A Case Study on Energy Overhead of Different IoT Network Stacks

Silvia Krug*†, Irida Shallari*, and Mattias O'Nils*
* *Department of Electronics Design*
Mid Sweden University
Sundsvall, Sweden
{silvia.krug, irida.shallari, mattias.onils}@miun.se
† *IMMS Institut für Mikroelektronik- und Mechatronik-Systeme gemeinnützige GmbH*
Ilmenau, Germany

***Abstract*—Due to the limited energy budget for sensor nodes in the Internet of Things (IoT), it is crucial to develop energy efficient communications amongst others. This need leads to the development of various energy-efficient protocols that consider different aspects of the energy status of a node. However, a single protocol covers only one part of the whole stack and savings on one level might not be as efficient for the overall system, if other levels are considered as well. In this paper, we analyze the energy required for an end device to maintain connectivity to the network as well as perform application specific tasks. By integrating the complete stack perspective, we build a more holistic view on the energy consumption and overhead for a wireless sensor node. For better understanding, we compare three different stack variants in a base scenario and add an extended study to evaluate the impact of retransmissions as a robustness mechanism. Our results show, that the overhead introduced by the complete stack has an significant impact on the nodes energy consumption especially if retransmissions are required.**

***Index Terms*—Internet of Things; Energy Consumption; Protocol Overhead Comparison; Experimental Observation; Analytical Evaluation;**

I. Introduction

Communication is often considered as one of the most costly features of wireless sensor nodes and several approaches exist to provide more energy-efficiency. Often, this however focuses on single protocols only. A coverage of the complete stack that defines the overall transceiver activity and thus the energy consumption for communications is missing yet required to obtain a more complete picture on communication introduced energy consumption.

Such a study requires the complete overhead in terms of packet encapsulation and additional packet exchanges to keep the network functional. This includes routing but also higher layer connection setup and maintenance. Besides that, depending on the stack configuration further utility functions such as time synchronization will require additional packet exchanges. As such, an isolated focus on only one protocol layer or only the application data is not sufficient to estimate the overall consumption.

Routing is one crucial aspect to the functionality of the network and plays a major role in the energy consumption besides the actual application communication. In [1], the authors study the effect of node failures on the energy consumption of the nodes as a result of route recalculations via simulations. They found that recalculations of the routing information triggered by node failures have a significant impact on all participating nodes in the network. However, route failures are not only triggered by node failures.

Unreliable links with a high Packet Error Rate (PER) can trigger similar behavior if nodes try to find better alternative routes [2], [3]. This functionality affects also nodes that do not participate in the routing process actively. A high PER also causes additional retransmissions for all other packet types as well. Therefore, the energy consumption is expected to further increase with higher PER.

When performing a long-term study of packet transmission of a leaf node in an outdoor wireless testbed in previous work, we found that the observed packet transmission vary quite dramatically over the observation period [4]. However, the energetic impact of the observed behavior remained open.

In this paper, we follow up on the study in [4] and analyze both the energy consumption of the different observations as well as the correlations between different stack components. Besides that, we analyze the overall energy consumption of different stack configurations that are used for the Internet of Things (IoT) and analyze how much energy is spend for utility tasks such as routing. The results show clearly, that the causes of energy consumption are more complex and cannot be related to a single protocol only.

The paper is organized as follows. In the next section, we briefly introduce the scenario used for our evaluation before discussing the covered IoT network stack variants in Section III. After these fundamentals, we perform an analysis of the energy consumption for the described stack variants in Section IV based on experimental and analytical studies. In Section V, we review the findings and relate them to previously reported results. Finally, we concluded the paper in Section VI where we also give our ideas for further studies.

II. Sample scenario

In [4], a simple Wireless Sensor Network (WSN) is studied, consisting of several sensor nodes that are collecting

978-1-5386-4980-0/19/$31.00 © 2019 IEEE

environmental parameters such as temperature, humidity, and pressure. Each sensor node transfers its data periodically to a predefined sink node or gateway. To reach the gateway, routing-enabled nodes can be used as relay. A sensor node can be either a routing-capable node that also acts as relay or a leaf node that does not participate in the routing process actively but requires a valid route in order to send its data to the gateway.

