Contiki-based IEEE 802.15.4 Node's Throughput and Wireless Channel Utilization Analysis

Muhammad Omer Farooq
Institute of Telematics
University of Luebeck, Germany
Email: farooq@itm.uni-luebeck.de

Thomas Kunz
Department of Systems and Computer Engineering
Carleton University, Canada
Email: tkunz@sce.carleton.ca

***Abstract*—In this paper, we analyse the impact of the Contiki Operating System (OS), and its Carrier Sense Multiple Access and Collision Avoidance (CSMA-CA) implementation on an IEEE 802.15.4 node's throughput and wireless channel utilization. The analysis is based on Contiki's Rime networking protocol stack, and its target is to determine an upper bound for the stated metrics. We explain that in Contiki with CSMA-CA as a MAC layer protocol, a node's throughput is limited to 8.1 kbps, at maximum, even without power saving features. In order to maximize a node's transmission capability, we modified Contiki's CSMA-CA implementation. A number of simulations are performed, and it is observed that with our modifications node throughput reaches 45 kbps, at maximum. Simulation results for estimating the channel capacity with our modified CSMA-CA MAC layer protocol show that the average per-node delay is low when the offered data load remains below 100 kbps. For an offered load of 100 kbps, the channel drops almost 20% of packets. Going beyond 100 kbps results in large latencies and significant packet loss. Results presented in this paper can serve as basis for the available bandwidth estimation in Wireless Sensor Networks (WSNs), QoS-based routing, and design of congestion control algorithm.**

***Index Terms*—IEEE 802.15.4 throughput; Wireless Sensor Networks; Channel Utilization.**

I. Introduction

Recent work on WSNs has led to their application in many real-time environments [1]: visual surveillance, assisted living, smart homes, and intelligent transportation to name but a few. The advent of wireless multimedia sensing nodes [2] enabled multimedia communication using the IEEE 802.15.4 standard, hence the formation of Wireless Multimedia Sensor Networks (WMSNs). Real-time applications require Quality of Service (QoS) provisioning in terms of delay, bandwidth, reliability, and network availability. In this work we focus on delay and bandwidth as QoS metrics. A good QoS provisioning framework needs to know: the overheads associated with the operating system and networking protocol stack on a node's ability to send and receive data. Moreover, it should have an idea about the maximum channel utilization and a node's transmission capability. It is important, primarily due to the following reasons: If a QoS framework is aware of maximum channel utilization, using a certain operating system and networking protocol stack, it can limit the amount of data into the network. This keeps delay within manageable limits, hence bounded delay. It can avoid congestion, and it can invoke congestion avoidance and control mechanisms at the appropriate times.

The contributions of this paper are fourfold. First, we highlight transmission rate limiting features of Contiki and its CSMA-CA implementation. Second, our modifications to the CSMA-CA MAC layer increases node throughput to 45 kbps. Third, we present IEEE 802.15.4 maximum channel utilization with Contiki. Forth, we show how delay increases as the offered data load in Contiki-based IEEE 802.15.4 WSN increases.

The remainder of this paper is organized as follows. Section II presents Contiki 2.5 event handling and CSMA-CA MAC layer behaviour. Section III presents our modifications to Contiki's CSMA-CA implementation. Simulation results are presented in Section IV, and this research is concluded in Section V.

II. Contiki 2.5 Event Handling and CSMA-CA MAC Layer Behaviour

Contiki is build around an event-driven kernel. The Contiki kernel comprises of an event scheduler that dispatches events to the running processes. In Contiki, processes run to completion, however, event handlers can use internal mechanisms for preemption. Contiki maintains a queue of pending events, and events are dispatched to target processes in a First In First Out (FIFO) manner. Interrupts can preempt an event handler, but to avoid synchronization issues, interrupts cannot post an event. To transmit a data packet, Contiki uses a callback timer. The callback timer takes an expiry time and a pointer to a function that acts as an event handler as arguments. When the timer expires, an event is stored in the event queue and the event handler is called eventually. If there are multiple events pending in an event queue and events are fired in a FIFO manner, it is possible that the event handler for a callback timer does not execute right away. This phenomenon limits the transmission capability of a node.

Whenever the MAC layer has a packet to transmit, it delays carrier sensing to $1/8^{th}$ of a second, using the null radio duty cycling algorithm. Afterwards, it performs carrier sensing and if no carrier is detected, the packet is transmitted. If reliability mode is enabled, the MAC layer waits for a predefined interval of time to detect an ACK, in Contiki 2.5 this interval is 6 real-time ticks for Tmote sky motes. When an ACK is detected, the system waits for another 10 real-time ticks. If no ACK is detected in the stated time interval, the system backs-off for a random amount of time. The random back-off interval depends on the Channel Check Interval (CCI) used by the radio duty cycling algorithm, which is 1/8th of a second for null radio duty cycling.

