Ad Hoc Networks 10 (2012) 339–355

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Practical packet combining for use with cooperative and non-cooperative ARQ schemes in resource-constrained wireless sensor networks

Damien O'Rourke [a,*], Conor Brennan [b]

[a] Autonomous Systems Laboratory, CSIRO, Brisbane, Australia
[b] The Rince Institute, Dublin City University, Dublin 9, Ireland

ARTICLE INFO

Article history:
Received 8 June 2010
Received in revised form 1 May 2011
Accepted 28 June 2011
Available online 23 July 2011

Keywords:
Packet combining
Diversity
Cooperative communications
ARQ
Wireless sensor networks

ABSTRACT

An improved *implementation* of a *post-detection* packet combining scheme, which is especially applicable to low power, resource-constrained sensor networks, is developed and practically implemented on popular off-the-shelf wireless motes. The algorithm can be used as part of protocols such as cooperative communications and hybrid-ARQ schemes which have been shown to be of major benefit for wireless communications. Using the packet combining implementation developed in this paper more than an 85% reduction in energy costs are possible over previous, similar approaches. Both simulated and practical experiments are developed in which packet combining is shown to offer up to approximately 2.5 dB reduction in the required Signal-to-Noise Ratio (SNR) for a desired Packet Error Rate (PER). This is a welcome result as complex schemes, such as maximal-ratio combining, are not implementable on many of the resource constrained devices under consideration.

1. Introduction

To help address the issue of energy consumption in WSNs this paper presents a highly efficient implementation of a simple but practical technique referred to as *post-detection* packet combining.[1] Packet combining can be used when the same data is received from multiple sources or from the same source over different time slots. The former case generally occurs in a promising technique referred to as cooperative communications, which has been shown to be of benefit for wireless networks [2] and is a viable alternative to Multiple-Input, Multiple-Output (MIMO) techniques which are not fully realisable on current WSNs due to their size, cost, and hardware limitations. The latter case can be simply referred to as a traditional Automatic Repeat Request (ARQ) system with packet combining [1].

Combining techniques such as Maximal-Ratio Combining (MRC) [3] or the use of Space Time Block Codes (STBCs) [4] are not viable options for commercially available architectures such as the popular Tmote Sky [5], MicaZ [6], and other similar devices, as their simplified hardware does not permit, for example, access to the analog portion of the transceiver. In addition, Channel State Information (CSI) at both the transmitter and receiver is generally not available and only partial information can be obtained at the receiver in most cases (in the form of Link Quality Indicator (LQI) and/or Received Signal Strength Indicator (RSSI)). As it is towards these low power, resource-constrained architectures that the current work is aimed, this paper therefore focuses on a *post-detection* (after demodulation) combining scheme rather than a *pre-detection* (before demodulation) combining scheme.

Although significant advances are likely to be made in the area of chipset design in the coming years the size of future WSNs are envisaged to become increasingly small in size (the original intention being on the order of a piece of dust [7]). It is reasonable to expect that there will

* Corresponding author. Tel.: +61 7 3327 4033.
E-mail address: damien.o'rourke@csiro.au (D. O'Rourke).

[1] Packet combining is also an effective method for achieving higher throughput [1].

1570-8705/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2011.06.013

remain a need for less complex protocols, despite these advances in CPU speed, available program memory, etc. Toward this end an efficient packet combining implementation referred to as *improved Packet Merging (iPM)*, a preliminary outline of which was provided by the authors in [8], is extended and thoroughly analysed in this paper. The analysis is carried out using both simulations and practical experiments on a commercially available mote platform. The results clearly demonstrate both the significant energy savings of iPM and the beneficial effects of the underlying packet combining scheme on a real system. The packet combining procedure used in iPM is based upon previous work in [9,10] and most recently [11]. The iPM algorithm uses available metrics such as RSSI/LQI combined with other novel techniques to significantly reduce the required complexity of the packet combining procedure.

1.1. Contributions

The aim of iPM is that it can be easily implemented on low-power, resource-constrained wireless sensor nodes. The purpose of such a scheme is to reduce the required energy consumption of a wireless node while maintaining the same performance levels or, alternatively, to obtain improved performance at the same energy consumption. As this paper is concerned with WSNs, and their inherent energy limitations, it is the former case that is of interest. The combining scheme is implemented in both a cooperative and non-cooperative setup to determine the gains possible; this is done both practically and through simulations. Although the concept of packet combining is not new, very few publications demonstrate practical evidence of its benefits and instead only document theoretical results. The aim of this work is not to make any new theoretical contributions to the field but to focus solely on the practical aspects and efficient implementation of previously proposed theoretical results. With this in mind the following contributions are made in this paper:

- A highly efficient implementation of a post-detection packet combining scheme based on the Cyclic Redundancy Check is developed. The improved packet combining scheme offers more than an 85% reduction in the number of iterations required to find the correct packet over current implementations. This leads to significant energy savings.
- A thorough energy analysis is carried out for a commonly used platform based on the IEEE 802.15.4 standard to help quantify these savings.
- The use of a second generator polynomial for the Cyclic Redundancy Check is shown to be of benefit in this scheme offering improved performance through the significant reduction of false-positives.
- The packet combining scheme is shown through simulations and practical experiments to have a positive effect on the energy consumption of a wireless node as it lowers the number of transmissions required to achieve a desired packet error rate. The experiments are carried out for both cooperative and non-cooperative scenarios, and for different packet sizes.