The periodic data transfer is implemented as a custom protocol that also sends occasional node health reports besides data. In addition to this, time synchronization between the nodes in the network is required. The Network Time Protocol (NTP) is used for this. Application data is only send when the node has both valid route and time references.

The nodes in the experiment are carefully deployed to locations, where they are within communication range of a suitable relay and able to reach the gateway. This is important later for the discussion on the impact of the observations.

III. Overview on Selected IoT Stack Variants

In order to achieve a periodic transfer of such environmental data, several IoT protocol options exist besides the custom implementation. Two prominent options are the Constraint Application Protocol (CoAP) and the Message Queuing Telemetry Transport (MQTT) protocol. For the later we consider the variant MQTT for Sensor Networks (MQTT-SN) that reduces some of the original overhead to make MQTT more feasible for constraint devices.

CoAP requires a polling like request-response communication scheme in which a sensor node acts as data provider or server, while the gateway or any other service act as client requesting data from it. MQTT-SN uses a publish-subscribe mechanism instead. In that case, the sensor node registers a topic with a broker and publishes data values to that topic whenever it has something to send. The broker then distributes the data to all registered subscribers of the topic. Subscribers are not necessarily nodes within the sensor network. In the MQTT-SN variant, a MQTT gateway can bridge the communication between the sensor node and the broker. Due to the different operation principles, we expect some differences in the number of packet exchanges required for each technology and thus also the required energy consumption.

To cover effects related to the networks utility functions such as routing and in this specific case time synchronization we add three stack implementations of three different sensor node operating systems to the comparison. The first is a TinyOS variant with HYDRO [5] as routing protocol. This setup was one of the first attempts to implement 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) in TinyOS. We use this old implementation because it was used for the real world experiments and TinyOS is still used for other applications as well.

The second variant is Contiki with RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) as routing protocol as defined in RFC [6]. This corresponds to the default 6LoWPAN stack based on 802.15.4. Similar to the TinyOS variant, NTP is needed to provide the required time synchronization.

As third variant, we chose OpenWSN [7]. It is also based on 802.15.4 and 6LoWPAN featuring RPL as routing protocol but employs the Time Synchronized Channel Hopping (TSCH) variant of the original MAC protocol. This ensured predictable timing for the transmissions and is thus an interesting candidate for timing sensitive applications in the IoT. Due to the inherent clock synchronization at MAC level, NTP is not required in addition. Besides that, OpenWSN reduced the packet overhead at each layer by removing as much redundant information as possible.

All three stacks are able to fulfill the requirements defined in the previous section but differ in the way they achieve that. In the following, we study how the different strategies effect the required energy for a transfer.

IV. Energy Consumption Analysis

To compare the energy amount required by each stack variant, we employ an analytical model[1] capturing the timing at medium access control (MAC) layer of 802.15.4 or 802.15.4e for TSCH, respectively. The model calculates the timing according to the standard definitions based on configurable header characteristics. Upper layer overhead is considered by adding the corresponding header information to the actual payload. Repeated encapsulation ensures that the overhead of the complete stack is considered.

Based on the stack description in the previous section, we calculate the ideal number of packets for data and utility functions covering the complete stack. To achieve this, we take the application data amount and calculate the required number of UDP and IP packets. Based on that as well as the MAC frame payload, we then calculate the number of 6LoWPAN fragments and MAC frames. These steps do consider differences in the Maximum Transmission Unit (MTU) as well as the required header information to estimate the actual amount of data the is transferred by the physical (PHY) layer. As a result, we obtain the number and size of the required transmissions at the PHY layer.

