Modelling and Performance Analysis of Wireless Sensor Networks Using Process Mining Techniques: ContikiMAC Use Case

Francois Despaux, Ye-Qiong Song, Abdelkader Lahmadi

Modelling and Performance Analysis of Wireless Sensor Networks Using Process Mining Techniques: ContikiMAC use case

François Despaux
LORIA, Université de Lorraine
Vandoeuvre-lès-Nancy, 54500, France
Email: francois.despaux@loria.fr

Ye-Qiong Song
LORIA, Université de Lorraine
Vandoeuvre-lès-Nancy, 54500, France
Email: song@loria.fr

Abdelkader Lahmadi
LORIA, Université de Lorraine
Vandoeuvre-lès-Nancy, 54500, France
Email: abdelkader.lahmadi@loria.fr

***Abstract*—In the current protocol stack for Internet of Things in general and wireless sensor network in particular, many devices rely on the ContikiMAC protocol at their MAC layer. This protocol is widely used and enabled by default for several industrial environments and time sensitive monitoring and control applications. However, few work exists regarding the performance of this protocol because it lacks of an underlying theoretical model for analysing its performance. In this paper, we propose a novel approach relying on process mining technique that aims to obtain a Markov chain model for networks running the ContikiMAC protocol. In particular, we present a comprehensive specification of the protocol and a Markov chain model obtained through the analysis and instrumentation of its reference implementation. We used the obtained Markov chain to analyze and estimate the end to end delay distribution for a multi-hops transmission with static routing. The approach can also be extended to a wide range of protocols.**

***Keywords*-MAC Protocols; Markov chain; Process Mining; Network performance.**

I. Introduction

Understanding the behavior and limitation of wireless sensor networks is important for estimating their performance metrics such as end to end delay, throughput, energy consumption, etc. Consequently, modelling the behavior of the networks becomes essential for estimating these metrics and further take decisions for improving the network performance. A lot of research work has been done to model the network through different methods, including analytical modelling and simulation based analysis. Due to its high dynamic nature, wireless sensor networks present a number of challenges which do not exist, or exist in rather different forms, in traditional wired networks. Therefore, modelling the behavior of such networks is challenging and not a straightforward task. Normally, proposed models abstract the reality in order to simplify the analysis and thus they are not accurate enough for estimating the performance parameters. Let's just take the example of the widely spread standard IEEE 802.15.4 MAC protocol [1]. In [12], authors present a Markovian model from which a set of performance results were obtained. However, a queue of size one was considered on each node, which does not represent the reality. A more complete model for this protocol was proposed by Misic et al. in [11] where many aspects of the protocol such as duty cycle and finite buffer size were considered and nodes were modeled as a $M/G/1/K$ queue system. However, the model lacks of a realistic radio channel model and capture effect model. Besides, the extension of the analysis to more complex multi-hop transmissions is not feasible.

Many other analytic models have been proposed in the literature in order to model the behavior of other duty cycled wireless sensor network MAC protocols such as S-MAC [18] , X-MAC [3] . Those models catch some main features of the protocols to give asymptotic performance trends in function of traffic load, allowing generally a qualitative rather than quantitative comparison of protocols. However, when one is interested by evaluating the performance of a network running a protocol, but not a protocol itself, it is still very difficult, to not say impossible, to apply those existing models. One of the reasons is that most of those models don't include neither network-related parameters (e.g. actual channel model, capture effect) nor actual implementation details (OS and implementation limits and optimizations). Otherwise the model would be too complex to be analytically resolvable. This difficulty is more stringent when considering multi-hop networks since the input flow to the forwarders is generally unknown and not necessarily Markovian. There exists only few work dealing with multi-hop network performance analysis. Two well-known alternative ways are simulations and testbed or field measurements. There are many tools for simulating wireless sensor networks but it is still difficult to take into account some network and implementation details (e.g. capture effect and the impact of OS). Measurements on actual networks can effectively capture all the features, but it is difficult to draw general conclusions since they are carried out for a particular case. Moreover some performance parameters are hard to be measured in practice.

