Noname manuscript No.
(will be inserted by the editor)

iFrag: Interference-Aware Frame Fragmentation Scheme for Wireless Sensor Networks

Ahmad Showail · Amr Elrasad · Ammar Meer · Anas Daghistani · Kamran Jamshaid · Basem Shihada

Received: date / Accepted: date

Abstract In wireless sensor networks, data transmission reliability is a fundamental challenge due to several physical constraints such as interference, power consumption, and environmental effects. In current wireless sensor implementations, a single bit error requires retransmitting the entire frame. This incurs extra processing overhead and power consumption, especially for large frames. Frame fragmentation into small blocks with individual error detection codes can reduce the unnecessary retransmission of the correctly received blocks. The optimal block size, however, varies based on the wireless channel conditions. In this paper, we propose an interference-aware frame fragmentation scheme called iFrag. iFrag effectively addresses the challenges associated with dynamic partitioning of blocks. We show through analytical and experimental results that iFrag achieves up to 3× improvement in throughput when the channel condition is noisy, while reduces the delay to 12% compared to other static fragmentation approach. On average, it shows 13% gain in throughput across all channel conditions used in our experiments. This significant improvement is due to dynamic nature of iFrag that minimizes the retransmission overhead by selecting the appropriate number of blocks in each data frame.

Keywords Wireless sensor networks; partial packet recovery; interference

1 Introduction

One fundamental challenge in wireless networks is the trade-off between large and small data frames. Large frames result in better channel capacity utilization, while small frames provide efficient error recovery (*i.e.,* only small fragments have to be retransmitted). Wired transmission mediums have low bit-error rates (BER), typically 10^{-15} to 10^{-12}, and thus embrace strategies utilizing large packet sizes to increase overall throughput. In contrast, the BER in a wireless network is orders of magnitude higher, typically 10^{-5} to 10^{-3}. Additionally, this BER may vary dramatically over short time intervals [1,6,13,19]. This makes it necessary to optimize the trade-off between throughput and error recovery in resource-constrained wireless sensor networks (WSNs).

WSNs typically use small data payloads. A frame transmission incurs additional overhead due to PHY and MAC layer headers. These headers include sender and receiver IDs, CRC for error detection, and other bytes for synchronization. The data link layer is responsible for partitioning the original payload into frames. If channel BER is low, frames with a large payload size can be used to better amortize the PHY and MAC layer overhead, thus improving network throughput. However, if BER is high, large frames are more likely to be corrupted, invoking a retransmission of the entire frame, and thus reducing network throughput. Generally, there is no unique optimal frame size proposed due to unpredictable channel quality [20,4]. In particular, TinyOS data link frames have been standardized to 29 bytes excluding the PHY and MAC layer headers.

As frame retransmission is inefficient, researchers in [8] came up with the concept of frame fragmentation for wireless sensors. The main idea is to divide the frame into several blocks, that are fixed in size, each of which is equipped with its own error detection code and block number. The number of fragments is predetermined and fixed regardless of the channel condition. We show that the overhead of the additional bytes reduces the network throughput when the channel condition is good.

CEMSE Division, KAUST, Saudi Arabia
E-mail: basem.shihada@kaust.edu.sa

Moreover, this approach does not differentiate between loosing data frames and loosing the acknowledgement frame sent by the receiver.

In this paper we are proposing an interference-aware frame fragmentation approach for sensor networks called iFrag. iFrag introduces dynamism to decrease the overhead of retransmissions under normal channel conditions. It dynamically chooses the block size that suits the channel condition observed in a specified time window. At fixed percentage of frame reception, the receiver sends a recovery frame which contains the number of correctly received blocks. Based on this information, the sender selects the optimal block size. As a result, the data link layer works with the optimum block sizes for the current channel condition. Additionally, iFrag solves the issue of unnecessary retransmissions due to the loss of the recovery frame by introducing a timeout at the receiver side. As we show in Section 5, iFrag achieves 3× more throughput and 12% less delay than the static fragmentation approach when the channel condition is noisy. On average, it achieves 13% gain in throughput across all channel conditions used in our experiments.

The idea of dynamic block size selection introduces several unique challenges. First, deciding when to switch between block sizes is challenging. Second, since there is no synchronization between the sender and the receiver, this switching mechanism must satisfy two requirements: mode discovery and data integrity. By mode discovery we mean that the receiver must have a mechanism to discover the change in the block size which is decided by the sender. Moreover, the data sent over various block sizes must be distinguishable for the receiver to preserve data integrity. Third, minimizing the expected overhead of implementing such dynamic mechanism. For iFrag to operate efficiently, its overhead must be less than its actual gain. Having these challenges in mind, iFrag was designed to be able to identify error patterns, identify best mode to choose, enable the receiver to know the received mode, minimize the overhead of dynamism, and preserve data integrity.

iFrag is implemented using TelosB [23] motes. Our experiments show that iFrag significantly boosts network throughput under both normal and high interference conditions. Moreover, iFrag improved the network end-to-end delay significantly especially under noisy channel. The performance of iFrag was evaluated under a realistic interference caused by nearby WiFi devices operating at the same frequency. Furthermore, we believe that iFrag has wider applicability beyond wireless sensor systems, as it could be applied to any other wireless standard.

To summarize, the main contributions of this work are as follows:

1. Introduce an accurate throughput model for iFrag
2. Introduce a multiple data frame retransmission avoidance method, harnessing data frame retransmission by retransmitting the recovery frame.
3. Design, implement, and evaluate a dynamic fine-grained error recovery mechanism that captures the channel condition and adapts the block sizes accordingly.

This paper is organized as follows. In Section 2, we present an executive summary of the related work. In Section 3, iFrag design is introduced. This includes frame structures as well as operation. The proposed model for the system is described in Section 4. In Section 5, the results of the experimental work are presented and thoroughly discussed. Finally the paper is concluded and future work is discussed in Section 6.

2 Related Work

Partial Packet Recovery techniques can be broadly classified into two categories based on the use of physical layer information in the recovery process. In fact, most of the techniques that use physical layer information for packet recovery [29,12,14,27] are both efficient and accurate. However, these require hardware modification to modify the PHY layer which makes them impractical. On the other hand, the other set of techniques do not require modifying the physical layer and hence easier to implement i.e. changes are made only at the MAC layer. However, many of these techniques face two challenges, namely: high computational cost and low prediction probability. We limit our discussion to methods with no physical layer support since these are the ones similar to our proposed work.

Partial Packet Recovery has been thoroughly studied in [25,2,9,22,8,15]. The proposed schemes were based on hard decision channel output that resolve some of the issues of data link layer protocols. In [22], the author proposed ARQ-With-Memory (MRQ) scheme. In this scheme, the receiver XORs erroneous packets to identify any existing errors. Then, the correct packet is retrieved by an exhaustive search to pass the checksum. This scheme is easy to implement, however it suffers from high computational intensity of packet recovery that is exponential with respect to BER and unnecessary retransmission of the entire packet which wastes the bandwidth. The authors in [8] target data link streaming in WSNs through Seda. The goal of Seda was to enhance the robustness and throughput of WSNs. In

Seda, the transmitter divides the packet from the network layer into four small sequenced data blocks which are then combined at the receiver side. For error detection, a 1 byte CRC is appended to each block. If erroneous blocks are received, the receiver initiates a recovery frame, which contains the sequence of the first incorrect block received and a binary block map for the consecutive blocks to request their retransmission. Once all the blocks are received correctly, they are reassembled for higher layers. One of Seda's limitations is the loss of feedback problem *i.e.* when the recovery block is lost. As a reaction, the sender periodically re-sends all the frames after the specified timeout. iFrag solves this issue by periodic sending of the recovery frame which is smaller in size than the data frame and has higher chance of being received correctly. the other limitation of Seda is the static assignment of the block size regardless of the channel condition. The authors of [15] proposed a similar static fragmentation approach for wireless local networks called fragment-based retransmission (FBR). In FBR, corrupted fragments will be given a secondary chance to be transmitted within the same channel access. Although the fact the authors mentioned that either 2 or 4 blocks could be used, there is no clue in the paper on how to favour one of these two schemes over the other. Moreover, it is not clear how the receiver will know the number of fragments sent by the sender to be able to correctly decode the frame. Finally, the extension of the sender transmission chance could degrade the network fairness significantly.