The transceiver activity (e.g. transmitting, receiving, idle, or sleep) is related to the number of transmissions required to transfer the data and the applied MAC scheme. For example, TSCH defines the activity sequence for each slot as depicted in Figure 1 with state durations and assigned power levels. Using this basic sequence and the payload size n of each PHY transmission, we are able to calculate the total duration spent in each state. The timing for 802.15.4 is calculated in a similar way, using ideal CSMA medium access to determine the timing information. Based on the timing, we calculate the energy consumption by multiplying the time spent in a specific state with the associated power P_X levels of real world hardware. This method gives a realistic yet somewhat optimistic energy value that can be seen as lower bound or ideal case.

[1]MatLab code available for download here [8]

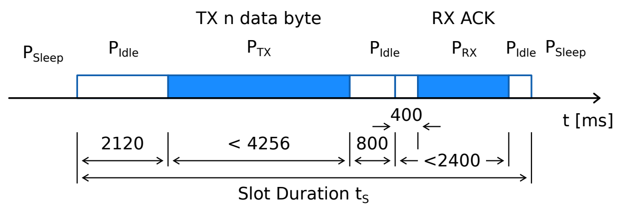

Fig. 1: TSCH slot timing to illustrate transceiver activity and its relation to data amount n to send

To apply the models to our study here, we have to identify the number of different transmissions and the size of each transmission. For the application data, we assume that the same amount as in the custom protocol is used. Therefore, each data packet carries 21 Byte of payload and status packets carry 17 Byte of payload. Due to the small data amounts, no fragmentation is required for any of the higher layer packets in this scenario. Table I lists the configured time intervals of the experimental setup with TinyOS as well as the ideal number of packets per day. For the other stacks, we configured similar intervals as far as possible to ensure comparability. In addition the table shows the average number of packets observed during the long term experiment presented in [4].

TABLE I: Example Interval Configuration

Function	**Interval**	**Packets per Day**		
		ideal	observed avg.	observed stdev
Routing Solicitation	1 d	1	45	105
Routing Updates	10 min	144	232	79
Time Synchronization	120 min	12	12	4
Application Data	15 min	96	98	13
Application Status	720 min	2	2	2

For all packet types, we calculate the required energy to transfer one packet based on power consumption values of the CC2538 transceiver [9]. This transceiver features the following power characteristics: $P_{TX} = 24\,\text{mA}$, $P_{RX} = 20\,\text{mA}$, $P_{Idle} = 13\,\text{mA}$, and $P_{Sleep} = 1.3\,\mu\text{A}$ Based on the resulting initial energy value and the configured timing or the actually observed number of packets, we then calculate the energy required to fulfill the application task for one day.

The number of observed packets shows a high variation (cf. Table I) between the best and worst case and an exact match of the ideal prediction does not occur. The distribution of the packets indicate an expected dependency on the network services, routing and time synchronization. In most cases, the data amount is however successfully transferred at the cost of multiple retransmissions. At this point our observation confirms the assumption in [1] that failures are a rather common observation in typical real-world WSN or IoT setups. This is even more crucial for the protocol design as the nodes in this study were positioned in a way that enables communication without frequent errors during the deployment phase. In any real world network, similar effects have to be expected.

In Figure 2, we show the resulting energy amount per day based on the observed packets during the long term experiment. The amounts are colored according to the respective functions. This figure shows a high variation in

