Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 398637, 9 pages
http://dx.doi.org/10.1155/2015/398637

Research Article

Contiki-Based IEEE 802.15.4 Channel Capacity Estimation and Suitability of Its CSMA-CA MAC Layer Protocol for Real-Time Multimedia Applications

Muhammad Omer Farooq[1] and Thomas Kunz[2]

[1]*Institute of Telematics, University of Lübeck, Building 64, 2nd and 3rd Floor, Ratzerburger Allee 160, 23562 Lübeck, Germany*
[2]*Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada K1S 5B6*

Correspondence should be addressed to Muhammad Omer Farooq; farooq@itm.uni-luebeck.de

Received 9 September 2013; Accepted 24 February 2014

Academic Editor: David Taniar

Real-time multimedia applications require quality of service (QoS) provisioning in terms of bounds on delay and packet loss along with soft bandwidth guarantees. The shared nature of the wireless communication medium results in interference. Interference combined with the overheads, associated with a medium access control (MAC) protocol, and the implementation of a networking protocol stack limit the available bandwidth in IEEE 802.15.4-based networks and can result in congestion, even if the transmission rates of nodes are well below the maximum bandwidth supported by an underlying communication technology. Congestion degrades the performance of admitted real-time multimedia flow(s). Therefore, in this paper, we experimentally derive the IEEE 802.15.4 channel capacity using an unslotted CSMA-CA MAC protocol. We experimentally derive channel capacity for two cases, that is, when the CSMA-CA protocol is working without ACKs and when it is working with ACKs. Moreover, for both cases, we plot the relationship of offered data load with delay and packet loss rate. Simulation results demonstrate that the parameters that affect the choice of a CSMA-CA MAC layer protocol are end-to-end delay and packet loss requirements of a real-time multimedia flow, data load within the interference range of transmitters along the forwarding path, and length of the forwarding path.

1. Introduction

High-resolution and multidimensional sensing characteristics of IEEE 802.15.4-based wireless multimedia sensor networks (WMSNs) have led to their application in many real-time scenarios: visual surveillance [1], assisted living [2], and intelligent transportation [3] to name but a few. The application domains of WMSNs suggest that such networks can generate real-time multimedia streams [4]. Real-time multimedia streams require quality of service (QoS) provisioning especially in terms of bounded delay and packet loss rate along with soft bandwidth guarantees. Bandwidth is a shared and scarce resource in WMSNs, and interference along with the overheads, associated with a medium access control (MAC) protocol, and a networking protocol stack's implementation further limit the available bandwidth. These effects result in congestion even if nodes' transmission rates are well below the bandwidth supported by the underlying communication technology. Congestion increases delay and packet loss and hence results in a performance degradation of admitted real-time multimedia flows. Therefore, it is crucial for a QoS provisioning framework to determine a threshold on bandwidth usage. To achieve this task, a QoS provisioning framework must determine the wireless channel capacity based on an underlying communication technology and the relationship of offered data load with the delay and packet loss rate.

The CSMA-CA MAC layer protocol standardized in the IEEE 802.15.4 specification can work in reliable and unreliable mode. Working in the reliable mode, IEEE 802.15.4's CSMA-CA protocol waits for a constant period of time after transmitting a data frame to receive an ACK frame. If an ACK frame is not received, the MAC layer backs off, and a retransmission attempt is made after a random exponential backoff

delay. This process increases delay, and apparently it seems that a flow's end-to-end throughput may decrease, primarily due to ACK overhead. Using IEEE 802.15.4's CSMA-CA MAC layer protocol in the unreliable mode can increase a node's transmission rate and packet loss rate, and at the same time it can also decrease a data frame's end-to-end delay, as nodes do not wait to receive an ACK frame. Unreliable CSMA-CA does not retransmit data frames; therefore, a flow's end-to-end throughput may not be predictable, primarily due to the characteristics of the wireless communication channel. QoS in computer networks is related with the predictability of the service being offered. In case of WMSNs, one QoS metric of interest is predictable end-to-end throughput. Therefore, considering the QoS requirements of real-time multimedia applications and the possible pros and cons of reliable and unreliable CSMA-CA MAC layer protocols, a thorough study of reliable and unreliable CSMA-CA protocols is required to select the appropriate (reliable or unreliable CSMA-CA) MAC layer protocol to support real-time multimedia applications effectively and efficiently in IEEE 802.15.4-based WMSNs.

The primary aim of this work is to determine the suitability of IEEE 802.15.4's unslotted CSMA-CA MAC layer protocol for real-time multimedia applications, that is, with respect to the requirements of real-time multimedia application. We determine whether CSMA-CA with ACKs is a better choice for real-time multimedia applications or CSMA-CA without ACKs. To accomplish this task we experimentally derive IEEE 802.15.4's channel capacity using both versions of the IEEE 802.15.4's CSMA-CA MAC protocol. Moreover, for both cases we experimentally derive the relationship of data load inside a network with delay and packet loss rate. Based on the channel capacity, delay, and packet loss rate, we decide on the suitability of the MAC layer protocol (protocols under study) for real-time multimedia applications.

The remainder of this paper is organized as follows. Section 2 presents related work. In Section 3, details of the Contiki operating system are presented. Experimental studies to derive the IEEE 802.15.4 channel capacity and the relationship of offered data load with delay and packet loss using both versions of IEEE 802.15.4's CSMA-CA MAC layer are presented in Section 4. Simulation results to validate statistics derived in Section 4 are presented in Section 5, and we conclude this paper in Section 6.