The concept of adaptively changing the size of frame fragments have been proposed in the literature. In [30], an adaptive subpacket scheme that optimizes the block size to maximize throughput is proposed. The size of each block is determined based on the SNR of the channel. However, the authors never mentioned how to inform the receiver about the newly assigned block size. Similar to iFrag, this approach is using block combining, *i.e.* transmitting new blocks along with the retransmission of the corrupted blocks. However, it is not possible to preserve data integrity without having a block number for each one of the blocks. The author of [26] proposed a segment-based retransmission scheme that use Luby-type erasure code to recover missing symbols. This work is based on the assumption that the transmitter has a precise knowledge of the channel BER and hence will select the segment size accordingly. However, this assumption is not feasible. Moreover, the feedback channel is assumed to be error-free and has no delay which is also not realistic. iFrag chooses the size of the block adaptively based on the transmission history. It deals with errors in the feedback channel by periodic sending of the recovery frame. Recently, the authors of [10] suggested to have an adaptive frame fragmentation scheme for wireless local networks (WLANs) named GEB which stands for gathering error-free blocks. The main purpose of GEB is to allow the sender to differentiate frame dropping due to collision from transmission losses due to noise and interference. In the case of the former, the contention window should be doubled where as it should not be affected by the bit errors in the frame. So if any block is received correctly a NAK will be sent to the receiver to inform it not to increase the contention window. In fact, this approach has two limitations. First, the error detection code is duplicated in both the frame level and block level which is an unnecessary overhead. Second, since GEB is not attaching a block number to every block, a partial retransmission would not be possible. Hence, the corrupted frame will be sent repeatedly until it gets received correctly. To the best of our knowledge, iFrag is the first work that implements adaptive partial packet recovery in real hardware and analyzes its performance through experimental evaluation.

Several papers have been studying the feasibility of WiFi-ZigBee coexistence. In [16], the authors proposed a bit level granularity scheme for sensor motes to survive WiFi interference through header and payload redundancy. Basically, the idea is to insert multiple headers at different places in the frame so as to increase the possibility of the header to be correctly received. On similar efforts, researchers in [11] proposed a scheme called WISE that predicts the length of WiFi white space and adjust the size of ZigBee frames accordingly. Other efforts are directed towards optimizing packet size in wireless sensor networks [20] [21] [5] [24]. The main idea is to change the frame size as the physical channel condition changes. We envision that iFrag can be easily integrated with these schemes since all of them do not apply the concept of partial packet recovery.

The idea of packet combining in wireless networks remains an interesting challenge. For example, SPaC [6] uses packet combining on the frame level. However, Seda is different from SpaC in the fact that the latter does not partition the frame into blocks. MRD [19] utilizes multiple access points to achieve packet combining. Basically, it combines the bits received at different motes to correct wireless errors. This work is similar to Seda in the fact that packets from the transmitter is divided into different blocks. Similarly, MRD also uses an error detection mechanism at the block level. Hence, exponential block combining trials might be possible. Moreover, a full retransmission of the whole frame will be the only solution if the combining efforts fail, which is very expensive. In reality, this scheme has less computational cost compared to [8], yet costly to be im-

plemented in the sensor motes with limited resources. Moreover, MRD requires an efficient access point selection which is considered a waste of bandwidth when there is data exchange between all the available access points.

Many other techniques are based on Forward Error Correction (FEC) [18,17,28]. It is well known that the knowledge of the channel BER is required for FEC methods to function. It has been shown in [13,1,19,6] that channels of any asynchronous wireless network suffers from unpredictable and frequent changes in channel condition. In addition, the high overhead of the required computation for FEC makes it impractical for resource constrained sensor motes. Alternatively, iFrag proposes a low computation technique that reduces packet retransmissions while improving network throughput and delay.

3 iFrag Protocol Design

In this section, we introduce the protocol design of iFrag. This includes frame structures, operation, and a discussion of the main challenges addressed in the proposed protocol design.

3.1 Overview

iFrag is a dynamic block size allocation protocol that adapts the block size based on varying channel conditions, leading to lower block loss rates and a significant reduction in block retransmissions. This improves data transmission reliability, resulting in high network throughput. Consider x bits of data to be transmitted between sender and receiver. The bit stream is grouped into one or more frames at the data link layer. Since iFrag was implemented on top of TelosB motes [23] which are equipped with 250 Kbps CC2420 radio [3], the overall frame size is limited by the wireless transceiver firmware to 128 bytes.

3.2 Motivation

Prior work in [8] highlighted the problem of wireless channel conditions variability through a BER measurement study. It proposed a streaming data link layer protocol named Seda, which focuses into de-coupling the trade-off between using large frame sizes to increase throughput and using small frame sizes to achieve effective error recovery. The basic idea is as follows: Seda divides the frame in four equal blocks, where each block

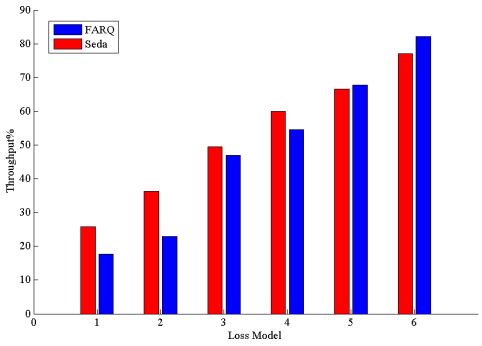


Fig. 1: Throughput of Seda and FARQ under various channel loss models

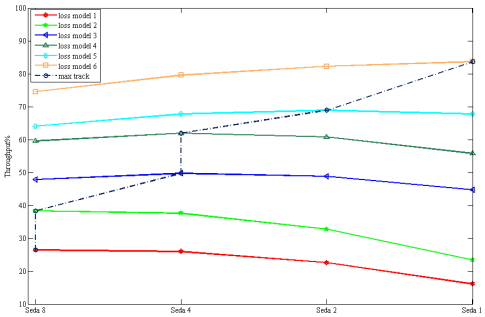


Fig. 2: Throughput of Seda with various block sizes under different channel loss models

has its own sequence number and CRC for error detection. This protocol transmits frames with the maximum size (100 bytes), which is limited by the buffer space of MicaZ platform [23]. Seda overcomes the large frame size problem and prevents retransmitting the whole frame by sending delayed automatic retransmission request frames called recovery frames. These recovery frames indicate the erroneous blocks that need to be retransmitted. However, Seda has several limitations when implemented. First, it has fixed block size of 27 bytes regardless of the channel condition. Second, Seda retransmits all data frames as a reaction to the loss of the recovery frame, which results in high redundancy overhead. Finally, Seda was only tested under situations where sensor motes were interfering with each other and never experimented under conditions where interference is caused by higher-power wireless devices.

In Fig. 1, we show a comparison between Seda and frame-based ARQ (FARQ) under various channel loss models proposed in [8] using MATLAB simulations. The channel BER decreases progressively from loss model 1 to 6, *i.e.*, loss model 1 represents the worst channel conditions, while loss model 6 represents an ideal chan-

nel. Specific channel attributes are described in Section 4. When the channel quality is good, FARQ outperforms Seda in loss models 5 and 6 since it has less overhead per frame, while Seda shows better performance in other loss models. However, in Fig. 2 we empirically evaluate four different Seda implementations, each with its own fixed block size, *e.g.*, Seda 8 uses 8 small blocks within a frame. We observe that Seda 8 outperforms all other Seda implementations (including traditional frame transmission with a block size of 1) under noisy channel conditions (loss models 1 and 2). As the channel conditions improve (loss models 3 to 6), Seda schemes using fewer blocks perform better. This shows that varying block sizes results in better channel utilization under different channel conditions. In the light of these observations, we propose iFrag, a scheme for dynamic partitioning of blocks based on current channel conditions.

