The ContikiMAC Radio Duty Cycling Protocol

Adam Dunkels
adam@sics.se
SICS Technical Report T2011:13
ISSN 1100-3154

December 2011

Abstract

Low-power wireless devices must keep their radio transceivers off as much as possible to reach a low power consumption, but must wake up often enough to be able to receive communication from their neighbors. This report describes the ContikiMAC radio duty cycling mechanism, the default radio duty cycling mechanism in Contiki 2.5, which uses a power efficient wake-up mechanism with a set of timing constraints to allow device to keep their transceivers off. With ContikiMAC, nodes can participate in network communication yet keep their radios turned off for roughly 99% of the time. This report describes the ContikiMAC mechanism, measures the energy consumption of individual ContikiMAC operations, and evaluates the efficiency of the fast sleep and phase-lock optimizations.

1 Introduction

Low-power wireless devices must maintain strict power budgets to attain years of lifetime. Of all components on a low-power wireless device, the wireless transceiver often has the highest power consumption. The wireless transceiver consumes as much power when passively listening for transmissions from other devices as it does when actively transmitting, so the transceiver must be completely turned off to save power. Since the device is not able to receive any data when the transceiver is turned off, a duty cycling mechanism must be used to turn the transceiver on every now and then. Over the years, numerous duty cycling mechanisms have been proposed (see e.g. Dutta and Dunkels for a more thorough review of duty cycling mechanisms [9]). This document describes the ContikiMAC duty cycling mechanism, the default duty cycling mechanism in Contiki 2.5

ContikiMAC is designed to be simple to understand and implement. ContikiMAC uses only asynchronous mechanisms, no signaling messages, and no additional packet headers. ContikiMAC packets are ordinary link layer messages. ContikiMAC has a significantly more power-efficient wake-up mechanism that previous duty cycling mechanisms. This is achieved by precise timing through a set of timing constraints. In addition, ContikiMAC uses a fast sleep optimization, to allow receivers to quickly detect false-positive wake-ups, and a transmission phase-lock optimization, to allow run-time optimization of the energy-efficiency of transmissions.

The mechanisms in ContikiMAC are inspired by existing duty cycling protocols. The idea of periodic wake-ups has been used by many protocols, such as B-MAC [21], X-MAC [1], and BoX-MAC [18]. The phase-lock optimization has been previously suggested by WiseMAC [11] and has since been used by other protocols as well [14]. The use of multiple copies of the data packet as a wake-up strobe has previously been used by the TinyOS BoX-MAC protocol [18].

The rest of this report is structured as follows. Section 2 presents the ContikiMAC mechanism and its underlying principles. Section 3 describes the implementation of ContikiMAC in Contiki 2.5. Section 4 evaluates the energy efficiency of ContikiMAC, both with micro benchmarks and in a data collection network. Related work is reviewed in Section 5 and Section 6 concludes the report.

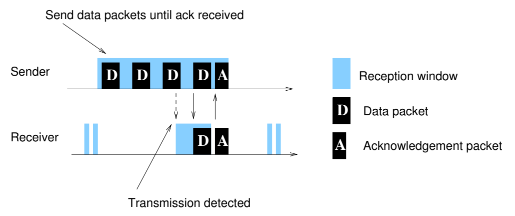


Figure 1: ContikiMAC: nodes sleep most of the time and periodically wake up to check for radio activity. If a packet transmission is detected, the receiver stays awake to receive the next packet and sends a link layer acknowledgment. To send a packet, the sender repeatedly sends the same packet until a link layer acknowledgment is received.

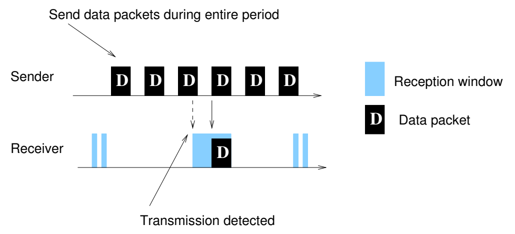


Figure 2: Broadcast transmissions are sent with repeated data packets for the full wake-up interval.

2 ContikiMAC

ContikiMAC is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. When the packet is successfully received, the receiver sends a link layer acknowledgment. To transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgment from the receiver. Packets that are sent a broadcasts do not result in link-layer acknowledgments. Instead, the sender repeatedly sends the packet during the full wake-up interval to ensure that all neighbors have received it. The principle of ContikiMAC is shown in Figure 1 and Figure 2.