In this paper we propose a novel approach for modelling the network behavior. Our approach combines the measurement-based and analytic approaches. Differently

[0]This work was partially supported by Quasimodo project under No. ANR 2010 INTB 0206 01 in France and No. NSFC 61061130563 in China

from the existing performance measurement methods which directly focus on the performance metrics, we first instrument the protocol code and record the protocol execution trace on network nodes (rather than sniffer's traffic trace) to capture both implementation details and network physical parameters. Considering that the approach requires a protocol execution to generate the log files one can ask why do not directly measure the performance from the execution output. The answer to this question is that we are interested in finding a Markov model for modelling the protocol behavior that will allow us to estimate the probability distribution of the e2e delay. A measuring approach will give us the average e2e delay and would require a lot of executions and samples in order to estimate the probability distribution. In the second step, we use the process mining approach to extract a Markov chain that more accurately models the network behavior. This Markov model can then be used to further evaluate the performance parameters such as delays of the network. Of course the extracted Markov chain model is traffic dependent. Nevertheless, a useful practice may be to generate traces for several scenarios with different traffic patterns (e.g. light, medium and heavy traffic) and network conditions, allowing to extract more general conclusions of the protocol behavior. Finally for computing the end-to-end delays in a large scale multi-hop transmission scenario, this approach allows to bypassing the difficulty of modelling the input flows of the forwarders (generally not Poisson arrivals), by directly using the Markov chain of those nodes.

The main contributions of this work are summarized as follows.

- We show how to make novel use of Processing Mining technique to extract Markov chains from protocol execution traces.
- Our approach is an alternative way for modelling the network that encompass phenomenons not taken into account by existing theoretical models.
- Performance metrics such as end to end delay can be computed from the extracted Markov chain, a not trivial issue when considering multi-hop transmissions.
- Even though ContikiMAC protocol [8] is widely spread nowadays, it lacks an underlying theoretical model to lean on. We present a comprehensive Markov model for modelling the ContikiMAC protocol (version 2.6).

In this work we focus on the network MAC layer for obtaining a comprehensive Markov model of a widely spreaded MAC protocol. Our approach can be extended to a wide range of protocols and different topologies. In a previous work [5], we have applied our approach to the standard IEEE 802.15.4 MAC protocol obtaining a comprehensive Markov chain model that allowed us to estimate the e2e delay distribution for a multi-hop transmission scenario. The reminder of this paper is organized as follows. Section II presents main related existing work on analytic modelling of duty-cycled MAC protocols. Section III gives a background of the ContikiMAC protocol, as well as the process mining approach. Our combined measurement-analytic methodology is presented in Section IV. Samples of results are presented and discussed in Section V. Finally Section VI gives concluding remarks and outlines future work.

II. Related Work

In this section we only focus on the review of the main analytic models of MAC protocols for wireless sensor networks. Most of them are developed for the standard IEEE 802.15.4 MAC protocol. Only a few work can be found for other MAC protocols such as S-MAC and X-MAC. As wireless sensor networks should now provide not only energy efficiency, but also good performance, the latency issues of MAC protocols become critical for delay sensitive applications. A survey on latency issues of duty-cycled MAC protocols is provided in [7]. Authors provided expressions for both one-hop and end-to-end delay. Although they can be used to estimate the protocol efficiency in terms of delay, these expressions for the one-hop delay do not consider the queuing delay, which is an important component that impacts in the whole end-to-end delay. Besides, expressions for the end-to-end delay assumes there is a single traffic in the network, limiting thus its practical use.

As far as MAC protocol is concerned, the existing models focus on the single node behavior. Most of the proposed solutions for IEEE 802.15.4 are based on Bianchi's Markov model [2], initially developed for IEEE 802.11 standard. This model have been extended for modelling the IEEE 802.15.4 MAC protocol under different assumptions. In [12], authors proposed a Markov chain approach for modelling the slotted version of the IEEE 802.15.4 MAC protocol and give performance results in terms of service time and delay for successful packet transmission. The contributions of this work are a Markov chain model for the IEEE 802.15.4 mac protocol including explicitly retry limits, acknowledgement mechanism and unsaturated traffic conditions. From this model they derived the discrete probability distribution of the packet delay. However, the validation of theoretical results was done by means of Monte Carlo simulations and not in a real environment. We have shown in [6] that non negligible issues arise when considering the real world platform that may affect the estimation done by theoretical models. Another limitation of this approach is that it does not consider the fact of having a packet queue on each node. Actually, the queue capacity on each node is fixed to one packet. The model does not consider duty-cycle. Therefore, this model is not suitable for modelling a real scenario where packets arriving from the upper layers or even from neighbor nodes must be stored and delayed if the current node is busy. Misic et al. [11] proposed a Markov chain model for the standard IEEE 802.15.4 MAC protocol considering a $M/G/1/K$ system queue model and superframe with both

active/only and active/inactive duty-cycle periods for a star topology (one hop). Expressions for the access delay, probability distribution of the packet service time as well as probability distribution of the queue length are presented. The limitation of this model is that all results were obtained for 1- transmission where a device sends a packet to a coordinator and waits for the acknowledgement. Even considering a $M/G/1/K$ queue system for the first node, taking into account that the output distribution of a $M/G/1/K$ is not Markov, it is not possible to extend the proposed model for multi-hop transmissions by chaining $M/G/1/K$ queue system. Instead, a $M/G/1/K \rightarrow G/G/1/K \rightarrow G/G/1/K \cdots G/G/1/K$ queue system must be considered. However, modelling this kind of queuing systems is not straightforward. As in Park model, results are not validated by a real experimental environment.

