Maximizing Throughput in ZigBee Wireless Networks through Analysis, Simulations and Implementations*

T. Ryan Burchfield and S. Venkatesan
Computer Engineering Program
University of Texas at Dallas
Richardson, Texas 75080
{timothy.burchfield@student., venky@}utdallas.edu

Douglas Weiner
Wireless Monitoring Solutions, Signal Technology
A Crane Co.
Plano, Texas 75093
doug.weiner@craneae.com

Abstract—ZigBee is a robust wireless communication standard managed by the ZigBee Alliance and based on the standard IEEE 802.15.4 physical and MAC layers. Certain applications of ZigBee, may be critically limited by its designation as a low data rate standard. This paper describes a three-phased approach for measuring and attaining maximal throughput in a ZigBee wireless network. The phases consist of: 1) practical calculations, 2) NS-2 simulations and 3) hardware implementations on Ember Corporation EM2420 based development equipment. The first two phases provide an approximate practical upperbound of 120kbps. The final phase is targeted to realizing maximal throughput in an actual hardware implementation. The results reveal maximum throughput for a ZigBee wireless network reaches 110kbps for well-refined hardware designs. Finally, a set of desirable traits is presented for future ZigBee hardware designs concerned with achieving maximum network throughput.

I. Introduction

The initial version of the ZigBee networking standard, published in 2004, met the goal of creating a reliable, cost-effective, low power, wirelessly networked monitoring and control platform targeted to home, business and factory automation applications. A typical application used to explain ZigBee's functionality is a light switch panel and various lights distributed throughout a home [1]. A ZigBee compliant radio and microcontroller is present at the light switch panel and each wirelessly controlled device. The ZigBee standard enables the light switch panel to dynamically discover new controllable devices, even of different manufacturer, and send pre-defined commands (on, off, dim) to those devices by utilizing ZigBee Alliance approved profiles. Each ZigBee Alliance approved profile describes the network configuration parameters and message formats necessary for devices of similar interest, e.g. lighting control, to communicate successfully. ZigBee can handle larger applications because of its multi-hop routing capabilities. If the device to be controlled is out of the reception range of the light switch, then other intermediate light switches or networked lights will cooperate to route packets to the final target.

A complex application becoming popular is a commercial/industrial wireless sensor network (WSN) used to monitor environmental operational conditions. Wireless sensor networks consist of a deployment of many independent data sensing nodes. Each sensing node is equipped with a radio transceiver and microcontroller capable of preliminary data processing and controlling wireless communication. These nodes are often deployed with a battery as their energy source so it is crucial that energy be used conservatively. Topologies of WSNs may change as nodes fail or relocate. Therefore, it is desirable to

*Supported by a contract from Wireless Monitoring Solutions, Signal Technology. A Crane Co.

have networks capable of adapting to network topology changes by supporting redundant mesh topologies. ZigBee is an ideal communication standard for WSNs because it is built on the power-efficient IEEE 802.15.4 specification and has a multi-hop routing algorithm capable of adapting to link failures while minimizing communication power consumption [1].

Event monitoring WSNs may detect conditions that cause demands for large amounts of data be quickly transferred through the network. Information from all sensors that detect an event may be crucial for end systems to determine the location and severity of the event, thus packets cannot be lost. Low latency is required where WSNs are part of control networks because an event representing an emergency must be quickly relayed to network egress points so that control measures may be dispatched to suppress the emergency. These heavy traffic patterns necessary for some WSN applications may not be feasible because ZigBee is based on the low data rate (up to 250kbps) IEEE 802.15.4 standard. Congestion, data framing and interference all contribute to a further decrease in the available throughput in ZigBee.

In order to measure throughput performance, this paper is organized in the following manner. Section II discusses background information about ZigBee, IEEE 802.15.4, and their general interactions. Section III illuminates a specific inefficiency in IEEE 802.15.4 and its impact on ZigBee. Section IV quantifies application throughput with a practical estimation based upon varying channel activity levels. Section V confirms the simulation platform's accuracy and simulates maximal throughput experiments. Finally, Section VI describes the software and hardware architecture necessary to realize near maximal throughput in a hardware implementation.

II. ZIGBEE MESSAGING OVERHEAD

Version 1.0 of the ZigBee specification is built on the IEEE 802.15.4-2003 standard PHY/MAC layers for low rate wireless personal area networks (WPANs). The PHY/MAC layers provided by the 802.15.4 standard facilitate network formation, management, node addressing and transmission scheduling among wireless nodes by concatenating extra headers with outgoing data frames [2]. Although the preamble and start frame delimiter segments transmitted by the PHY layer are not headers containing data in the typical sense, they require a finite amount of airtime for every frame's transmission so they shall be considered part of the header for simplicity. The IEEE 802.15.4 standard constrains the maximum packet size to 133 octets by dictating the maximum physical service data unit length (largest data unit handed down from MAC) is equal to 127 (`aMaxPHYPacketSize`) octets. The remaining 6 octets correspond to the overheard of the preamble, start frame delimiter and frame length field that are prepended to the packet. In total, PHY/MAC headers occupy 17 octets of overhead in each data frame of the format depicted in Figure 1. Overhead added by the MAC layer is subject to change due to variable length addressing functionality. This occurrence is discussed at detail in Section III.

ZigBee implementations add three additional headers to the outgoing data frames to perform the following services. The ZigBee network (NWK) layer provides services for devices to join and leave a network, apply security to data frames and discover and maintain routes between devices. The ZigBee application services (APS) layer provides functionality necessary for devices to maintain bindings, which are device groupings based upon application communication needs. Finally, the ZigBee application framework (AF) layer identifies a device's potential services as dictated by a given AF profile. In total, ZigBee headers occupy 15 octets of overhead for each data frame. The complete IEEE 802.15.4 and ZigBee frame structures are depicted in Figure 1.