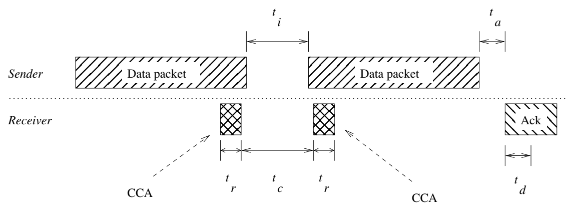


Figure 3: The ContikiMAC transmission and CCA timing.

2.1 ContikiMAC Timing

ContikiMAC has a power-efficient wake-up mechanism that relies on precise timing between transmissions. ContikiMAC wake-ups use an inexpensive Clear Channel Assessment (CCA) mechanism that uses the Received Signal Strentgh Indicator (RSSI) of the radio transceiver to give an indication of radio activity on the channel. If the RSSI is below a given threshold, the CCA returns positive, indicating that the channel is clear. If the RSSI is above the threshold, the CCA returns negative, indicating that the channel is in use.

The ContikiMAC timing is shown in Figure 3. The timing requirements from the figure are:

t_i: the interval between each packet transmission.

t_r: the time required for a stable RSSI, needed for a stable CCA indication.

t_c: the interval between each CCA.

t_a: the time between receiving a packet and sending the acknowledgment packet.

t_d: the time required for successfully detecting an acknowledgment from the receiver.

The timing must satisfy a number of constraints. First, t_i, the interval between each packet transmission, must be smaller than t_c, the interval between each CCA. This is to ensure that either the first or the second CCA is able to see the packet transmission. If t_c would be smaller than t_i, two CCAs would not be able to reliably detect a transmission.

The requirement on t_i and t_c also place a requirement on the shortest packet size that ContikiMAC can support,

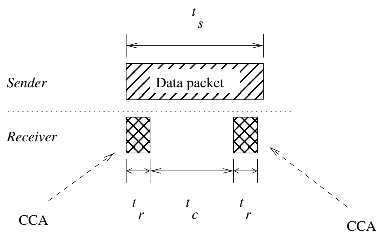


Figure 4: A packet transmission must be long enough so that it does not fall between to subsequent CCAs.

as shown in Figure 4. For ContikiMACs two CCAs to be able to detect the packet, a packet transmission cannot be so short that it falls between the CCAs. Specifically, t_s, the transmission time of the shortest packet, must be larger than $t_r + t_c + t_r$.

When a CCA has detected a packet transmission, ContikiMAC keeps the radio on to be able to receive the full packet. When a full packet has been received, a link-layer acknowledgment is transmitted. The time it takes for an acknowledgment packet to be transmitted, t_a, and the time it takes for an acknowledgment packet to be detected, t_d, establishes the lower bound for the check interval t_c.

We can now construct the full ContikiMAC timing constraints as

$$t_a + t_d < t_i < t_c < t_c + 2t_r < t_s. \qquad (1)$$

With an IEEE 802.15.4 link layer and a specific radio transceiver, some of the variables in Equation 1 are given as constants. First, t_a, the time between a packet reception and the acknowledgment transmission, is defined by the IEEE 802.15.4 specification as 12 symbols. In 802.15.4, one symbol is 4/250 milliseconds long, giving $t_a = 48/250 = 0.192$ milliseconds. Second, an IEEE 802.15.4 receiver can reliably detect the reception of the acknowledgment after the 4-byte long preamble and the 1-byte start of frame delimiter is transmitted, which takes 40/250 miliseconds. Thus, $t_d = 40/250$. Finally, t_r is given by the data sheet of the CC2420 radio transceiver as 0.192 milliseconds.

With the constants for substituted, Equation 1 becomes

$$0.352 < t_i < t_c < t_c + 0.384 < t_s. \qquad (2)$$

The remaining variables, t_i, t_c, and t_s can now be chosen. Equation 2 gives a lower bound on $t_s > 0.736$ milliseconds, which sets a limit on the smallest packet size we can handle. With a bitrate of 250 kilobits per second, this means that packets must be at least 23 bytes long, including preamble, start of frame delimiter, and length field, which leaves 16 bytes of packet data.

To ensure that all packets are larger than the smallest packet size, packets may be padded with additional framing to ensure a minimum packet size. Alternatively, if the network layer is able to ensure that packets never go below a given size, no framing is needed. For example, in the case of an IPv6 network layer, packets with full IPv6 headers will always be longer than the smallest ContikiMAC packet size on a IEEE 802.15.4 link layer. With 6lowpan IPv6 header compression, packets may become smaller, but ensuring a smallest packet size is simple: do not compress the header of IPv6 packets that are smaller than a given threshold.