1.2. Paper organisation

The next section discusses background material and relevant work. Section 3 then discusses general packet combining and analyses the gains achievable through more efficient combining algorithms using simulations. Section 4 reviews the preliminary improved version (iPM) developed by the authors in [8]. Section 5 discusses an initial experimental setup used in both [8] and the current work. Packet combining issues, such as hidden errors and false-positives, are studied in Section 6 with the help of the experimental results. Section 7 looks at further improvements to iPM and provides experimental results to show the benefits of the proposed techniques. Section 8 looks at energy measurements carried out on a popular IEEE 802.15.4-compliant wireless mote. Section 9 carries out experiments to obtain PER curves on the wireless motes and shows the benefits of it on a real system. Finally, Section 10 concludes the paper and discusses future work.

2. Background and related work

Packet combining was first described by Sindhu [9] using forward error correction in bursty channels. An incremental redundancy combining technique called *code combining* was then developed by Chase [12] and was designed with very noisy channels in mind. Code combining concatenates the received packets to form codewords from increasingly longer and lower-rate codes. In [13] a scheme is described whereby the transmitter sends a packet to L different nodes within a cluster. Code combining is then used at the cluster head in order to correct any errors that may have arisen.

In [10] an ARQ scheme with packet combining is presented that is suited to the resource-constrained devices under consideration in this paper. The particular scheme, referred to as Extended ARQ (EARQ), has a throughput for smaller packet sizes that is sufficiently tight to an upper bound for type-II ARQ schemes (up to a very high BER). The combining method in EARQ uses a modulo-2 sum of the received packets in order to assist in error correction (see Section 3). In [11] an interesting protocol called Simple Packet Combining (SPaC) is described that makes use of this combining procedure.

The SPaC protocol is designed specifically for low-rate, low duty cycle sensor networks and has been shown to offer a lot of promise for these type of devices. When a node receives two or more corrupt packets, a packet combining procedure is attempted. Either a *packet merging* or *packet decoding* operation is performed: the latter can be thought of as a form of coded cooperation [14] when used in a cooperative sense or of type-II hybrid ARQ when used in a non-cooperative sense [1]. The packet decoding procedure uses a systematic, invertible block code and sends a parity packet if the first (non-parity) packet is received with errors. This allows for the use of redundancy only if required and is hence a form of incremental redundancy.

The packet merging part of the SPaC protocol is the same as the combining procedure presented in [10]; the name change is required to distinguish it from packet decoding. Packet merging is shown to be somewhat infe-

rior to packet decoding: the authors explaining this by suggesting that packet merging is essentially a repetition code. Packet merging is, in fact, not a repetition code as it makes use of the frame check sequence (used for error detection in a cyclic-redundancy check) to correct the packets. A repetition code uses a majority voting scheme and it is thus impossible to perform a decoding operation based only on two packets – exactly what packet merging does. However, with an improved packet merging scheme the SPaC protocol shows even greater potential as an error control technique for WSNs. The rest of this paper develops and analyses a significantly improved packet merging protocol which can then be used to greatly enhance protocols such as SPaC or, if simplicity of implementation is desired, to act as a standalone error-correction process (ignoring altogether the packet decoding aspect of SPaC).

3. Packet merging

This section discusses the packet merging procedure and simulation results pertaining to the gains achievable using it. It should be emphasised that packet merging is a *post-detection* combining scheme. Modern transceivers used in WSNs generally cannot implement *pre-detection* combining schemes such as MRC as no access to the analog portion of the receiver is given. The theoretical underpinnings of packet merging are examined in particular in [10] but also in [15–17]. A receiver decision model (in which multiple retransmissions are combined on a pair-by-pair basis) is fully described along with a probabilistic model of the merging scheme. The analysis is mainly concerned with the overall throughput of a system using packet merging where it was compared to the upper bound of type-II hybrid ARQ. The simulation to follow examines the scheme from the perspective of the achievable gains, i.e. the saving in SNR required to achieve a particular PER. This is motivated by energy being a concern in WSNs. In particular it offers motivation for the need of an improved merging algorithm by quantifying the gains achievable. In later sections, an energy analysis is carried out along with results on a real system, the latter being of interest as previous papers have only considered an analytical analysis.

3.1. Procedure

For packet merging, the destination accepts two packets of the same type and creates a merged error mask which will be referred to as an "Ambiguity Vector" (V_A):

$$m'_S \oplus m'_R = e_S \oplus e_R = V_A. \quad (1)$$

The variables m'_S and m'_R represent the two received, corrupt data vectors from the Source (S) and Relay (R) respectively (or both from the source depending on the specific protocol used), and e_S, e_R are, respectively, the corresponding error vectors. The ambiguity vector V_A shows in which bit positions the two packets differ. It will contain a 1 in the positions where an error has occurred in either of the data packets, and a 0 where no error has occurred or where one occurred in the same bit position of both packets. This last case is not correctable using packet merging and is referred to as a *hidden error* (Section 6.1). The Hamming weight of the ambiguity vector (denoted by $w(V_A)$) is the total number of ones in the ambiguity vector and is used to determine the feasibility of the search, as will be seen.

Once V_A is obtained, it is then used with one of the packets, m_{ch} (i.e. the chosen packet m'_S or m'_R), and the Frame Check Sequence (FCS) to try to obtain the original packet. This is done by modifying m_{ch} in each bit position where there is a potential error (as indicated by V_A), and checking whether the resulting packet matches the FCS.

