Large-Scale Performance Evaluation of the IETF Internet of Things Protocol Suite for Smart City Solutions

Javier Isern, August Betzler*, Carles Gomez†,
Ilker Demirkol†, Josep Paradells†
Urbiotica, Barcelona, Spain
*I2CAT Foundation, Barcelona, Spain
†Department of Network Engineering, UPC, Barcelona, Spain
javier.isern@urbiotica.com
august.betzler@i2cat.net
{carlesgo, ilker.demirkol, josep.paradells}@entel.upc.edu

ABSTRACT

The Internet of Things (IoT) intends to interconnect massive amount of heterogeneous, smart devices, with the goal of interweaving the virtual world with the physical world. Smart Cities are typical IoT application domains, comprising networks with large number of sensors that survey environmental data in order to provide different services, such as City Mobility solutions that facilitate on-street parking and traffic flow monitoring. Standards and specifications defined by the Internet Engineering Task Force (IETF), like IPv6, the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) adaptation layer, and the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) are cornerstones of the IoT. In this paper we carry out large-scale experimental evaluations in an IoT testbed of 251 nodes to analyze the performance of the IETF IoT protocol suite in such a large-scale network. We define a City Mobility Solution application, using traces from a commercial Smart City deployment and the commonly employed Contiki implementation of IETF IoT protocol suite. The results show that the out of the box Contiki IoT protocol stack is not capable of delivering a satisfying performance. However, after a thorough analysis of the initial results, a set of improved parameter configurations is derived that allows the network to achieve much higher performance. Among others, improvements of 60.39% in PDR and 63.67% in delay are achieved. Furthermore, the paper presents and discusses the technical solutions and best-practice guidelines for a specific City Mobility solution being developed by Urbiotica, a company with ample expertise in Smart City deployments.

Keywords

Internet of Things, 6LoWPAN, RPL, CoAP, Smart City, City Mobility, Contiki

1. INTRODUCTION

During the last decade, the Internet of Things (IoT) has experienced a significant growth, fueled by the evolution of wireless technologies and the development of miniaturized, low-cost wireless devices that are capable of connecting to the Internet. In IoT communications, devices interchange information, such as sensor readings, often finding their main applications in automatized large scale deployments. One salient scenario for such large scale deployments are Smart Cities [10], which benefit from the standardization process led by the Internet Engineering Task Force (IETF) for protocols that are used in the IoT.

One decade ago, the IETF decided that IPv6 would be used as the network protocol for constrained devices such as the aforementioned ones. IPv6 offers an address space large enough to support the plethora of devices that form the IoT and it has built-in support for network auto configuration. Further, to support IPv6 on constrained, low-power devices with limited hardware capacities, the IETF has specified the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) adaptation layer [12], which enables fragmentation of IPv6 frames and IPv6 header compression. The routing mechanisms proposed by the IETF to establish multihop network topologies is the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [14]. At the application layer, the IETF has developed a lightweight RESTful protocol called the Constrained Application Protocol (CoAP) [13].

To our best knowledge, no experimental large-scale evaluations of IoT stacks in testbeds or real Smart City environments have been published so far. Several studies carry out evaluations in network simulators or discuss existing deployments. For example, a proposal for the use of an IoT Stack in an urban environment is made by Zanella et al. in [15]. After providing a survey of the technologies required for an urban IoT system, an existing Smart City deployment is discussed. Yet, the authors only describe the deployment, rather than carrying out an experimental performance analysis. Rothenpieler et al. [11] follow a different approach: They propose an architecture for point-to-point networks, where personal devices (like smart phones) dynamically connect to each other to manage the transport of sensor data. While this proposal is discussed throughly, the authors do not implement their approach to validate it in a real deployment.

In order to determine whether a common IoT communication protocol stack is capable of delivering a satisfying performance for Smart Cities, and to provide a reference for future Smart City deployments as part of the IoT, in this paper we perform large-scale experimental evaluations for a City Mobility solution that is composed of on-street parking [2] and traffic flow monitoring applications. The first one monitors the availability of parking spaces, whereas the second quantifies the number, speed and type of vehicles driving in the streets. We choose this City Mobility solution for our evaluations, since it is based on communications between constrained devices and it generates large amount of data when comparing with other type of City solutions such as pollution, waste, sound or lighting monitoring.

The experimental evaluations of the City Mobility solution are carried out in the large-scale *IoTLab* testbed [5], using 251 nodes and a testbed layout matching real world Smart City deployments, permitting an accurate analysis of Smart City scenarios. Use of such public and remotely accessible IoT testbed also permits the reproducibility of the experiments. Traffic patterns have been derived from actual smart city deployments. To our best knowledge, this is the first time that such a large-scale IoT-based experimental evaluation of the IETF IoT protocol suite is investigated.

The node operating system is Contiki, a widely used platform that provides the de-facto implementation of the IETF IoT protocol suite. We analyzed the effect of several IoT stack parameters on different performance metrics including application layer metrics such as the packet delivery ratio (PDR), the end-to-end delay and lower layer metrics such as frame drop rate and topology change rate.