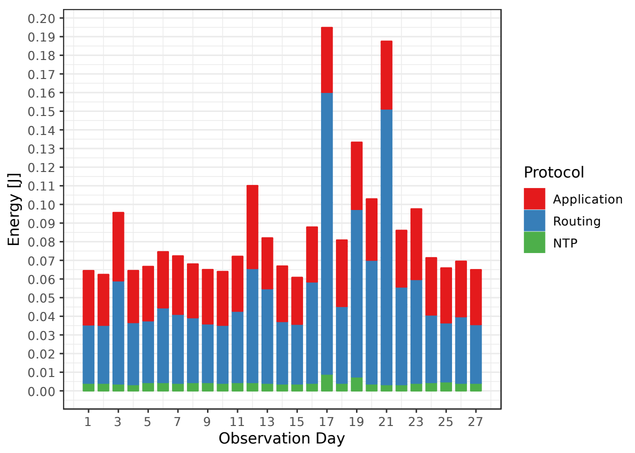

Fig. 2: Observed Total Energy Consumption over 27 Days

the total required energy. Between the best and worst case exists a difference of a factor three resulting directly from more packets that are required for a similar task. Overall, the variance is not as extreme but still significant for some days. When analyzing these observations, one has to consider that the nodes were carefully positioned in order to ensure good connectivity. The high variations in our observations indicate that dynamic aspects in the propagation conditions can have a significant impact.

To analyze the variation further, we show the energy consumption per utility function in Figure 3. We indicate the

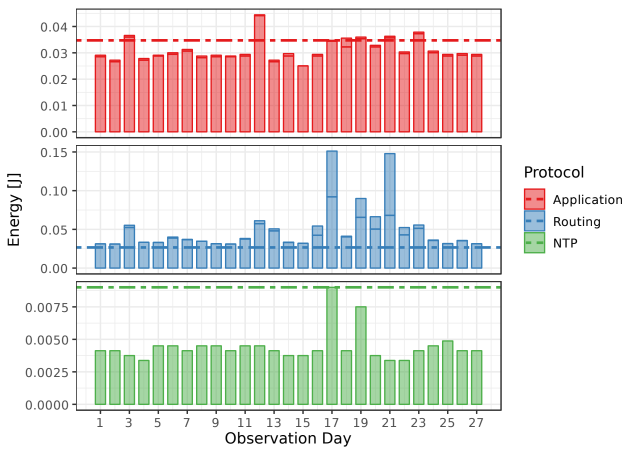

Fig. 3: Observed Energy Consumption per Utility Function over 27 Days

ideal energy amount as a dashed line, in addition. Here, it becomes obvious that the routing functions show the highest variation as expected based on the values in Table I due to the variation in the packet number. The high variation results from robustness mechanisms of the protocols employed. If packets get lost, they get retransmitted several times before giving up and eventually requesting an alternative route. However, routing information is also subject to potential packet loss.

The second observation is that both NTP and the application are not reaching the ideal level, indicating that in these cases the number of actually sent packets is lower than expected. This results directly from the condition, that both a valid route and time synchronization have to be available before sending application data. It also indicates, that there are not many retransmissions for these two functions.

Finally, we look at the packet types for the best (day 1) and worst case (day 17) observations. Figure 4, depicts the detailed number of observed packets per function for these two cases and the ideal case. This comparison confirms that the

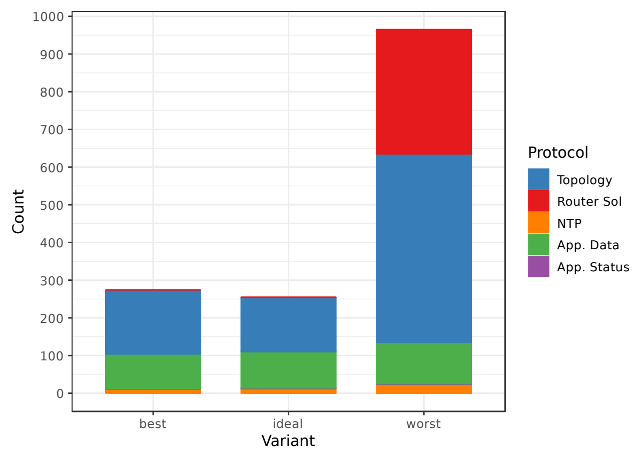

Fig. 4: Comparison if Best and Worst Case to Expected Ideal Case

number of actual packets does not match the ideal expectation even in the best case. In both cases, the number is higher indicating retransmissions for all packet types. In the best case scenario, the ratio of missing or additional packets per type is between 5 to 8 %. Such an even distribution is not present in the worst case scenario. There, the network overhead for routing and time synchronization is much higher indicating more complicated propagation conditions during that day. The propagation conditions therefore cause the high energy consumption.