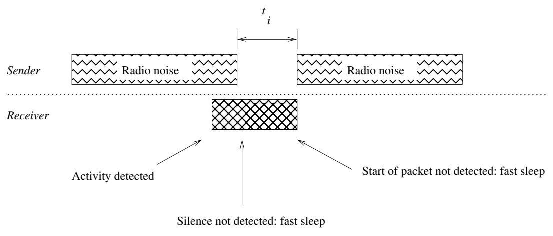
The ContikiMAC implementation in Contiki 2.5 uses the following configuration:

- $t_i = 0.4$ milliseconds,
- $t_c = 0.5$ milliseconds, and
- $t_s = 0.884$ milliseconds.

2.2 Packet Detection and Fast Sleep

The ContikiMAC CCAs do not reliably detect packet transmissions: they only detect that the radio signal strength is above a certain threshold. The detection of a radio signal may either mean that a neighbor is transmitting a packet to the receiver, that a neighbor is transmitting to another receiver, or that some other device is radiating radio energy that is being detected by the CCA mechanism. ContikiMAC must be able to discern between these events and react properly.

If a neighbor is transmitting a packet to the receiver, the receiver should stay awake to receive the full packet and transmit a link layer acknowledgment. Other nodes, which detect the packet, could quickly go to sleep again. Potential receivers cannot go to sleep to quickly, however, as they must be able to receive the full packet. The naive way to determine how long to be awake when a CCA has detected radio activity is to stay awake for $t_l + t_i + t_l$,



t_i
Sender
Radio noise
Radio noise
Receiver
Activity detected
Start of packet not detected: fast sleep
Silence not detected: fast sleep

Figure 5: The ContikiMAC fast sleep optimization: if a silence period is not detected before t_l, the receiver goes back to sleep. If the silence period is longer than t_i, the receiver goes back to sleep. If no packet is received after the silence period, even if radio activity is detected, the receiver goes back to sleep.

where t_l is the transmission time of the longest possible packet. This ensures, if the CCA woke up during the start of the packet, that the full packet will be received by the receiver.

The fast sleep optimization lets potential receivers go to sleep earlier if the CCA woke up due to spurious radio noise. The fast sleep optimization leverages knowledge of the specific pattern of ContikiMAC transmissions as follows. First, if the CCA detects radio activity, but the radio activity has a duration that is longer than the maximum packet length t_l, the CCA has detected noise and can go back to sleep. I.e., if the activity period is not followed by a silence period. Second, if the radio activity is followed by a silence period that is longer than the interval between two successive transmissions, t_i, the receiver can go back to sleep. Third, if the activity period is followed by a silence period of the correct length, followed by activity, but no start of packet could be detected, the receiver can go back to sleep. The process is illustrated in Figure 5.

2.3 Transmission Phase-Lock

If we assume that each receiver has a periodic and stable wake-up interval, the sender can use knowledge of the wake-up phase of the receiver to optimize its transmission. A sender can learn of a receiver's wake-up phase by making note of the time at which it saw a link layer acknowledgment from the receiver. Since the receiver must have been awake to be able to receive the packet,

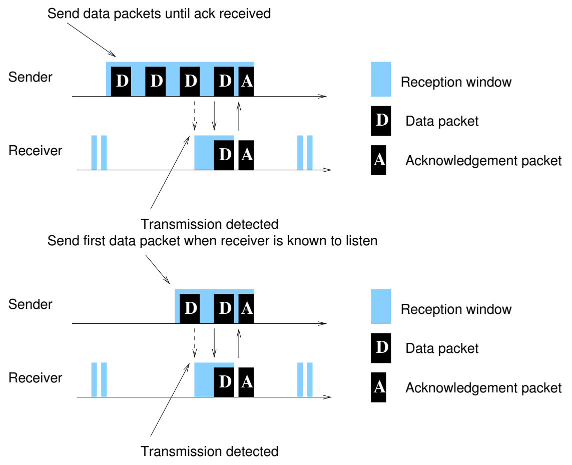


Figure 6: Transmission phase-lock: after a successful transmission, the sender has learned the wake-up phase of the receiver and subsequently needs to send fewer transmissions.

the sender can assume that the reception of a link layer acknowledgment means that the sender has successfully transmitted a packet within the receiver's wake-up window and thus that the sender has found the receiver's wake-up phase. After a sender has learned the phase of a receiver, the sender can commence its successive transmissions to this receiver just before the receiver is expected to be awake. The process is illustrated in Figure 6.