III. Payload Limitation in IEEE 802.15.4-2003

The IEEE 802.15.4 standard incorporates two device addressing modes, commonly referred to as short addressing and long addressing. Long addressing consists of a 64-bit hardware address unique to every 802.15.4 radio transceiver while short addressing is 16 bits and assigned to a node after joining a given network. Since the IEEE 802.15.4 standard is for low data rate PANs, short addresses are favored. To accommodate addressing by means of short or long address the MAC layer header designates address fields to be of 0, 2, or 8 octets in length. Once a short address has been assigned, the IEEE 802.15.4 standard specifies that if the source's short address is known when sending a packet, the sender shall stamp the packet using its short address in preference to the long address [2]. Similarly, the same applies when addressing a packet with the destination's address. Thus in a ZigBee network where short addresses are known, 4 octets (2 16-bit short addresses) out of a maximum of 16 octets (2 64-bit long addresses) are utilized by the MAC header for device addressing.

In certain instances, a device is a member of multiple PANs or multiple PANs operate within the same region. IEEE 802.15.4 MAC layer accommodates this through two specified fields: source PAN ID and destination PAN ID. Collectively, these allow devices to reject messages that are not of local interest. Each of these 16-bit fields may be optional. Specifically, the IEEE 802.15.4 standard designates that if the source and destination PAN ID are equal and the PAN ID compression sub-field within the frame control field of the MAC header is set then the source PAN ID is dropped and only the destination PAN ID is present. Thus, MAC header length varies. In a single PAN ZigBee network only the destination PAN ID is in outgoing packets, thus two out of a maximum of four octets of overhead are utilized.

The maximum MAC header length is 25 octets denoted by the constant `aMaxMACFrameOverhead`. There is no formal definition of minimum overhead within the 2003 standard. The maximum supported MAC layer payload is defined by `aMaxMACFrameSize` in the following formula:

`aMaxMACFrameSize = aMaxPHYPacketSize – aMaxMACFrameOverhead = 127 – 25 = 102.`

This definition is significant, as it restricts the available room for payload within a packet to 102 octets by allocating 25 octets to fields within the MAC header, regardless of whether those fields are used or not. For the typical ZigBee packet described in Figure 1, 11 of the maximum 25 (`aMaxMACFrameOverhead`) MAC header octets are used, leaving the remaining 14 octets to be unusable for application data. Thus, the actual packet transmitted over the air (including preamble, start frame delimiter and frame length field) is 119 octets long instead of 133 octets. For efficiency, the largest packet must be transmitted. Taking into account ZigBee header lengths (15 octets), the available space for application payload is limited to 87 octets (`aMaxMACFrameSize` – 15) compared to the desired 101 octets as determined by maximum physical transmission unit length.

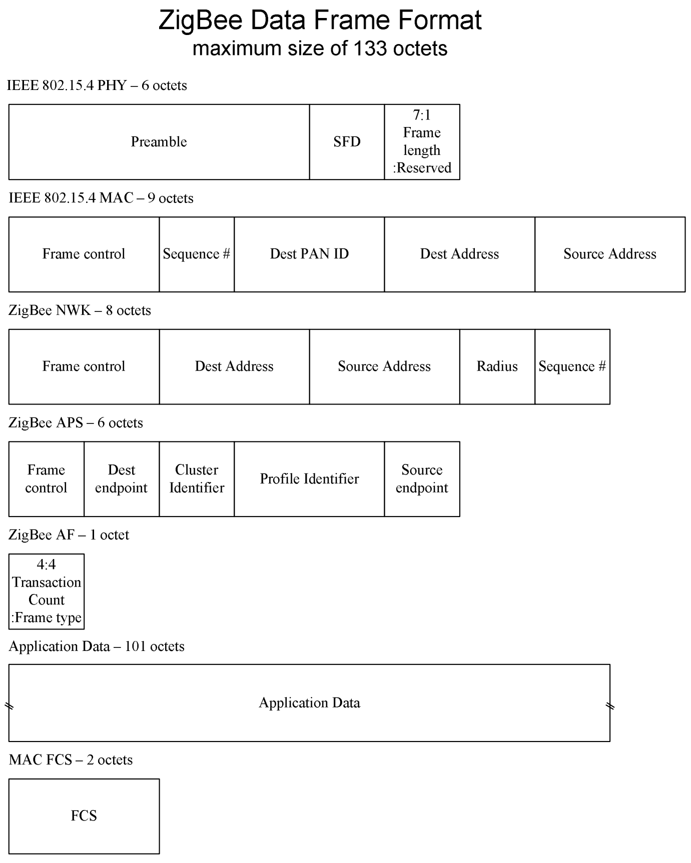


Figure 1. ZigBee/IEEE 802.15.4 frame formats

In September of 2006 a revision to the IEEE 802.15.4 standard, IEEE 802.15.4-2006, was published by the Institute of Electrical and Electronics Engineers [3]. IEEE 802.15.4-2006 replaced the `aMaxMACFrameSize` variable with a new pair of variables described below:

```
aMaxMACPayloadSize=aMaxPHYPacketSize – aMinMPDUOverhead = 127 – 9 = 118
aMaxMACSafePayloadSize=aMaxPHYPacketSize-aMaxMPDUUnsecuredOverhead = 127–25 = 102.
```

The new pair of variables gives application developers the opportunity to recover the octets unused by MAC layer at the expense of backwards compatibility with prior implementations. The first variable, `aMaxMACPayloadSize`, recognizes the minimum MAC header size and only allocates that many octets to the header, leaving 118 octets to the MAC payload. This allows for the creation of maximal length physical service data frames of 133 octets regardless of the addressing fields used. The second variable `aMaxMACSafePayloadSize` holds the original 102-octet value of `aMaxMACFrameSize` for the purpose of notifying developers that MAC payloads of length greater than 102 octets may not be handled properly by systems strictly compliant to IEEE 802.15.4-2003.