All the above-mentioned work only deals with single hop case, so they cannot be readily used for evaluating multi-hop networks. [17] is one of the rare work dealing with multi-hop transmissions. A more general framework is proposed for including both channel, MAC and routing characteristics in the analysis. By considering the TinyOS default CSMA/CA MAC protocol (similar to IEEE802.15.4), each node is modeled by a Geom/PH/1/M queue. The e2e delay distribution is obtained and compared to both simulations and measurements. This is the most achieved work. Its extension for dealing with dynamic duty-cycled MAC protocol is, however, not obvious. Authors in [16], propose a system that can automatically infer a protocol state machine from real-world traces. However, their approach is based on network traces where normally no information regarding the underlying MAC protocol behavior is present. Besides, the suitability of this approach for estimating performance parameters such as e2e delay is not clear since the output of the system is a state machine where no information concerning the sejourn time on each state is available. Finally, authors in [9] developed a theoretical framework to estimate the end-to-end delay in a networked system using frequency-domain modelling and analysis where they shown that their approach is more scalable and allows analysis of compositional networked systems. In this paper, we apply this methodology to compute the e2e delay.

III. Background

A. ContikiMAC

ContikiMAC is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions. If the packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. Receiver sends then the corresponding acknowledgement. To transmit a packet the sender repeatedly sends its packet until it receives the corresponding acknowledgement. Broadcast packets are sent during the full wake-up interval and do not result in link-layer acknowledgement.

ContikiMAC has a power-efficient wake-up mechanism that relies on precise timing between transmissions. Figure 1 shows how timing is conceived in ContikiMAC where:

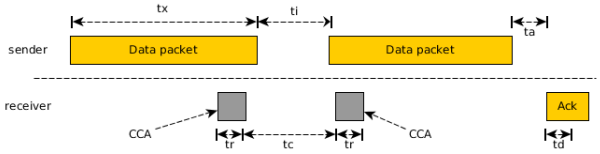

Figure 1: ContikiMAC transmission and CCA timing.

- t_i is the interval between transmissions.
- t_r is the CCA interval.
- t_c the interval between each CCA.
- t_a the time between receiving a packet and sending the corresponding acknowledgement.
- t_d the time for successfully detecting the acknowledgement.
- t_l the maximum packet length.
- t_s the shortest packet length. It must satisfy that $t_s > t_r + t_c + t_r$
- t_x the packet size. It must satisfy that $t_s < t_x < t_l$

Timing in ContikiMAC must satisfy the following constraint:

$$t_a + t_d < t_i < t_c < t_c + 2t_r < t_s.$$

ContikiMAC has also a mechanism called *fast sleep* which allows receivers to go sleep earlier if the CCA woke up due to spurious radio noise. This happens if:

- CCA detects activity but the activity is longer than t_l.
- radio activity is followed by a silence period longer than t_i.
- the activity period is followed by a silence period of the correct length followed by activity but no start of packet is detected.

B. Process Mining

Process mining has been widely applied in lots of fields and is an analysis method to construct models automatically through analysing the event logs. It can be considered as a branch of data mining. Traditional data mining methods aim at forecasting system behaviors while process mining at constructing whole process models. Mining algorithms are a key aspect in process mining. Many algorithms has been proposed in order to construct process model from event log files. Van der Aalst [13] proposes the α and β algorithms for discovering a workflow model based on Petri nets and developed the ProM mining tool [14]. Authors in [10] propose a method with derivation and statistics which uses Stochastic Task Graphs as the intermediate to obtain a workflow model. In [15], authors propose a sequence clustering algorithm for processes with high diversity of

behavior. The algorithm consists in dividing the log into clusters in order to analyse reduced sets of cases and find a Markov chain model for constructing the process model. The algorithm was implemented as a plugin in ProM. In this work we make use of this algorithm in order to obtain a Markov chain for modelling the network behavior. Given a set of clusters c_k, the algorithm starts by randomly initializing the state transition probabilities. The second step is to assign each sequence to the cluster that can produce the higher probability (see [15] for the probability expression). New transition probabilities are computed for each cluster and then the algorithm repeat the assignment of sequences to cluster until the cluster models do not change. In our case we consider only one cluster composed by a set of sequences, each of them consisting of states and transitions of the MAC protocol state machine. Therefore, the algorithm is reduced to find the frequences of transitions between states within the set of sequences.