After every successful packet transmission, the CSMA-CA MAC layer with null duty cycling waits for 1/8th of a second to transmit the next packet in the MAC layer queue. Therefore, CSMA-CA with null radio duty cycling can only transfer Γ bps, where $\Gamma = \sum_{i=1}^{8} \gamma_i \times 8$, and γ_i is the total size of the i^{th} data packet in the number of bits. A transmission rate of Γ bps is only possible if MAC layer ACKs are disabled, otherwise node throughput will further degrade. The maximum frame size supported by IEEE 802.15.4 is 127 bytes, hence the maximum achievable node's throughput is 8.1 kbps.

III. Modifications To Contiki's CSMA-CA Implementation

To enhance network throughput, we modified the time interval for which a CSMA-CA MAC layer waits to perform the clear channel assessment. The CSMA-CA MAC layer uses Contiki's callback timer (ctimer) mechanism to invoke the function responsible for performing the CSMA-CA MAC layer activities corresponding to a packet transmission. Our modification was to set the value of the callback timer to 0 (makes sense when reliability mode is disabled), so that, in case there is a packet in the MAC layer queue, Contiki's CSMA-CA immediately performs clear channel assessment, and transmits packet if no carrier is detected, otherwise a node switches to the back-off mode. Moreover, Contiki 2.5 transmits unicast packet straight away as broadcast packet, if the MAC layer queue is full, bypassing all other packets. This feature seemed strange to us, therefore we have disabled that. Secondly, we are interested in determining the total number of packets transferred by a node. To accomplish this task our system keeps a record of the number of packets transmitted by a MAC layer. Our focus is to determine an upper bond on a node's throughput. Therefore, we have disabled MAC layer ACKs. To support accurate channel capacity estimation, system collects corrupted and interfered data frames information from Cooja's radio model being used. Thirdly, we are interested in measuring average per-packet delay at the MAC layer. Our system keeps a record of the time that each packet spends in the MAC layer queue. To determine the average per-packet delay per second, the total queuing delay per second is divided by the number of packets transmitted per second. Finally, to support higher data rate applications we have increased the MAC layer queue size so that it can store 20 packets.

IV. Experimental Results

In this section, we present results pertaining to node's throughput, average channel capacity, and delay's relationship with offered data load, for Contiki-based IEEE 802.15.4 WSNs. To obtain mentioned results, we performed a number of simulations. General simulation parameters are shown in Table I.

TABLE I
General Parameters for Simulation

Parameter Name	**Value**
MAC layer	CSMA-CA
MAC layer reliability	Disabled
Radio duty cycling algorithm	Null radio duty cycling
Radio Model	Undirected Graph Model
MAC layer queue size	30 packets
Bit rate	250 kbps
Node transmission range	50 meters
Node carrier sensing range	100 meters
Total packet size	127 bytes
Simulated node type	Tmote sky

A. *Upper Bound on Node Throughput Using Contiki*

To analyse the impact of the Contiki operating system and the networking protocol stack on the node throughput, we conducted simulations with different scenarios. We used two nodes; one node acts as a transmitter and the other acts as a receiver. The purpose of using two nodes with maximum sized packet is to get an upper bound on a node's throughput. In different simulation scenario, the application transmits 20, 30, 40, 50, and 60 packets per second and transmission continues till the transmitters transmits 200, 300, 400, 500, and 600 packets respectively. Fig. 1 shows a comparison of the application transmission rate and the MAC layer throughput for the three lower per-node packet rates.

Fig. 2 compares application transmission rate and the MAC

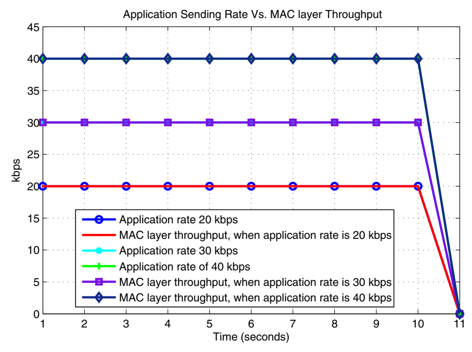

Fig. 1. Application data generation rate and MAC layer throughput

layer throughput when the application tries to send 50 and 60 kbps data respectively.

We next repeated this simulation scenarios. The difference between the two experiments is that now the application transfers data to the MAC layer in burst mode, and the MAC layer queue has a capacity to store all packets transferred to the MAC layer. Fig. 3 and Fig. 4 show application rate vs.