The ZigBee Alliance has recognized the addressing flexibility presented in IEEE 802.15.4-2006 and incorporated these changes into the latest version of the ZigBee standard released December 1, 2006 [4].

IV. Practical Estimation of ZigBee Throughput

Figure 2 assimilates several procedures from the IEEE 802.15.4 standard specification to illustrate how a node must send large (> 18 octets) packets when an ack is required. The first procedure, carrier sense multiple access with collision avoidance (CSMA-CA), dictates how 802.15.4 devices shall gain access to the wireless channel. The length of time required to execute CSMA-CA and the probability of CSMA-CA terminating without granting channel access increase with the activity level. The remaining steps of a packet's transmission are executed if the CSMA-CA procedure grants channel access and are assumed to be error free, so they cannot have variable transmission times for a given packet. Throughput calculations first estimate the average time required to execute CSMA-CA then calculate the time required to complete the remaining four steps of the transmission process for a packet of a given length. Practical estimation of throughput in a ZigBee network is governed by the following assumptions:

- *There are no transmission errors resulting in a corrupted or lost data packet.*
- *Each data packet requests acknowledgement upon reception.*
- *The packet structure and field sizes under analysis are as defined in Figure 1. Specifically, the application payload,* `AppPayload`*, is equal to 101 bytes.*
- *The destination node is a single hop away; routing tables are precompiled.*
- *Time is modeled from the radio's perspective in symbols, which can be converted to seconds by dividing by the radio symbol rate of 62,500 symbols per second.*

A. *Transmission Phase 1 (CSMA-CA)*

Within the MAC layer, the CSMA-CA procedure dictates how IEEE 802.15.4 devices gain transmit access to a wireless channel by first listening to determine if another device is currently transmitting. Figure 3 depicts the unslotted portion of the CSMA-CA algorithm, used for non-beacon enabled PANs,

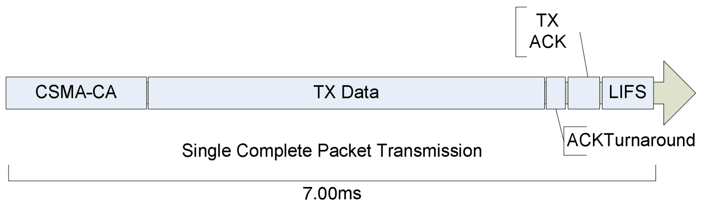


Figure 2. IEEE 802.15.4 packet transmission procedure with estimated duration for `P`$_{\texttt{inactive}}$ `= 0.9.`

as described by the IEEE 802.15.4 standard. Steps (2) and (3) require variable amounts of time to execute depending upon channel activity. The value of `BE` has upper and lower bounds specified to be `macMinBE` (3) to `macMaxBE` (default 5) by the IEEE 802.15.4 standard. The maximum number of iterations to gain channel access is also defined as `macMaxCSMABackoffs` (default 4). These bounds allow the CSMA-CA procedure's correlation to a packet's transmission to be modeled as the Markov chain in Figure 4.

If the algorithm does not gain access to the channel within `macMaxCSMABackoffs` attempts, a channel access failure is declared and the transmission is cancelled. Traversing Markov chain from the 'TX Request' start state to the 'Access Failure' final state yields the probability of a channel access failure. For demonstration, this paper shall set $P_{inactive}$ to be 0.9 for throughput calculations.

$$P_{access\_failure} = (1 - P_{inactive})^{macMaxCSMABackoffs} = (0.1)^4 = 0.0001$$

With the exception of 'Access Failure', each CSMA-CA state has a nonzero symbol time. For this estimation we shall consider the average case to obtain results relevant to practical implementations. Average delay for a given state in the CSMA-CA procedure can be calculated by multiplying the probability of entering the state and the time penalty incurred for visiting that state. Average delay for executing an iteration of the CSMA-CA procedure is the sum of the average delays for each state. The computation process is performed in the following example:

The average symbol time for a given CCA stage, CCA_i is defined as,

$symbols_{CCAi}$ = $backoff\_duration_i$ + cca_duration = $backoff\_duration_i$ + 8 symbols.
From step (2), let $X_i = random(2^{BEi} - 1)$ then,
$backoff\_duration_i = E(X_i) * aUnitBackoffPeriod = E(X_i) * 20$ symbols.

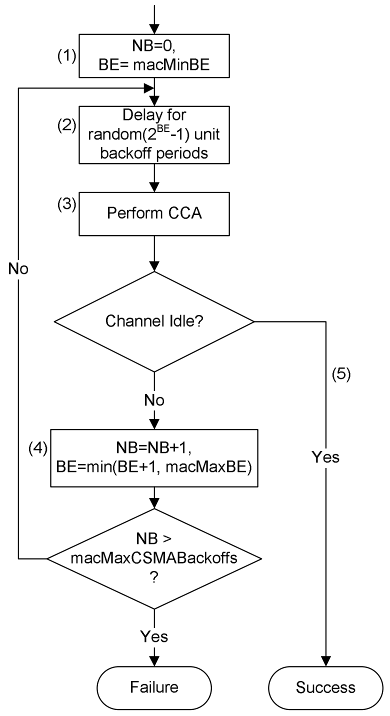


Figure 3. IEEE 802.15.4 CSMA-CA procedure [3].

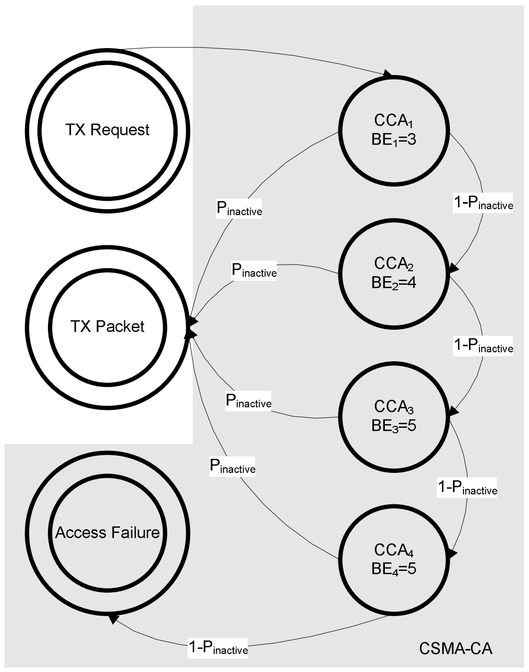


Figure 4. Finite state machine, based upon a Markov chain, representing the relation of CSMA-CA to the packet transmit process.