2. Related Work

In [5] an analytical throughput analysis of the slotted CSMA-CA MAC layer protocol of IEEE 802.15.4 is presented. An analytical model typically requires simplifying assumptions to produce results. In real WSNs, these assumptions may not be true [6]; hence, such analysis may not accurately predict achievable throughput. Analytical models only consider the working of the IEEE 802.15.4 slotted CSMA-CA protocol; therefore, such analytical models do not predict throughput from an application's perspective. That is, an application normally runs over an operating system, and data transmitted by an application program traverse the networking protocol stack. Therefore, the operating system architecture and networking protocol stack overhead have an impact on a node's ability to transmit data. An operating system's design is affected by the resources available on the target hardware. Sensor nodes are severally resource-constraint devices; therefore, a WSN operating system designer has to consider these limitations. For example, memory available on a contemporary Tmote sky node is 10 KB; hence, from the networking protocol perspective the operating system may allocate small buffers for the networking protocols [7]. Networking protocols are invariably implemented using timers and events; therefore, the way events are handled in an operating system has an impact on a node's throughput. Therefore, if an analytical approach states that IEEE 802.15.4 slotted CSMA-CA can achieve a certain throughput, we cannot conclude that the stated throughput can be achieved by user applications. Hence, a good analysis must consider all the factors that limit the channel capacity.

WSN testbed-based throughput measurements of IEEE 802.15.4 are presented in [8, 9]. Furthermore, [9] also presents simulation-based throughput measurements of IEEE 802.15.4. Both research papers focus on overall channel capacity and do not consider the node level throughput. The testbed results reported in [8, 9] show that the upper limit on channel throughput is in the range of 35 to 40 kbps. The simulation-based results reported in [9] show that the maximum channel capacity is approximately 65 kbps. It is important to note that in [9] simulations were performed using NS 2.34, which does not capture the impact of an operating system on the channel capacity.

In [10, 11], a throughput analysis of IEEE 802.15.4 with the Guaranteed Time Slot (GTS) algorithm is presented. The model for the IEEE 802.15.4 GTS algorithm is built with the help of OPNET Modeler [12] and simulations were performed using the same simulator. Both papers focus on maximum channel throughput. Again, limitations imposed by the WSN operating systems are ignored; hence, the reported maximum channel capacity may not be an accurate estimate from an application's perspective.

Most of the related work is focused on estimating the overall channel capacity. As per authors knowledge no work focuses on analyzing the impact of an operating system's networking protocol stack's implementation and offered load on a node's throughput and per-packet delay.

3. Contiki Operating System Overview and Its Unslotted CSMA-CA Implementation

In this section, we discuss the design of the Contiki operating system for WSNs. The discussion in this section helps us to understand the Contiki features that limit the transmission capability of a node. Afterwards, we discuss Contiki's CSMA-CA implementation in detail, and we examine how it limits the transmission rate of a node. We conclude this section with a discussion of our modifications to the Contiki's CSMA-CA implementation to increase a node's transmission capability.

3.1. Contiki Overview. Contiki is a lightweight open source OS written in the C programming language for WSNs [13]. It follows a modular architecture, and it is built around an event driven kernel. Contiki provides preemptive multitasking at the process level, but, to yield the processor to another thread, the running thread has to invoke the yield function explicitly. In other words, if a running thread continues to run, waiting threads are not scheduled.

The Contiki kernel comprises an event scheduler that dispatches events to the running processes. Process execution is triggered by events dispatched by the kernel to the process or by a polling mechanism. When an event is dispatched to a process, it runs 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. Furthermore, the communication stack overhead, for example, copying a message and adding headers to a message, adds further delay.

3.2. The Contiki 2.5 CSMA-CA MAC Layer. When a CSMA-CA MAC layer receives a packet for the upper layer, it enqueues the packet in a MAC layer buffer. If the packet received at the MAC layer is a broadcast packet, it is not enqueued; rather, the MAC layer broadcasts it straight-away, without performing carrier sensing. In case of a unicast packet, if no space is available in the MAC layer queue, the unicast packet is treated as a broadcast packet; hence, the packet is transmitted without performing carrier sensing. If a unicast packet is treated as a broadcast packet, it is not retransmitted, in case of data packet collision or corruption.

Whenever the MAC layer has a packet to transmit, it delays carrier sensing by 1/8th 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. The real-time timer on a Tmote sky mote ticks 16,384 times per second. 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 backoff 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. If the MAC layer is about to transmit a packet and it senses that the channel is busy, it backs off for a random amount of time, as stated above. The CSMA-CA makes three retransmission attempts and if unsuccessful, the packet is dropped.

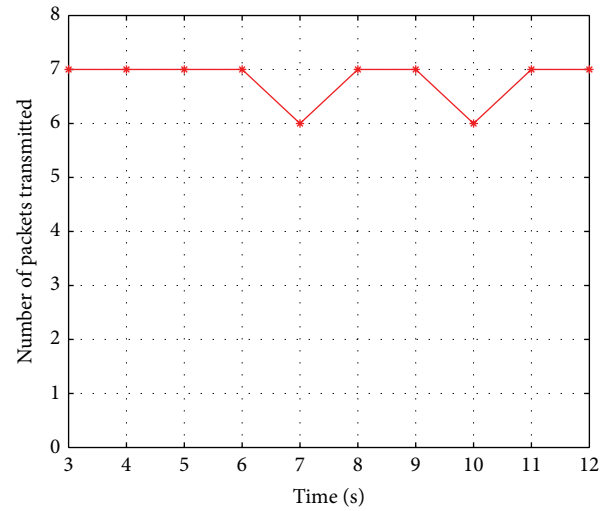

Figure 1: Packets transmitted per unit time using default CSMA-CA and null duty cycling algorithm.