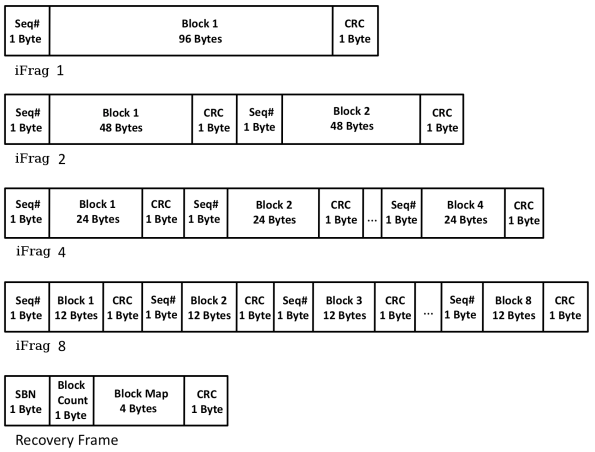
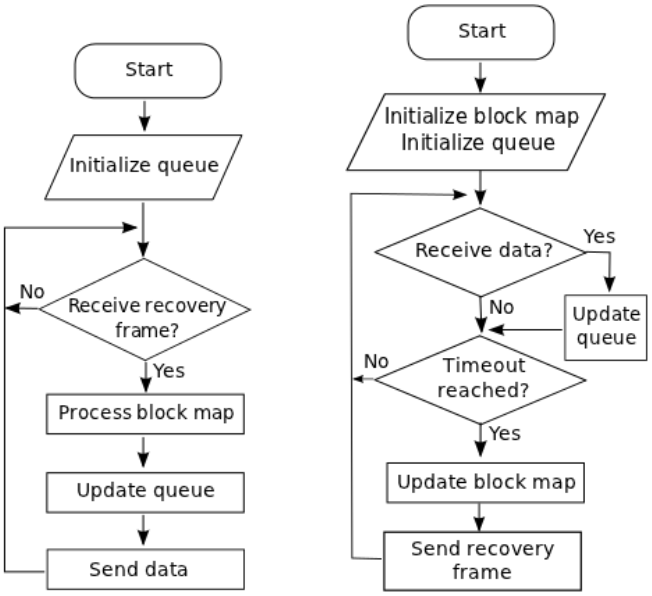


Fig. 3: iFrag frame formats

3.3 Frame Structure

iFrag defines data and recovery frames. The structure of these frames is illustrated in Fig. 3. Depending on the iFrag mode, data frames are composed of one or more blocks. Each block has two additional bytes, one each for sequence number and block CRC. A 1 byte block sequence number can represent values from 0 to 255. Our experiments show that this range of sequence numbers is sufficient and is unlikely to overlap in the block numbers even at extremely noisy channel conditions. It is worth noting that sequence number step increment depends on iFrag mode (*e.g.,* in iFrag 1 the sequence number increases by 8 while in iFrag 8 it increases by one). This is further described in Section 3.5. As a proof of concept, iFrag was implemented with four different modes, namely iFrag 1, iFrag 2, iFrag 4 and iFrag 8, where 1, 2, 4 and 8 represent the number of data blocks in the frame. For example, if iFrag 4 is used, then the data frame will contain four different data blocks. These blocks may not be consecutive as explained later in Section 3.4. Although the fact that our implementation of iFrag uses only 4 modes, we envisioned that more modes could be supported. It is also important to note that the amount of data sent in each frame is fixed to 96 bytes. This size was chosen because iFrag 8 appends 16 extra bytes in noisy channel conditions (1 byte sequence number and 1 byte CRC for each block). The maximum frame size is 128 bytes in which 16 bytes are reserved as PHY and MAC headers. iFrag uses the remaining 96 bytes for data. If header CRC is to be inserted, then the amount of data will be 95 bytes instead. We decided not to include header CRC to save this overhead. Note that padding will be added to the last block when the transmitted data size is not a multiple of the block size. A special block sequence number is reserved to indicate the frame last block. The byte that is next to this special block sequence number specifies the length of the data in that last block.



(a) Sender side (b) Receiver side

Fig. 4: iFrag operation flow chart

iFrag uses recovery frames which are sent from the receiver to the sender to acknowledge the received blocks and report missing or corrupted blocks. The recovery frames consist of four fields: Start Block Number (SBN), Block Map, Block Count, and CRC. SBN field is of 1

byte size and is used to flag the first missing or corrupted block. In case that all blocks are received correctly, SBN points to the next expected block. The Block Map field is of 4 bytes size and is used to indicate the status of the consecutive blocks following the SBN. Each bit represents a single block. After each session (*i.e.,* consists of sending 4 frames as in[8]), the receiver reports the number of correctly received blocks to the sender. The sender uses the Block Count field to calculate the Packet Reception Ratio (PRR) which is used by iFrag to decide on which mode it should operate on. The CRC component is used to check the correctness of the recovery frame. Both Block Count and CRC are of 1 byte each.

3.4 iFrag operation

iFrag sender and receiver sides operation are illustrated in Fig. 4. The sender and the receiver agree on the supported iFrag modes during the neighborhood discovery phase. The sender transmits a specific number of consecutive frames each session (4 frames in our implementation), and waits for the recovery frame from the receiver. At the end of this session, the receiver sends the recovery frame to acknowledge the received blocks and request the missing ones. The loss of the recovery frame is not sufficient to give reasonable measure of channel condition due to the bursty nature of error patterns in wireless channels. As a result, no data will be sent until the recovery frame is received. This design was inspired by the fact that the frames will be lost or corrupted if the channel is noisy. Hence, instead of re-sending the data frames again, that are often large, we limit our retransmission to the smaller recovery frame, which has higher chances to be delivered correctly. Moreover, resending the data frames before obtaining the recovery frame may result in re-sending some of the correctly received blocks which is considered as an additional overhead and will affect the network goodput.

In the case of corrupted or lost recovery frame a new one will be sent after $timeout_{recovery}$ period. This period is set to a value greater than one RTT added to the transmission delay for the whole session. $timeout_{recovery}$ should be long enough to allow the recovery frame to be delivered and processed in the sender side, allow the sender to send all the frames required, and to give a chance for the last frame to make it to the receiver. If all frames were lost, then the receiver needs to re-send the recovery frame. A small margin is added to the timer to insure the consideration of processing and queuing time in the sender and receiver side. After sending all of the required blocks, the sender then sends an end message. If the end message is lost or corrupted, the receiver ends the connection if it does not receive any new data for a predetermined period of time $timeout_{end}$. iFrag sender and receiver sides operation are described in Algorithm 1 and Algorithm 2 respectively.