Then for a given CCA_i, according to the random function, X_i is a random variable of standard uniform distribution with parameter $b = 2^{BEi} - 1$. Therefore,

$E(X_i) = (2^{BEi} - 1) / 2$.

The CCA durations are calculated by these means with the following results:

$symbols_{CCA1} = 78$ symbols
$symbols_{CCA2} = 158$ symbols
$symbols_{CCA3} = 318$ symbols
$symbols_{CCA4} = 318$ symbols.

The symbol time required to traverse the CSMA-CA algorithm and in turn *attempt* to transmit a single packet can be computed given the probability, $P_{inactive}$, of declaring the channel to be idle for a given CCA operation and the symbol time, $symbols_{TX\text{-}Phase2}$, required to complete the remaining transmission steps of: TX Packet, ACKTurnaround, TX ACK and IFS. Therefore, by traversing the state machine transitions, the average time to complete one iteration, of unknown outcome, of the transmission process can be calculated as:

$symbols_{iteration} = CCA_1 + (1-P_{inactive})*(CCA_2 + (1-P_{inactive})*(CCA_3 + (1-P_{inactive})*CCA4)) + symbols_{TX\text{-}Phase2}*P_{access\_granted}$.

The number of iterations required for an 'access granted' outcome is modeled as the number of Bernoulli trials required to obtain a 'success' outcome, which is a geometric distribution. The geometric distribution requires one parameter, the probability of a success for a given trial, which shall be $P_{access\_granted}$. Therefore, based on the properties of the geometric distribution, the expected number of iterations required to gain channel access is,

$P_{access\_failure} = (1-P_{inactive})^4$
$P_{access\_granted} = 1 - P_{access\_failure} = 1 - (1-P_{inactive})^4 = 1 – (0.1)^4 = 0.9999$

$E(iteration\_count) = 1/P_{access\_granted}$.

Finally, leaving the average symbol time required for a successful transmission of a single packet,

$symbols_{transmit} = symbols_{iteration} * E(iteration\_count) = symbols_{iteration}/P_{access\_granted}$
$symbols_{transmit} = (CCA_1 + (1-P_{inactive})*(CCA_2 + (1-P_{inactive})*(CCA_3 + (1-P_{inactive})*CCA4)))/P_{access\_granted} + symbols_{TX\text{-}Phase2}$.

This formula can be separated to represent the two distinct transmission phases with the introduction of a new variable, $symbols_{CSMA\text{-}CA}$.

$symbols_{CSMA\text{-}CA} = (CCA_1 + (1-P_{inactive})*(CCA_2 + (1-P_{inactive})*(CCA_3 + (1-P_{inactive})*CCA4)))/P_{access\_granted}$

$symbols_{transmit} = symbols_{CSMA\text{-}CA} + symbols_{TX\text{-}Phase2}$.

For $P_{inactive} = 0.9$, $symbols_{CSMA\text{-}CA}$ evaluates to,

$symbols_{CSMA\text{-}CA} = (78 + (0.1)*(158 + (0.1)*(318 + (0.1)*318)))/0.9999 = 97.31$ symbols.

B. *Transmission Phase 2 (TX Packet)*

The second phase of packet transmission is conditionally executed if the CSMA-CA procedure grants channel access. Assuming that no data or acknowledgment packets are lost, the time to execute this phase of transmission varies only with respect to the data packet's size. The largest packet size permitted within 802.15.4 will afford the highest throughput to overhead ratio so each data packet is assumed to contain 101 application data bytes plus 32 bytes overhead (133 bytes). The total time required to execute phase 2 is the sum of its components:

$symbols_{TX\text{-}Phase2} = symbols_{tx} + symbols_{turnaround} + symbols_{tx\text{-}ACK} + symbols_{IFS}$.

1) *Transmission Time (TX Data)*

The physical layer transfers packets after assembling them into the PPDU structure. Thus, the physical layer concatenates 6 additional octets for a total of 133 octets to be transmitted. The actual symbol time required for transmission is:

$\mathtt{length_{PPDU}} = 133$ bytes
$\mathtt{length_{symbol}} = 4$ (bits/symbol)
$\mathtt{symbols_{tx}} = \mathtt{length_{PPDU}}$ (bytes) * 8 (bits/byte) / $\mathtt{length_{symbol}}$ (bits/symbol)
$\mathtt{symbols_{tx}} = 133*8/4 = 266$ symbols.

2) *Turnaround Time (ACKTurnaround)*

After the last octet of a packet is received at the destination, the IEEE 802.15.4 standard specifies that a node shall require no more than `aTurnaroundTime` symbols to switch the RF transceiver from receive to transmit mode to transmit an acknowledgement. Thus for the practical case,

$\mathtt{symbols_{turnaround}} = \mathtt{aTurnaroundTime} = 12$ symbols.

3) *Acknowledgment Transmission Time (TX ACK)*

The transmission time required for an acknowledgement frame is calculated similarly to that of the data frame calculated two steps earlier. The one modification of the procedure is the reduction of $\mathtt{length_{PPDU}}$ to 11 bytes as specified by standard acknowledgment frame format of 5 bytes belonging to MAC headers and 6 bytes belonging to PHY headers.

$\mathtt{symbols_{tx\text{-}ACK}} = 11*8/4 = 22$ symbols.