In the SPaC protocol the chosen packet, m_{ch}, is randomly selected and an exhaustive search is used where every possible error candidate is tested – henceforth referred to as the *Brute-force* method. This leads to the following possibilities: (i) A unique error pattern yielding the correct FCS is obtained in which case a success is declared, (ii) Multiple candidate error patterns yield the correct FCS in which case a failure is declared, and (iii) none of the candidate patterns match the original error. This last event is only possible in the case of hidden errors. The purpose of exhaustively testing every possible error candidate is to identify the second scenario, in which case it is impossible to tell which candidate is the correct one.

The maximum allowable value of $w(V_A)$, denoted by $w(V_A)_{max}$, is influenced by the computational cost of the merging operation and the percentage of hidden errors that might occur. The value of $w(V_A)_{max}$ strongly influences the performance of the packet merging algorithm. With a Brute-force search, however, the value is limited due to computation complexity. As will be seen, a reduced search is possible with packet merging leading to a significant reduction in energy consumption or improved throughput performance through an ability to increase the value of $w(V_A)_{max}$.

3.2. Performance

This section studies the performance of packet merging. It does this through the development of a simulation which is used, in particular, to study the effect varying $w(V_A)_{max}$ has on the overall combining gain, but also to compare these results with the optimal pre-detection combining performance of MRC. The simulation was used as it offered a more controlled environment in which to study packet merging (e.g. in a slow Rayleigh fading with no interference) and to help motivate the need for the improved merging procedure developed in Section 4. Practical results on packet merging will be studied later in the paper (Sections 8 and 9).

As has been noted previously, the type of combining generally assumed in theoretical papers on cooperative communications is MRC which, unfortunately, is normally not possible on resource-constrained devices. As a benchmark, however, the performance of an MRC system is also considered. It should be stressed that MRC is only being used to help gauge the performance of packet merging. The latter is not meant as a replacement for MRC but as a substitute when more powerful combining techniques, such as MRC, are not feasible.

3.2.1. Comparison of ARQ schemes

For the following analysis it is informative to consider four simple cases. The first case is referred to as *traditional ARQ (tradARQ)*. In this case the source node repeats the information when the original packet received at the destination is incorrect. If the second packet is correctly received a success is declared, if not a failure is declared. The second case is referred to as *traditional ARQ with combining (tradARQC)*. In this case the source node again resends the data, if an error occurs. However, upon receiving an erroneous packet on the second attempt, rather than declaring a failure, a combining (either MRC or packet merging) operation takes place. If the combining operation fails only then is a failure declared. The third and fourth cases are analogous to the first and second, respectively, and are referred to as *cooperative ARQ (coopARQ)* and *cooperative ARQ with combining (coopARQC)*. For these cases, rather than the source retransmitting the data the relay is requested to do so, provided it has correctly decoded the packet from the source. For the case when the relay has not correctly decoded the packet it is the source that retransmits.

To help simplify the analysis a total of two transmissions are allowed: the original transmission and one retransmission. If multiple retransmissions are required the procedure described in [10], where combining is carried out over successive pairs, can be used to extend this further if required.

3.2.2. Analytical model

In [18] an interesting analytical analysis is carried out where the Packet Error Rates (PERs) obtainable using an MRC system are presented for each of the four cases described earlier. The analysis presented is of relevance for this paper as it enables a comparison between the gains achievable with packet merging and those of MRC. The modulation scheme considered in [18] is Binary Phase Shift Keying (BPSK) and a packet size of 1080 bits is used. The channel model used is a slowly varying Rayleigh fading channel and the SNR values quoted for a given link are thus average values, with the instantaneous value for any given packet varying accordingly. Due to the slow fading nature of the channel the instantaneous SNR is assumed to remain constant over the course of two transmissions.

To assist in dealing with the relevant quantities, the following notation is used in the next section: $\bar{\gamma}_{i,j}$ is used to represent the average SNR between nodes i and j, the variable $\gamma_{i,j}$ is used to represent the instantaneous SNR, and $i,j \in \{s,r,d\}$, which represents the source, relay and destination channels, respectively. The subscripts in $\bar{\gamma}_{i,j}$ will be dropped when multiple quantities are equal and no confusion results, e.g. $\bar{\gamma}_{s,d} = \bar{\gamma}_{r,d} = \bar{\gamma}$.

3.2.3. Simulation

The simulation consisted of a three node architecture: a Source (S), a Relay (R) and a Destination (D). The channel model used was slow Rayleigh flat fading with Additive White Gaussian Noise (AWGN). The channel gain remained constant during the time to transmit two consecutive packets. A simplified block diagram of the simulated environment is shown in Fig. 1. A Pseudorandom Noise (PN) sequence was used to create a data packet which was then modulated using BPSK. The modulated signal was transmitted over the channel. The received signal was then demodulated and the accuracy of the packet determined by comparing it with the original data. If the packet was received correctly no further action was taken. However, if the packet was received incorrectly then the data was retransmitted on the same channel in one case (traditional ARQ), and on an independent fading channel in the second case (cooperative ARQ).

The simulation was validated by comparing its output to results obtained in [18]. This was achieved by setting the packet size to 1080 bits and simulating the transmission of 100,000 packets for each value of average SNR. As with the analytical model developed in [18] $\bar{\gamma}_{s,d} = \bar{\gamma}_{r,d}$, both of which were varied from 0 to 20 dB. The source–relay channel was assumed to be perfect for the cooperative scenario in order to reduce the complexity of the simulation. A comparison was carried out using only the tradARQ case and coopARQ case as the simulator was not designed to implement maximal-ratio combining. Fig. 2 shows the results of the tests where it can be seen that the simulation curves agree closely with those of the analytical model.