IV. Methodology

In this section we introduce the design and implementation of our process mining approach. The process mining tool used in this work is the ProM data mining tool version 5.2. As we said before, we focus our attention on the network MAC layer so the analysis done here studies the underlying mac protocol of the network. However, and since the process mining takes in consideration the network traces, the approach can be easily extended to cover the whole network behavior. In Figure 2 we can see a flow diagram showing the steps to follow in order to obtain the Markov chain model from the protocol traces. In the next subsections we describe each of the components of the flow diagram.

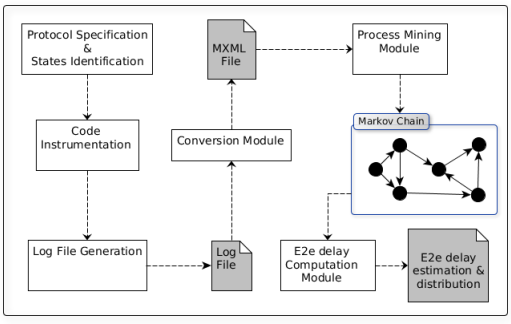

Figure 2: Step by step flow diagram of the analysis and modelling process.

A. Protocol Specification and States Identification

The first and one of the most important step in this process is the protocol specification and states identification. Each protocol has a specification from where we can obtain the main aspects of it. Usually, a flow diagram showing the main states and transitions between them is provided. Based on the specification of this protocol we can identify the main states (backoff, sensing, transmitting, ack received, etc.) and transition between them. All these states should be taken from the corresponding protocol specification. In our case, we have created the corresponding Contiki flow diagram from the specification of the protocol as shown in Figure 3 to identify both states and transitions.

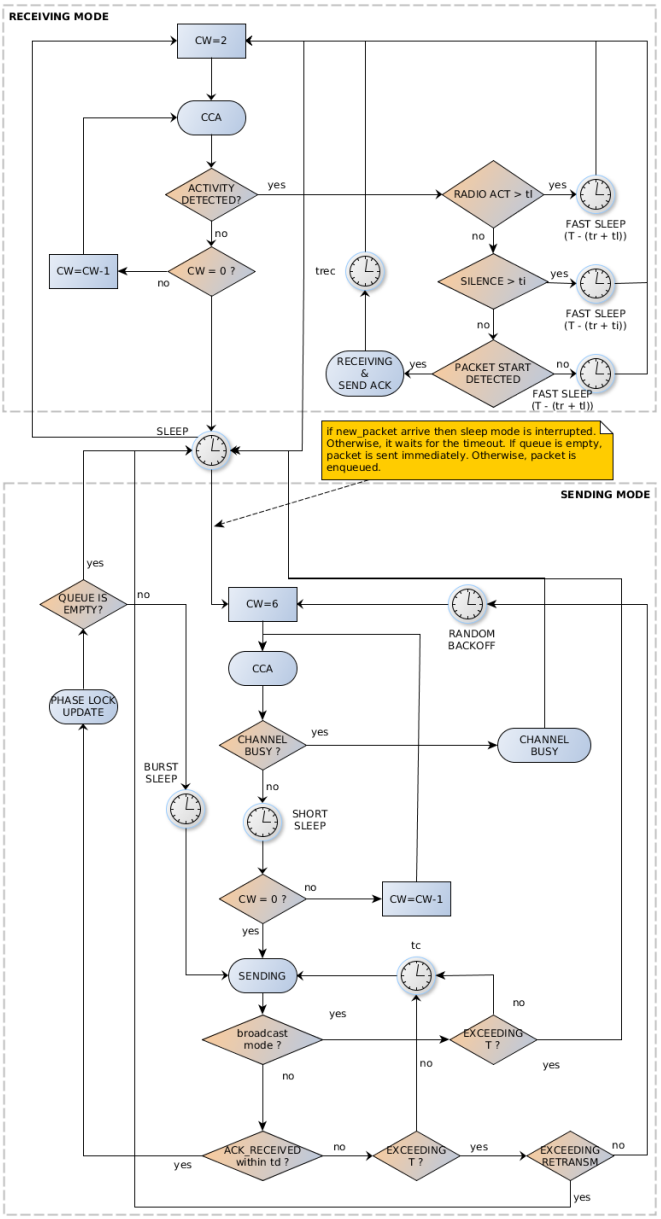

Figure 3: ContikiMAC flow diagram.

B. Code Instrumentation and Log File Generation

Once we have identified each state and transitions in the protocol the next step is to generate the log files in such a way that all states and transitions previously identified are present in the protocol execution output (log file). In order

to generate the log file it would be necessary to identify each state in the protocol implementation so that each state will appear in the protocol execution and therefore in the generated log file. In this way, the procedure consists in printing a line each time a change from one state to another is found during the execution. Since the Contiki code is written in C, is enough to add a *printf* command in the code whenever a new state is reached by the execution. In this way, we will obtain a trace log from the execution where each state should be present and transitions are represented by the previous and next state of the current state, that is to say, if state X is followed by state Z and preceded by state Y in the log file, then we have both Y $\rightarrow$ X and X $\rightarrow$ Z transitions. We also keep track of the timestamp in order to estimate the sejourn time spent on each state.