3 Implementation

The ContikiMAC implementation in Contiki 2.5 uses the Contiki real-time timers (rtimer) to schedule its periodic wake-ups to ensure a stable behavior even if many underlying processes are running. The real-time timers preempt any Contiki process at the exact time at which they are scheduled. The ContikiMAC wake-up mechanism runs as a protothread [6] that is scheduled by a periodic real-time timer. This protothread performs the periodic wake-ups and implements the fast sleep optimization.

Transmissions are driven by an ordinary Contiki process. If a wake-up is scheduled to occur when the radio is busy during a transmission, the wake-up timer schedules a new wake-up after another wake-up interval without performing the wake-up.

The phase-lock mechanism is implemented as a separate module from ContikiMAC, to allow it to be used by other duty cycle mechanisms, such as the Contiki X-MAC [1] implementation. The phase-lock mechanism maintains a list of neighbors and their wake-up phases. The ContikiMAC transmission logic records the time of each packet transmission. When a link layer acknowledgment is received, it notifies the phase-lock module with the transmission time of the last packet. This time is used as an approximation of the wake-up phase of the receiver.

Before commencing a transmission, the ContikiMAC transmission logic calls the phase-lock module to check if it has a recorded wake-up phase of the intended receiver. If so, the phase-lock code queues the packet to be transmitted and sets a callback timer (ctimer) at the time of the expected wake-up of the receiver. ContikiMAC will then resume the transmission when the callback occurs. The transmission will then be significantly shorter than a normal transmission, because it occurs just before the neighbor is expected to be awake. Reducing the length of the transmission thus reduces radio congestion.

If a neighbor whose phase is known has rebooted, or if its clock has drifted far enough away from its previous wake-up phase, transmissions to the neighbor will fail. To protect from this, ContikiMAC maintains a count of failed transmissions for each known neighbor. After a fixed number of failed transmission (16 in Contiki 2.5), the neighbor is evicted from the list of known neighbors. Likewise, if no link layer acknowledgment is received within a fixed time (30 seconds in Contiki 2.5), regardless of the number of transmissions, the neighbor is evicted.

4 Evaluation

This report evaluates two aspects of ContikiMAC: the energy consumption of the individual ContikiMAC operations and the power efficiency of ContikiMAC in a data collection sensor network. In addition to the results presented here, we have used ContikiMAC in much recent work. For more ContikiMAC performance results, the reader is referred to Dunkels et al. [3]; Duquennoy, Österlind, Dunkels [7]; Duquennoy et al. [8]; Kovatch, Duquennoy, Dunkels [16]; Lundén and Dunkels [17]; and Tsiftes and Dunkels [24].

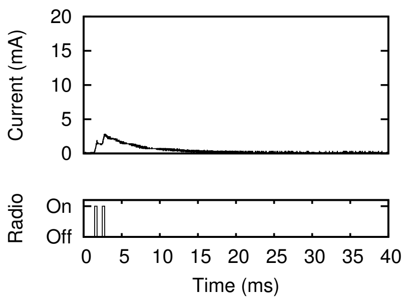


Figure 7: A ContikiMAC wake-up with no signal detected. The two CCAs are seen in the lower graph.

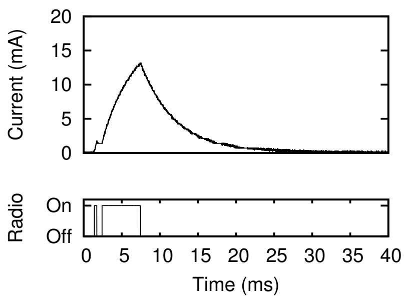


Figure 8: A ContikiMAC wake-up with radio activity detected and where the fast sleep optimization quickly turns the radio off.

4.1 Micro Benchmarks

We measure the energy consumption of the individual ContikiMAC operations by measuring the current draw of a Tmote Sky mote [22] running ContikiMAC. We measure the current draw with an oscilloscope by measuring the voltage over an 100 Ω resistor connected in series with the Tmote Sky power source. We also instrument ContikiMAC to register the state of the radio on one of the Tmote Sky I/O pins, with a high current indicating that the radio is on and a low current indicating that the radio is off, and measure the state of the pin with the same oscilloscope. All measurements use ContikiMAC with a wake-up fre-

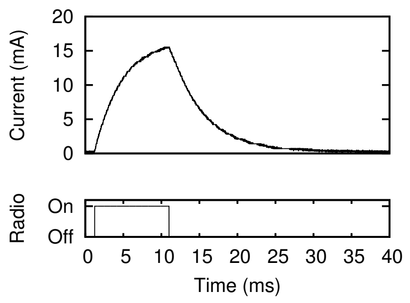


Figure 9: Broadcast reception: wake-up, packet detected, broadcast packet received.