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 ith data packet in bytes. A transmission rate of Γ bps is only possible if MAC layer ACKs are disabled; otherwise, node throughput will further degrade.

We have performed simulations to validate that, using the default CSMA-CA implementation with null duty cycling, demonstrating that a node running Contiki can only transmit 8 kbps at maximum. Simulations were performed using the Cooja WSN simulator [14]. Two Tmote sky nodes are used; one node sends ten packets per second to the other node. The size of each data packet is 127 bytes, including packet headers, and the MAC layer queue can store 6 full size packets. Nodes are using Contiki's Rime communication stack with Cooja's undirected radiograph model. The two-node WSN is simulated for fourteen seconds, and MAC layer ACKs are disabled. If the MAC layer queue is full, the MAC layer broadcasts packets without performing channel sensing. We drop such packets at the MAC layer because broadcasting a packet without performing channel sensing can cause interference and hence might lead to decreased network throughput.

To accurately estimate throughput, we keep track of interfered and corrupted packets by adding code to Cooja's undirected radio graph mode. Hence, the simulator detects and reports interfered and corrupted data packets, and such data packets are not considered when we estimate a node's throughput. In the first set of experiments reported here, though, note that there are no collisions as we only have one transmitter. The MAC layer keeps track of all transmitted packets. Figure 1 shows the number of packets transmitted per unit time between three to twelve seconds of simulation time.

Figure 1 shows that a node can only transfer 7 packets per second. The size of each packet was 127 bytes; therefore, the node's throughput in this case was 6.94 kbps. This confirms the fact that after every successful transmission a node waits for 1/8th of a second. An OS has to add headers to an application data and schedule a packet transfer, and, in case of CSMA, there is a need to perform clear channel assessment; therefore, a node cannot exactly transfer eight packets per second. The important lesson we take away from this experiment is that a node's transmission ability is limited in terms of number of packets, using the default setup. We conclude that the delay between two successive transmissions along with the FIFO event dispatching mechanism, and events run to completion semantics, reduce node level throughput.

3.3. Modifications to the Contiki Operating System. To enhance network throughput, we modified the time interval for which a CSMA-CA MAC layer waits to perform the clear channel assessment. For the null radio duty cycling algorithm, this time is calculated as (1/CCI) seconds. The value of CCI, in case of null duty cycling, is 8. 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, so that, in case there is a packet in the MAC layer queue, Contiki's CSMA-CA immediately performs clear channel assessment and transmits a packet if no carrier is detected; otherwise, a node switches to the backoff mode. It is shown through simulations that our modification increases a node's throughput; hence, nodes in a network can transmit more data compared to what nodes can transmit with Contiki's original CSMA-CA implementation. The rationale for using a CCI value of 8 is to ensure fairness, that is, reducing the rate at which nodes transmit leaves more bandwidth for other nodes. If our modifications are coupled with an admission control procedure, not only do we increase nodes' throughput, but also the admission control will impose its notion of fairness. Moreover, Contiki 2.5's feature of sending unicast packet straight away as broadcast packet, if the MAC layer queue is full, bypassing all other packets, seemed strange; therefore, as mentioned before, we have disabled that.

Secondly, 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 for transmitted packets 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.

We are interested in studying Contiki's CSMA-CA implementation with and without ACKs. In the latter case, it is possible that two or more nodes sense that the wireless channel is idle; hence, they may start transmission at the same time. This results in a collision of the transmitted packets. If our simulator considers such packets as delivered, we would end up overestimating the channel capacity. Therefore, we added logic to Cooja's undirected radiograph model (URGM) (our simulations use URGM), to keep track of the total number of collided and corrupted data packets per second.

Table 1: General parameters for simulation.

Parameter value	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 frame size	127 bytes
Simulated node type	Tmote sky

4. IEEE 802.15.4 Channel Capacity Estimation

To estimate the IEEE 802.15.4-based WSN channel capacity and relationship of delay and packet loss rate with offered traffic load, we simulated a WSN 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 total offered data load in the network from 20 to 220 kbps (offered data 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. Table 1 lists general simulation parameters.

From Table 1 it can be seen that we are using full size IEEE 802.15.4 packets to estimate the IEEE 802.15.4 channel capacity. If we use short IEEE 802.15.4 addressing mode, 102 bytes of application data is carried in a MAC layer frame using the Rime protocol stack of Contiki operating system. Invariably, multimedia applications generate lots of data; therefore, it is not rare that such applications utilize the maximum possible packet size of 102 bytes in each transmitted packet.

4.1. IEEE 802.15.4 Channel Capacity Estimation. Figure 2 shows the average channel throughput with respect to the offered data load. In case of CSMA-CA without ACKs, the rate at which the channel throughput increases till the offered data load reaches 100 kbps is almost linear with offered data load. When the offered data load ranges from 100 to 180 kbps, the slope of the line showing average channel throughput increases more slowly. This is primarily due to the distributed nature of the CSMA-CA protocol. As data load in the WSN increases, each node has more data to send; hence, nodes frequently contend for channel access and in this process nodes go to backoff mode more frequently. It results in an increased delay in getting access to the channel; hence, channel throughput increases slowly. In fact, in simulation results we have observed that an offered data load of 180 kbps acts as a threshold point in terms of offered data load, after

which channel throughput starts to decrease. This is primarily due to the CSMA-CA protocol and its backoff mechanism.