C. Model Extraction and End to End Delay Computation

1) Conversion Module: The propose of this module is to explain the conversion process, that is to say, how to extract a Markov chain model from the protocol execution code. The obtained log file give us the information regarding the states and transitions between states within the protocol execution. The log file can be grouped in a set of sequences, each one delimited by an initial state and a final state. The initial state is unique and is asociated to the moment a packet arrives to the node and is added to the node's queue. The final states may be two: when a packet is acknowledged or when a packet is discarded due to a transmission failed. In this way, we can see the log file as a set of sequences. An example of a sequence in a log file is shown in Figure 4. Then, the approach consists in processing each sequences

```
8685    ID:2    ENQUEUING
8686    ID:2    CCA1
8687    ID:2    SLEEP
8796    ID:2    CCA2
8768    ID:2    SLEEP
8879    ID:2    CCA3
8892    ID:2    SLEEP
9002    ID:2    CCA4
9005    ID:2    SLEEP
9102    ID:2    CCA5
9103    ID:2    SLEEP
9208    ID:2    CCA6
9210    ID:2    SLEEP
9310    ID:2    SENDING
9512    ID:2    ACK_RECEIVED
9700    ID:2    ENQUEUING
.....
.....
```

Figure 4: A sequence in a log file.

identifying the states (CCA, SLEEP, etc) within it and the transition between the states in the sequence. A transition from a particular state S is determined by the next state in the sequence. The idea is to determine the number of times (frequency) a transition from one state to another appears in the whole set of sequences. Therefore, after finishing processing the sequences we would have a set of frequencies of transitions from each state to the other ones. This set of frequencies will give us the probability of transition between a particular state to another one allowing also to generate a Markov chain model. This procedure is done by means of a process mining tool (ProM), concretely by the sequence clustering algorithm detailed before. The ProM tool needs a specific file format in order to generate the Markov chain. Then, this module will translate the network event log file into a readable MXML file that would be consumed by the process mining tool identifying the set of sequences previously mentioned. Therefore, it would receive the event log file as the input and after processing it sequentially it will translate it in MXML format.

2) Events Mining and Model Extraction: The environment in which this work is based is the ProM process mining tool version 5.2. The output of the parsing step done by the *Conversion* module provides us the input for the ProM tool which is the event log file in MXML format. We make use of the sequence clustering technique in order to find a Markov chain model from the event log file since it can be considered as a sequence of states. An implementation of this algorithm is provided by ProM. Then, we are able to find the Markov chain model by loading the MXML file generated by the *Conversion* module and applying to it the sequence clustering algorithm.

3) End to End Delay Computation Module: Finally, we implement the Performance Computation module which will compute the end to end delay between the source and the destination node. In order to estimate this performance parameter we make use of the approach presented in [9] taking the obtained Markov chain as the input. The Markov chain will give us the information of the transitions between states of the protocol together with the corresponding transition probabilities. Then, is it possible to compute the *Probability Transition Matrix* P. From P and the Laplace transform of the sejourn time distribution on each state we are able to compute the *Adjacency Matrix* A defined in [9]. Once obtained A, we proceed to compute the vector $\vec{A}^r_{i,d}$ representing the delay distribution in frequency domain of all connected paths of length r, $r = \{1, 2, 3, ...\}$, from state i to destination d. Then, the e2e delay distribution in time domain can be found by means of the *Inverse Laplace Transform* and the set of vectors $\vec{A}^r_{i,d}$ for $r = \{1, 2, 3, ...\}$, as we will see in next section.

V. Results & Discussions

A. Scenario Configuration

In order to carry out the experimentation we have set a testbed with TelosB motes. We consider an in-tandem topology as seen in Figure 5. We have chosen a small example in order to ilustrate our methodology. However, it can be easily extended to more complex scenarios. The distance between device-router and router-coordinator is set to one meter. Both device and router have a queue length of four packets. Packets are generated at the device as unicast

packets and are sent to the router which will then forward them to the coordinator. We set three scenarios by varying the Poisson arrival rate to the device: 1, 2 and 10 packets per second. The packet size is set to 77 bytes (60 bytes of payload + 17 bytes of header) and the queue length on each node was set to four packets. Both router and coordinator send the corresponding acknowledgement once receiving a packet from device and router respectively.