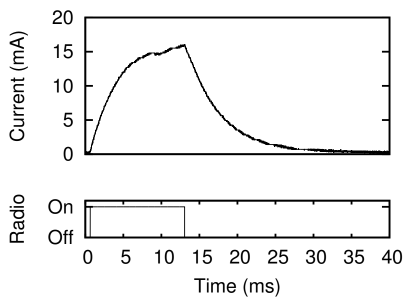


Figure 10: Unicast reception: wake-up, packet detected, unicast packet received

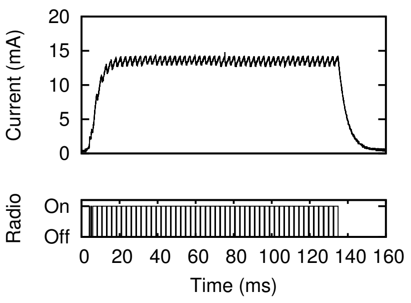


Figure 11: Broadcast transmission.

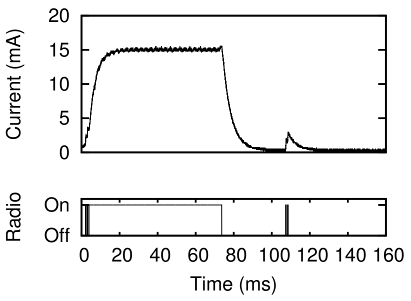


Figure 12: Non-synchronized unicast transmission (with subsequent wake-up at 110 ms

quency of 8 Hz, which results in a wake-up interval of 125 ms.

Figure 7 shows the current draw of a ContikiMAC wake-up that did not result in any packet reception. In the lower graph, we see that the radio is turned on twice, to perform the two CCAs of the ContikiMAC wake-up. Figure 8 shows a ContikiMAC wake-up where the second CCA detected spurious radio activity. The radio is then kept on for a while longer, until the fast sleep optimization turns off the radio.

Figure 9 and Figure 10 shows a broadcast reception and a unicast reception, respectively. In both cases, the radio was turned on as part of the ContikiMAC wake-up mechanism and the first CCA detected radio activity. The radio was then kept on until the packet was received. We see that the radio is turned on longer in the unicast reception case. This is because of the acknowledgment transmission that is done as part of the unicast packet reception.

The current draw of transmissions are shown in Figure 11 through Figure 13. Figure 11 shows the current draw of a broadcast transmission. A broadcast transmission must wake up and deliver its packet to all neighbors. It therefore runs for a full wake-up interval. Since a broadcast transmission does not expect any link layer acknowledgment, the transmitter can turn off its radio between each packet transmission to save power, which can be seen in the figure. Figure 12 show the current draw of a unicast transmission to a previously unknown neighbor. In this case, the neighbor's wake-up occurred after roughly 60 ms, which caused the transmitter to repeatedly send its packet some 70 ms. At the start of the transmission, the initial clear channel assessment can also be seen. Subsequent transmissions to the same receiver can

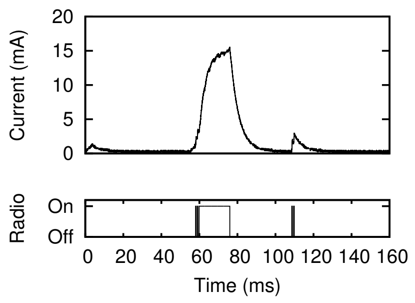


Figure 13: Synchronized unicast transmission (with subsequent wake-up at 110 ms)

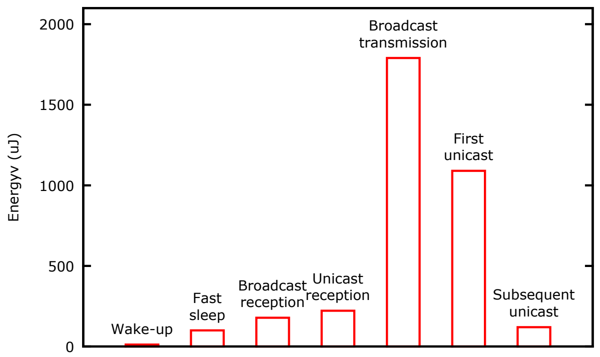


Figure 14: The energy consumption of the individual ContikiMAC operations.

now be optimized to start at the expected wake-up time of the neighbor, as seen in Figure 13, which shows how the number of transmissions are reduced because of the phase-lock optimization.