3.5 Discussion

The idea of dynamic block size selection introduces a number of challenges. One of these challenges is to know when to switch between iFrag modes without synchronization between the sender and the receiver. This switching mechanism must satisfy three requirements: mode discovery, data integrity, and minimum overhead. In the first requirement, the receiver must have a mechanism to discover the change in the mode which is decided by the sender. In second requirement, the sent data over various modes must be distinguishable for the receiver. In final requirement, minimize the expected overhead of implementing such dynamic mechanism, which is a logical consequence of having iFrag. For iFrag to operate efficiently, its overhead must be less than its actual gain (*i.e.,* the overall performance of iFrag should outperform all other static Seda modes). Having these challenges in mind, iFrag was designed to be able to identify error patterns, identify best mode to choose, enable the receiver to know the received mode, minimize the overhead of dynamism, and preserve data integrity. Each of these challenges is solved by implementing a specific technique as detailed below.

iFrag uses the percentage between the number of sent and correctly received blocks within 5 sessions (*i.e.,* 20 frames) to correctly identify the channel error pattern. The sender knows how many blocks of each type have been sent. In addition, the receiver knows how many blocks have been correctly received. The receiver reports the number of correctly received blocks to the sender in every recovery frame. The sender infers the channel condition based on this PRR. If the channel is noisy, iFrag reduces the block size to provide increased reliability. On the other hand, if the channel is good, iFrag increases the block size to reduce overhead. Finally, when the channel quality falls between these extremes, iFrag simply continues using the current mode. Our implementation supports transitioning from one mode to the next in a gradual manner. Since interference conditions may be transient, we do not immediately jump across multiple iFrag modes. Such fast transitions may produce system instability, and lead to increase in header overhead per block, resulting in performance degradation in the next session.

iFrag relies on PRR to decide whether the channel is good or noisy using two thresholds for each operating

Operational Mode	Channel Condition
iFrag 1	PRR=100%
iFrag 2	80%≤PRR<100%
iFrag 4	50%≤PRR < 80%
iFrag 8	PRR<50%

Table 1: Thresholds for switching from one iFrag mode to the other

mode, namely $threshold_{good}$ and $threshold_{bad}$. Therefore, iFrag has 6 thresholds in total. These thresholds are based on PRR of the previous session and could be chosen based on statistical channel condition analysis at the early stage of the connection (initialization phase). In our implementation, the thresholds are fixed to the values shown in Table 1 such that the transition between one iFrag block size to the other can produce less overall overhead based on the mathematical model discussed in Section 4. The choice of these thresholds is experimentally verified as shown in Section 5. In fact, the thresholds could be fine tuned through a dynamic learning component. However, this was not used because of the implementation overhead with minimal expected improvements.

iFrag allows switching between modes on the fly. The receiver determines iFrag mode based on the received frame size. This is feasible because each mode has different number of blocks, hence different frame size. For example, in iFrag 1 the frame is composed of only 1 block (*i.e.,* only two additional bytes, one for block sequence number and the other for block CRC). On the other hand, iFrag 8 will consist of 8 blocks and hence 16 additional bytes. As a result, the receiver infers the mode based on the size of the received frame. This in turn minimizes the need for a control frame that may be exchanged between the sender and the receiver each time the sender decides to change iFrag mode.

Additionally, iFrag design preserves data integrity. Whenever the mode of transmission is being changed to different block size, the data sent by the sender should be correctly identified by the receiver. The following example shows how data integrity can be lost. iFrag sends blocks 77 and 78 in iFrag 2 mode and block 77 is lost. If iFrag switches to iFrag 1 mode because of higher PRR then block 77 will be sent as a 96 bytes instead of 48 bytes which affects data integrity. Similar situation will occur if the transition was in the opposite direction, *i.e.*, switching from larger block sizes to smaller block sizes. The problem worsens if iFrag is going on multiple consecutive transitions while some blocks are not yet received correctly. It can be seen that not having consistent numbering scheme for block numbers may lead to incorrect behavior. In order to solve this problem, several design choices were made. First, is to standardize the block sizes and the block numbering convention. In other words, all block numbers refer to the first small iFrag 8 block that contains the data in the beginning of the block. The remaining part of the block contains the next consecutive iFrag 8 blocks without appending ad-

Algorithm 1: iFrag Sender

```
Initiate connection and inform receiver of supported modes
Divide network layer packet into blocks
Add block sequence number and CRC, reframe and handoff packet to MAC layer for transmission
if recovery frame received then
    Update PRR
    if Session is starting then
        Select iFrag mode:
        if PRR > threshold_good then
            Switch to the next mode with bigger block size
        else if PRR < threshold_bad then
            Switch to the next mode with smaller block size
        Reset PRR
    else
        Keep using the same mode
    Retransmit requested blocks (with new blocks if any) as determined from the BlockMap field of the recovery frame
```

Algorithm 2: iFrag Receiver

```
Connection establishment (know Sender supported modes)
Send recovery frame that includes BlockMap and SBN to request for frames from the Sender
if data frame received then
    Identify iFrag mode using frame size
    Identify correct blocks through CRC (each block contains consecutive bytes)
    if all blocks are correctly received then
        Re-assemble blocks into a network layer packet and hand-off to network layer
    else if some blocks are corrupted then
        reconstruct recovery frame accordingly
        Buffer correctly received blocks
Send recovery frame after each session or after timeout_recovery period, whichever earlier
Stop sending recovery frame when End Message received or when no new data has been received for the period of timeout_end
```

ditional bytes for block number and CRC (recall, block of type iFrag 2 numbered with block number 77). This means that this block contains data of iFrag 8 block 77 in the beginning of the iFrag 2 block. Because iFrag 2 block size is 4 times larger than iFrag 8, three consecutive iFrag 8 blocks, namely block 78, 79 and 80, are added. Consequently, if the data for next blocks have not yet been sent, the next iFrag 2 block will be numbered 81 and contains 81, 82, 83, 84 data. In this approach, changing the mode will not affect data integrity because iFrag will be sending consecutive data blocks within every block number and CRC. The idea of having the smallest block as the actual indicator is similar to saying that data are represented in chunks of 12 bytes where each has its own number.

In case where the transition happens from smaller to bigger blocks, the proposed solution may impose some overhead since iFrag might be sending some additional blocks that were previously received correctly. For example, consider having iFrag operating in iFrag 4 mode. Each block now contains 2 consecutive blocks of data. Suppose that blocks 0 up to 30 were sent (block 30 contains both 30 and 31 block numbers) and that block 20 and block 30 were lost (*i.e.,* iFrag 8 block numbers 20, 21, 30, 31 were lost). The receiver sends the corresponding block map. If the sender continues to operate on the same mode, then the first two iFrag 4 blocks sent would be 20 and 30, containing the corresponding data without problems. Even if the decision was made to reduce the size to iFrag 8, then the next 4 blocks to be sent would be 20, 21, 30, 31. However, if the mode was changed to iFrag 2, then an overhead is introduced, since iFrag is now forced to send consecutive bytes within the block itself. This means that the first two iFrag 2 blocks to be sent would be 20, 30 and then continuing with 34. This is because now block 20 contains 20, 21, 22, 23 where two are needed and two are overhead. The other block which is 30 contains also 4 consecutive blocks 30, 31, 32, 33. It is worth noting that blocks 32, 33 are actually not an overhead since they are correctly received for the first time. This will make the next block to be sent to start from the next unsent block which is 34 instead of 32 as 32, 33 were already sent. In fact, the overhead for such rare cases is around 50% or less on average.

iFrag suffers from certain overhead while being in transition between different modes with lost blocks pending retransmission. For example, consider the following case: iFrag sends blocks in iFrag 8 mode and just sent blocks 0 to 7 in the last frame. By assuming that block 6 was lost and iFrag decided to switch to iFrag 4, two consecutive blocks with a single block sequence number will be sent (*i.e.,* block number 6 and 7 will be sent again). As block 7 was correctly received from the previous session, it is considered as an overhead and eventually discarded. In practice, this overhead is very small as shown in our experimental results presented in section 5.

m	Number of blocks in a frame
P_g	Probability of being in good state
P_b	Probability of being in bad state
P_{bb}	Transition probability from the bad state to the bad state
P_{bg}	Transition probability from the bad state to the good state
P_{gb}	Transition probability from the good state to the bad state
P_{gg}	Transition probability from the good state to the good state
N_b	The mean size of error cluster which starts and ends with corrupted bits followed by more than or equal to 200 consecutive error-free bits
N_g	The mean size of inter cluster that is nothing but the number of error-free bits between two consecutive error clusters
$P(B)$	Probability of having sequence of B bits without errors
$P_s(n, B, i)$	Probability of having sequence of B bits without errors given that the chain started at state s and will visit this state n times through i transitions between states
e_b	Bit error probability in bad state
H	Frame header size (bits)
F	Frame size (bits)
K	Block size (bits)
R	Recovery frame size (bits)
HAR	Header acceptance rate
FAR	Frame acceptance rate
KAR	Block acceptance rate
RAR	Recovery acceptance rate

Table 2: Terms used in the system model

4 System Model

In this section, we present an analytical model for iFrag. We use Gilbert-Elliot hidden Markov model [7], as shown in Fig. 5, to derive an expression for having error-free chunks of bits, then we use these probabilities to estimate the throughput. We show that iFrag outperforms Seda analytically and using MATLAB simulations. Table 2 summarizes the terms used in our model.