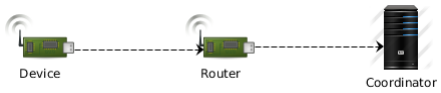

Figure 5: Scenario setup using TelosB nodes.

B. Resulting Markov chain

In order to obtain the Markov chain model we apply the methodology described in previous section to the concerned nodes in the network. We proceed then to obtain both device and router logs and by applying the procedure described before we get the corresponding Markov chain model for each node in the path. Due to the lack of space and the dimensions of the Markov chains, we give the reference [4] to each of them. However, in order to give a description, we present the Markov chain obtained for the device for the scenario $\lambda = 1$, which is shown in Figure 6. Here, the square blocks represent the sequence CCA1→SLEEP→CCA2→SLEEP→ $\cdots$ →CCA6→SLEEP →SENDING. A dotted line between two states S_1 and S_n in a block means that all transitions between intermediate states are equals to 1. Otherwise, if some of the transitions are lower than 1 (meaning that there is a transition to the CHANNEL BUSY state), then we show the state together with the transition probabilities to the next states. For the Markov chains shown in [4], two subscripts were added to each state in order to represent the number of collisions and number of times the channel was found busy. Then, each state in the Markov chain has the *STATE_k_l* format where k represents the current number of collisions and l the number of times the channel was found busy. For instance, being in state *CCA2_0_0*, if the channel was found busy then we increment the variable l and the Markov chain moves to the *CHANNEL BUSY_0_1* to retry the channel assessment.

C. End to End Delay Estimation

Now it is time to estimate the e2e delay in two hops from devices to the coordinator. As we explained before, based on the obtained Markov chain model and the transition probabilities between each state we are able to compute the *Probability Transition Matrix* P. From P and the estimation of the sejourn time distribution e_i on each state of the Markov chain we compute the *Adjacency Matrix* A as

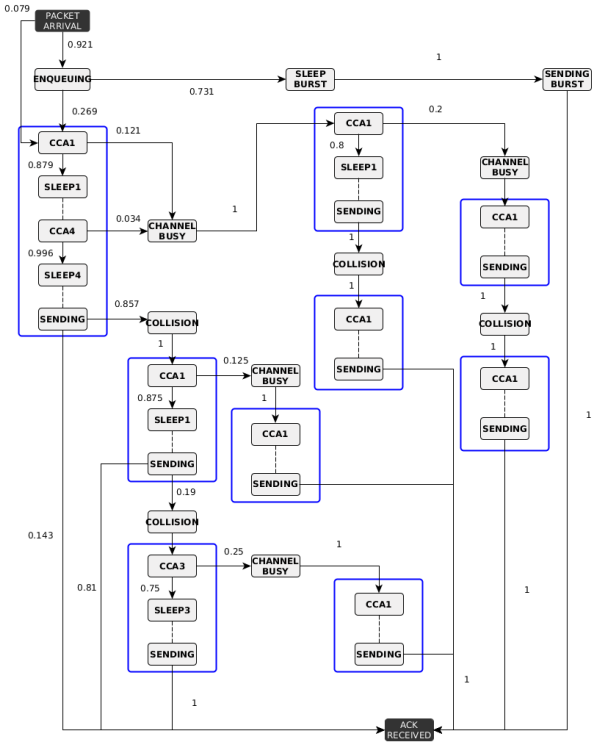

Figure 6: Obtained Markov chain: single device.

follows

$$A = \begin{pmatrix} 0 & p_{12}e_1 & 0 & \cdots & 0 \\ 0 & 0 & p_{23}e_2 & \cdots & 0 \\ \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad (1)$$

where e_i represents the Laplace transform of the sejourn time distribution on state i and p_{ij} the transition probability from state i to j taken from P. To find e_i, we compute the empirical average sejourn time γ_i obtained by analysing the traces of the protocol. Then, we make the assumption that the sejourn time on state i follows a negative exponential distribution of parameter γ_i. Therefore, the Laplace transform of the sejourn time distribution is

$$e_i = \frac{\gamma_i}{\gamma_i + s} \quad (2)$$

Once obtained A, we proceed to compute the vector $\vec{A}^r_{i,d}$ representing the delay distribution of all connected paths of length r, $r = \{1, 2, 3, ...\}$, from state i to destination d (ACK RECEIVED) which is computed as follows:

$$\vec{A}^r_{i,d} = A \cdot \vec{A}^{r-1}_{i,d} \quad (3)$$

where $\vec{A}^1_{i,d}$ is the vector containing the delay distribution in one-hop from state i to the destination. This vector is a non null vector since there is always one or more states directly connected to the destination state. Being s the source state