4) *Interframe Spacing Time (LIFS)*

Since a finite amount of processing time is required for the MAC sublayer to complete a packet's reception, the IEEE 802.15.4 standard has designated an interframe spacing (IFS) period that follows the transmission of a data frame. The IFS period can take a minimum value designated by one of two constants, `aMinLIFSPeriod` or `aMinSIFSPeriod` according to the following statement:

```
If $\mathtt{length_{MPDU}} \le$ aMaxSIFSFrameSize
then,
$\mathtt{symbols_{IFS}} \ge$ aMinSIFSPeriod = 12 symbols
else,
$\mathtt{symbols_{IFS}} \ge$ aMinLIFSPeriod = 40 symbols.
```

For the scenario under consideration,

$\mathtt{symbols_{IFS}} = \mathtt{aMinLIFSPeriod} = 40$ symbols.

C. *Total Transmission Time*

Given the calculations completed thus far, the average time required for transmission of a single packet in a lightly loaded network ($\mathtt{P_{inactive}} = 0.9$) is computed in Table IV..C. The number of times that the transmission process can be repeated within a second is easily computed given the average transmission symbol time, $\mathtt{symbols_{sum}}$. Finally, throughput is computed given the average number of packets transmitted per second.

TABLE I. PACKET TRANSMISSION TIME BREAKDOWN

Procedure	**Identifier**	**Symbols**	**Milliseconds**
CSMA-CA	$\text{symbols}_{\text{CSMA-CA}}$	97.31	1.56
TX Packet	$\text{symbols}_{\text{tx}}$	266	4.26
ACKTurnaround	$\text{symbols}_{\text{turnaround}}$	12	0.19
TX ACK	$\text{symbols}_{\text{tx-ACK}}$	22	0.35
LIFS	$\text{symbols}_{\text{IFS}}$	40	0.64
Sum	$\mathbf{symbols_{sum}}$	**437.31**	**7.00**

$\text{packets}_{sec} = \text{rate}_{symbol} / \text{symbols}_{sum} = 62{,}500/437.31 = 142.92$ (packets/s)
throughput_{kbps} = AppPayload (bytes/packet) * 8 (bits/byte) * packets_{sec} (packets/s)
$\text{throughput}_{kbps} = 101*8*142.92 = 115.5$kbps

Thus maximum throughput for single hop transmission in a lightly loaded, non-beacon enabled PAN is approximately 115.5kbps. This includes allowances for overhead created by ZigBee packet headers; therefore, it is an appropriate estimate for an upperbound of throughput in a ZigBee wireless network.

D. *Effect of Channel Activity on Throughput*

Our throughput estimation method can be performed for any channel activity level as depicted in Figure 5, which displays a set of curves representing the degradation in maximum throughput with increased channel activity. The accuracy of such predictions is evaluated using NS-2 simulations in Section V. *It can be observed, from Figure 5, that maximum throughput reaches approximately 120kbps for IEEE 802.15.4-2006 acknowledged transmissions when $P_{inactive}$ approaches 1.*

V. SIMULATED ZIGBEE THROUGHPUT

Version 2.28 or later of the network simulator NS-2 is equipped with an IEEE 802.15.4 PHY/MAC simulation module provided by Zheng and Lee [5][6]. Ramachandran [7] has published a set of modifications and accuracy improvements for the IEEE 802.15.4 module in NS-2, which are also applied to the experimental platform. This module can be used to simulate throughput in a ZigBee network because for single hop throughput purposes, ZigBee is merely a set of headers placed above the application payload in IEEE 802.15.4 data packets.

Verification of the simulation platform indicated that the IFS period following a packet's transmission was not implemented at the transmitting node as per IEEE 802.15.4. The PHY/MAC layers from Zheng and Lee implement the IFS period at the receiver's MAC layer as a delay between the reception of the packet and the receipt notification to the upper layer. This implementation of the delay is consistent with the reason for incorporating an IFS delay into the standard. However, this delay does not enforce the condition imposed by standard to prevent nodes from overloading their neighbors. Our simulations include the insertion of the appropriate IFS delay between successive transmissions from a single node.

The network topology used for the simulation consists of two nodes (labeled {0} and {1}) separated by a distance of 25 meters. The DumbAgent routing protocol is specified because only single hop transmissions are considered. Node {0} is started first as non-beaconing PAN coordinator, followed by node {1} as a full function device. At time 30 (seconds) node {1} constructs a constant bit rate (CBR) traffic flow to node {0}.

CBR traffic in NS-2 requires two parameters, packet size and packet rate. The packet size designated for the CBR flow is 116 bytes. (101 bytes of application payload plus 15 bytes of ZigBee headers; see Figure 1.) To ensure that maximum throughput is achieved, the packet generation rate is set to be higher than what IEEE 802.15.4 can service. The simulated link layer buffers packets that cannot be transmitted immediately. When the link layer queue at node {1} is full, no outgoing packets are accepted and attempts to schedule new packets are dropped until a pending packet transmission is completed. The packet rate set for simulation is a single packet every 3.0 milliseconds, less than half of the estimated 7.0 milliseconds required for complete transmission. This packet rate is selected for its ability to expose potential flaws in the throughput estimation technique that could result in a significantly lower throughput estimation than actual capabilities.

The simulation is halted after 90 seconds of CBR activity. Throughput is obtained by parsing the trace file with two procedures. First dropped packets are eliminated. Next, the length of simulated ZigBee headers is subtracted from the CBR packet sizes to determine actual application payload in each packet. Cumulative application payload transferred is then counted by procedure two. *Throughput for a single* *hop, non-beacon enabled wireless network reached a maximum of 120kbps, matching the estimation derived in Section IV.* Throughput for various channel activity levels is simulated by modifying the CCA procedure to return a true result only when the channel is detected to be idle and a probabilistic condition associated with the desired channel activity level ($P_{inactive}$) is satisfied. Overall accuracy of the simulation platform and our estimations is evaluated by comparing the graphs, which depict throughput versus channel activity level, obtained from estimations in Figure 5 to the graphs obtained from NS-2 simulations in Figure 6. Figure 6 contains measured/calculated data points at the 10% intervals for $P_{inactive}$. Indistinguishable results testify to the accuracy of the two devised throughput measurement techniques and the feasibility of such throughput levels.

