modified PNs models including stochastic-time for one single signalized intersection. Dotoli and Fanti [4] have built a colored timed PN with a deterministic modular framework, in which parts of the system, and even parts of the subsystems, can be specified and analyzed separately. Examples using modularity are given in Soares and Vrancken [5], in which a p-timed PN is used for the control of a traffic signal in both main road and side streets. However, formal characteristics of PNs (*e.g.*, deadlock and liveliness) havent been discussed. Moreover, PNs suffer from a lack of analysis and verification tools. To overcome these limits of PNs, we propose UPPAAL timed automata for design and verification of coherent state of cross road's traffic light. UPPAAL is a timed-based modelling software with a graphical user interface. It is the result of the research works of two universities UPPsala University in Sweden (UPP) and AALborg University in Denmark (AAL) [2].

In **Web0**, thermal cameras and on-street wired sensors detect vehicles and pedestrians in order to adapt the cycle of traffic light control systems. However, such a solution can be expensive. In addition, the system uses only its local view of the environment to detect the arrival of a vehicle. Other solutions use recent technologies such as wireless sensors devices to limit the cost of thermal cameras and reduce the time needed to deploy sensors. In **tlig_decentralized_2014** and **rose_internet_2015**, the authors propose an adaptive system based on local wireless communication between lights and vehicles. But such a solution requires a global interconnection between all road's users and infrastructure. This problem comes from the rigid definition of technologies' standards. Our work is not only limited to establish WSN, but it is scalable to interconnect heterogeneous wireless technologies through the Internet. The obtained WSN intends to meet multiple QoS requirements of IoT applying the MQTT protocol. In **Silva2018**, the latency of MQTT has been evaluated by calculating the average round-trip delay between two clients located in two different continents. However, the evaluation has been limited to the impact of messages' size. In our work, we consider the period of generated messages, and we calculate the RTT delays from WSN to Cloud IoT plateform.

In [1] **difebbraro_trafficresponsive_2006** [3] [5], the authors focus only on the structural analysis of their models and the transitions between colors of traffic lights. However, the implementation of their models as a service in the Internet of smart cities has not been discussed. Moreover, their methodology is not tested with any real traffic lights' Testbed.

III. Related work

IV. Use Case and Model Design

Our objective is to build a robust Testbed that behaves like a real urban traffic light control system. As presented in section II, a crossroad traffic light design has been proposed based on PNs [1]. Authors demonstrated the necessity to monitor checkpoints like traffic light transitions from red to green and define critical control points to ensure that the transition model is correct. Authors have identified which signal indication sequence optimizes the overall system performance. We were inspired from this work by adding the vulnerability of wireless network *i.e.* packet loss. Indeed, this model is theoretical and static, and would not model entirely traffic light control system. Thus, we propose a UPPAAL timed automata for modelling and verification of the system.

A. Use case

We consider the use case of traffic lights at an intersection crossed by high priority vehicle like ambulances, fire-fighters or public transportation systems. Crossing delays are important in such use cases, when the goal is to travel in the city from two locations without experiencing traffic jams. We consider the case of a high priority vehicle approaching a traffic light sign on red state. The detection of priority vehicles via (RFID or touch sensor) triggers the transmission of notification to road signs' network asking for a switch of traffic light to green. In that situation, it should be possible to interrupt the usual cycle of a crossroad.

In order to be as close as possible to a real urban traffic light, we prototyped the closest Testbed of a crossroad in Paris. We took an actual crossing point with its dimension and static timers. Fig. 1 shows IoT interconnection of four traffic lights, high priority vehicles and roads. Our system is described by one intersection (or crossing) of two roads A and B, with two traffic lights by road. The signs and roads can use heterogeneous technologies as presented with different colors.