In case of CSMA-CA with ACKs, an interesting observation is that, up to an offered data load of 60 kbps, packet drop rate is almost 0%. The primary reason for 0% packet drop rate till the offered data load is 60 kbps is that MAC layer ACKs are enabled, and if an ACK for a transmitted frame is not received, the frame is retransmitted. Moreover, in this case, the channel throughput almost remains constant at approximately 93 kbps from an offered data load of 120 kbps to an offered data load of 200 kbps. The reason for this phenomenon is that the MAC layer retransmits lost/corrupted data frames. The MAC layer was not able to retransmit all lost frames due to the backoff and ACKs overhead. The channel is saturated at an offered load of 220 kbps as the channel throughput drops to 86 kbps.

For an offered data load between 20 and 100 kbps the average channel throughput of CSMA-CA with ACKs is higher compared to the average channel throughput of CSMA-CA without ACKs. For an offered data load between 120 and 180 kbps, CSMA-CA without ACKs offers higher channel throughput and lower packet loss rate. The higher packet loss rate, and decreased throughput in case of CSMA-CA with ACKs between an offered data load of 120 to 180 kbps are due to the following reasons: (a) Figure 2 shows that an increase in the offered load results in increased packet loss rate, therefore causing a higher number of retransmissions, and (b) an increased offered data load normally means more packets transmitted per second, hence more time a node has to wait for ACKs. Channel throughput results shown in Figure 2 show a step decrease in the channel's throughput for an offered data load in excess of 180 kbps, in case of CSMA-CA without ACKs. On the other hand, there is a gradual decrease in the channel's throughput for an offered data load of in excess of 200 kbps, in case of CSMA-CA with ACKs.

CSMA-CA with ACKs offers a packet loss rate of 0% as long as the offered data load is between 0 and 60 kbps. In all other cases, an increase in offered data load implies higher packet loss rate. Therefore, if the only parameter of interest for real-time multimedia application is strict reliability (low packet loss rate), CSMA-CA with ACKs is the only choice, and the system must limit the amount of data within the interference range of transmitters along the forwarding path to 60 kbps. A real-time multimedia application can tolerate end-to-end packet loss rate of 5% [15]; in this case, CSMA-CA without ACK is only a feasible choice if total data load within the interference range of nodes along the forwarding paths is less than 30 kbps, and preferably there must be no more than one intermediate node between the source-destination pair. Assuming that end-to-end packet loss rate of 5% is the only requirement and there are multiple intermediate nodes between source-destination pairs CSMA-CA with ACKs is the only choice, total data load within the interference range of nodes along the forwarding paths must not exceed 60 kbps.

Figure 3 shows the relationship of delay with offered data load. In case of CSMA-CA without ACKs, it can be observed that average per-packet delay does not increase a lot as long as the offered data load ranges between 0 and 100 kbps. Beyond that point, average per-packet delay

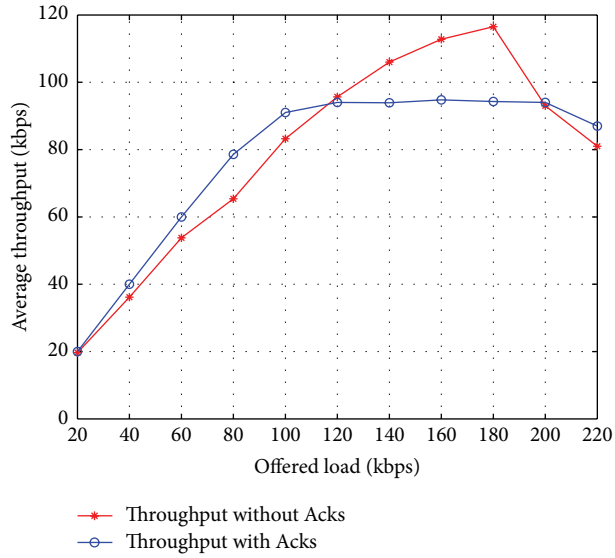

Figure 2: Offered data load versus throughput.

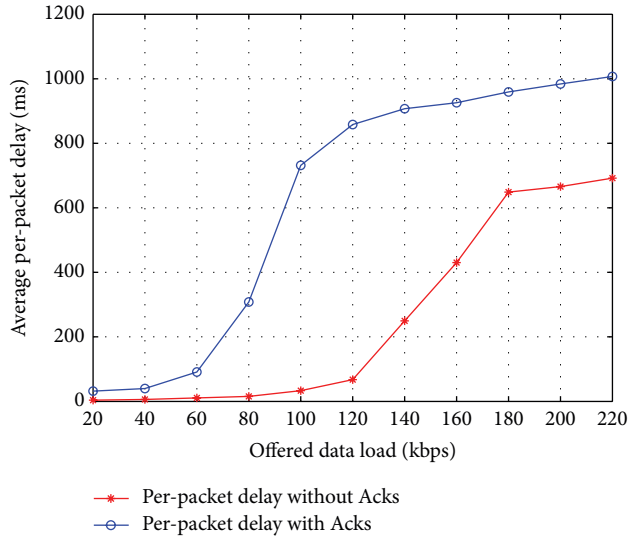

Figure 3: Offered data load versus average per-packet delay.

increases sharply. From Figures 2 and 3, it can be concluded that the channel capacity in case of CSMA-CA without ACKs is 118 kbps, and it is achieved at an offered data load of 180 kbps. This confirms that operating below the bandwidth supported by the underlying communication technology results in congestion and hence increased delay and packet loss rate.