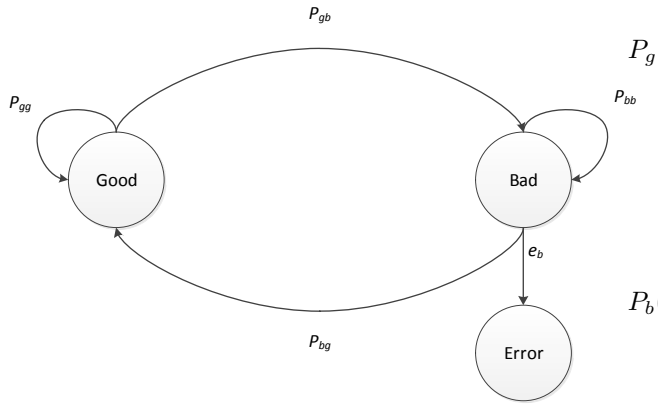


Fig. 5: Gilbert-Elliot wireless channel model

In [8], the authors defined the channel model probabilities as,

$$P_g = \frac{N_g}{N_g + N_b}$$
$$P_b = \frac{N_b}{N_g + N_b}$$
$$P_{bg} = \frac{1}{N_b} = 1 - P_{bb}$$
$$P_{gb} = \frac{1}{N_g} = 1 - P_{gg}$$

The probability that a sequence of B bits is received correctly, $P(B)$, is given by:

$$P(B) = P_g P_{gg}^{B-1} + P_b P_{bb}^{B-1}(1-e_b)^{B-1} + \sum_{i=1}^{B-1}\sum_{n=k_1}^{k_2} P_g(n,B,i) + \sum_{i=1}^{B-1}\sum_{n=k_1}^{k_2} P_b(n,B,i) \quad (1)$$

where $k_1 = 1 + i - \lceil \frac{i}{2} \rceil$, and $k_2 = B - \lceil \frac{i}{2} \rceil$.

As per the definition of $P_g(n,B,i)$ in Table 2, there are $\lceil \frac{i}{2} \rceil$ transitions from good to bad states and $i - \lceil \frac{i}{2} \rceil$ transitions from bad to good states. Also, there are k_1 segments (sequences) of bits in good states and $\lceil \frac{i}{2} \rceil$ in bad states. This leads to $n - k_1$ transitions from good state to good state and $B - n - \lceil \frac{i}{2} \rceil$ transitions from bad state to bad state. The good state segments' lengths must sum to n, so there are $\binom{n-1}{k_1-1}$ different combinations. Similarly there are $\binom{B-n-1}{\lceil \frac{i}{2} \rceil - 1}$ combinations of segments of bad states. As a result, $P_g(n,B,i)$ can be derived as:

$$P_g(n,B,i) = \binom{n-1}{k_1-1}\binom{B-n-1}{\lceil \frac{i}{2} \rceil - 1} P_g P_{gg}^{n-k_1} P_{gb}^{\lceil \frac{i}{2} \rceil} P_{bg}^{i-\lceil \frac{i}{2} \rceil} P_{bb}^{B-n-\lceil \frac{i}{2} \rceil}(1-e_b)^{B-n} \quad (2)$$

Similarly, we derive $P_b(n,B,i)$ as:

$$P_b(n,B,i) = \binom{n-1}{k_1-1}\binom{B-n-1}{\lceil \frac{i}{2} \rceil - 1} P_b P_{bb}^{n-k_1} P_{bg}^{\lceil \frac{i}{2} \rceil} P_{gb}^{i-\lceil \frac{i}{2} \rceil} P_{gg}^{B-n-\lceil \frac{i}{2} \rceil}(1-e_b)^{n} \quad (3)$$

Using this approach we derive the header, the frame, the block, and the recovery acceptance rates as follows,

$$HAR = P(H) \quad (4)$$
$$FAR = P(F) \quad (5)$$
$$KAR = P(K) \quad (6)$$
$$RAR = P(R) \quad (7)$$

Using the above equations, we calculate the overall throughput of FARQ, Seda, and iFrag as follows,

$$Throughput_{FARQ} = \frac{F - H}{\frac{F}{FAR*RAR} + \frac{R}{4}} \quad (8)$$

$$Throughput_{Seda\ m} = \frac{m*(K-16)}{\frac{m*K}{HAR*KAR*RAR} + \frac{H}{HAR*RAR} + \frac{R}{4}} \quad (9)$$

$$Throughput_{iFrag\ m} = \frac{m*(K-16)}{\frac{m*K}{HAR*KAR} + \frac{H}{HAR} + \frac{R}{4*RAR}} \quad (10)$$

The last equation is for iFrag during m mode. To evaluate the average throughput for iFrag, we will assume that the channel alternates between different L loss models and the victor $\mathbf{T} = [T_1, T_2, ..., T_L]$ represents the time percentage the channel stays in each loss model. Per Section 3, our design is capable of choosing the iFrag mode, m, that maximizes the throughput, neglecting the time period needed to switch between modes, we can evaluate the average iFrag throughput as:

$$Throughput_{iFrag} = \sum_{l=1}^{L} max\{Throughput_{iFrag\ 1}, ..., Throughput_{iFrag\ m}\} T_l$$

Loss Model	Mean Error Cluster Size (bits)	Mean Inter-Cluster Size (bits)	e_b	Overall BER
1	250	1000	0.4	0.08
2	100	1000	0.4	0.036
3	386	3234	0.43	0.045
4	120	3234	0.36	0.013
5	386	9690	0.4	0.015
6	0	∞	0	0

Table 3: Loss models

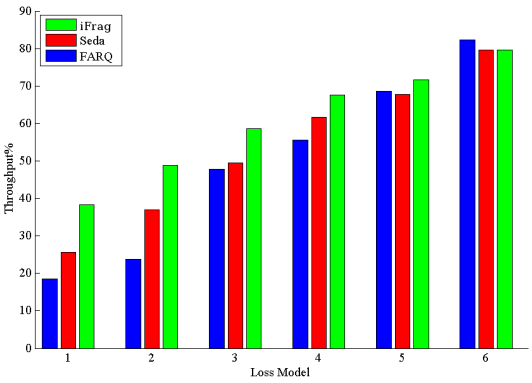


Fig. 6: FARQ, Seda, and iFrag analytical results, m=4

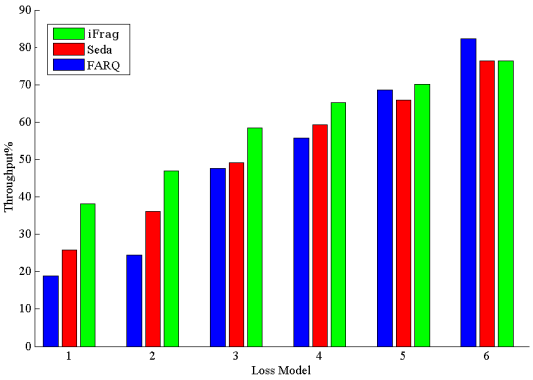


Fig. 7: FARQ, Seda, and iFrag simulation results, m=4

(11)