The roads are connected to the Internet through sensors, which are able to detect the arrival of high priority vehicles. The detection is performed by Touch sensors driven by WSN nodes. Note that signals on each road should always have different colors (or states). Of course, when the road A traffic light is on green, the state of the traffic light on the opposite road, must be on red and vice versa. To access the Internet, all messages sent by those objects will go through a Border

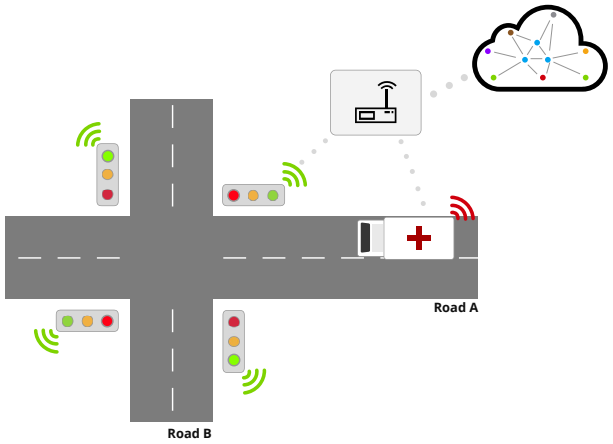

Figure 1. Use case illustration

Router (BR) device and a Middleware or Edge computer. The Middleware is connected to the Internet and forwards all packets to a Cloud. Therefore, the collected data in the Cloud allow a Middleware to decide for the future state of lights. Then, BR disseminates this decision through the network to the actuators.

B. Design Model

Our design is based on UPPAAL model checker software. It specifies a graph of states with clocks and data variables. It allows us to model how our system works and simulates all possible traffic lights states. The modelling allows us to cover all possible cases of lights change of our IoT-UTLC. It is also a tool for verifying formally the consistancy of traffic light changes: GREEN to YELLOW states, YELLOW to RED states and RED to GREEN states. We simulated our system by three automata shown in Fig. 3, Fig. 2 and Fig. 4 and available at (https://github.com/IoT-UTLC/Resources). We proved that our model worked without deadlock and starvation. It means that in our system, there is always a transition to go to the next state. It proves that the system will not stop functioning over time. Incoherent situations, like four signals on GREEN, must not happen.

Fig. 2 highlights the model of traffic lights and describes their behavior. The change of light's state is based on requests and confirmation exchange between lights and the Middleware. We defined two roles for the traffic lights: one is set to master mode and the second one is set to slave mode. Masters send requests every 60 seconds to change their states while considering the current ones. Line 6 of Algorithm 1 shows the condition when a light should change its state. CLK state represents clock or time progress. *reqAG* and *reqAR* are respectively the requests to ask for GREEN and RED states on road A. The same is specified for the road B. When it receives a confirmation from the Middleware, a cycle is started to send the signal with the desired state. Every 30 seconds, the traffic lights can change their states, starting with GREEN light and then to YELLOW light for 3 seconds after that switching to RED. They can also start from RED and then change to GREEN state after additional 3 seconds. We add this extra delay to avoid a dangerous situation when a GREEN state is on the two roads A and B at the same time. Even if we have messages lost due to the wireless nature of the network, our UPPAAL models ensure that this situation doesn't arise. It has been introduced after experiencing a latency between the WSN and the Cloud platform (see section VI). When GREEN or RED states are actuated, confirmation messages are sent to the Middleware. The pseudo Algorithm 1 summarizes this mechanism.

Algorithm 1: Traffic light

```
1  init_60s_timer(); while true do
2      if end_timer() then
3          send_request_new_state();
4          reset_timer();
5      end
6      if msg_received_red() and my_state is green then
7          change_state(yellow);
8          wait();
9          change_state(red);
10         send_confirmation_to_middleware();
11     end
12     if msg_received_green() and my_state is red then
13         change_state(green);
14         send_confirmation_to_middleware();
15     end
16 end
```

The model in Fig. 3 defines the different possibilities in terms of internal cycles depending on the request made by traffic lights (*reqAG*, *reqBG*, *reqAR* *reqBR*). Mainly, the Middleware sends the message to the Cloud and waits for its response. Then, it sends messages to traffic lights master and slave to change their state following this order: every light go to RED before setting GREEN signals. It also uses acknowledgments from the traffic lights to ensure that the new state has been set. In order to ensure these two features, we used a system to retain messages if the IoT Cloud Platform send GREEN states before RED states (see line 3 of Algorithm. 2). Algorithm 2 shows a simple description showing how the Middleware confirms the order of traffic lights changes.