The results show that stability and network performance strongly depend on the settings of the protocol stack parameters and that the out of the box configuration of the analyzed communication protocol stack is not capable of delivering the necessary performance required by the evaluated Smart City use case. We then derive a set of improved parameter configurations for the communication protocol stack that allows the network to achieve a high performance for a wide-range of traffic load investigated. With the recommended parameter configuration, we improve PDR by 60.39% and delay by 63.67%, reaching the values of 96.06% and 788 ms of PDR and delay, respectively.

The remainder of the paper is organized as follows: In Section 2 we define the Smart City use case considered for the communication protocol analysis in this paper. The description of how the use case is implemented in the testbed is described in Section 3. In Section 4 the experimental results for the default configuration of the protocol stack are presented. Also, changes to increase the performance of the protocol stack are proposed and the results for these improved settings are exposed. Conclusions and proposals for future lines of work are given in Section 5.

2. CITY MOBILITY SCENARIO

The evaluations carried out in this paper focus on the experimental analysis of a Mobility solution for Smart Cities. In this section we identify typical network characteristics of this use case based on topology and traffic information obtained from real Smart City deployments.

In this process we determine a representative network topology, identify different logical roles of the devices in the network that comprises the City Mobility solution, and derive a set of exemplary data traffic patterns. In the following subsections we carry out a detailed evaluation of each of these characteristics, eventually explaining how they are carried over to our experimental evaluations performed in the IoTLab testbed.

2.1 Use Case and Topology

A prevalent use case for Smart City networks is the large-scale deployment of wireless sensor nodes (SNs) to carry out monitoring tasks, such as the detection of vehicle traffic flows, the supervision of available parking spaces, or the collection of environmental data such as noise, temperature or air pollution.

For these purposes, SNs are strategically deployed on the streets, buildings, or other physical structures, such as traffic lights or lamp posts. For the experimental evaluations carried out in this paper, we choose a City Mobility scenario that combines two use cases: The monitoring of on-street parking spaces [2] and the measurement of traffic flows. In this scenario, the SNs use their sensors to detect vehicles based on the changes these cause in magnetic fields. Equipped with low-power radios, the SNs transmit this data over the air to a border router (BR) that is the gateway responsible of forwarding the data to servers located in the cloud for further processing.

The deployment of such a solution imposes several challenges on the design of the network. For instance, the SNs are placed underneath the surface of the street, which strongly limits their radio transmission range. Furthermore, the SNs are reduced function devices (or *leaf* nodes), thus they are not serving as possible relay for other nodes' data. Since the SNs are battery-powered devices, they apply Radio Duty Cycling (RDC) to reduce consumption and to operate for as many years as possible.

In order to connect all the SNs of the network to the BR, a backbone of routing-capable transport nodes (TNs) is deployed in such Smart City solutions. The TNs serve as relay nodes for data packets that need to travel across the network. To achieve the largest coverage possible and at the same time network stability, the TNs are placed strategically, so that every SN can connect to one or more TNs or directly to the BR.

The location of the nodes in a deployment therefore depends on their transmission ranges and the layout of the streets. In modern cities, the layout of the streets often follows the pattern of a grid with horizontal and vertical streets intersecting at angles of 90 degrees and where 'isles' of buildings are located between the streets. Examples for such street layouts are the Manhattan district of New York or the Eixample in Barcelona, and similar layouts can be found in many other cities.

2.2 Data Generation Model

In this subsection we define the type and volume of data that will be generated in the experimental evaluations across the network. The data traffic patterns are based on statistics from real deployments of on-street parking and traffic flow applications.

We define two types of SNs, which generate notification messages that report either the change of a parking space availability or the detection of a vehicle in a street lane, depending on the type of sensor. The number of packets sent by a SN depends on how many cars enter or leave a

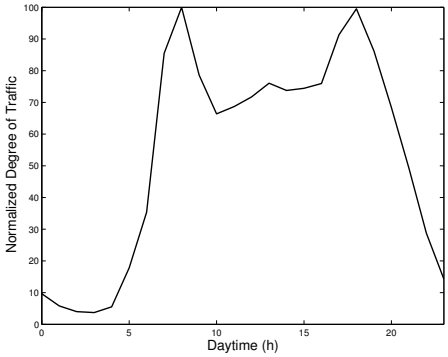

Figure 1: Average number of SN events measured during one day, averaged over a week and normalized with respect to the maximum number of events detected.

Table 1: Number of Events per Hour Originated by Each Node for LOW, MEDIUM, and HIGH Traffic Loads.

Type of Node	LOW	MEDIUM	HIGH
Parking Space	5.76	9.6	14.4
Traffic Flow	42.35	80	360
Transport Node	6	6	6

parking space or the amount of cars that circulate in the streets, respectively.