Having validated the simulation, packet merging was then implemented. Plots of PER versus average SNR for two different values $w(V_A)_{max}$ are shown in Fig. 3a and b. From these figures it can be seen that two different types of gains are observable: combining gain, obtained by enhancing either traditional ARQ or cooperative ARQ using a combining process, and diversity gain, obtained through the use of independent fading channels. Each gain is a difference in the SNR required for a given PER. The former, however, is between the ARQ and ARQC curves for each

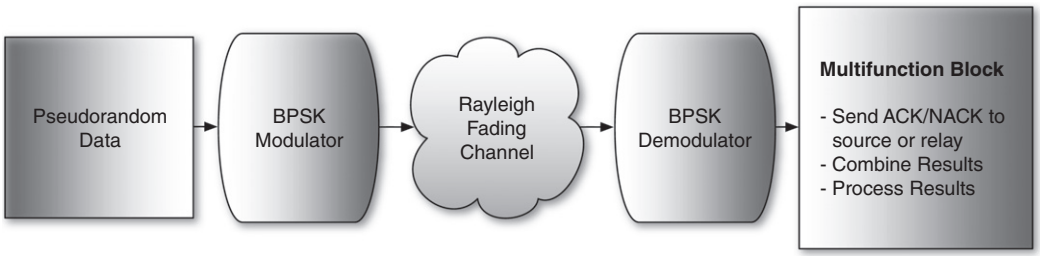

Fig. 1. Simplified block diagram of the simulation setup using BPSK.

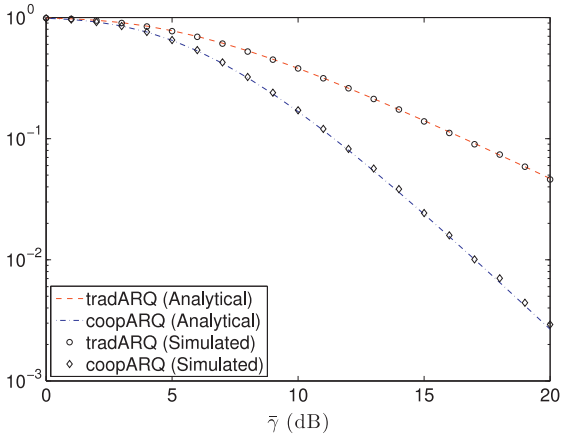

Fig. 2. Comparison of the simulated versus the analytical results for a BPSK system over a slow Rayleigh fading channel with a packet size of 1080 bits. The source–relay channel was assumed to be perfect.

scheme whereas the latter, between the traditional ARQ scheme and the cooperative ARQ scheme (without considering combining). This paper is concerned mainly with the gain due to combining and any references to the gains obtainable will be with respect to this type of gain - diversity gains will be explicitly referred to as such.

The combining gains are determined not only by $\bar{\gamma}_{s,d}, \bar{\gamma}_{r,d}$ and $\bar{\gamma}_{s,r}$ but also by the size of $w(V_A)_{max}$. Larger values of $w(V_A)_{max}$ offer increasingly large gains (as it permits the correction of a larger amount of errors); however, the computational complexity also increases. The gains obtainable for $w(V_A)_{max} = 10$ were determined using the results shown in Fig. 3a. As also noted in [18], traditional ARQ achieves a larger combining gain than cooperative ARQ. However, the overall gain, which includes a diversity gain, is much larger in the latter case. The gain improvement obtained when using $w(V_A)_{max} = 15$ is evident from Fig. 3b.

The effects of $w(V_A)_{max}$ on the combining gains is shown in Fig. 4. For traditional ARQ the gains obtained for $w(V_A)_{max} = 10$ and $w(V_A)_{max} = 15$ are 1.39 dB and 1.86 dB respectively. In the cooperative case they are .42 dB and .61 dB with the perfect source–relay channel. Of course, if diversity is taken into account the overall gains of the cooperative system (compared to tradARQ) would be those shown in Fig. 4 *plus* a gain of approximately .3–6.5 dB due to diversity – higher values of SNR leading to larger gains.

The assumption of a perfect source–relay channel means that cooperation is always available to be leveraged (where necessary) and that the system never resorts to traditional ARQ. The combining gain of the traditional system is always larger than that of the cooperative system for a perfect source–relay channel (Fig. 4). The simulations show that if an imperfect channel is assumed the combining gain of the cooperative case would be larger than .42 dB (for the same value of $w(V_A)_{max}$) in the regions of mid-SNR tending towards that of the traditional ARQ schemes in the limit of high SNR as the system reverts back to traditional ARQ on occasion. This compensates somewhat for the loss of diversity due to the imperfect source–relay channel.