We introduced the IoT Cloud Platform model shown in Fig. 4. It simulates the subscription mechanism according to messages sent by the Middleware to update collected data. We defined two states RED and GREEN without a transition state like YELLOW state defined for the traffic lights. The condition *touchA* indicates if the road A detects the high priority vehicle. The name *touch* is related to the type of sensor integrated in our prototype (see next section). The Cloud confirms to the Middleware that the state is changed by sending a message *confirmA*.

Exchanged messages within our WSN are based on IEEE 802.15.4 stack. And our Middleware defines QoS levels of

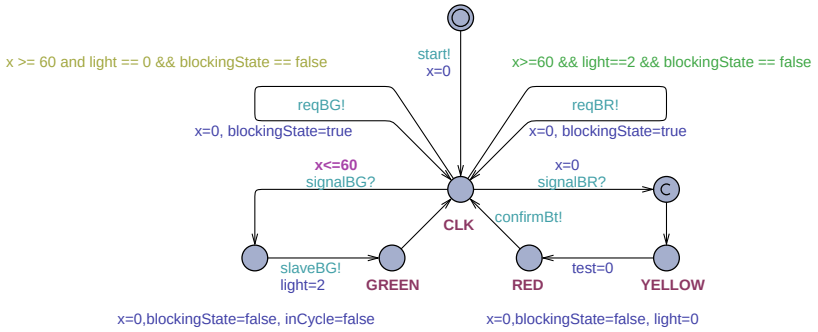

Figure 2. Model of our Traffic Lights in UPPAAL

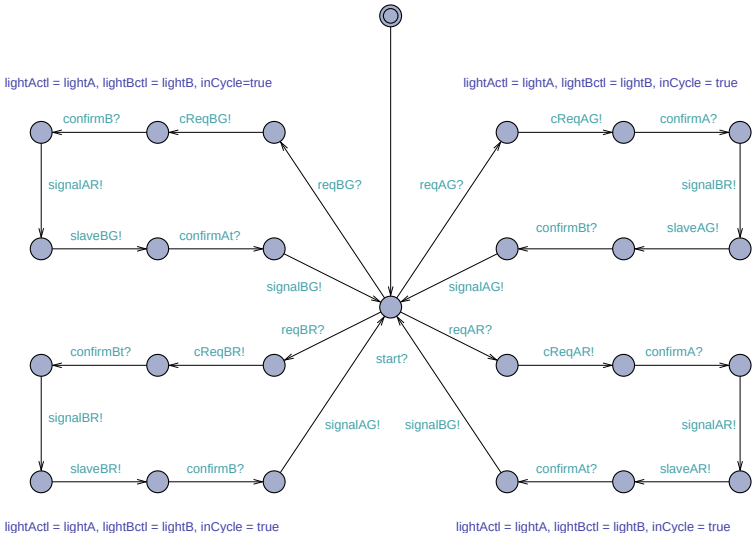

Figure 3. Model of our Middleware in UPPAAL

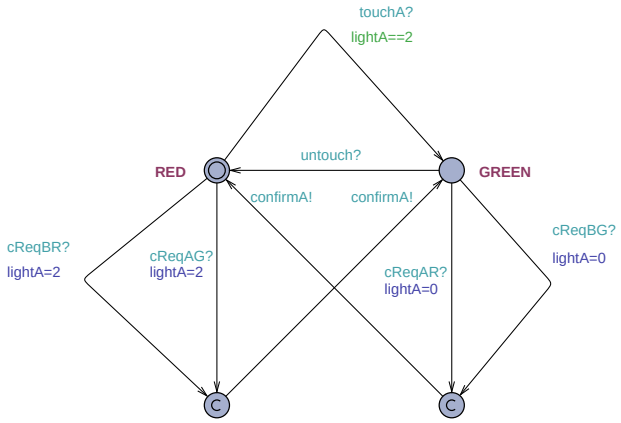

Figure 4. Model of our Cloud variables in UPPAAL

exchanged messages via MQTT protocol.