Figure 3 also shows the relationship of delay with offered data load when the MAC layer ACKs are enabled. It can be observed that the average per-packet delay in this case is much higher as compared to the average per-packet delay when the MAC layer ACKs were not enabled. From an offered data load of 60 to 120 kbps there is a sharp increase in the average per-packet delay, and the maximum channel capacity in this case is 94 kbps.

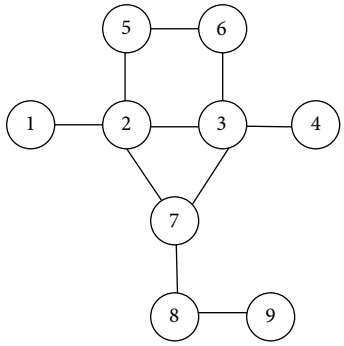

Figure 4: Simulated network topology.

Table 2: Nodes' interference set.

Node ID	Interfering nodes
1	1, 2, 3, 5, and 7
2	1, 2, 3, 4, 5, 6, 7, and 8
3	1, 2, 3, 4, 5, 6, 7, and 8
4	2, 3, 4, 5, 6, and 7
5	1, 2, 3, 4, 5, 6, and 7
6	2, 3, 4, 5, 6, and 7
7	1, 2, 3, 4, 5, 6, 7, 8, and 9
8	2, 3, 7, 8, and 9
9	7, 8, and 9

A real-time multimedia flow can tolerate end-to-end packet delay of 250–300 ms [16]. Figure 3 shows per-hop average per-packet delay; therefore, end-to-end delay depends on the number of hops between the source and the destination node. If a real-time multimedia flow requires 5% end-to-end packet loss rate and end-to-end delay should remain within 250–300 ms, using CSMA-CA without ACKs is only possible if there is at most one relay node between the source and the destination node, and data load is limited to at most 30 kbps. In the same scenario, CSMA-CA with ACKs can fulfill a flow's requirement even if the source and the destination nodes are three hops away, and data load must be limited to 50 kbps. In general, the choice of a CSMA-CA protocol depends on an application's requirements, forwarding path's length, and the data load within the interference range of transmitters along the forwarding path.

5. Performance Evaluation

In this section, we simulate a multihop WSN and we assume that nodes inside the network generate real-time multimedia flows. The main purpose of these simulations is to validate the conclusions we drew from the results presented in Section 4. We run separate simulations to validate the results for CSMA-CA without ACKs and the results for CSMA-CA with ACKs. In each simulation scenario, average per-packet delay (total time spent by a packet in the MAC layer queue), and node's average throughput between the simulation time interval of 20 to 100 seconds are measured at each node. Figure 4 shows the simulated network topology. Table 1 lists general simulation parameter. Simulations are performed on the Cooja WSN simulator. Moreover, Table 2 shows nodes within the interference range of each node present in the simulated network.

5.1. Results Verification of IEEE 802.15.4's CSMA-CA Protocol without ACKs. The test-case to validate the results for CSMA-CA without ACKs assumes that a flow can tolerate some packet loss (up to 8% per-hop packet loss) but requires bounded delay (per-hop packet delay less than or equal to 30 ms). As per the experimental result shown in Figures 2 and 3, offered data load inside the network should not exceed 60 kbps to provide an acceptable level of service. To validate the results presented in Figures 2 and 3, we created three simulation scenarios. Each simulation scenario is repeated three times, and we report average results in this section.

5.1.1. Scenario 1. Table 3 summarizes the flows in Scenario 1. It is evident from Figure 5 that the average per-packet delay at each node is less than 30 ms corresponding to Scenario 1. Data loads within the interference range of nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are 40, 50, 50, 30, 40, 30, 50, 40, and 20 kbps, respectively. Figure 2 does not plot average throughput corresponding to the offered data load of 30 kbps. Considering Figure 2, it can be observed that the average throughput for an offered data load between 20 and 40 kbps increases almost linearly. Therefore, we can use (1) to estimate packet loss rate at an offered data load of 30 kbps via linear interpolation.

Consider

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}. \tag{1}$$

In this case, $x_1 = 20$, $x_2 = 40$, $y_1 = 19.7$, and $y_2 = 38$. Solving (1) for the given values yields $y = 0.915x + 1.4$. The average throughput for an offered data load of 30 kbps is 28.85 kbps; therefore, per-hop packet drop rate is 3.3 percent. Similarly, Figure 2 does not plot average throughput for an offered data load of 50 kbps, but we can solve (1) using the closest two data points: $x_1 = 40$, $x_2 = 60$, $y_1 = 38$, and $y_2 = 55$. Solving (1) for these values yields $y = 0.85x + 4$. Therefore, the estimated per-hop packet drop rate is 7% for an offered data load of 50 kbps. The average number of bits that node 4 anticipates to receive as per the result present in Figure 2 can be calculated as $(10 \times 0.95 \times 0.93 \times 0.93) = 8.21$ kbps. Simulation results show that node 4 received 9.2 kbps. Hence, the per-hop packet loss rate for flow A is certainly below 8 percent. Similarly, node 9 must receive $(10 \times 0.95 \times 0.93) = 8.83$ kbps. Simulation results show that the average number of bits received by node 9 is 9.80 kbps. Therefore, the requirements in terms of delay and per-hop packet loss rate are met for both flows.