(PACKET ARRIVAL), when we look at $A^r_{s,d}$ we find the delay distribution in r hops from source to destination. Then, the whole delay distribution in the frequency domain can be computed as follows:

$$D_{f-dom} = \sum_{r=1} A^r_{s,d} \quad (4)$$

The first order derivative of 4 evaluated in $s = 0$ will give us the average delay $\bar{d}$ of the Markov chain. This procedure is applied for both router and device Markov chains obtaining $\bar{d}_{device}$ and $\bar{d}_{router}$. Then, the average e2e delay from device to coordinator is the sum of these values

$$\bar{D}_{e2e} = \bar{d}_{device} + \bar{d}_{router} \quad (5)$$

The whole e2e delay distribution in frequency domain is the product of 4 for both device and router

$$D_{e2e(f-dom)} = D_{f-dom}(Dev) \times D_{f-dom}(Rout) \quad (6)$$

Finally, the whole e2e delay distribution in time domain can be computed as follows

$$D_{e2e(t-dom)} = \textit{InverseLaplaceTransform}(D_{e2e(f-dom)}) \quad (7)$$

Table I shows the comparision between the measured e2e delay and the one computed by the Performance Computation Module for the three defined scenarios.

	Empirical Av. Delay (sec)			Computed Av. Delay (sec)		
λ	Device	Router	e2e	Device	Router	e2e
1 p/sec	0.11	0.125	0.236	0.12	0.13	0.25
2 p/sec	0.166	0.266	0.432	0.181	0.264	0.44
10 p/sec	0.235	0.38	0.615	0.241	0.383	0.62

Table I: Empirical and computed e2e average delay.

	% Packet dropped	
λ	Device	Router
1 p/sec	0	0
2 p/sec	5	13.7
10 p/sec	10.5	52.7

Table II: Buffer drop rate.

Results show that for the three scenarios the computed e2e delay is almost the same as the one measured from the traces of the protocol showing the suitability of the approach for estimating the e2e delay. Figure 7 shows the e2e delay distribution for the three scenarios. As we can see, as the arrival rate increases the e2e delay also increases. This is due to the fact that for low traffic rate ($\lambda = 1$ packets per second), the queuing delay is almost nonexistent. On the other hand, when considering an arrival rate of $\lambda = 10$ packets per second, the e2e delay increases due to the queuing delay since packets arrive at a high rate and must be stored in the buffer until the packet currently in process is sent to the next hop. To ilustrate this we can take a look at Table II where we show the percentage of packets dropped due to the fact that the buffer was full. As we can see, for $\lambda = 1$ there are no dropped packets meaning that each time

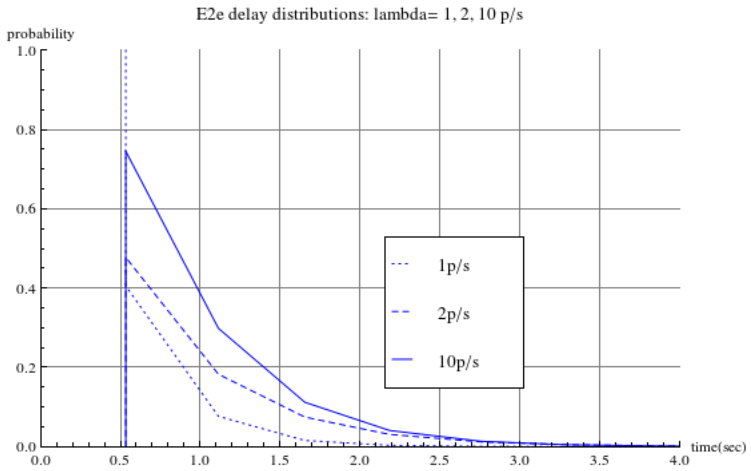

Figure 7: Probability density function of e2e delay for packet arrival rate $\lambda = 1, 2, 10$ p/s.

a packet arrived to the buffer, either it was empty or with enough capacity to store it. Then, the queuing delay in this scenario has a low impact in the whole e2e delay. On the other hand, when considering $\lambda = 10$ packets per second, we can see that 10% and 53% of packets were dropped at the device and router respectively. That means that, at some moments during the execution of the protocol, both buffers were full and thus the device queueing delay in this case is not negligible and will impact in the whole e2e delay, as seen in Figure 7.