V. Prototyping

We have prototyped the wireless sensors and actuator's network of traffic lights and roads on a mockup [1] of real intersection in Paris with a scale of 1:68. Our specifications have been defined considering the low-cost and energy efficiency of the solution. This Testbed is a proof of concept of not limited to our use case as it is scalable for other applications. For example, additional sensors of fine particules could be implanted bringing correlation between traffic jam and pollution.

Fig. 5 shows the architecture of our IoT-UTLC with three layers. From left to right, we have the WSN layer with connected traffic lights actuators, sensors and IEEE 802.15.4 transceivers. The second part is the gateway of the WSN ensured by the BR and the Middleware *i.e.* Python script launched by host computer. The last layer is the Ubidots IoT Cloud Platform. It is an open source solution used to collect and analyze WSN data.

A. 6LoWPAN, Contiki OS, Re-Mote and Border Router

Our WSN is an IPv6 LowPower Wireless Personal Area Network (6LoWPAN) based on IEEE 802.15.4 stack. It is well adapted to embedded wireless devices with energy aware constraint and for its capabilities to define a mesh topology. Contiki Os[2] has been used to implement networks' functions

[1] https://github.com/IoT-UTLC/Resources/wiki

[2] http://www.contiki-os.org/

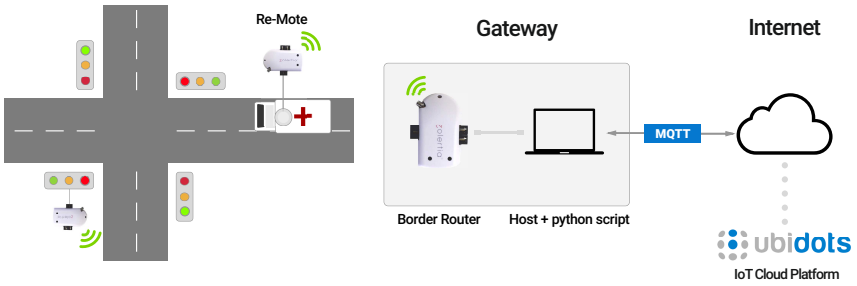

Figure 5. Architecture of Iot-UTLC

Algorithm 2: Middleware confirmation

```
if message_received() then
    if is_green() then
        retain_msg(); //green then red
    end
    else
        if retained_msg_exist() then
            update_to_red(); //green then red
        end
        else
            update_to_red(); //red then green
        end
    end
end
while true do
    confirmation_red_lights();
    update_to_green();
    confirmation_green_lights();
end
```

such as send, receive and data processing. It is an embedded operating system with large open source community. It supports Zolertia's Re-Motes [3] and implements recent IEEE 802.15.4 standard specifications. It also includes protocols such as RPL, CoAP and MQTT. Furthermore, developer community is active and makes available source codes examples in order to help developing quickly new applications.

Re-motes are compatible with our WSN specifications and our design model. They are wireless devices with ultra-low power operation mode. This choice has been motivated by long radio range of its IEEE 802.15.4 CC1200 transceiver, which transmits in the frequency band of 868-915 MHz. In addition, each Re-Mote has analog and digital ports with a possibility to connect several analog sensors and actuators. A Re-mote can be driven by a computer and become a sink and/or BR as well as a gateway between the 6LoWPAN network and the computer.

To implement the previous model described in Section

[3]https://github.com/Zolertia/Resources/wiki/RE-Mote

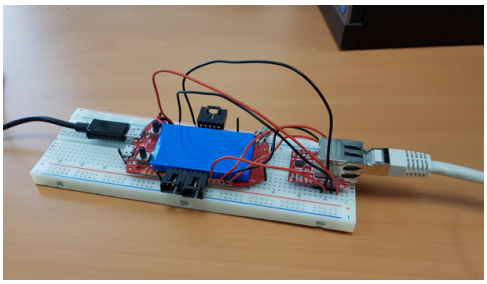

Figure 6. BR and sink combined on one board

??, we used six Re-motes: one for the BR, four to control the traffic lights and one Re-Mote to detect the arrival of a high priority vehicle near a crossing point. For simplicity, we choose a touch sensor as a detecting device of priority vehicle. We have developed four types of programs running on a Re-mote: traffic lights signs, sensors, high priority vehicle detecting device and BR function. Sensors send periodically information to the IoT Cloud Platform with temperature, pressure or any relevant information that can be sensed. As mentioned in the previous section, traffic lights are sub-divided into two modes: slaves and masters. Masters nodes are the only ones to request the Middleware to change its lights state and slaves simply change its state depending on received packets. These roles are defined to reduce the overhead of network, redundancy and collisions, for instance. Masters send periodically packets to request a change of state to the Middleware which forwards them to a Cloud platform.