The larger cooperative gain obtainable for the imperfect source–relay channel can also be understood as follows. In tradARQ and coopARQ a packet is only declared in error if *both* the first and the second transmissions are in error. In tradARQC and coopARQC a packet is only declared in error if both the first and second transmissions are in error *and* the packet merging operation fails. If packet merging always fails then the PER of each scheme reverts back to that of no combining, i.e. the tradARQ and coopARQ cases, respectively. Fig. 5 shows a plot of the percentage of ambiguity vectors, for each SNR, with a Hamming weight less than or equal to 15; both the traditional case and the cooperative case are shown. For each value of SNR, the cooperative situation produces, on average, 9.5% fewer ambiguity vectors with $w(V_A) \leqslant w(V_A)_{max} = 15$ than the traditional case. The achieved gain will therefore also be less as there will be fewer occasions when packet merging is used. However, as the source–relay channel degrades from perfection, an increasing number of retransmitted packets come from the source node creating more opportunities for a successful merging operation.

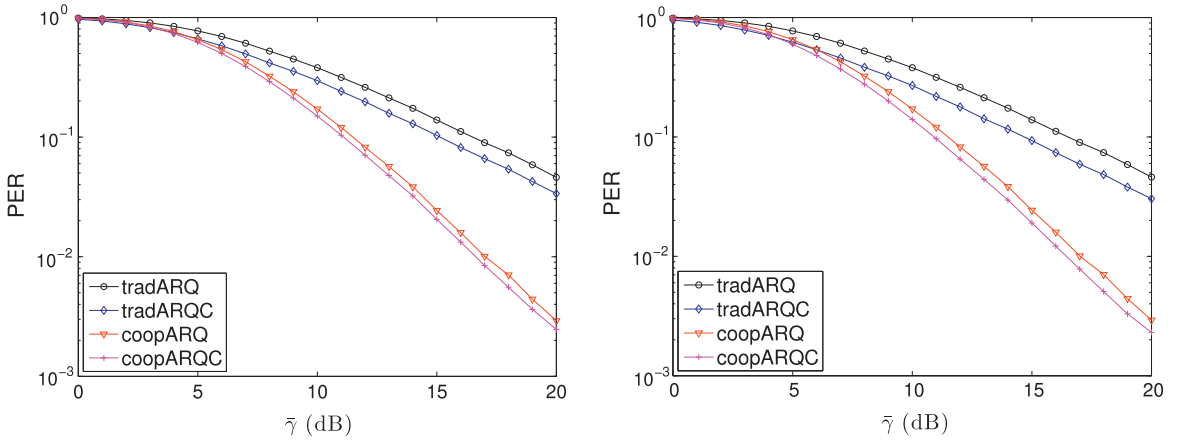

Fig. 3. Packet error rate curves for a BPSK system with a packet size of 1080 bits.

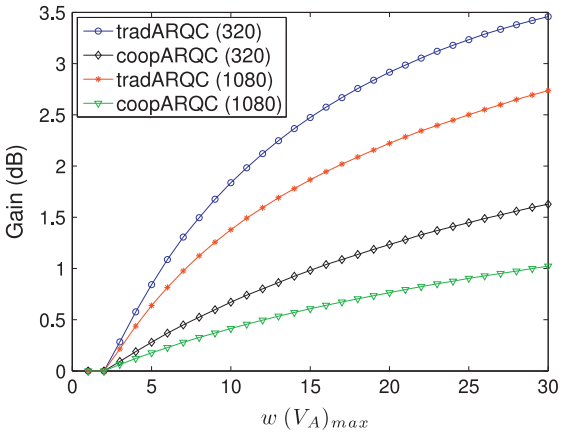

Fig. 4. Gains achievable using Packet Merging (PM) over different values of $w(V_A)_{max}$ for two different packet sizes and a perfect source–relay channel.

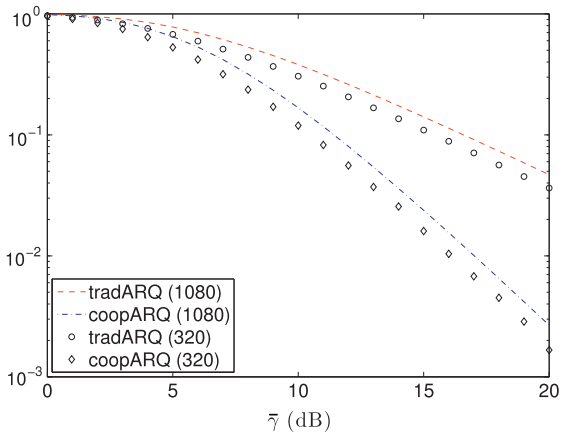

Fig. 6. Comparison of two BPSK systems transmitting over a slow Rayleigh fading channel with packet sizes of 320 bits and 1080 bits. The source–relay channel is assumed to be perfect.

From the analytical results in [18] the gain achievable with one retransmission, MRC, and a packet size of 1080-bits is approximately 2.58 dB. It is interesting to note from Fig. 4 that the gain of the traditional packet merging system with the same packet size actually exceeds that of the corresponding MRC system with $w(V_A)_{max} \geqslant 27$. In reality, however, it would not be feasible to process such a large number of errors. For more realistic levels of $w(V_A)_{max}$ the gains are not as large as MRC but it still motivates the need for a more efficient merging procedure.