By computing the areas under the graphs in Figure 7 through Figure 13, we can compute the energy consumption of each operation. The result is shown in Figure 14. We see that the cost of a broadcast transmission is many orders of magnitude higher than the cost of the wake-up. This is good: the wake-up is the most common operation in ContikiMAC—executed many times per second—and therefore should be significantly less expensive than the other operations.

Armed with the information in Figure 14, we can now compare the cost of the ContikiMAC wake-up operation with the wake-up operation of other duty cycling mechanisms. Table 1 shows the cost of a wake-up in ContikiMAC, in the Contiki X-MAC implementation [1], and the duty cycling mechanism by Hui and Culler [14].

Table 1: Comparison of the energy consumption of the wake-up operation.

Protocol	**Energy (uJ)**
X-MAC [1]	132
Hui and Culler [14]	54
ContikiMAC	12

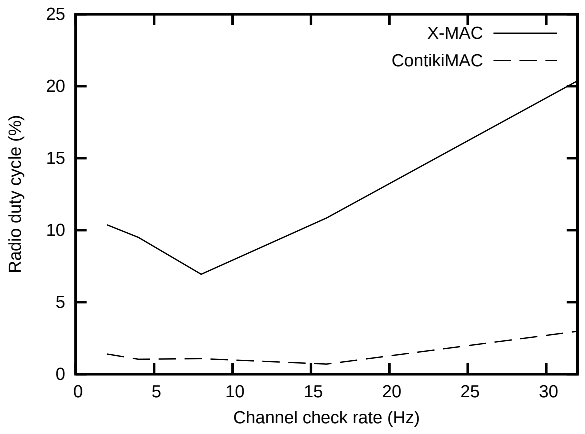


Figure 15: The radio duty cycle in a data collection network with path loss, with X-MAC and ContikiMAC, as a function of the wake-up frequency (in the graph called channel check rate).

4.2 Network Power Consumption

To evaluate the network power consumption of ContikiMAC and the efficiency of its optimizations, we run a set of simulations in the Contiki simulation environment. The Contiki simulation environment consists of the Cooja network simulator and the MSPsim device emulator. MSPsim provides a cycle-accurate Tmote Sky emulation, with a symbol-accurate emulation of the CC2420

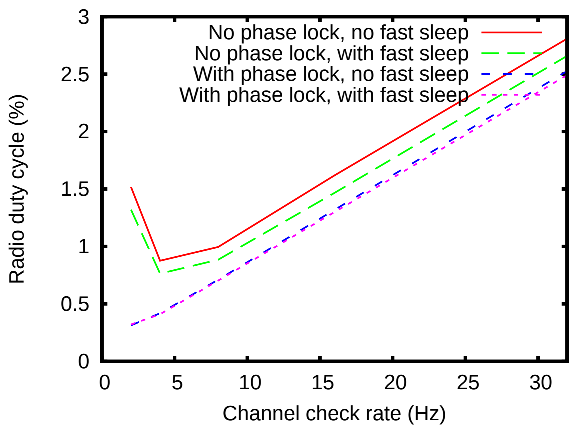


Figure 16: The network radio duty cycle with ContikiMAC, averaged for all nodes a the network without path loss.

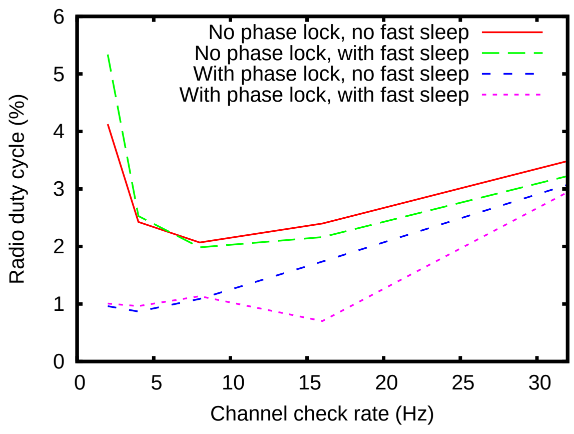


Figure 17: The network radio duty cycle with ContikiMAC, averaged for all nodes in a network with path loss.

radio transceiver. It enables the study of the behavior of ContikiMAC in a timing-accurate and controlled environment.