Figure 1 depicts the normalized number of events registered commonly by nodes per hour during a day in a real Smart City deployment of Urbiotica company [1]. Consider that the maximum value depends on the type of SN. From these statistics we derive 3 degrees of traffic loads (LOW, MEDIUM, and HIGH): During night time, the notifications produced by both type of sensors are LOW since there are almost no cars circulating on the streets, nor parking or leaving parking spaces. In the morning hours the number of detected vehicles increases and reaches a peak value. A peak is also measured during the late afternoon. We associate these peaks to HIGH traffic. As MEDIUM traffic we define the mean number of packets transmitted over the course of one day. Apart from the traffic generated by the SNs, each TN generates control messages containing different types of network statistics every 10 minutes. Table 1 indicates the number of packets generated by the SNs and the TNs.

All the messages are sent to the BR where, in the real deployments, the data is forwarded to the servers. The time between the generation of notification messages in the experimental evaluations carried out are based on an exponential distribution, using the values provided in Table 1 as mean values.

3. EXPERIMENTAL SETUP AND CONFIGURATION OF THE CONTIKI STACK

In the previous section the use case, the network topology and the traffic patterns of the City Mobility solution to be evaluated in this paper have been specified. In this section we introduce the IoTLab testbed and the experimental setup used for the large-scale evaluations of the Contiki IoT stack and we explain how the Smart City use case deployment is mapped to the testbed.

[1]http://www.urbiotica.com/

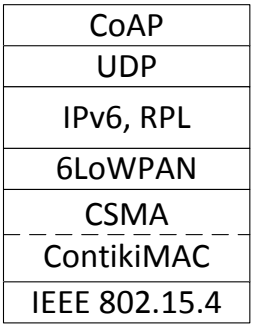

Figure 3: The IoT communication protocol stack implemented in Contiki.

3.1 The Grenoble IoTLab Testbed

The experimental evaluations are carried out in the indoor IoTLab testbed located in Grenoble, France [5]. The Grenoble testbed is composed of a total of 938 motes, of which 384 are *M3 Open Nodes* [4], representing the largest available fraction of motes of the same type in the testbed. The M3 nodes incorporate an IEEE 802.15.4-based AT86RF231 radio transceiver operating at 2.4 GHz and the STM32F103RE MCU that offers 64 kByte of RAM and 512 kByte of ROM.

The large number of M3 nodes available to carry out experiments and their sufficient memory capacities provide an ample play for large-scale network evaluations. More importantly, the layout of the M3 testbed favors the analysis of Smart City scenarios, since the lanes of nodes in the testbed represent intersecting streets with nodes similar to real deployments. Figure 2 shows a graphical representation of the locations of the M3 motes in the testbed that are used for the experiments.

All the nodes are located in an environment vulnerable to external conditions, for example interference from other wireless devices and/or moving obstacles or persons that affect the radio links. Similar phenomena are characteristic of real Smart City deployments.

3.2 The Contiki Protocol Stack

The nodes in the testbed are programmed with Contiki, implementing a fully IPv6-based IETF protocol suite for the IoT. The protocol layers implemented by Contiki are depicted in Fig. 3. In the following, we introduce each of the protocol layers of the Contiki stack and explain its mechanisms from top to bottom.

Contiki comes with Erbium CoAP [7], which implements CoAP as detailed in RFC 7252 [13]. CoAP is a Representational State Transfer style (RESTful) protocol designed for IoT communications. It is a lightweight alternative to HTTP with its own set of options and mechanisms well suited for networks of constrained devices that have very limited hardware capacities in terms of memory, processing power, and radio technology. CoAP offers the methods GET, PUT, POST, and DELETE to manipulate resources on servers. CoAP operates over UDP, which in Contiki is implemented by the *uIP* TCP/IP stack. Since UDP does not provide reliability at the transport layer, CoAP allows the option of confirmable messages, requiring an acknowl-

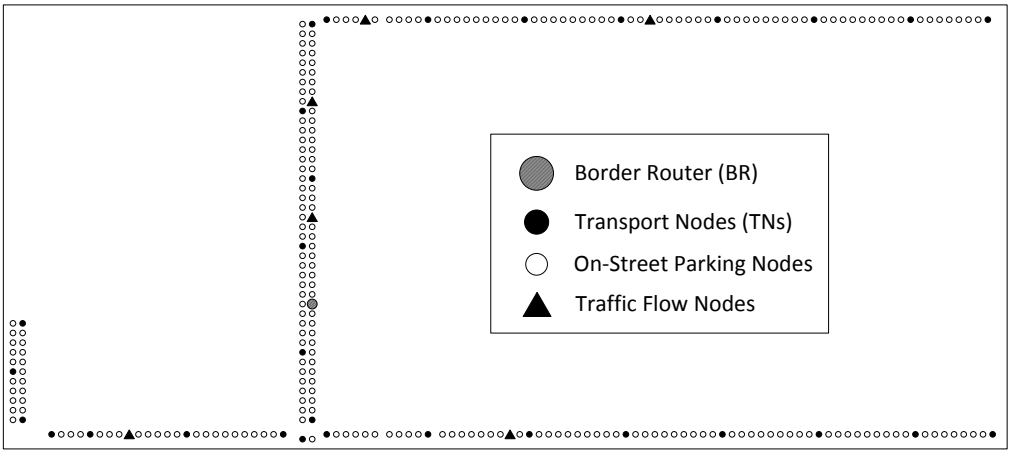

Figure 2: Map of the M3 motes in the Grenoble testbed that have been used for the experiments.

edgment (ACK) to confirm the reception at the destination. If no ACK is received the CoAP message is retransmitted. By default, up to 4 retransmissions are allowed before the CoAP message is dropped.