The over proportional amount of routing traffic indicates another important fact related to a system perspective of the energy consumption. With the given stack configuration the application sends data only if it has a valid route and time synchronization. If many routing requests are required to fulfill that base assumption, no data is be send. Therefore, an evenly distributed PER on the lower layer cannot be translated directly to required retransmissions on higher layers. For the evaluation of energetic effects there, another approach is required that considers the dependability on the network utility functions.

To analyze if the other application protocols perform better, we compare the three stack variants with the discussed application protocols for the ideal case. Figure 5 shows the results. Here, different strategies in overhead reduction / utility

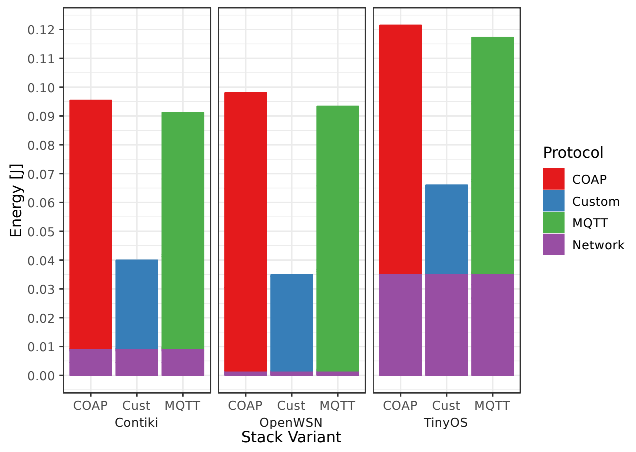

Fig. 5: Comparison of Energy Consumption of Different Stack Variants

functions become obvious. TinyOS shows the highest effort for utility functions and OpenWSN the least. Differences regarding the application data result from different encapsulation strategies and messaging schemes of the stack variants.

Comparing the consumption for application data, it is surprising that OpenWSN shows a high energy consumption for the application packets. This results in overall higher consumption than Contiki, even though OpenWSN otherwise shows a reduced utility overhead due to efficient header compression schemes and build in time synchronization. In this case, the reduction at network layer does not have the desired effect to reduce the nodes overall energy consumption. The custom protocol on-top of TinyOS shows a relatively good performance in comparison, even if the network load otherwise is the highest.

Regarding protocol robustness, one should also consider the impact of lost packets on the request-response or publish-subscribe schemes employed by the standardized protocols. In case of request-response scheme, the server will have to resend its request, if it does not receive a response. The reason for the missing packet could be that either the request or the response are lost. In the latter case, the energy consumption for one exchange is doubled at the node, while it is the same in the first case. Since publish-subscribe does not necessarily include such a feedback mechanism unless the sensor node subscribes to its own topic. In that case, any lost application protocol packet would result in at least doubled energy at the node. A careful design of the communication exchanges in a error-prone environment is essential to avoid unexpected energy expenses.

The more standardized protocols enabling easy web access to the sensor data come at a significant extra cost in terms of energy consumption. This is always required if protocol standardization is applied to ensure interoperability. An alternative could be to define and standardize novel protocols with a focus on cross layer energy efficiency.

V. Discussion

Many works try to estimate the lifetime of sensor nodes and the corresponding networks. However, the results are often based on periodic data sampling and static packet error rates as for example in [10]. Our results however show that real world scenarios are more complex and a realistic lifetime estimation has to consider the variation of the PER over time and dependencies between different network functions, as each of them can cause additional retransmissions and increased transceiver activity. Such models are currently however missing.

Our results show the energy consumption of a single leaf node only. Such a node does not actively take part in the route finding process except for the solicitation messages to find a suitable relay. For nodes that act as routers, the observed fluctuations will have an even bigger impact on the required packet exchanges and thus their energy consumption. This finding complements the results in [1], where only the route finding process itself is studied.