VI. ZigBee Hardware Throughput

In Phase 3, we measure maximum attainable throughput using ZigBee compliant hardware. The prospects for coaxing hardware to reach maximum throughput in excess of 120kbps appear to be slim given the findings presented by Ashton [8]. Ashton sought to expose a baseline for performance in a ZigBee network by considering the correlations between the number of hops, latency and throughput. Although Ashton's results are specific to the following platform, the experiment is indicative of ZigBee's performance because the platform is fully compatible with the ZigBee standard. Ashton measured 46kbps to be the maximum single hop throughput using the following assumptions and experimental setup:

- Software application is not specially configured for purposes of throughput measurement.
- Application payload is equal to 91 bytes.
- Ember Corporation EM250 System on Chip (SoC), integrated radio and 16-bit microcontroller chip, devices realize ZigBee hardware platform.
- No security (encryption) is in use.

A. *UTD Test Model*

In disjunction with previous tests, our hardware implementation assumes the following:

- Software application can be specially configured for purposes of throughput measurement.
- Application payload is equal to 101 bytes.
- Ember Corporation EM2420 ZigBee compliant radio transceiver is paired with Atmel AVR based ATmega128L 8-bit microcontroller provided by EM2420 developer kit, which is based on a functionally separated two-chip design (Figure 8).
- No security (encryption) is in use.
- The non-beaconing, mesh based, IEEE 802.15.4 communication mode is utilized.

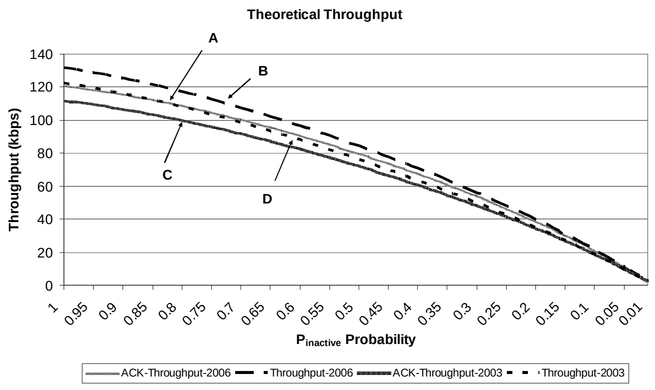


Figure 5. Pratical estimation of maximum throughput of ZigBee packets versus channel activity level for: A) IEEE 802.15.4-2006 with MAC ACK, B) IEEE 802.15.4-2006 no MAC ACK, C) IEEE 802.15.4-2003 with MAC ACK, D) IEEE 802.15.4-2003 no MAC ACK.

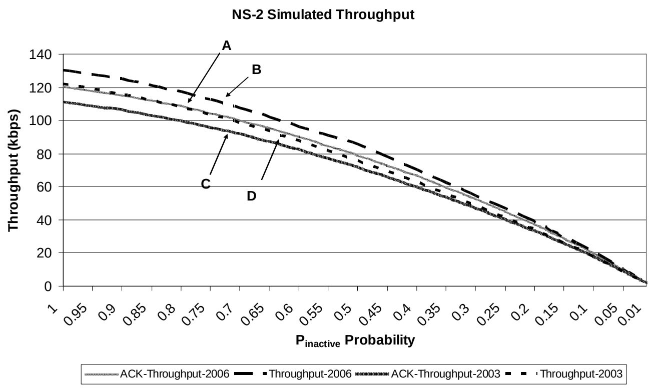


Figure 6. NS-2 simulated maximum throughput of ZigBee packets versus channel activity level for: A) IEEE 802.15.4-2006 with MAC ACK, B) IEEE 802.15.4-2006 no MAC ACK, C) IEEE 802.15.4-2003 with MAC ACK, D) IEEE 802.15.4-2003 no MAC ACK.

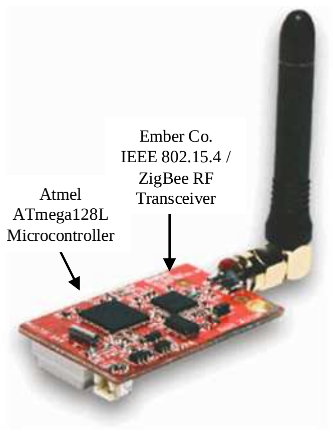

Ember Co.
IEEE 802.15.4 / ZigBee RF Transceiver
Atmel ATmega128L Microcontroller

Figure 8. Ember EM2420 based radio communication module used for hardware throughput measurements.

In notable contrast between the two models, the first assumption for our model is valuable because enhancements observed by refining software configurations can provide insight essential for designing better performing systems. The procedure written to test maximum throughput contains five steps illustrated by the flowchart in Figure 7. *Initial ZigBee compliant hardware measurements of throughput using ZigBee messages yield a result of 53kbps.* At this point, it is certain that radio airtime is not being utilized to its fullest potential as described by practical estimations. Software refinements are necessary to enhance throughput.