5.1.2. Scenario 2. Table 4 summarizes the flows in simulation Scenario 2. This scenario has the same two flows as Scenario 1, with an additional flow from node 5 to node 4 (flow C), sending data at a rate of 5 kbps. In this scenario, the maximum data loads that can be ideally observed

Table 3: Simulation Scenario 1.

Flow ID	Source node	Destination node	Start time (Sec)	Pkts/Sec	Total packets to transmit
A	1	4	4	10	1000
B	7	9	10	10	1000

Table 4: Simulation Scenario 2.

Flow ID	Source node	Destination node	Start time (Sec)	Pkts/Sec	Total packets to transmit
A	1	4	4	10	1000
B	7	9	10	10	1000
C	5	4	15	5	500

(i.e., if no packets were lost) at nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are 45, 60, 60, 40, 50, 40, 60, 40, and 20 kbps, respectively. Data load within the interference range of nodes 2, 3, and 7 is maximum, that is, 60 kbps with respect to the chosen threshold level. In our experiments, the average number of bits received at node 4 is 12.96 kbps. Node 4 receives 8.24 kbps for the flow A and 4.72 kbps for flow C. From Figure 2, we can see that, at the offered data load of 40 kbps, the packet drop rate is approximately 5 percent. Figure 2 does not plot average throughput for the offered data load of 45 and 50 kbps, but again we can use (1) to estimate average packet loss at an offered data load of 45 and 50 kbps, as done before.

The estimate of the average throughput for an offered data load of 45 kbps is 42.25 kbps; therefore, in this case, we estimate that the per-hop packet drop rate is 6 percent. The per-hop packet drop rate for the offered data load of 50 kbps is 7 percent. If we use these derived values, the average number of bits that node 4 expects to receive for flow A is $(10 \times 0.94 \times 0.92 \times 0.92) = 7.66$ kbps, which is lower than 8.24 kbps, and the average per-packet delay at nodes 1, 2, and 3 is less than 30 ms as shown in Figure 3; hence, the QoS requirements of flow A are met. Similarly, node 4 should at least receive $(5 \times 0.93 \times 0.95) = 4.42$ kbps for flow C. But our results show that node 4 has received 4.72 kbps, which is more than 4.42 kbps. Moreover, Figure 5 shows that the average per-packet delay at nodes 5 and 6 is less than 30 ms. Hence, the requirements of flow C are also fulfilled. Node 9 is the destination of flow B, and, as per our simulation results, it has received on average 9.80 kbps. Considering the results presented in Figure 2 and the data load within the interference range of nodes 7 and 8, node 9 should have received 8.74 kbps on an average. In this case, node 9 has received 12 percent more data compared to the results presented in Figure 2. The average per-packet delay as per Figure 5 is less than 30 ms at nodes 7 and 8; hence, the requirements of flow B are also met.

In this scenario, the average number of bits received at destination nodes is more than what is anticipated (Figure 6). In both scenarios, we have observed that the packet drop rate is lower than what is shown in Figure 2. A plausible explanation is that we derived the expected number of received bits based on the assumption that the offered load is equal to the sum of all flows without any packet loss. In reality, as soon as a packet is dropped, the offered load is reduced, resulting in a lower offered load and hence a lower packet loss rate, increasing the actual observed packet delivery rate.

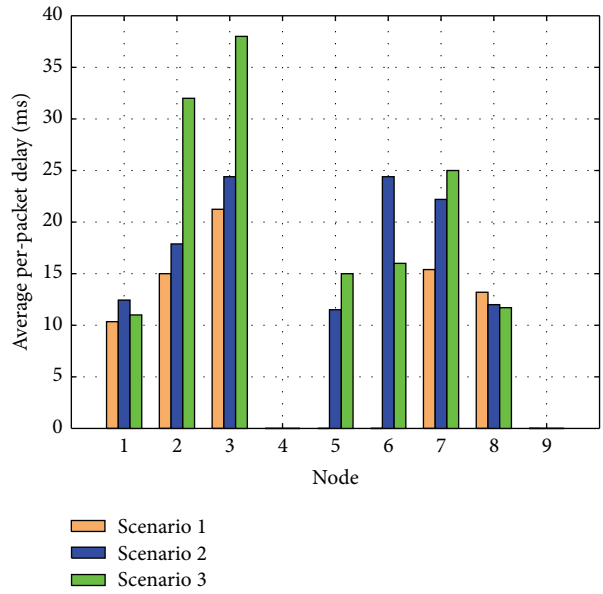

Figure 5: Nodes' average per-hop packet delay.

One can consider this as a positive development, because requirements of real-time multimedia flows are fulfilled. But at the same time, one can argue that our threshold of 60 kbps is overly conservative; hence, we are missing out opportunities to admit more flows in a network. Therefore, to analyze the impact of operating marginally above the threshold, let us consider Scenario 3.