The dependability of applications on utility functions of the network (here routing and time synchronization) is well known. Studies on the energy efficient protocol operation, however do not focus on these effects. In [11], the authors study the energy consumption of TSCH isolated, for example. Failure handling leading to retransmissions or other expensive recalculations are typically not in focus of energy evaluations as well. While it is important to realize efficient protocol operation, the shown cross layer effects have to be considered for an overall energy efficient node design. This involves partially the selection of protocols and their configuration. However to achieve interoperability, the the development of further holistic approaches to conserve energy throughout the stack are required.

Especially if the application activity patterns vary as e.g. discussed in [12] or [13], the energy variations due to channel uncertainty can have a significant impact on the overall node lifetime. During both the energy evaluations and the design of the stack architecture, the communication subsystem has to be considered as a whole with all required functions and their dependencies. This leads to cross layer energy aware design.

VI. Conclusions

Based on a real world experiment with in general well connected sensor nodes, we observed an unexpectedly high variance in the packet retransmissions. In this paper, we analyzed the energetic impact of these variations and discussed the need to add cross layer effects to the energetic characterization of sensor nodes in order to enhance the lifetime predictions. As a result, it became clear that simple PER distributions are not able to cover dependencies between different layers that however have significant impact on the number of attempted retransmissions. Instead, we suggest a more holistic view on the communication subsystem including all layers of the protocol stack.

From such a holistic view point, it is possible to gain a better understanding of challenges arising from network uncertainty and interdependencies of the networking functions. With that, we are able to derive more realistic bounds on the required energy enabling on one hand better estimations of the energy requirements for communications and on the other hand provide insights for the development of novel protocol with this holistic bounds in mind to reduce the energy consumption.

As part of our future work, we plan to analyze the cross dependencies further by integrating them into our energy evaluation framework. This should enable us to study the impact of dynamic network effects on the energy consumption of a nodes communication subsystem as well as trade-offs to local processing in such cases.

References

[1] U. Kulau, S. Müller, S. Schildt, A. Martens, F. Büsching, and L. Wolf, "Energy Efficiency Impact of Transient Node Failures when using RPL," in *18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2017, pp. 1–6.

[2] K. D. Korte, A. Sehgal, and J. Schönwälder, "A Study of the RPL Repair Process Using ContikiRPL," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2012, pp. 50–61.

[3] K. Heurtefeux, H. Menouar, and N. AbuAli, "Experimental Evaluation of a Routing Protocol for WSNs: RPL robustness under study," in *9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2013, pp. 491–498.

[4] S. Krug, A. Schreiber, and M. Rink, "Poster: Beyond Static Sending Intervals for Sensor Node Energy Estimation," in *12th Workshop on Challenged Networks (CHANTS)*. ACM, 2017.

[5] A. Tavakoli, J. Hui, and D. Culler, "HYDRO: A Hybrid Routing Protocol for Lossy and Low Power Networks," March 2009. [Online]. Available: http://tools.ietf.org/pdf/draft-tavakoli-hydro-01.txt

[6] T. Winter, P. Thubert, A. B. J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Standards Track), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[7] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister, "OpenWSN: A Standards-Based Low-Power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[8] S. Krug. (2018) MatLab Functions for Cost Estimation of IoT Data Transfers. [Data set]. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-34861

[9] *CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications*, Texas Instruments Inc., Apr. 2015, rev D.

[10] É. Morin, M. Maman, R. Guizzetti, and A. Duda, "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things," *IEEE Access*, vol. 5, pp. 7097–7114, 2017.

[11] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. Pister, "A Realistic Energy Consumption Model for TSCH Networks," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 482–489, 2014.

[12] U. Kulau, J. van Balen, S. Schildt, F. Büsching, and L. Wolf, "Dynamic Sample Rate Adaptation for Long-Term IoT Sensing Applications," in *3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016.

[13] I. Shallari, S. Krug, and M. O'Nils, "Architectural evaluation of node: server partitioning for people counting," in *12th International Conference on Distributed Smart Cameras (ICDSC)*. ACM, 2018, p. 1.