B. *Refinement 1: Remove excess functions*

First, we check if all non-transmission related sources of delay can be eliminated. To eliminate these factors, steps (1), (2), (3) and (5) need only to be executed once for a given test session. Step (4) shall remain in a loop repeatedly scheduling the same packet buffer for transmission. In this configuration, no processing time is spent allocating, filling or deallocating space for packets. *This modification resulted in a nearly 2x increase to 110kbps.* The throughput increase seems to indicate that at least one of the isolated steps is consuming a large quantity of processing time. Step (2) is easily eliminated from the possible causes by only running it once per execution while leaving (1), (3), (4) and (5) in the throughput loop. The result of this test was equal to that of the original 53kbps. Thus, step (2) is not the bottleneck. Other combinations of removed steps cannot be tested because of finite memory limitations. The 110kbps throughput obtained using this refinement is considered the standard as the highest throughput that can be achieved using this hardware platform in the given experimental environment. The 10kbps gap between estimated throughput and this experimental value is projected to be due to a combination of environmental factors (noise, interference, etc.) and intrinsic hardware platform limitations (processor/RF transceiver latency, etc.). The origins of possible hardware platform limitations are investigated in the remaining refinements.

C. *Refinement 2: Decrease interrupt service latency*

On completion of a transmission, the Ember EM2420 radio notifies the AVR microcontroller of the event with an interrupt. The interrupt service routine is designed to quickly set a flag indicating the change in state and then resume user application. The user application is responsible for calling an interrupt management procedure that checks the status of various flags to perform actions based upon recent interrupts. In the case of a 'transmission complete interrupt', the radio interrupts the AVR and a corresponding flag is set. Upon the next application call of the interrupt management procedure, the status of the completed transmission is relayed to the application and scheduling buffers are processed to begin the next packet transmission. Previous throughput measurements had one such call to the interrupt management procedure per throughput loop iteration. This refinement measures throughput improvement by calling the

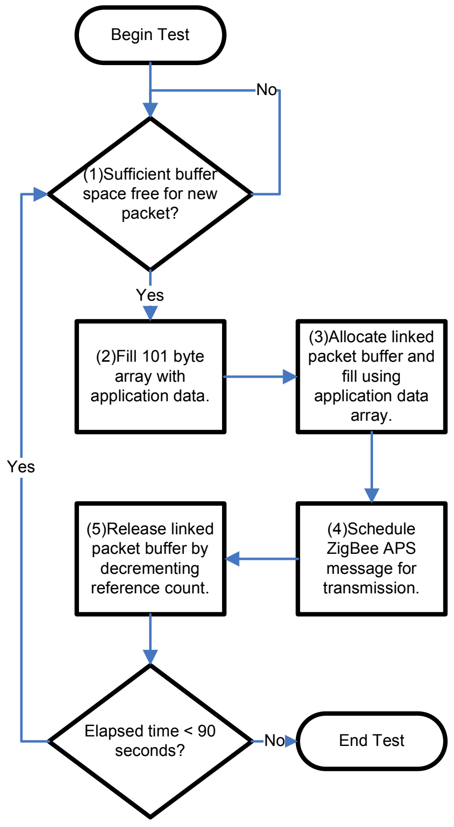

Begin Test
No
(1)Sufficient buffer space free for new packet?
Yes
(2)Fill 101 byte array with application data.
(3)Allocate linked packet buffer and fill using application data array.
Yes
(5)Release linked packet buffer by decrementing reference count.
(4)Schedule ZigBee APS message for transmission.
Elapsed time < 90 seconds?
No
End Test

Figure 7. Throughput measurement data send routine.

interrupt manager two or more times per throughput loop iteration. *A maximum throughput of 80kbps occurred with two interrupt manager calls per loop iteration.* This refinement is highly desirable because it produces a significant throughput improvement and is minimally invasive to the application. However, refinement 2 falls well short of the previously discovered throughput maximums. ZigBee does not dictate the manner in which interrupts shall be handled so this discussion is based upon our ZigBee compliant hardware platform. However, the lessons learned from this refinement are needed to understand the effectiveness of refinement 3 and provide useful insight to the design of throughput efficient wireless implementation.

D. *Refinement 3: Increase available packet buffers*

Refinement 3 is invasive to the application to gain an understanding of the bottleneck that limits throughput to 80kbps for unique packets while allowing 110kbps for repeated transmission of the same packet. In the Ember implementation of the ZigBee stack, packet buffers consist of a series of linked 32-byte buffers. Packet buffers that are large enough to store an entire packet are created in step (3) by linking the necessary number of link buffers in a linked list like structure. Relatively tight memory constraints present in embedded microcontrollers dictate strict limits on the number of link buffers that can be allocated. A packet of maximum size requires five link buffers for packet storage plus overhead. The default configurations used in previous tests allocate 24 32-byte linked buffers, thereby buffering a maximum of 4 packets. In this configuration, step (1) of the throughput loop may find insufficient memory space to create and schedule a new packet for transmission.

Refinement 3 increases the buffer count to 45 32-byte buffers, therefore holding a maximum of 9 packets, at which point step (1) empirically yields all ‘yes’ results. The desired outcome is for the application to schedule more packets simultaneously, reducing the probability that the radio will have idle time. Refinement 3 is applied in addition to refinement 2. *Increasing the number of available link buffers drastically increases application throughput to 108kbps.* This refinement nearly matches the effectiveness of refinement 1, which allocated a single static packet buffer. Thus, four completely buffered packets are insufficient to prevent idling of the radio due to latency in deallocating packet buffers. This problem is explored next.

1) *Buffer allocation/deallocation process*

Each packet buffer is allocated using a reference counter initially set to one by step (3). Step (4) increments the reference count to indicate that the buffer is needed by the radio. Step (5) then decrements the reference count to signal that the application is finished with the packet buffer. At this point, the reference count is greater than zero so the buffer is unavailable for reallocation. Upon completion of the transmission, the radio issues an interrupt to the microcontroller. The microcontroller interrupt service routine sets a transmission complete flag. On calling the interrupt management routine, the application is notified, via callback function, of the completed packet transmission and is passed a reference to the original packet buffer. The application may perform additional processing now. On returning from the application callback the interrupt management routine decrements the reference count of the packet buffer to zero. The individual links of the packet buffer are now available for subsequent allocation. This process shows that packet buffers are not freed immediately after transferring the packet to the radio for transmission. The buffers are freed at the end of an application call to the interrupt management routine, which bears no direct correlation to the time at which transmission is completed.