One of the main performance measures that was not taken into consideration in [8] is the delay from both the frame and the block perspective. We will present an analytical approach aiming to estimate the average number of transmission trials to receive a correct frame via FARQ, Seda, and iFrag. For FARQ, the average transmission trials is the mean of a geometric random variable with success probability, p_{FARQ}, given by:

$$p_{FARQ} = FAR * RAR \tag{12}$$

Consequently, the average transmission trials, ATT_{FARQ}, equals

$$ATT_{FARQ} = \frac{1}{p_{FARQ}} \tag{13}$$

The average transmission trials of ATT_{Seda} of m blocks can be obtained with a modified approach. The transmission trials needed to have a complete correct frame are the maximum transmission trials of each block, assuming that all of the m blocks start within the same transmission trial. The transmission trials of each block is also a geometric random variable, X_{block}, with success probability, p_{block}, given by:

$$p_{block} = HAR * KAR * RAR \tag{14}$$

The cumulative distribution function (CDF) of X_{block}, $F_{X_{block}}(n)$ is

$$F_{X_{block}}(n) = 1 - (1 - p_{block})^n, n = 1, 2, 3, ... \tag{15}$$

The transmission trials needed to have a correct frame via Seda is a random variable, X_{Seda}, given by the maximum of m independent random variables,X_{block} as follows

$$X_{Seda\ m} = max\{X_{block1}, X_{block2}, ..., X_{blockm}\} \tag{16}$$

The CDF of X_{Seda} will be

$$F_{X_{Seda\ m}}(n) = (F_{X_{block}}(n))^m \tag{17}$$

Knowing the CDF of X_{Seda}, we can estimate ATT_{Seda} as

$$ATT_{Seda\ m} = E(X_{Seda\ m}) = \sum_{n=0}^{\infty} 1 - F_{X_{Seda\ m}}(n) \tag{18}$$

A similar approach can be used for iFrag as follows

$$p_{block+} = HAR * KAR \tag{19}$$

$$F_{X_{block+}}(n) = 1 - (1 - p_{block+})^n, n = 1, 2, 3, ... \tag{20}$$

$$X_{iFrag\ m} = max\{X_{block+1}, X_{block+2}, ..., X_{block+m}\} \tag{21}$$

$$F_{X_{iFrag\ m}}(n) = (F_{X_{block+}}(n))^m \tag{22}$$

$$ATT_{iFrag\ m} = E(X_{iFrag\ m}) = \sum_{n=0}^{\infty} 1 - F_{X_{iFrag\ m}}(n) \tag{23}$$

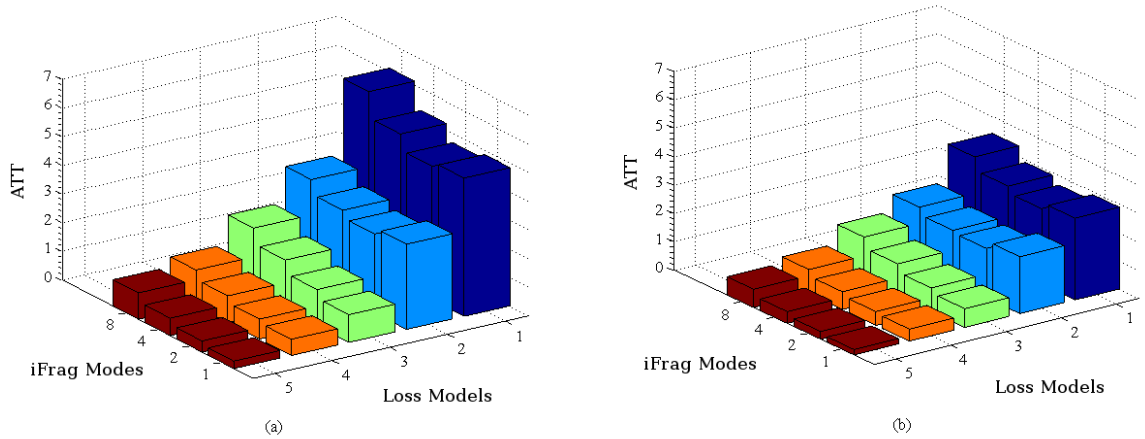


Fig. 8: Average Transmission Trials (ATT): (a) Seda (b) iFrag

In order to give a fair comparison between Seda design and iFrag, we use the same loss models chosen by the authors in [8]. These models, presented in Table 3, were obtained experimentally to identify the main parameters of the loss model, namely mean error cluster size, mean inter-cluster size, and BER. We added a loss model 6 to represent the ideal channel case.

The throughput results for the analytical model are shown in Fig. 6. It can be seen that iFrag outperforms both Seda and FARQ in all of the loss models, except for loss model 6 which represents the ideal channel. In such conditions, FARQ reduces the transmission overhead compared to Seda and iFrag and achieves the highest throughput. The reason behind iFrag lead is that it harness sending the recovery frame till it is received correctly. Although one might think that harnessing means more overhead and consequently less throughput, but in fact the opposite is true as a whole correct frame might be sent again due to loss of a recovery frame.

Simulations were done via MATLAB and the results are shown in Fig. 7, in which we simulated iFrag in static mode. We assumed that the channel variability is the main controller for both throughput and delay. We also assumed that choosing some parameters for channel variability will be not comprehensive to compare our simulations results to Seda results for two reasons. First, we do not have accurate measure or a concrete mathematical model on how channel is changing from loss model to another. Second, Seda were tested under fixed channel model. The results shows that static iFrag is better than Seda in all loss models. In order to be fair, we limit our iFrag to be static in both analysis and simulations. Note that we compare the dynamic iFrag to Seda in the implementation section, where we implemented both schemes and test over various channel conditions.

Through simulation, Fig. 8 gives us insight about how ATT behaves versus both loss model and iFrag mode. For all loss models and iFrag modes, iFrag has less ATT than Seda, especially for high BER models. This means that iFrag encounters less delay than Seda. It is also more energy efficient for sensor nodes as it saves energy by reducing the overall transmitted data and consequently prolonging the battery life of the motes.

5 Performance Analysis

In this section we present iFrag experimental results under varying channel conditions. We also show the effect of high power interference on the efficiency of iFrag packet recovery.

5.1 Experimental Setup

iFrag was examined using TelosB [23] motes in an office environment. TelosB motes are equipped with Chipcon-CC2420 radio [3] which is compatible with IEEE 802.15.4 (ZigBee) standard and are capable of sending/receiving 250 Kbps. The implementation was carried out using TinyOS 2.1.1. TelosB motes operate in the 2.4 GHz ISM band and interfere with WiFi devices. To limit the impact of this interference, our experiments were performed at night when the wireless traffic is minimal.

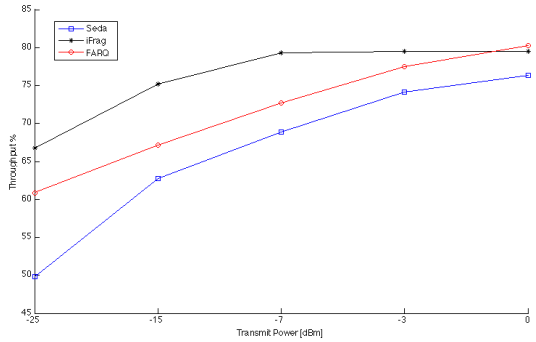


Fig. 9: Throughput of iFrag vs. Seda vs. FARQ without interference

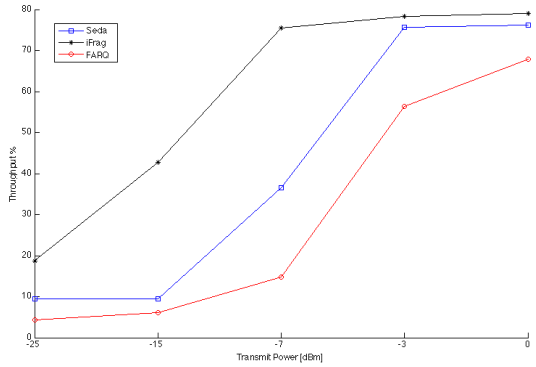


Fig. 10: Throughput of iFrag vs. Seda vs. FARQ with interference

The motes were $1m$ apart from each other and constantly powered through USB connections to avoid low power issues. We disabled MAC-layer automatic CRC to allow corrupted packets to be passed to iFrag implementation. To evaluate iFrag under various channel conditions, every experiment was repeated under two interference settings. The first set of experiments did not impose any interference. The second set of experiments were done while transferring a large file between two Linux boxes equipped with IEEE 802.11g wireless cards. These boxes were placed $15m$ apart and used a transmit power of 20 dBm. We used WiFi as a source of interference because of this high transmit power. We anticipate that our results will also hold in environments with lower level of interference. Further, WiFi interference already exists in many WSN deployments such as smart buildings and traffic control applications.