A more commonly used packet size for WSNs is about 320 bits [19]. The simulation was run again, with this packet size, for both tradARQ and coopARQ (i.e. without any combining). Fig. 6 shows the resulting plots compared with those of a tradARQ and coopARQ system using a packet size of 1080 bits. The system with the packet size of 320-bits outperforms that with the larger packet size of 1080-bits. This is to be expected as the probability of obtaining one or more bits in error in a packet increases with packet

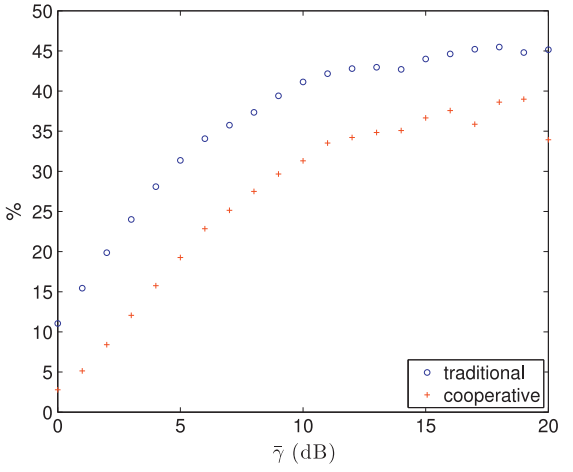

Fig. 5. The percentage of ambiguity vectors obtained for each SNR with a Hamming weight less than or equal to 15. In this case $\bar{\gamma} = \bar{\gamma}_{s,d} = \bar{\gamma}_{r,d}$.

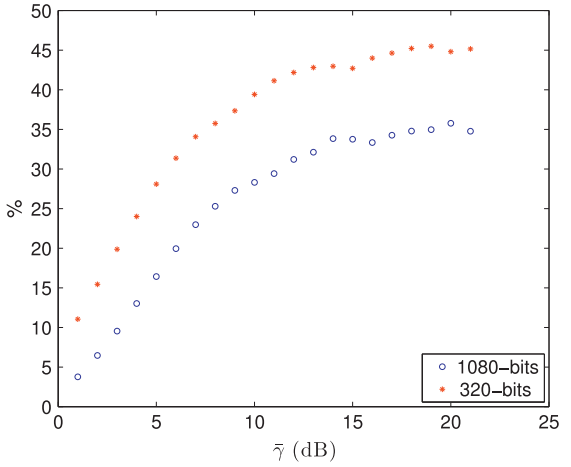

Fig. 7. Percentage of ambiguity vectors obtained using tradARQC with Hamming weights less than or equal to 15. Both 320-bit and 1080-bit cases are considered.

size. The percentage of ambiguity vectors that have Hamming weights less than or equal to $w(V_A)_{max}$ therefore increases for smaller packet sizes (as shown in Fig. 7 for the traditional ARQ case).

Fig. 8 shows the PERs obtained for tradARQ, tradARQC, coopARQ, coopARQC for a value of $w(V_A)_{max} = 15$. Interestingly, combining works better for the smaller and more commonly used packet size [19–21]. This can be seen when plotting the gains obtained over all values of $w(V_A)_{max}$ (Fig. 4).

A summary of the gains obtainable in each case is provided in Table 1. For the traditional case the gains at $w(V_A)_{max} = 10$ and $w(V_A)_{max} = 15$ are 1.84 dB and 2.47 dB respectively. This can be compared with the gains of 1.39 dB and 1.86 dB for the system with a packet size of 1080-bits. In the cooperative case the gains are .67 dB and .98 dB with a perfect source–relay channel. This can be compared with the gains of .42 dB and .61 dB for the cooperative case using a packet size of 1080-bits and a

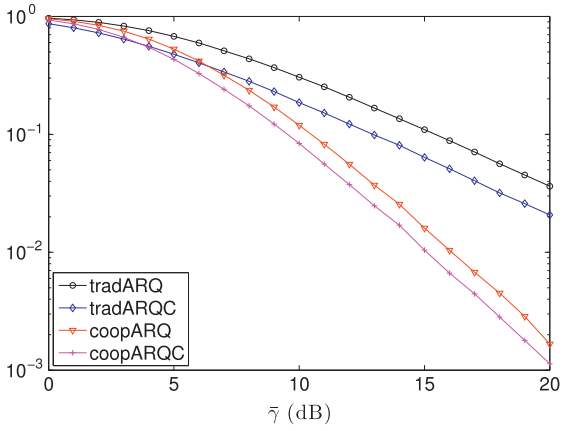

Fig. 8. Packet error rate curves for a BPSK system with a packet size of 320 bits and $w(V_A)_{max} = 15$.

Table 1
Gains (in dB) for each method used when two transmission attempts are permitted.

Packet Size	1080 bits		320 bits	
$w(V_A)_{max}$	10	15	10	15
Traditional ARQ	1.39	1.86	1.84	2.47
Cooperative ARQ	0.42	0.61	0.67	0.98

perfect source–relay channel. These results motivate a more efficient merging procedure than the simple Brute-force discussed earlier. The next section discusses ways of achieving this.

4. Improved Packet Merging (iPM)

The authors developed two improved packet merging algorithms in [8], where preliminary results were presented. The first algorithm, referred to as Simple Packet Merging (sPM), chooses the same packet as m_{ch} on each run of the algorithm possibly based on some *a priori* topological information. As there are a total of $2^{w(V_A)} - 2$ possible error candidates the algorithm runs through each possible candidate (e_c) from 1 to $2^{w(V_A)} - 2$ in unit increments, referred to as an "incremental search", modifying m_{ch} and determining whether the following is true:

$$\mathrm{CRC}(m_{ch} \oplus e_c) \oplus \mathrm{FCS} = 0, \tag{2}$$

where CRC(x) calculates the FCS for the value x. In contrast to the Brute-force search method sPM will actually stop searching once a packet is found that satisfies Eq. (2). An exhaustive search is only carried out if either no such packet is found or if the packet happens to occur at the very last possible iteration. This particular stopping method is referred to as a *conditioned-stop*.