The Contiki implementation of RPL is the mechanism used to establish the multihop network topology. RPL constructs a Destination Oriented Directed Acyclic Graph (DODAG) and uses this graph to route data traffic, applying the Trickle algorithm [6] to regulate the transmission of DODAG Information Objects (DIOs), used for the DODAG construction and maintenance. RPL has been designed and optimized to route data (e.g. obtained by sensors) through the DODAG towards a special node called the root, which collects the data. RPL nodes use Objective Functions (OFs) to compute the network topology and routes.

In order to join a DODAG, a node either can wait to receive DIO messages from nearby nodes or it can send a DODAG Information Solicitation (DIS) to request DIO messages from a subset of neighboring nodes. Each node in a DODAG selects a DODAG parent set, which is composed of the nodes that provide connectivity to the rest of the nodes in the DODAG. An important concept in RPL is the Rank property, which abstracts the distance between a DODAG node and the DODAG root. In order to calculate the Rank property, the OF is used, considering metrics and constraints such as ETX, Latency, Hop-Count (HP), Link Quality Level (LQL) or Remaining energy.

Contiki RPL operates in the so called storing mode, where each node maintains the routing information necessary to reach child nodes. In this mode, if a routing-capable node receives a packet destined for another node it checks if it knows the next hop towards the intended destination. Otherwise, the packet is forwarded to its default parent, i.e., one hop in direction of the DODAG root (in this case, the BR). This process is repeated (up to reaching the BR), until a node has a valid routing table entry for the intended destination, and that node forwards the packet following the routing table entry.

To support the use of IPv6 over the low-power IEEE 802.15.4 physical layer, Contiki implements the 6LoWPAN adaptation layer, which provides IPv6 header compression, fragmentation and reassembly, as well as an adapted version of Neighbor Discovery. Underneath the 6LoWPAN layer, Contiki implements an optional carrier sense multiple access (CSMA) mechanism. To avoid collisions when sending data frames, devices may use CSMA to sense the channel for ongoing transmission and to back off the transmission in case of detecting a busy channel. As default Radio Duty Cycle (RDC) mechanism, Contiki implements ContikiMAC [3]. In ContikiMAC, the radio of the device is only turned on either to start a data transmission or to periodically check for possible incoming data. To transmit data with ContikiMAC, packets are strobed until either a MAC layer ACK is received or until the maximum burst duration limit is achieved. The frequency with which the nodes check for data and the maximum duration of the data bursts is determined by the parameter *Channel Check Rate* (CCR). At the bottom of the stack, Contiki implements the IEEE 802.15.4 physical layer which allows a data transmission rate of 250 kbps in the 2.4 GHz band.

The Contiki protocol stack comes with a large set of default configuration parameters. For the first part of the evaluations carried out in this paper we refer to the default configuration of Contiki as the *Default Configuration*. In Section 4 some of the most relevant parameters of the stack are highlighted and their impact on the performance is evaluated.

3.3 Testbed Setup

As explained in Section 2.1, there are 4 types of nodes in the City Mobility solution use case: the BR acts as the bridge between the local network and the Internet; TNs form the data transport backbone, and two variations of SNs (are used for parking space and traffic flow monitoring). Except for a few details, the configuration of the Contiki stack is the same for all types of nodes.

The data generated by the nodes is transmitted as payload of confirmable CoAP messages. If the data is not acknowledged by the destination, the message is retransmitted. This is necessary to compensate packet losses due to congestion or bad links, given the fact that CoAP operates over UDP that does not provide reliability. By default, up to four retransmissions are allowed for each CoAP message. We set the RPL routing table sizes and the maximum number of storable RPL neighbors to be large enough to avoid the corresponding tables to fill, which can cause malfunctioning of routing. Contrarily to the BR and TNs, SNs are not mains powered, which requires them to apply a RDC

to save energy. By default, Contiki applies the ContikiMAC RDC mechanism, where nodes can enter the sleep mode while not sensing the channel, receiving or transmitting data. Whereas to maintain the communication compatibility with the SNs, the BR and TNs also apply the frame transmission pattern as defined by ContikiMAC, they can keep their radio turned on permanently while they are not transmitting. The nodes communicate over IEEE 802.15.4 channel 26 to avoid possible external interference sources like Wi-Fi transmissions. The BR and TNs transmit at 3 dBm since it is the maximum power available for this hardware. On the other hand, to model the real case scenario where SNs are buried into the streets as explained in Section 2.1, we configure them to use 0 dBm of transmission power.