2) *Solutions*

Refinement 3 is an undesirable long-term solution. The 45 buffers with overheard occupy nearly 45% of the AVR’s 4KB internal data memory. Three solutions are proposed to alleviate the effects of the packet deallocation bottleneck.

1) Tightly couple packet deallocation with the interrupt service routine responsible for servicing interrupts from the radio. Immediately upon receiving an ‘acknowledgment received’ or ‘transmission complete’ interrupt, deallocate the packet buffer.

2) Create additional packet buffering capabilities at the radio transceiver. The application may not require access to a packet’s contents after it schedules the packet for transmission. It would be advantageous for the radio to autonomously handle all retransmission attempts. With this, the microcontroller can deallocate a packet buffer as soon as a copy is transferred to the radio.

3) Add an external RAM to the hardware design. A small external RAM will add little cost to hardware designs while providing a large amount of space suitable for storing packets.

E. *Delivery order*

Since refinements 2 and 3 result in maximum throughput, these refinements are applied as the standard system. Unique sequence numbers are placed in the first two bytes of payload for every data packet in the throughput measurement. These sequence numbers provide a means for detecting out of order delivery and failed delivery of packets when repeating throughput tests. *Subsequent tests indicated that for the network model in this paper all scheduled packets arrived at the destination, in order.* Repeated experiments reliably produced the same result.

VII. SUMMARY AND CONCLUSIONS

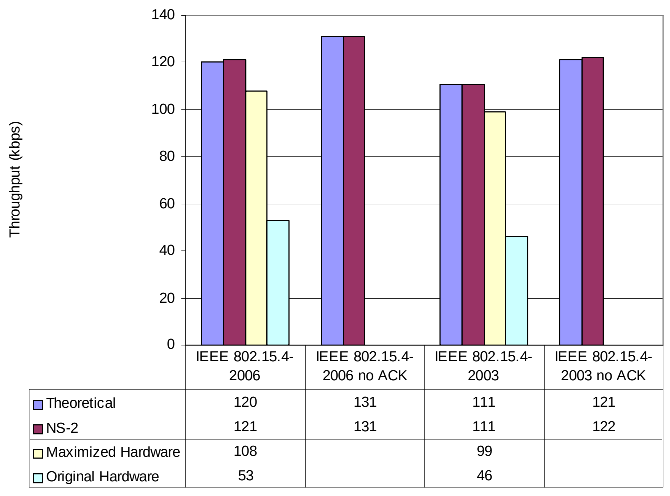
We limited our study to two ZigBee devices. The network may now be expanded to many participants to determine 1) the effect of multi-hop paths on throughput, 2) reliability in a congested network, 3) delivery order, 4) latency and 5) maximum network throughput. Our results show that these parameters can be accurately measured using a properly programmed hardware platform or by simulation.

A final summary of all measurable throughput performance characteristics is provided for comparison in Figure 9. The first and third columns display clear throughput improvements achieved using off the shelf hardware designs paired with software customizations. The equality of estimated versus simulated throughput measurements across all four scenarios argues for both techniques’ accuracy. Experiments indicate that the maximum potential throughput has been reached by implementing the techniques described in this paper. Other designs stand to increase throughput and benefit from this work by acknowledging the critical design criteria as observed within this paper and consisting of interrupt service times, available memory and processor-transceiver communication latency.

The value of these experiments comes from establishing an analytical and empirically verified upperbound for actual application throughput of a ZigBee network. The theoretical foundations presented here provide a means for system designers to evaluate their design with regards to its fulfillment of ZigBee’s maximum potential.

ACKNOWLEDGEMENT

We thank our colleagues Maulin Patel and Qasim Javed for their valuable input regarding ZigBee, IEEE 802.15.4 and techniques for utilizing Ember Corporation ZigBee development products.



	IEEE 802.15.4-2006	IEEE 802.15.4-2006 no ACK	IEEE 802.15.4-2003	IEEE 802.15.4-2003 no ACK
■ Theoretical	120	131	111	121
■ NS-2	121	131	111	122
□ Maximized Hardware	108		99	
□ Original Hardware	53		46	

Figure 9. ZigBee throughput maximums obtained from various measurement platforms and improvements.

REFERENCES

[1] ZigBee Alliance Document 053474r06: ZigBee Specification, December 2004.

[2] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2003, IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2003.

[3] Institute of Electrical and Electronics Engineers, Inc., IEEE Std. 802.15.4-2006, IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). New York: IEEE Press. 2006.

[4] ZigBee Alliance Document 053474r13: ZigBee Specification, December 2006.

[5] Network Simulator 2 – NS2, http://www.isi.edu/nsnam/ns/.

[6] J. Zheng and M. J. Lee. (2001, April 1, 2001). Low rate wireless personal area networks (LR-WPANs), NS2 simulation platform. *2005(10/19).* Available: http://ees2cy.engr.ccny.cuny.edu/zheng/pub/

[7] I. Ramachandran. (2006, 02/07/2006). Changes made to the IEEE 802.15.4 NS-2 implementation. *2006(10/19)*. Available: http://www.ee.washington.edu/research/funlab/802_15_4/ns2_changes.pdf;http://www.ee.washington.edu/research/funlab/802_15_4/changed_files

[8] S. Ashton, Ember Corporation. Zigbee network performance, typical results and implications for application design. Presented at ZigBee Developers' Conference 2006.

[9] Ember Corporation. (2006, 06/29/2006). EM250 radio communication module, technical specification. *2006(10/19)*. Available: http://www.ember.com/pdf/EM250/120-2001-000C_EM250_RCM.pdf

[10] Ember Corporation. (2004, 07/21/2004). EM2420, 2.4 GHz IEEE 802.15.4 / ZigBee RF transceiver, technical specification. *2006(10/19)*. Available: http://www.ember.com/pdf/EM2420datasheet.pdf