The second algorithm, referred to as Improved Packet Merging (iPM), steps through the error candidates in a more structured manner. Rather than using an incremental search, as was done for sPM, it first tests all error candidates with a Hamming weight of one, then all those with a Hamming weight of two and so on using the conditioned-stop method (see Fig. 9). As a further improvement, a metric such as LQI or RSSI is used to choose, in real time, the best packet to use as m_{ch}. Some comments on the practical aspects of this last improvement are in order. The effectiveness of this decision will naturally depend on the particular metric used, the choice of which is limited by the capabilities of the device employed. Although the platform used in the experimental setup (Section 5) provided access to both LQI and RSSI, the specific implementation itself was somewhat of a limitation providing only an average value over the first eight symbol periods of the packet. Future transceiver designs may provide more detailed information, such as one RSSI value per symbol, which should significantly improve the decision making process.

sPM search	iPM search
0000...0001	0000...0001 } w(e_c) = 1
0000...0010	0000...0010
0000...0011	0000...0100
0000...0100	:
:	0000...0011 } w(e_c) = 2
1111...1100	0000...0101
1111...1101	0000...1001
1111...1110	:
1111...1110	1111...1110 } w(e_c) = w(V_A) - 1

Fig. 9. Comparison of the search methods for sPM and iPM.

A summary of the three outlined schemes is now provided for comparison:

1. Brute-force
   (a) Selection criterion:
   - Random.

   (b) Search method: Incremental and exhaustive.
2. sPM
   (a) Selection criterion:
   - *Case 1:* Always choose m'_S.
   - *Case 2:* Always choose m'_R.

   (b) Search method: Incremental and conditioned.
3. iPM:
   (a) Selection criterion:
   - If $M_{m'_R} > M_{m'_S}$, choose m'_R; else choose m'_S.

   (b) Search method: Improved and conditioned.

where $M_{m'_x}$, $x \in \{S, R\}$, is the chosen metric RSSI, LQI or some other available quantity. It was shown in [8] that iPM achieves about a 70% improvement over the Brute-force search and a 21% improvement over sPM; we will therefore only consider iPM from this point forward.

4.1. Incremental CRC

Despite iPM's significant improvements over a complete search it is still desirable to reduce the computational complexity of the packet merging algorithm further. An

incremental CRC technique is described in [11] which is essential for the efficient operation of the merging procedure and is used in the experiments described. Further information can be found in [11].

5. Experimental setup

The experiments carried out in [8] were extended in this work to examine the new improvements to the algorithm (to be described in Sections 7.2 and 7.1). In this section the experimental setup used in both [8] and the current work is described in more detail. The setup used a three node architecture: a Source (S), a Relay (R) and a Destination (D). The particular devices used were the popular Tmote Sky motes [5] which incorporates a 2.4 GHz spread-spectrum radio compliant with the IEEE 802.15.4 standard [22]. Three different experiments were undertaken at three separate transmit power levels: −25 dBm, −15 dBm and 0 dBm (referred to as Ex1, Ex2 and Ex3 respectively). The main aim of each setup was to produce packets with large numbers of errors in order to test packet merging in these scenarios. To this end, the nodes were positioned such that the interuser channel (i.e. from S to R) was good, but the uplink channels (i.e. from S to D, and R to D) were bordering on the sensitivity levels of the receiver. The different power levels used allowed for increasing distances between the nodes while meeting the above requirement. The SNR at the receiver is more strongly influenced by the surrounding environment if the transmit power and, consequently, the distance are increased.

A number of different environments were tested for the experimental setup before deciding on a cluttered utility room that was found to give the most useful data (i.e. the largest packet error rates). This room contained metal shelves, a number of toolboxes, a washing machine, two bicycles and other similar items. The source and relay nodes were placed close to one another within this room and were not moved throughout the entire experiment. The placement of the receiving node was varied for each transmission power: as the power levels were increased it was necessary to increase the separation distance between the receiving node and the source–relay pair. This was done in order to maintain a situation where the receiver operated at the fringe of its radio range (from both source and relay) and involved using node separations from 1 to 30 m. At the smaller distances the communication would have been dominated by a large LOS component while at large separations (involving the receiver being placed in separate rooms), there would have been no LOS but increased levels of multi-path (due to changes in the surrounding environment).

The number of packets sent for each experiment was 500,000. The data sent was test data and was therefore available to the receiver for comparative purposes. This allowed, for example, an explicit evaluation of e_S and e_R which normally would not be possible. The size of the transmitted packets was dependent upon whether the source or relay was transmitting. As the entire source packet (header + payload + FCS) was retransmitted by the relay, the relay's packet was necessarily 13 bytes longer than the transmitters (relay's header + entire source packet + relay's FCS). For these experiments, therefore, the source packet contained 27 bytes (216 bits) in total whereas the relay's packet contained 40 bytes (320 bits). The combining operation was carried out over the full 27 bytes of the source packet (and the 27 byte payload of the relay packet). The experiments described in Section 3.2 further considers *source* packet sizes of 40 bytes (320 bits). Although the source header was included in the combining operation (for completeness), whether or not this is possible, or necessary, will depend upon the particular network topology, the protocols used, and the particular layer of the protocol stack in which combining is attempted.