In the topology studied, as depicted in Figure 2, the BR is located at the center of the grid, as it should be in any Smart City application. Further, to facilitate the connectivity between the SNs and the BR, TNs are placed in regular intervals in lanes and at the crossings, like they would be placed in the streets of a real deployment. Since the TNs have a limited transmission range, it is necessary to guarantee that the TNs are close enough to each other in order to form the data transport backbone. Via experimental evaluations, we determine that the guaranteed transmission range of TNs is 10 grid nodes in a straight line. Thus, at least every 10 grid nodes a TN is placed. Above that, at every crossing as many TNs as the number of branches joining are placed to increase connectivity between the lanes. This is due to the fact that there are no nodes located at the intersections in the testbed and some of the TNs are not within line of sight of each other. After applying these settings to our testbed topology, we end up with a network of 220 SNs that rely on a transport backbone of 30 TNs to communicate with the BR.

With the described network setup we measure an average network depth of 3.3 hops during the experiments. We take a snapshot of the network topology during a test run we measure the distance of each node to the BR. Figure 4 depicts the percentage of nodes that are at a certain distance from the BR.

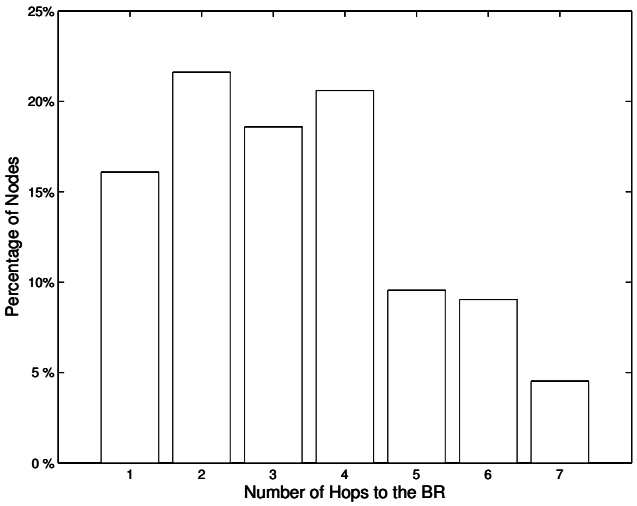

Figure 4: Snapshot of the percentage of nodes with different distance hops from the BR. Nodes that are 7 or more hops away from the BR appear as 7 hops in the statistic.

Table 2: Results for the Default Configuration at MEDIUM Traffic Load. The Average Frame Drops and Parent Changes are Given per Minute and per Node. Average and 95% Confidence Intervals are Given for Each Metric.

Metric	Result
PDR	$59.89 \pm 2.84\%$
Packet Delay	2164 ± 174 ms
Frame Drops	3.2 ± 1.96
Parent Changes	1.5 ± 0.08

4. EVALUATION RESULTS

This section presents the results of the experimental large-scale evaluations of the Smart City application scenario. First the results obtained for the Default Configuration are presented, followed by a discussion of how its performance can be improved for large-scale Smart City applications by adapting the protocol stack settings. Based on this initial analysis, a set of improved parameter settings for the Contiki protocol stack is determined and experiments are repeated with the updated settings. For each configuration analyzed, experiments with a duration of 60 minutes are repeated 5 times.

4.1 Performance Metrics Used

The performance of the different communication protocol stack configurations is measured in terms of two well-known application-layer performance metrics: the end-to-end PDR and the end-to-end delay. Other metrics, like the parent change rate of RPL and the number of frame drops, are also taken into account as performance indicators.

The overall PDR of a test run is calculated by dividing the total number of successfully delivered CoAP packets at the application layer by the number of CoAP packets generated by the SNs and TNs. The PDR describes how successful the communication protocol stack is in transferring packets through the network. On the other hand, the end-to-end delay is the total time between the creation of a CoAP packet and its processing at the destination node (in our setup, the BR). The end-to-end delay indicates the time it takes for the network to react to events, such as car presence, and is an important metric for a real time monitoring.

Besides these two main performance metrics, we also measure the number of times nodes change their RPL parent, where frequent changes are an indicator for instability. Further, we count the number of frames dropped at the MAC layer and use this information as an indicator for network congestion.

4.2 Evaluation of the Default Configuration

We apply the default configuration parameters and run the experiments for the MEDIUM traffic load, which corresponds to the average traffic expected over the course of a day. The results for these experiments are given in Table 2. A PDR of only 59.89% is achieved, meaning that a substantial part from the application layer data generated and transmitted from SNs and TNs is lost, in spite of using reliability mechanisms. We also observe a long average delay that exceeds 2 s. We identify two main reasons for the poor performance of the Default Configuration: a high degree of congestion that leads to a large amount of frame drops and

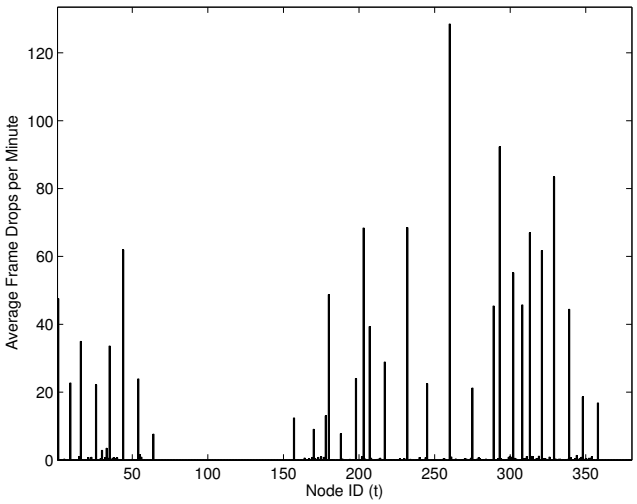

Figure 5: Average frame drops per minute and per node for the Default Configuration. The unique node IDs used are assigned by the testbed. Nodes with 0 drops are either not part of the experiment or do not suffer frame losses.

an unstable behavior of RPL that increments congestion.