BR node behaves differently compared to the other Re-motes. The entries of its routing table are the list of Re-motes that pass through it. It reroutes every packet it receives from its neighboring to host computer (or sink), which creates a connection to the IoT Cloud platform. Two options are possible to create our BR: i) separate the BR and sink and ii) combine both on the same device. In our development, we worked on how to implement the sink and the BR nodes on the same Re-Mote board. Fig. 6 shows a prototype of the combined BR and sink, both connected to an ethernet interface. Indeed, if the border router becomes an Ethernet router, there will no longer be any connection between the host/sink machine and the IoT Cloud platform. Every Re-mote

is able to connect independently to the IoT Cloud platform. This approach has some advantages, such as the autonomy of the devices, but it generates an overhead requiring extra synchronization packets' exchange. Therefore, we separate the sink and BR, since this solution is more flexible and resilient for our Testbed.

B. MQTT and UBIDOTS

Fig. 7 presents the layers of our UTLC network. From bottom to up, the WSN network sense and/or detect, process and actuates traffic lights. The second layer manages the 6 LowPan addressing and routing of packets throughout an IEEE 802.15.4 network. The Edge Computing is the Middleware between the WSN and the Cloud platform. For the setup of our UTLC, we start by establishing the access network of WSN. The next step is to connect this network to Core network. MQTT protocol controls three levels of QoS of exchanged packets from the WSN to the chosen Ubidots [4] Cloud platform. It adopts IntServ approach for supporting quality of service in the network, it tags incoming packets in the border routers with different levels of priority. Core routers read incoming packets headers and queue them according to their priority, packets with a high priority are sent faster compared to low priority ones.

MQTT ensures the QoS and publish/subscribe mechanisms through a broker. The broker behaves as a server by filtering messages and organizing them in topics, which are strings used to filter messages and define the hierarchy of our data structure. They allow us to organize how to receive multiple data from sensors such as temperature, up time, battery status and how to display them and obtain a real-time glance of our system. It gets its messages from publishers and sends any modifications to entities, which that subscribed to the updated topics. We used this mechanism with the Middleware in order to publish messages to the broker and get from the main topic the new values of the subscriber.

The QoS feature of MQTT protocol manages network resources by handling retransmissions and guarantees the delivery of messages. It allows more control on messages by defining the level of guarantee. By default, the QoS is defined by three levels. The first one, level 0, is 'At most one'. Level 1 is 'At least one' where there is an acknowledgment to let the sender know that its packet has been received. Finally, level 2 'Exactly once' is the highest verification level with a request/response flows to ensure that only one message will be delivered and processed by the receiver. In our case, we applied levels 1 and 2 using *paho.mqtt.client* Python library. For example, publisher of high priority data such as touch sensor has to indicate the highest level of QoS by the code shown bellow. We shared our implementation and its source codes at https://github.com/IoT-UTLC/contiki.

```
payload = json.dumps({"RoadA": data, "RoadB": 0})
res, mid = conn.publish(MQTT_URL_PUB, payload,
          qos=int(QoS)) # QoS is QoS level to use
```

[4]https://ubidots.com/

We experienced significant latency of high priority messages when we tested of IoT-UTLC mockup. Therefore, assessments of the MQTT protocol in our case provided significant information about its efficiency.