We run a set of simulations with a 20-node simulation topology. All nodes run Contiki and the Contiki Collect protocol. The Contiki Collect protocol, which is part of the Contiki Rime stack [4], is an address-free data collection protocol that builds a tree rooted in one or more sinks, towards which packets are routed. The performance of Contiki Collect has been experimentally shown [15] to be similar to other data collection protocols, such as the TinyOS Collection Tree Protocol [12]. The nodes send a data packet towards the sink once every 120 seconds. Every transmission is sent with 31 hop-by-hop retransmissions. Each node sends 100 packets towards the sink. The simulation is run until all data packets have been received by the sink. In all simulations, Contiki Collect was able to successfully deliver all packets to the sink.

The purpose of the simulations is both to measure the typical radio duty cycle that can be achieved with ContikiMAC and to measure the effect of the fast sleep and phase-lock optimizations. We vary the wake-up frequency and the simulated loss levels. We run one set of simulations with no path loss: packets are not lost due to radio fading, but only due to collisions with other packets. The second set of simulations use a path loss model where the probability of a packet loss is proportional to the square of the distance between the sender and the receiver.

We measure the radio duty cycle with Contiki's Powertrace tool [2]. We use the radio duty cycle as a proxy for the power consumption of the network, as the radio transceiver has a linear power draw that depends on its on-time [5, 22].

We first compare the performance of ContikiMAC with that of X-MAC [1] in a network with path loss. We expect the power consumption of X-MAC to be significantly higher than that of ContikiMAC due to the more costly wake-up mechanism in X-MAC. Figure 15 shows the result: the power consumption of ContikiMAC is significantly lower for all wake-up frequencies in the experiment.

Next, we measure the efficiency of the individual ContikiMAC optimizations. We run the simulations with the ContikiMAC optimizations switched on and off. The results are shown in Figure 16 and Figure 17. Figure 16 shows the radio duty cycle when there is no path loss. We see that the radio duty cycle increases with the wake-up frequency: with more wake-ups, the total power consumption of the network increases. We also see that the fast sleep and phase-lock optimizations significantly reduce power consumption.

Figure 17 shows the results in the network with path loss. We see that the phase-lock and fast sleep optimizations are more efficient in the face of loss. This is because of a phase-locked transmission being shorter than non-phase-locked transmissions, leading both to less energy being spent on transmissions and to less radio congestion.

5 Related Work

The high power consumption of the radio transceiver is a well-known issue that has spurred much work on radio duty cycling. Radio duty cycling mechanisms can be divided into two main categories: synchronous and asynchronous. Synchronous mechanisms depend on neighboring nodes being synchronized with each other whereas asynchronous mechanisms do not depend on any a priori synchronization. Asynchronous mechanisms can further be subdivided into sender-initiated and receiver-initiated mechanisms. In sender-initiated mechanisms, the sender initiates communication between a sender and a receiver, whereas in receiver-initiated mechanisms, the receiver initiates communication. ContikiMAC is a sender-initiated asynchronous mechanism. The literature provides many examples of mechanisms from these categories as well as hybrid mechanisms that combine features from more than one of the categories.

Examples of synchronous protocols as the early work on S-MAC [26] and T-MAC [25] as well as the more recent TSMP [20]. In S-MAC and T-MAC, nodes periodically wake up in a scheduled manner such that communication can take place when adjacent nodes are awake. The wake-up schedules are arranged to avoid overlapping and medium contention. In TSMP, time is divided into 10 ms long slots. Nodes are given a schedule of when to be awake. Nodes wake up briefly at the start of each slot to listen for any activity on the radio medium. If activity is detected, the radio is kept on longer to be able to receive incoming packets.

Asynchronous protocols have the advantage of not requiring synchronization and the research community has explored many different variants of asynchronous protocols. Early work on sensor network architectures found a simple asynchronous mechanism called low-power listening [13] in which nodes periodically wake up to sample the medium for a wake-up tone. If a wake-up tone is found, the radio is kept on to receive a transmission. To send a packet, the sender first transmits a wake-up tone to wake its neighborhood up. Later variants of this scheme used scheduled transmissions to avoid sending a too long wake-up tone [11]. The low-power listening scheme was moved to a packetizing radio with X-MAC [1]. In X-MAC, the wake-up tone is composed of a series of strobe packets. When the receiver wakes up, it sends a link-layer acknowledgment to the sender to indicate that it is awake and ready to receive the data packet. Others have subsequently improved upon these protocols [21, 14]. ContikiMAC is highly similar to existing low-power listening protocols but has a significantly more effective wake-up mechanism due to the precise timing between each data packet transmission.