5.2 Experimental Results

In all experiments, throughput was calculated under various channel conditions. A sender mote sends 1000 frames to a receiver mote in each run of the experiment. Results are then averaged over five runs. As described earlier, each data frame consists of 96 bytes of data, one byte per block for the block sequence number, one byte per block for CRC and 16 bytes of physical and MAC layer headers. In our iFrag implementation, each frame may have one, two, four, or eight blocks.

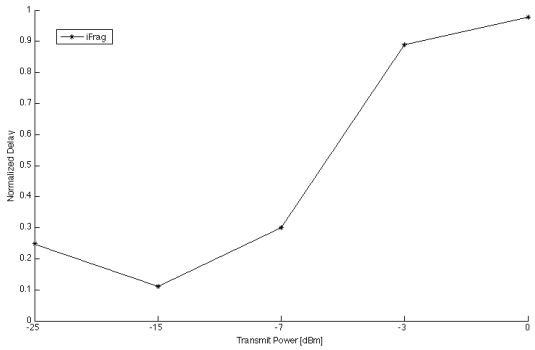


Fig. 11: Average delay per frame of iFrag vs. Seda. The y-axis shows the normalized iFrag delay with respect to Seda

In Fig. 9, iFrag is compared with FARQ and Seda in terms of throughput under normal channel conditions. To make a fair comparison, the implementation of FARQ is similar to Seda except for the fact that it uses a single block per frame. iFrag maintains constantly higher throughput compared to Seda, showing an average of 13% throughput improvement across all channel conditions. As expected, FARQ performs slightly better than iFrag when using the highest transmit power only. This happens due to the fact that FARQ imposes less overhead than iFrag as the latter starts initially with several blocks per frame and adjusts itself after the first session. In Fig. 10 the experiments were repeated while imposing interference from WiFi nodes. Similarly, iFrag maintains higher throughput compared to Seda. FARQ is always performing worse than both iFrag and Seda due to retransmissions caused by the interference. iFrag shows a $3\times$ increase in throughput compared to Seda in bad channel condition. iFrag and Seda throughput performance remains almost the same when the transmission power is high. This is because the imposed interference is not affecting the motes while they are using high transmission powers. As the power of the motes is reduced, iFrag outperforms Seda. This improvement is attributed to the dynamic nature of iFrag that smartly minimizes the overhead by selecting the appropriate number of blocks in each data frame.

One of the major improvements that allows iFrag to outperform Seda is the choice of re-sending recovery frame when it is lost or corrupted instead of re-sending

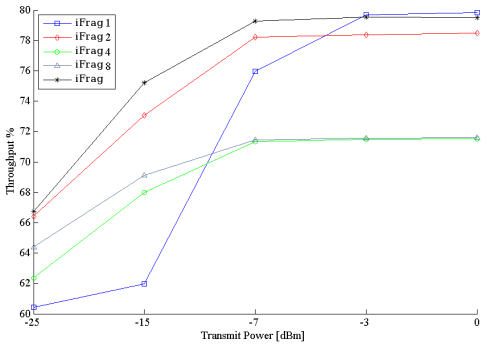


Fig. 12: Throughput of iFrag vs. static iFrag without interference

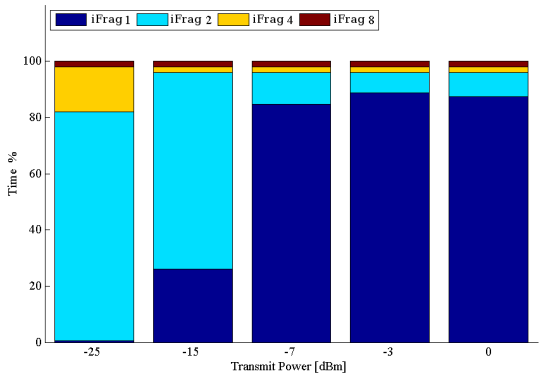


Fig. 13: Percentage of time iFrag is spending in each mode without interference

the same data. The recovery frame is about five times smaller than a data frame, and thus has a higher probability of being received correctly. The recovery frame prevents the sender from retransmitting data that has already have been received correctly. This limits retransmission overhead. It also makes iFrag more energy efficient by limiting redundant transmissions by battery-operated sensor nodes. Our experimental results are in agreement with the analytical and simulation results in the case of ideal channel condition presented in Section 4. It is challenging to compare other simulation loss models with the experimental results because it is difficult to generate the same simulated loss models, as interference is unpredictable.

Fig. 11 compares the network end-to-end delay of iFrag versus Seda. It shows the normalized network delay under various channel conditions. The quality of the channel is altered by reducing motes transmission power. iFrag experiences similar delay to Seda when the channel quality is very good. However, iFrag starts to outperform Seda as the interference is increased. The peak performance happens at -15 dBm in which iFrag

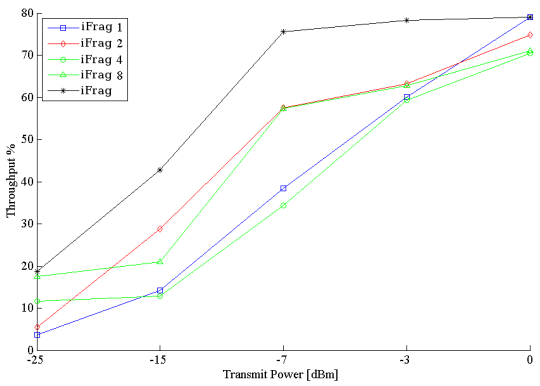


Fig. 14: Throughput of iFrag vs. static iFrag with interference

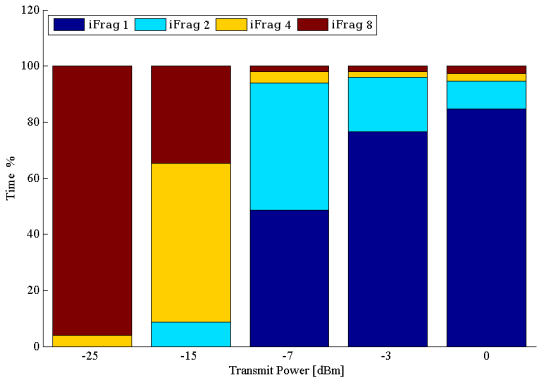


Fig. 15: Percentage of time iFrag is spending in each mode with interference

manages to reduce the delay to only 12% compared to Seda. Although iFrag is not targeting network delay reduction, it is interesting to see that it performs better than Seda when the channel is suffering from interference. This improvement is attributed to the fact that iFrag receiver retransmits the recovery frame in case it has been lost or corrupted. As discussed earlier, this recovery frame is smaller in size than the data frame, and hence easier to be delivered correctly. As a result, the sender will trigger a new session of data sending as soon as it receives a recovery frame. Alternatively, the sender in Seda will wait for a predefined amount of time before sending the already sent data in case of recovery frame loss or corruption. In extreme interference conditions, iFrag suffers from frequent losses which result in performance degradation, increasing the normalized delay to 25% compared to Seda.