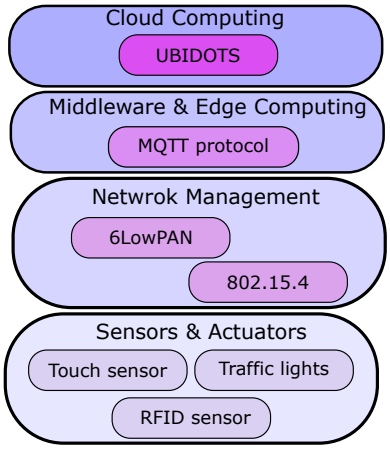

Figure 7. UTLC network layers

VI. Results

In this section, we report experimental results of urban traffic-light control system based on MQTT protocol. We report the limitations of our solution. To test the efficiency of MQTT protocol, we made 2 scenarios, in the first scenario, the frequency packets sending is 1 packet per 10s, the second scenario, the frequency packets sending is 1 packet per 1s. The measured delays have been taken into account between the Middleware (Edge) to the Cloud platform (Ubidots) throughout the Internet. Note that we don't know the routes and the routers that our packets will go through. We measured the Round Trip Time (RRT) for the packets exchanged between the two sides. In each scenario, more than 100 values have been taken. Tests have been reproduced for two different hours and days.

Fig. 8 shows the measured Cumulative Distribution Function (CDF) of RTT delays for the two levels of QoS, levels 1 and 2. We consider three probabilistic distribution functions (Normal, Gamma and Logistic) in the experimental results in order to characterize MQTT performance. The obtained correlation allows us to define a representative empirical model. Table I details the results of a correlation matrix between distributions and experimental results. We can see that logistic distribution **STEPHENS1979** fits better with our measured RTT values for the two scenarios. The standard logistic law is of parameters 0 and 1. Its distribution function of a random variable x is the sigmoid following the expression:

$$F(x) = \frac{1}{1+e^{-x}}, \text{where } x \in [-\infty, +\infty] \qquad (1)$$

Fig. 9 highlights the empirical model of CDF featuring the RTT of MQTT protocol. As can be seen, RTT protocol is more efficient when the number of packets is greater than 35% of

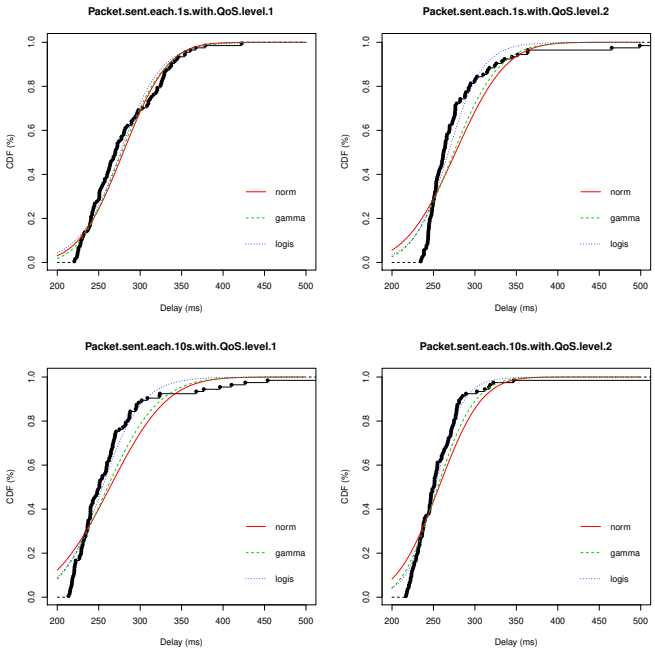

Figure 8. Normal, Gamma and Logistic distribution

Table I. Correlation between distributions and empirical results

	norm	gamma	logis
1s with QoS level 1	172.12074	175.2950	**185.4433**
1s with QoS level 2	159.59630	172.8193	**189.7002**
10s with QoS level 1	146.85668	161.2369	**175.3682**
10s with QoS level 2	176.28502	192.6108	**204.3235**