The MAC layer provides a limited queue size for frames that are about to be transmitted over the radio. Similarly, it limits the number of neighbors for which frames can be stored. If either of these limits is exceeded when a new frame needs to be transmitted, the new frame is dropped. In the analyzed scenario, the types of frames that are generated are either CoAP frames at the application layer or RPL frames required to maintain routing information. A high degree of frame losses at the MAC layer therefore not only causes a malfunction of the application, it also prevents RPL from working correctly, i.e., the correct execution of its routing mechanisms. Figure 5 shows the average number of frames dropped per minute by each one of the nodes, where the majority of the nodes with large number of frame drops are TNs. Mainly, TNs suffer from congestion since they act as relay nodes that forward packets for other nodes. Some of these TNs suffer from heavy packet losses of up to 128 packet drops per minute, which indicates that there is heavy congestion in their neighborhood. Nodes that interconnect branches or are close to the BR are more affected than others.

Also, a high number of RPL parent changes is observed, which can heavily impact the network performance: After every parent change, the Trickle timer is reset, therefore increasing the DIO generation rate for the affected node. If many nodes keep changing their parents throughout the experiments, the number of DIOs is permanently high (the network does not reach or maintain a 'stable state').

As a result of the MAC layer drops and the frequent parent changes, the Default Configuration is offering poor performance in the analyzed use case. In the following section we address the issues observed to provide improved settings for the Contiki stack in order to enhance the performance.

4.3 Evaluation of the Smart City Protocol Stack

Contiki's Default Configuration of IETF IoT protocol suite performs poorly in the analyzed City Mobility application as a result of two major issues: MAC layer packet drops and RPL instability. In order to improve the performance

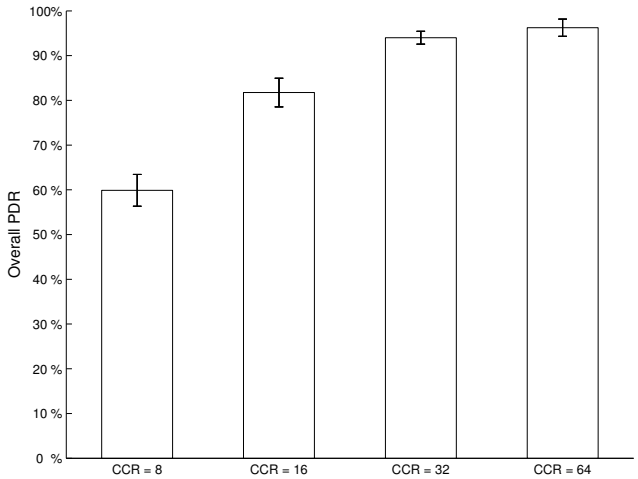

Figure 6: Average PDR values and 95% confidence intervals measured for different settings of CCR at MEDIUM traffic.

of the Contiki stack, we propose alternative settings of the stack that are adapted to the characteristics of the analyzed network.

4.3.1 Determining a Recommended Configuration

A closer look at the behavior of the MAC layer mechanisms reveals that the setting of the Channel Check Rate (CCR) parameter heavily determines the performance of the MAC layer. Carrying out experiments with a MEDIUM traffic load and with the CCR parameter set to the default value of 8 and the alternative values of 16, 32, and 64, reveals that the performance in terms of PDR can be greatly improved with larger CCR values, as shown in Fig. 6.

The increase of PDR observed with each doubling of the CCR can mainly be related to the fact that the nodes wake up more frequently to check for transmissions, increasing the responsiveness of the nodes and decreasing one-hop delays. Apart from that, the transmission burst duration is reduced, decreasing the potential time a node occupies the medium until it obtains a MAC layer ACK, effectively reducing the degree of channel congestion.

However, the increase of performance by incrementing the CCR comes at an important cost: with higher CCRs, the duty cycles of the SNs become noticeably longer, since the SNs wake up more often to check for incoming data packets. This applies even if the network is silent and there are no ongoing transmissions.

To determine an optimized setting of the CCR parameter for our evaluations, we refer to work by Dunkels [3] and Ali [1] who analyze the effect of the CCR on the performance of IoT networks in detail. Both sources come to the conclusion that a value of 16 is either an optimal setting (Dunkels) or that it should be the upper limit (Ali). Increasing the CCR further would heavily decrease the longevity of the nodes due to a much higher duty cycling. Since the SNs deployed in the Smart City scenario network need to achieve very long lifetimes to avoid costly battery replacements, we follow these guidelines and use a CCR of 16.