5.1.3. Scenario 3. Table 5 summarizes the flows in simulation Scenario 3. This scenario further extends Scenario 2 by increasing the data rate of flow C, so that some nodes experience an offered load (as the sum of the transmission rates of all nodes in the interference range) above the 60 kbps threshold. In this scenario, the maximum data loads that can be observed at nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are 47, 64, 64, 44, 54, 44, 64, 40, and 20 kbps, respectively. For simplicity, we assume that the packet loss rate corresponding to the offered data load of 44, 47, 54, and 64 kbps is the same as for the offered data load of 45, 45, 55, and 65 kbps, respectively. The per-hop packet drop rate corresponding to the offered data load of 45 kbps is 6 percent, as derived in Scenario 2. We need

Table 5: Simulation Scenario 3.

Flow ID	Source node	Destination node	Start time (Sec)	Pkts/Sec	Total packets to transmit
A	1	4	4	10	1000
B	7	9	10	10	1000
C	5	4	15	7	700

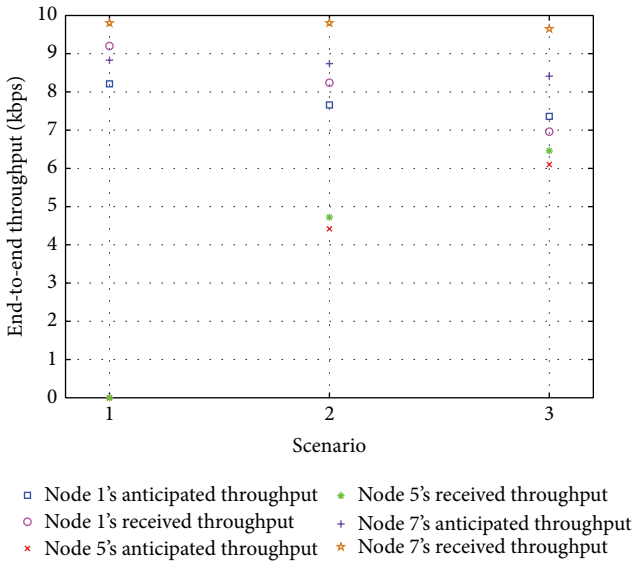

Figure 6: Anticipated versus received end-to-end throughput.

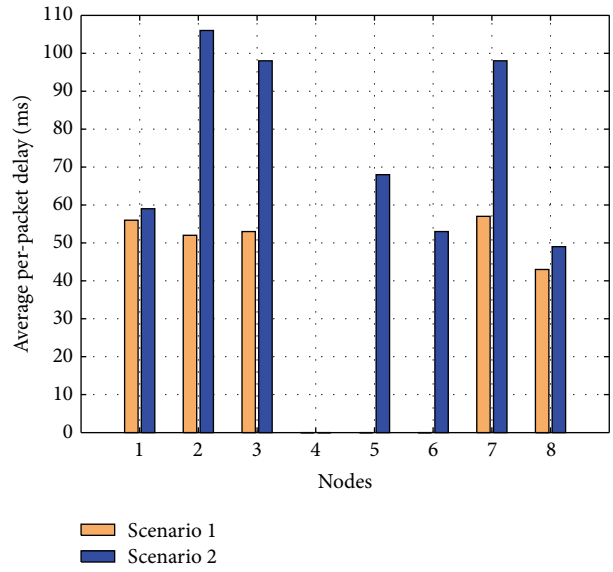

Figure 7: Nodes' average per-hop delay (ACKs mode).

to solve $y = 0.85x + 4$ to estimate the per-hop packet loss rate corresponding to the offered data load of 55 kbps, which is 7.72 percent. Similarly, for the offered data load of 65 kbps we need to solve (1) with values $x_1 = 60$, $x_2 = 80$, $y_1 = 55$, and $y_2 = 65$ resulting in $y = 0.5x + 25$. Hence, the per-hop packet loss rate when the offered data load is 65 kbps is 11.5 percent. Node 4 has received 13.41 kbps, 6.96 kbps for flow A and 6.46 kbps for flow C.

The average number of bits that node 4 should expect to receive for flow A is $(10 \times 0.94 \times 0.885 \times 0.885) = 7.36$ kbps. Similarly, as per Figure 2, the average number of bits that node 4 expects to receive for flow C is $(7 \times 0.923 \times 0.94) = 6.10$ kbps. Our results show flow A has suffered more packet loss; moreover, Figure 5 shows that the average per-packet delay at nodes 2 and 3 has significantly increased and now exceeds the target of 30 ms per hop. Therefore, in this case the flow A experiences degradation in its performance. Node 9 can expect receiving $(10 \times 0.885 \times 0.95) = 8.41$ kbps, but in simulation node 9 receives 9.65 kbps, which is 15 percent more than what is expected. Nevertheless, we have shown that exceeding 60 kbps deteriorates the performance of at least one real-time multimedia flow. Furthermore, Figure 5 shows increased delay especially at nodes 2 and 3 which further supports the tightness of the chosen threshold presented in this paper.

5.2. Results Verification of IEEE 802.15.4's CSMA-CA Protocol with ACKs. The test-case to validate the results for CSMA-CA with ACKs assumes that a flow cannot tolerate packet loss, and it requires per-hop packet delay of less than 70 ms. As per the experimental results shown in Figures 2 and 3, offered data load inside the network should not exceed 50 kbps to provide an acceptable level of service. We created two simulation scenarios. Each simulation scenario is repeated three times, and we report average results in this section.

5.2.1. Scenario 1. Table 3 summarizes the flows in Scenario 1. It is evident from Figure 7 that the average per-packet delay at each node is less than 70 ms corresponding to Scenario 1. Data loads within the interference range of nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are 40, 50, 50, 30, 40, 30, 50, 40, and 20 kbps, respectively. As per Figure 2, the per-hop packet drop rate is 0% till the data load exceeds 60 kbps. In our simulation results, the end-to-end throughput of both flows is 100%. Therefore, the results corresponding to the first simulation scenario validates our determined statistics for the wireless channel under study.