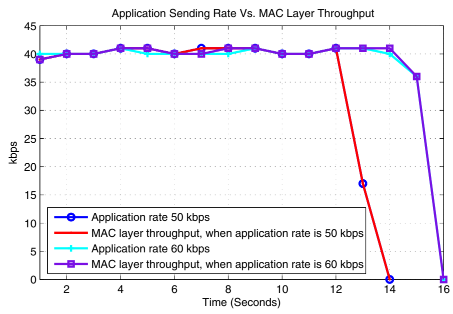

Fig. 2. Increased application data generation rate and MAC layer throughput

MAC layer throughput w.r.t. burst transfer mode. It can be observed that maximum achievable node's throughput is 45 kbps.

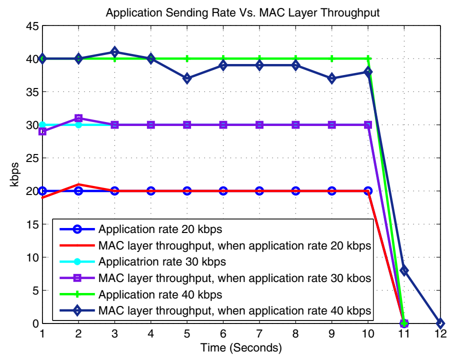

Fig. 3. Application data sending rate (burst mode) and MAC layer throughput

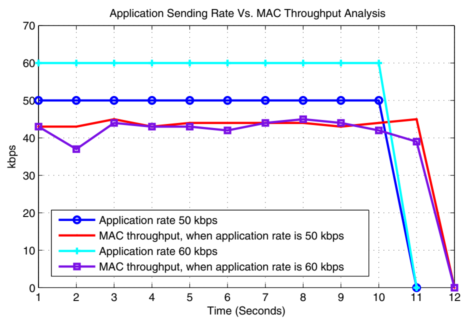

Fig. 4. Increased application data generation rate (burst mode) and MAC layer throughput

B. *IEEE 802.15.4 Channel Capacity Estimation Using Contiki*

To estimate the IEEE 802.15.4 WSN channel capacity and delay relationship with the offered traffic load, we simulated a WSN network with eleven nodes. All nodes are within the transmission range of each other. Ten nodes act as transmitters and one node acts as a receiver. We increase the offered load in the network from 20 to 200 kbps (offered load is uniformly distributed among 10 transmitters). Each simulation scenario is repeated three times, and averaged results are reported to account for the random nature of the CSMA-CA protocol. Fig. 5 shows average channel throughput w.r.t. offered load, and Fig. 6 shows the relationship of delay with the offered load. It can be observed that the channel saturates when the offered data load is in excess of 180 kbps. Moreover, going beyond 100 kbps results in excessive delay and packet drop rate.

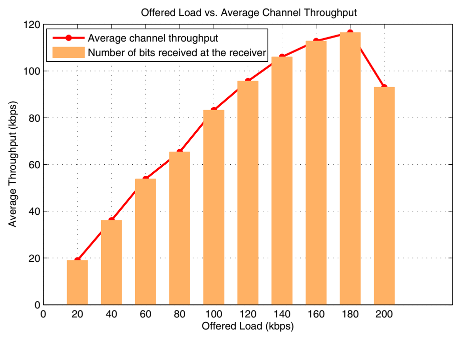

Fig. 5. Offered data load vs. throughput

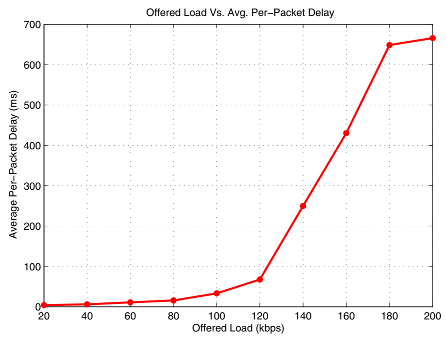

Fig. 6. Offered data load vs. average per-packet delay

V. Conclusions

Contiki limits a node's transmission capability, primarily due to its event handling mechanism and implementation of the networking protocol stack. We experimentally derived an upper bound on a node's transmission capability and wireless channel utilization. Knowing a node's transmission capability and the wireless channel utilization can help to design better QoS-based routing protocols, admission control mechanism, and congestion detection and avoidance techniques.

References

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In 47^{th} *Design Automation Conference*, DAC '10, pages 731–736, 2010.

[2] A. Rowe, A. Goode, G. Dhiraj, and N. Illah. CMUcam3: an open programmable embedded vision sensor. Technical report, Carnegie Mellon Robotics Institute, 2007.