To quantify the advantages of dynamism in iFrag, the dynamic version was compared to a static version. Static iFrag means that the number of blocks in iFrag

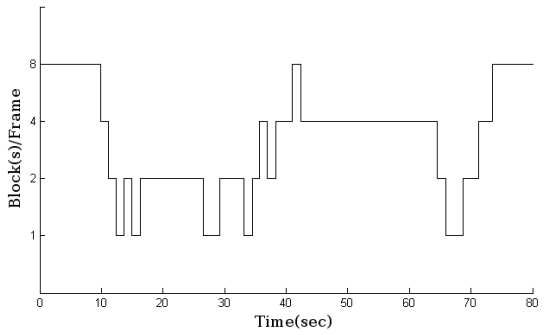


Fig. 16: iFrag transitions between modes over time

is fixed to either 1, 2, 4 or 8 blocks. Initially, the experiments were done without interference. Fig. 12 shows that iFrag outperforms all other static ones. The only exception is in iFrag 1, which is performing slightly better than iFrag under perfect channel (*i.e.,* no loss due to high transmission power and no interference). The reason for this behavior is the initial mode of iFrag. It always starts with 8 blocks in each frame and it requires at least one session to discover that the channel is good and adjust its mode. Another reason comes from environment interference. Fig. 13 represents the percentage of time iFrag is spending on each mode. Even with the motes at the best transmission power, iFrag still spends around 10% of its time in iFrag 2. The main reason for this observation is the selection mechanism of threshold for switching between iFrag 1 and iFrag 2. In current implementation, 100% PRR is required to switch to iFrag 1, forcing iFrag to switch to iFrag 2 after each session with one or more lost packets.

The dynamic vs. static experiment was repeated under bad channel condition. As shown in Fig. 14, iFrag results in higher throughput than all other schemes. Fig. 15 shows that iFrag is spending more than 80% of the time in iFrag 1 when the channel is good. On the other hand, iFrag never switches back to iFrag 1 when the transmission power is -25 dBm or -15 dBm since the channel is considered noisy.

We studied the dynamic behavior of iFrag over time. Fig.16 shows the transitions between different iFrag modes according to the PRR. In this experiment, we set the transmission power to -7 dbm and set the distance between the motes to $1.5m$. The results show that the transitions occur gradually one-step at a time. In each step, the protocol stabilizes for a minimum of one session (*i.e.,* reception of five distinct acknowledgements). The protocol always starts with 8 blocks per frame until the block loss ratio reaches the specified threshold. It then switches to iFrag 4 *i.e.* four blocks per frame, and then to iFrag 2 and so on, one step at a time. When the channel quality deteriorates, iFrag attempts to operate on larger number of blocks one step at a time as well.

6 Conclusion and Future Work

In this paper, we presented iFrag, a dynamic data streaming approach that is suitable for resource constrained wireless sensor networks. iFrag works mainly by dividing the frame into several blocks, each with its own CRC check. At the receiver side, if any block is found to be corrupted, then that block will be retransmitted along with new blocks, if any. Basically, iFrag features two enhancements over the static partial packet recovery approaches. First, it limits unnecessary data retransmissions by allowing the receiver to periodically send the recovery frame. Second, it dynamically selects the number of blocks in each frame based on the channel condition observed during previous session. We show through analytical, simulation, and experimental evaluation that iFrag can increase the throughput by 13% on average while reducing the delay to 12% compared to other static fragmentation approach.

There are interesting avenues for further work in this area. Sending a frame with blocks of variable length instead of fixed, identical length brings additional challenges that need to be addressed. Also, machine learning techniques may be used to dynamically select thresholds for switching between modes based on current channel conditions. Finally, the concepts behind iFrag may be extended to IEEE 802.11/b/g/n networks for enhancing their performance.

References

1. D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM, pages 121–132, 2004.
2. A. Avudainayagam, J. Shea, T. Wong, and X. Li. Reliability exchange schemes for iterative packet combining in distributed arrays. In *Wireless Communications and Networking, IEEE WCNC*, pages 832 –837, 2003.
3. CC2420. CC2420 Data Sheet. `http://www.ti.com/`.
4. W. Dong, X. Liu, C. Chen, Y. He, G. Chen, Y. Liu, and J. Bu. DPLC: Dynamic packet length control in wireless sensor networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.
5. W. Dong, X. Liu, C. Chen, Y. He, G. Chen, Y. Liu, and J. Bu. DPLC: Dynamic packet length control in wireless sensor networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, March 2010.
6. H. Dubois-Ferrière, D. Estrin, and M. Vetterli. Packet combining in sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, SenSys, pages 102–115, 2005.

7. J.-P. Ebert and A. Willig. A Gilbert-Elliot bit error model and the efficient use in packet level simulation. *Technical Report, TKN-99-002*, 1999.
8. R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys, pages 209–222, 2006.
9. B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller. Maranello: practical partial packet recovery for 802.11. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI, pages 14–14, 2010.
10. Y. hua Zhu, H. Xu, K. kai Chi, and H. Hu. Accumulating error-free frame blocks to improve throughput for IEEE 802.11-based WLAN. *Journal of Network and Computer Applications*, 35(2):743 – 752, 2012.
11. J. Huang, G. Xing, G. Zhou, and R. Zhou. Beyond co-existence: Exploiting WiFi white space for ZigBee performance assurance. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 305 –314, oct. 2010.
12. K. Jamieson and H. Balakrishnan. PPR: partial packet recovery for wireless networks. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM, pages 409–420, 2007.
13. A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding link-layer behavior in highly congested IEEE 802.11b wireless networks. In *Proceedings of the ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, E-WIND, pages 11–16, 2005.
14. S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *Proceedings of the ACM SIGCOMM conference on Data communication*, pages 401–412, 2008.
15. C.-F. Kuo, H.-W. Tseng, and A.-C. Pang. A fragment-based retransmission scheme with QoS considerations for wireless networks. In *Proceedings of the 2007 international conference on Wireless communications and mobile computing*, IWCMC '07, pages 225–230, 2007.
16. C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi interference in low power ZigBee networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 309–322, 2010.
17. K. C.-J. Lin, N. Kushman, and D. Katabi. ZipTx: Harnessing partial packets in 802.11 networks. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, MobiCom, pages 351–362, 2008.
18. S. Lin and P. Yu. A hybrid arq scheme with parity retransmission for error control of satellite channels. *IEEE Transactions on Communications*, 30(7):1701 – 1719, 1982.
19. A. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, MobiCom, pages 16–30, 2005.
20. E. Modiano. An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols. *Wirel. Netw.*, 5(4):279–286, July 1999.
21. Y. Sankarasubramaniam, I. Akyildiz, and S. McLaughlin. Energy efficiency based packet size optimization in wireless sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 1 – 8, 2003.
22. P. Sindhu. Retransmission error control with memory. *Communications, IEEE Transactions on*, 25(5):473 – 479, may 1977.
23. TelosB. Memsic Corporation. `http://www.memsic.com/`.
24. M. Vuran and I. Akyildiz. Cross-layer packet size optimization for wireless terrestrial, underwater, and underground sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 226 –230, April 2008.
25. M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proceedings of the ACM SIGCOMM conference on Data communication*, pages 3–14, 2009.
26. A. Willig. Memory-efficient segment-based packet-combining schemes in face of deadlines. *IEEE Transactions on Industrial Informatics*, 5(3):338 –350, 2009.
27. G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi. Beyond the bits: cooperative packet recovery using physical layer information. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom, pages 147–158, 2007.
28. J. Xie, W. Hu, and Z. Zhang. Revisiting partial packet recovery in 802.11 wireless lans. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys, pages 281–292, 2011.
29. J. Zhang, H. Shen, K. Tan, R. Chandra, Y. Zhang, and Q. Zhang. Frame retransmissions considered harmful: improving spectrum efficiency using micro-ACKs. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, Mobicom '12, pages 89–100, 2012.
30. Y. Zhou and J. Wang. Optimum subpacket transmission for hybrid ARQ systems. *Communications, IEEE Transactions on*, 54(5):934 – 942, May 2006.