In the next step, we determine improved configurations for RPL parameters in order to reduce the number of control messages generated during the experiments.

The Trickle algorithm [9] is the main responsible mechanism for determining when and whether RPL messages need to be transmitted. To decrease the RPL traffic overhead, there are several parameters of the Trickle algorithm that can be adjusted and that have a direct impact on the number of generated routing packets. We focus on a subset of Trickle parameters that have the most relevant impact on the amount of generated messages: The minimum Trickle interval size (Imin), the RPL DIO redundancy constant (K) and the Parent Switch Threshold (PST) defined by the Objective Function used, which is the Minimum Rank with Hysteresis Objective Function [8].

Trickle controls a timer that determines the size of the interval during which a node may perform a DIO transmission. The value of the interval is set to Imin during the initialization of the algorithm or after a node changes its RPL parent. By default, Imin in Contiki is set to 4.096 s. Each time the Trickle timer expires, if no inconsistencies (i.e, topology changes) are detected, the interval is doubled, up to the maximum value determined by Imax[2]. This means that as time passes and if the network is stable, the number of messages generated by Trickle continuously decreases.

Thus, by increasing the Imin value, it is possible to reduce the number of DIO messages not only during the network initialization phase, but also when there are RPL message drops. While the initialization of the network is a unique event during an experiment, changes of parents are observed repeatedly over the course of an experiment. In order to reduce the network congestion as reaction to such an event, we increase Imin to 8.192 s, the upper limit recommended by Ali et al. [1].

Another important parameter that passively determines the number of generated control messages is the RPL DIO redundancy constant (K). Each TN keeps track of the number of DIO transmissions in its vicinity and each TN's own DIO transmissions are suppressed if at least K DIO transmissions are overheard within the current interval.

Therefore, a reduction of K potentially decreases the RPL control overhead by limiting the number of transmitted DIOs within a neighborhood [6]. In a dense network with static nodes it is reasonable to assume that the network topology is close to stable, thus a lower DIO dissemination rate seems acceptable. In Contiki, as in the RPL specification, K is set to 10 by default. According to the Trickle specification [9], in general a value between 1 and 5 is suggested. We choose a value of 3 for the improved settings.

Finally, the PST parameter controls the hysteresis used when deciding which node is the preferred parent. Therefore, it can be used to reduce the rate of parent changes and hence, to increase the network stability. Table 3 gives a summary of how the improved values discussed in this subsection are set in comparison to the Default Configuration. We name this configuration the *Recommended Configuration*.

4.3.2 Recommended Configuration Results

We run tests with the Recommended Configuration for the 3 traffic loads defined in Table 1 for the City Mobility solution, the results of which are given in Table 4.

Comparing with the MEDIUM traffic PDR results, using the Default configuration in Fig. 6, the PDR of the Recommended Configuration improves by 60.39%. At the same time, the delays decrease by 1.38 s, representing an improvement of 63.67%. These results evidence significant network performance improvements.

[2]In Contiki up to 10 interval doublings are allowed.

Table 3: Contiki Default Parameter Changes to Reach the Recommended Configuration Proposed.

Parameter	Default	New Config.
IMIN	12	13
K	10	3
PST	ETX_DIVISOR/2	ETX_DIVISOR/1
MAC Frame Queue	8	10
MAC Neighbor Queue	2	5

Table 4: Results for the Recommended Parameter Configuration. Average Values are Given with 95% Confidence Intervals. Frame Drops and Parent Changes are Given per Minute and per Node.

Traffic	Low	Medium	High
PDR	$95.88 \pm 0.65\%$	$96.06 \pm 0.72\%$	$96.07 \pm 0.06\%$
Delay (ms)	745 ± 54	788 ± 53	859 ± 64
Frame Drops	0.63 ± 0.33	0.58 ± 0.34	0.64 ± 0.37
Parent Changes	0.86 ± 0.05	0.91 ± 0.05	0.98 ± 0.04

The results also indicate that, even though delay and drop rates increase as the traffic load increases, the PDR only decreases marginally. This is due to the application layer reliability mechanism. At a higher traffic load, a greater number of retries is necessary due to frame losses. These retries allow to maintain a high PDR, at the expense of a delay increase.

The network stability is also improved as the parent change rate decreases because of the adjusted value of the RPL Objective Function PST. The average parent change rate for each node decreases by 39.33% in comparison with the results measured for MEDIUM traffic with the Default Configuration. Again, the different traffic loads do not lead to noticeable differences in the mean parent changes rate. As a further benefit, the Recommended Configuration also helps reducing RPL overhead since a parent change leads the involved node to reset its RPL interval to Imin, thus increasing the RPL DIO message count.