VI. Conclusions & Future Work

In this paper we have presented an approach for extracting empirical Markov chain models from network protocol traces by means of process mining techniques. An empirical Markov chain model was obtained for ContikiMAC mac protocol. The contributions of our work can be enumerated as follows:

- We are able to obtain a Markov chain model from any protocol by analysing the protocol output (traces). Since this is an empirical approach we think that the obtained model is more realistic and accurate for representing the exact behavior of the protocol with regard to the theoretical models (as the one proposed in [11]) since we are considering all the events that may arise during the execution of the protocol. Then, this model encompass phenomenons such as capture effect and also the issues introduced by the underlying operating system which are not negligible and have an impact in the protocol's execution, as we have shown in [6]. This issues are not taken into account in theoretical models leading to wrong estimations of the performance parameters.
- The obtained Markov chain together with the sejourn time on each state allowed us to define the matrix A. By means of A we can compute the set of vectors $\vec{A}^r_{i,d}$ for

$r = \{2, 3, 4....\}$, and then we were able to estimate the e2e delay from source to destination. We have mentioned that, in general, existing mathematical models are conceived for a star topology where delay and other performance parameters are found for the case of 1-hop transmission and the extension to include multi-hop transmissions is not trivial. With our approach we overcome this problem by proposing a way for estimating the e2e delay for a multi-hop transmission scenario.

- Contrarily to the standard IEEE 802.15.4 mac protocol, and in spite of the spread of the ContikiMAC protocol, no theoretical model exists in the literature for modelling this protocol. Therefore, we contribute to the literature by presenting a comprehensive Markov model for ContikiMAC [4]. We were also able to obtain a more specific description of the protocol. From its specification we were able to obtain a conceptual model by identifying the states and transitions from one state to another to best understand how protocol works. This approach can also be extended for many other protocols which have not been yet deeply studied.

The approach is suitable for analysing high dynamic protocols. However, it is imperative to have the source code of the protocol to be able to instrument it. As a future work, we plan to extend the approach to some others not deeply explored protocols such as iQueue-MAC [19] and to focus our attention in the obtention of some other performance parameters from the Markov chain model. We also expect to set up a scenario by varying the distance between nodes in order to show how our approach takes into consideration the capture effect as well as to set up a scenario considering dynamic routing such as RPL.

REFERENCES

[1] IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - specific requirement Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, 2007.

[2] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE J.Sel. A. Commun.*, 18(3):535–547, September 2006.

[3] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM.

[4] Francois Despaux, Ye-Qiong Song, and Abdelkader Lahmadi. Markov Chain Results. http://qoswsanquasimod.gforge.inria.fr/.

[5] Francois Despaux, Ye-Qiong Song, and Abdelkader Lahmadi. Towards performance analysis of wireless sensor networks using process mining techniques. In *ISCC 2014*.

[6] Francois Despaux, Ye-Qiong Song, and Abdelkader Lahmadi. On the gap between mathematical modeling and measurement analysis for performance evaluation of the 802.15.4 mac protocol. In *12th International Workshop on Real Time Networks RTN13*, 2013.

[7] Messaoud Doudou, Djamel Djenouri, and Nadjib Badache. Survey on latency issues of asynchronous MAC protocols in delay-sensitive wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 15(2):528–550, 2013.

[8] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, December 2011.

[9] Wenbo He, Xue Liu, Long Zheng, and Hao Yang. Reliability calculus: A theoretical framework to analyze communication reliability. ICDCS '10, pages 159–168, Washington, DC, USA, 2010. IEEE Computer Society.

[10] Joachim Herbst and Dimitris Karagiannis. Workflow mining with inwolve. *Comput. Ind.*, 53(3):245–264, April 2004.

[11] Jelena Misic and Vojislav Misic. *Wireless Personal Area Networks: Performance, Interconnection, and Security with IEEE 802.15.4*. Wiley Publishing, 2008.

[12] P. Park, P. Di Marco, C. Fischione, and K. H. Johansson. Delay analysis of slotted IEEE 802.15. 4 with a finite retry limit and unsaturated traffic. In *IEEE Global Communications Conference*, page 18, 2009.

[13] W.M.P. van der Aalst, A.J.M.M. Weijter, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16:2004, 2003.

[14] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In Gianfranco Ciardo and Philippe Darondeau, editors, *ICATPN*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer, 2005.

[15] Gabriel M. Veiga and Diogo R. Ferreira. Understanding spaghetti models with sequence clustering for ProM. In *Business Process Management Workshops*, page 92103, 2010.

[16] Yipeng Wang, Zhibin Zhang, Danfeng Daphne Yao, Buyun Qu, and Li Guo. Inferring protocol state machine from network traces: A probabilistic approach. ACNS'11, pages 1–18, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] Yunbo Wang, Mehmet C. Vuran, and Steve Goddard. Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks. *IEEE/ACM*, 20(1):305–318, February 2012.

[18] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *INFOCOM 2002.*, volume 3, page 15671576, 2002.

[19] Shuguo Zhuo, Zhi Wang, Ye-Qiong Song, Zhibo Wang, and Luís Almeida. iqueue-mac: A traffic adaptive duty-cycled mac protocol with dynamic slot allocation. In *SECON*, pages 95–103, 2013.