Receiver-initiated protocols have a shorter history than sender-initiated protocols. Low-Power Probing (LPP) is perhaps the first example of a receiver-initiated protocol for low-power wireless [19]. In LPP, when a node intends to send a packet, it turns on its radio to listen for probes from potential receivers. When the sender hears a probe from the intended receiver of the packet, it transmits its packet. RI-MAC [23] is a similar but more effective mechanism of the same type. A-MAC [10] makes the wake-up signal more efficient, making both idle power consumption lower and transmissions more effective

6 Conclusions

This report presents the ContikiMAC radio duty cycling mechanism for low-power wireless networks, the default radio duty cycling mechanism in Contiki 2.5. ContikiMAC is designed to be simple to understand and implement, uses only asynchronous and implicit synchronization, and requires no signaling messages or additional headers. ContikiMAC uses a simple but elaborate timing scheme to allow its wake-up mechanism to be highly power efficient, a phase-lock mechanism to make transmissions efficient, and a fast sleep optimization to allow receivers to quickly go to sleep when faced with spurious radio interference. Measurements show that the wake-up mechanism is significantly lower than for existing duty cycling mechanisms and that the phase-lock and fast sleep mechanisms reduce the network power consumption between 10% and 80%, depending on the wake-up frequency of the devices in the network.

Acknowledgments

The author wishes to thank (in alphabetical order) Mariano Alvira, Simon Duquennoy, Joakim Eriksson, Niclas Finne, Marcus Lundén, David Kopf, Fredrik Österlind, and Nicolas Tsiftes, for inspiration, help, and expertise. This work was funded by the SSF as part of the Promos project and the EU Commission as part of the NOBEL and Calipso projects.

References

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, 2006.

[2] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. Technical Report T2011:05, Swedish Institute of Computer Science, March 2011.

[3] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2011.

[4] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, November 2007.

[5] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Cork, Ireland, June 2007.

[6] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Boulder, Colorado, USA, November 2006.

[7] S. Duquennoy, F. Österlind, and A. Dunkels. Lossy Links, Low Power, High Throughput. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011.

[8] S. Duquennoy, N. Wirström, N. Tsiftes, and A. Dunkels. Leveraging IP for Sensor Network Deployment. In *Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, Chicago, IL, USA, April 2011.

[9] P. Dutta and A. Dunkels. Operating systems and network protocols for wireless sensor networks. *Philosophical Transactions of the Royal Society A*, 370(1958):68–84, January 2012.

[10] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, November 2010.

[11] A. El-Hoiydi, J.-D. Decotignie, C. C. Enz, and E. Le Roux. wiseMAC, an ultra low power MAC protocol for the wiseNET wireless sensor network. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2003.

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, CA, USA, 2009.

[13] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.

[14] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, North Carolina, USA, November 2008.

[15] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, M. Durvy, J. Vasseur, A. Terzis, A. Dunkels, and D. Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011.

[16] M. Kovatsch, S. Duquennoy, and A. Dunkels. A Low-Power CoAP for Contiki. In *Proceedings of the Workshop on Internet of Things Technology and Architectures (IEEE IoTech 2011)*, Valencia, Spain, October 2011.

[17] M. Lundén and A. Dunkels. The Politecast Communication Primitive for Low-power Wireless. *ACM SIGCOMM Computer Communication Review*, 41:31–37, April 2011.

[18] D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking. Technical Report SING-08-00, Stanford University, 2008.

[19] R. Musaloiu-E., C-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, St. Louis, Missouri, USA, 2008.

[20] K. Pister and L. Doherty. TSMP: Time Synchronized Mesh Protocol. In *Proceedings of the IASTED International Symposium on Distributed Sensor Networks (DSN08)*, Orlando, Florida, USA, November 2008.

[21] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Baltimore, MD, USA, 2004.

[22] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Los Angeles, CA, USA, April 2005.

[23] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[24] N. Tsiftes and A. Dunkels. A database in every sensor. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, November 2011.

[25] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Los Angeles, California, USA, November 2003.

[26] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, USA, June 2002.