5.2.2. Scenario 2. Table 4 summarizes the flows in simulation Scenario 2. In this scenario, the maximum data loads that can be ideally observed at nodes 1, 2, 3, 4, 5, 6, 7, 8, and 9 are 45, 60, 60, 40, 50, 40, 60, 40, and 20 kbps, respectively. Data load within the interference range of nodes 2, 3, and 7 is 60 kbps, and it is above the chosen threshold. In our experiments, the end-to-end throughput of all three flows is perfect, and it is in accordance with the results presented in Figure 2. The data load within the interference range of nodes 2, 3, and

7 is 60 kbps which is above the chosen threshold, and if we consider per-hop packet delay results presented in Figure 7, it can be observed that per-packet delay at these nodes is in excess of 70 ms. Hence, simulation results for this scenario demonstrates that our determined thresholds are tight.

6. Conclusions

In this paper, we experimentally determined the IEEE 802.15.4-based wireless channel capacity using the unslotted CSMA-CA MAC layer protocol. Furthermore, we showed the relationship of offered data load with the per-hop packet delay and per-hop packet loss rate. Based on the experimental results presented in this paper, we demonstrated that the CSMA-CA protocol without ACKs offers lower end-to-end delay compared to the CSMA-CA protocol with ACKs. Furthermore, we showed that the CSMA-CA protocol with ACKs offers 0% packet loss rate if the data load within the interference range of a node is within 60 kbps. If the data load within the interference range of a node is below 120 kbps, the CSMA-CA protocol with ACKs offers better throughput as compared to the CSMA-CA protocol without ACKs. The CSMA-CA protocol without ACKs achieves better throughput if the data load is between 120 and 160 kbps, as compared to the CSMA-CA protocol with ACKs. We conclude that the choice of a suitable IEEE 802.15.4 CSMA-CA MAC layer protocol depends on the requirements of a real-time multimedia flow, data load within the interference range of transmitters along the forwarding path, and the length of the forwarding path. The experimental results can help to derive tight bounds on the offered data load, if the path length and QoS requirements of a flow are known. Hence, the relationship of offered data load with the packet loss rate and delay can be a useful information for a flow admission control.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. Chitnis, Y. Liang, J. Y. Zheng, P. Pagano, and G. Lipari, "Wireless line sensor network for distributed visual surveillance," in *Proceedings of the 6th ACM International Symposium on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '09)*, pp. 71–78, October 2009.

[2] N. Li, B. Yan, G. Chen, P. Govindaswamy, and J. Wang, "Design and implementation of a sensor-based wireless camera system for continuous monitoring in assistive environments," *Personal and Ubiquitous Computing*, vol. 14, no. 6, pp. 499–510, 2010.

[3] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Proceedings of the 47th Design Automation Conference (DAC '10)*, pp. 731–736, June 2010.

[4] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 921–960, 2007.

[5] T.-J. Lee, H. R. Lee, and M. Y. Chung, "MAC throughput limit analysis of slotted CSMA/CA in IEEE 802.15.4 WPAN," *IEEE Communications Letters*, vol. 10, no. 7, pp. 561–563, 2006.

[6] S. W. Golomb, "Mathematical models—uses and limitations," *Simulation*, vol. 4, no. 14, pp. 197–198, 1970.

[7] M. O. Farooq and T. Kunz, "Operating systems for wireless sensor networks: a survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011.

[8] A. Kumar, P. G. Namboothiri, S. Deshpande, S. Vidhyadharan, K. M. Sivalingam, and S. A. V. Satya Murty, "Testbed based throughput analysis in a wireless sensor network," in *Proceedings of the 18th National Conference on Communications (NCC '12)*, pp. 1–5, February 2012.

[9] J. Edwards, F. Demers, M. St-Hilaire, and T. Kunz, "Comparison of ns2.34's ZigBee/802.15.4 implementation to Memsic's IRIS Motes," in *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC '11)*, pp. 986–991, July 2011.

[10] P. Jurčík, A. Koubâa, M. Alves, E. Tovar, and Z. Hanzálek, "A simulation model for the IEEE 802.15.4 protocol: delay/throughput evaluation of the GTS mechanism," in *Proceedings of the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '07)*, pp. 109–116, IEEE, Istanbul, Turky, October 2007.

[11] S. Fan, J. Li, H. Sun, and R. Wang, "Throughput analysis of GTS allocation in beacon enabled IEEE 802.15.4," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, pp. 561–565, July 2010.

[12] OPNET Modeler, http://www.opnet.com/solutions/network_rd/modeler.html.

[13] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki—a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, November 2004.

[14] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proceedings of the 31st Annual IEEE Conference on Local Computer Networks (LCN '06)*, pp. 641–648, November 2006.

[15] C. Sarr, C. Chaudet, G. Chelius, and I. G. Lassous, "Bandwidth estimation for IEEE 802.11-based ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 10, pp. 1228–1241, 2008.

[16] Telecommunication Standardization Sector of ITU, *ITU-T Recommendation G.114: Transmission Systems and Media : General Recommendations on the Transmission Quality for an Entire International Telephone Connection : One-Way Transmission Time*, International Telecommunication Union, 1994.