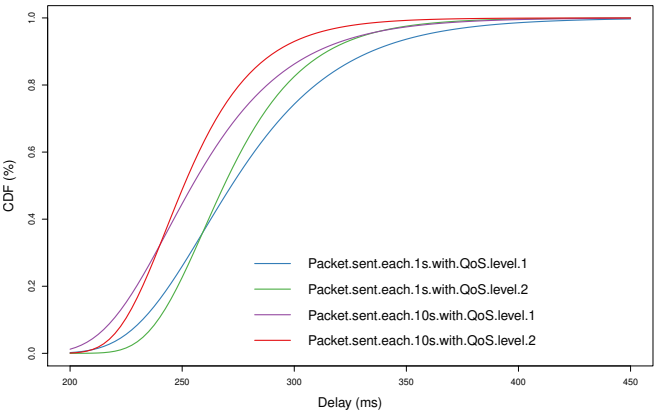

Figure 9. Cumulative distribution function of RTT delay for two QoS levels

the total number of packets sent. This can be explained by the fact that priority queues are useful when queues of QoS are full. In that case, the packets with a highest priority level will reach their destination with low latency. Furthermore, when we increase the number of packets sent to Ubidots at one per second, the MQTT still offers the same efficiency with suitable delays. The found latency of up to 400 ms would be problematic for real world safety applications which require at most 100 ms **Chen2017**.

Although our Mockup is innovative by combining IEEE 802.15.4, 6LowPAN, MQTT protocol and Edge Computing, the performances of our solution depend on external parameters. These parameters are related to Internet Service Provider and all packet's routes throughout Internet network from the Middleware to Ubidots. Thus, real implementation of such a system should be done by introducing a private Cloud near data sources.

VII. Conclusion

In this paper, we proposed an Urban Traffic Light Control (IoT-UTLC), considering its architectures elements and tools used to build an IoT lockup. We reported three main contributions in this work: i) Modelling through UPPAAL of crossing's traffic lights, ii) Prototyping of IoT Edge Computing, iii) Performance assessment of MQTT protocol. Traffic lights control has been taken as an IoT network. WSN has been deployed on motes running Contiki Os and exchanging IEEE802.15.4/6LowPAN packets. MQTT was the QoS protocol between our WSN and Ubidots Cloud platform. UPPAAL model checker design ensured that lights' colors change is adaptive to the arriving of a priority vehicle.

Our experiments have investigated the relationship between the MQTT protocol and the traffic flow congestion. Our results showed that the MQTT is efficient when the number of packets sent exceeds 35% of the total number. The packets with the highest level of QoS has low latency than other packets. The protocol remains efficient since the delay of priority packets decreases when the network overhead increases. However, found latencies of up to 400ms is higher than the expected one for vehicular safety application. Thus, the proposed IoT architecture and protocols must be improved to consider the safety requirements.

While we are still developing our prototyping, we plane to integrate other use cases such as smart buildings and industrial IoT. As a near future work, we plane to extend our experiments with private Cloud towards a real Fog Computing.

Acknowledgement

We would like to thank our colleague Sebti Mouelhi, associate professor at ECE Paris, who provided us insight and expertise on UPPAAL.

References

Others

[1] Y. Huang, Y. Weng, and M. Zhou, " Modular Design of Urban Traffic-Light Control Systems Based on Synchronized Timed Petri Nets ", *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 530–539, Apr. 2014, 00050.

[2] A. David, K. G. Larsen, A. Legay, M. Mikuionis, and D. B. Poulsen, " Uppaal SMC Tutorial ", *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, Aug. 2015, 00176.

[3] A. D. Febbraro, N. Sacco, and D. Giglio, " On Using Petri Nets for Representing and Controlling Signalized Urban Areas: New Model and Results ", in *2009 12th International IEEE Conference on Intelligent Transportation Systems*, 00018, Oct. 2009, pp. 1–8.

[4] M. Dotoli and M. P. Fanti, " An Urban Traffic Network Model via Coloured Timed Petri Nets ", *IFAC Proceedings Volumes*, 7th International Workshop on Discrete Event Systems (WODES'04), Reims, France, September 22-24, 2004, vol. 37, no. 18, pp. 207–212, Sep. 1, 2004, 00101.

[5] M. dos Santos Soares and J. Vrancken, " A Modular Petri Net to Modeling and Scenario Analysis of a Network of Road Traffic Signals ", *Control Engineering Practice*, Special Section: Wiener-Hammerstein System Identification Benchmark, vol. 20, no. 11, pp. 1183–1194, Nov. 1, 2012, 00017.