5. CONCLUSIONS AND FUTURE WORK

Configuring an IoT stack to achieve a satisfying performance in a Smart City environment is not a trivial task. Evaluations performed of IoT networks in real large-scale testbeds or deployments are not available in the literature. In this paper we carry out an experimental evaluation of a large-scale City Mobility application with different traffic loads and node roles. We performed it in a large-scale testbed to carry out experiments with the Contiki communication protocol stack.

The results obtained demonstrate that with the default configuration parameters used by Contiki the network does not achieve an acceptable performance in the analyzed City Mobility solution losing more than the 40% of the information generated. We find the ContikiMAC and RPL layers of the Contiki protocol stack to have a significant effect on the overall network performance.

By combining the modifications of the ContikiMAC and RPL layers, a recommended stack configuration with im-

proved parameter settings is derived for the analyzed City Mobility solution. This Recommended Configuration achieves a PDR and end-to-end delays of 96.06% and 788 ms respectively for the MEDIUM traffic load. This represents an improvement of more than 60% in both performance metrics when comparing with the Default Configuration analysis. Further, we achieve to improve the network stability by reducing the parent change rate in 39.33%. These results demonstrate that networks based on the IETF IoT protocol suite are adequate for being incorporated in the City Mobility solution.

Finally, we derive a series of guidelines that should be taken into account when constructing the TN-backbone for a Smart City scenario:

- The position of the BR and TNs is critical for the network performance. If the backbone network built by the TNs is too dense, bad positioned or too disperse, it can heavily decrease the PDR. Main causes can be high degrees of congestion if the backbone is too dense or a missing variety of links if the nodes are bad positioned or dispersed.

- The TNs located close to the BR should be stable and there should be a variety of TNs to ensure that even in case that one fails, the rest of the network remains reachable. This is critical due to the chain-like network that is automatically constructed by RPL and due to the generally linear layout of streets, where few alternative routes are available.

- The density of the TN backbone network depends on the transmission ranges that can be achieved from one TN to another. If the connection quality between two TNs is not sufficient to establish a stable connection, inserting another TN in between those two nodes might be necessary to assure connectivity.

It can be demonstrated that with the improved settings proposed in this paper, the basic requirements for a large-scale IoT City Mobility solution are met. Yet, the performance analysis reveals that there seems to be enough room for further improvements via the adjustment of other protocol stack parameters. In future work, we plan to carry out a comprehensive study considering a wide set of parameter and mechanism settings at all protocol stack layers, looking at various Smart City network topologies and use cases. Further improvements, such as a higher reduction of RPL traffic overhead and advanced congestion control mechanisms, not only could benefit the performance of the analyzed City Mobility solution but the performance of large-scale IoT networks in general.

6. ACKNOWLEDGMENTS

This work was supported in part by the Spanish Government's Ministerio de Economía y Competitividad under grant number RYC-2013-13029, through project TEC2012-32531, FEDER, the Spanish Government's Ministerio de Industría, Energía y Turismo, under the program AEESD 2013, through the project Smart Gateway Evolution and by the Generalitat de Catalunya, Departament d'Economia i Coneixement through the project of Doctorat Industrial.

7. REFERENCES

[1] H. Ali. *A Performance Evaluation of RPL in Contiki.* PhD thesis, MS thesis, Blekinge Institute of Technology, 2012.

[2] J. Chinrungrueng, U. Sunantachaikul, and S. Triamlumlerd. Smart parking: An application of optical wireless sensor network. In *Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on*, pages 66–66. IEEE, 2007.

[3] A. Dunkels. The contikimac radio duty cycling protocol. 2011.

[4] Fit/IoT-LAB. M3 Open Node. https://github.com/iot-lab/iot-lab/wiki/Hardware_M3-node, June 2014.

[5] E. Fleury, N. Mitton, T. Noel, and C. Adjih. FIT IoT-LAB: The Largest IoT Open Experimental Testbed. *ERCIM News*, (101):14, Apr. 2015.

[6] H. Kermajani, C. Gomez, and M. H. Arshad. Modeling the message count of the trickle algorithm in a steady-state, static wireless sensor network. *Communications Letters, IEEE*, 16(12):1960–1963, 2012.

[7] M. Kovatsch, S. Duquennoy, and A. Dunkels. A low-power coap for contiki. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2011)*, Valencia, Spain, Oct. 2011.

[8] P. Levis. The minimum rank with hysteresis objective function. Technical report, 2012.

[9] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The trickle algorithm (rfc 6206). Technical report, 2011.

[10] J. Paradells, C. Gomez, I. Demirkol, J. Oller, and M. Catalan. Infrastructureless smart cities. use cases and performance. In *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[11] P. Rothenpieler, B. Altakrouri, O. Kleine, and L. Ruge. Distributed crowd-sensing infrastructure for personalized dynamic iot spaces. In *Proceedings of the First International Conference on IoT in Urban Space*, pages 90–92. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014.

[12] Z. Shelby and C. Bormann. *6LoWPAN: The wireless embedded Internet*, volume 43. John Wiley & Sons, 2011.

[13] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP) (RFC 7252). http://www.rfc-editor.org/info/rfc7252, June 2014.

[14] T. Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.

[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *Internet of Things Journal, IEEE*, 1(1):22–32, 2014.