The protocol used was essentially the simple Adaptive Decode-and-Forward protocol (simple AdDF) described in [23] (with some slight modifications): the source node broadcast its message (m_S) to R and D in phase one. The relay having decoded each received message (m_R), only forwarded it in phase two if its CRC check passed. Upon reception, the destination carried out another CRC check on the received versions of m_S and m_R. If either passed then no further action was taken; however, if both failed then the error correction algorithm was only applied if the following was true:

$$(m'_S \oplus m'_R \neq 0) \quad \text{and} \quad (\mathrm{FCS}_S \oplus \mathrm{FCS}_R = 0), \tag{3}$$

where FCS_S and FCS_R represent the frame check sequence of the source and relay messages respectively. In the basic scheme described in this section[2] $\mathrm{FCS}_S = \mathrm{FCS}_R$ and will therefore be referred to simply as FCS. Corrections were not attempted either if the Hamming weight of the ambiguity vector was greater than 10; i.e. $w(V_A) > 10$. In general, however, the maximum allowable value of $w(V_A)$, denoted by $w(V_A)_{max}$, is influenced by the computational cost of the merging operation and the percentage of hidden errors that might occur. The value of 10 was chosen in this case as the experimental results showed a similar number of packets (approximately 2.2%) containing hidden errors for each $w(V_A)$ ranging from 1 through 10. For larger values of $w(V_A)$, up to around 15, the number of hidden errors increased only slightly. Beyond this value, however, there was a significant increase. Also, as the computational complexity of iPM is less than the merging used in SPaC (where $w(V_A)_{max} = 6$) it is possible to allow for a larger value of $w(V_A)_{max}$. In some circumstances values of $w(V_A)_{max}$ larger than 10 may be possible; this will be discussed further in Section 3.2.

Results pertaining to iPM itself obtained from these experiments can be found in [8]. The rest of this paper makes use of these results along with new results to further improve and analyse iPM and packet merging in general. Any modifications to the setup will be described as needed. Before looking at the benefits of packet merging it is informative to understand some of its drawbacks; the experimental results will be used to help quantify these next.

[2] Section 7.1 enhances the protocol by using two FCS polynomials.

6. Packet merging issues

As with all error correction schemes there is a limit to the amount of errors that can be corrected. This section looks at the concepts of *hidden errors* and *false-positives*. While a method is described for dealing with false positives, the concept of hidden errors is only briefly touched upon here but is more thoroughly dealt with in [10,11].

6.1. Hidden errors

As mentioned previously, the presence of hidden errors is an issue for packet merging. These occur when the same bit is in error in both packets used for combining (i.e. m'_S and m'_R) and therefore cancel each other out. Hidden errors are not correctable by packet merging in its basic form and in fact cause the algorithm to run through all possible iterations before determining an inability to decode. This situation is clearly wasteful of resources and some method of at least detecting their presence is desirable.

Fig. 10 shows the percentage of ambiguity vectors with hidden errors for a particular weight obtained in experiment 3 (i.e. 0 dBm transmit power with corresponding large distance between source and destination) of Section 5. Setting $w(V_A)_{max} = 10$, the percentage of hidden errors that would have occurred within the allowed range is approximately 2% whereas setting $w(V_A)_{max} = 15$ increases it slightly to 3.9%. The actual number of hidden errors will naturally depend on the particular environment the nodes are placed in and the likelihood of obtaining larger values of $w(V_A)$. In [11] the number of hidden errors and false positives (Section 6.1.1) were shown to be less of an issue in choosing the value of $w(V_A)_{max}$ than the computational efficiency involved in the Brute-force search.

An undetected hidden error means the algorithm will have to test every possible error candidate before realising an inability to decode. Although this is expensive in terms of both energy and time the Brute-force search method actually does this anyway, regardless of the presence of hidden errors (the advantage, of course, being its ability to detect of false-positives). Using iPM, the presence of hidden errors will simply revert the search method to that of a Brute-force search in the absence of false-positives. The improved efficiency of iPM will offset this cost and the particular value of $w(V_A)_{max}$ will depend on the setup and environment. In general the greatly improved efficiency of iPM permits larger values of $w(V_A)_{max}$ than would be possible with a Brute-force search. For the results shown in Fig. 10 $w(V_A)_{max} = 10$ seems to be a good choice although larger values may also be possible; It will be shown in Section 3.2 that the SNR gains possible with packet merging are a function of the value of $w(V_A)_{max}$ with larger values leading to a larger gains. The effects of setting $w(V_A)_{max} = 15$ will also be considered from both an energy and gains perspective.

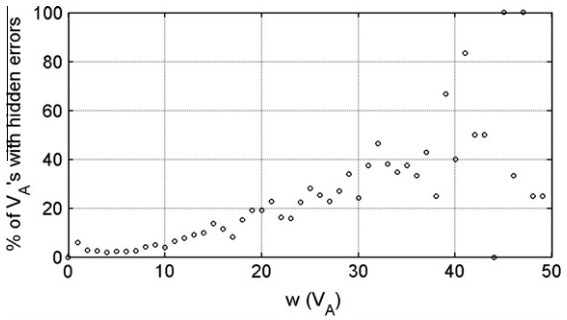

Fig. 10. For each Hamming weight, this graph shows the fraction of ambiguity vectors that contained hidden errors at a particular weight, out of the total at that weight.

Table 2 shows the average number of hidden errors (e_h) found as a percentage of a) the number of ambiguity vectors with $w(V_A) \leqslant 10$, and b) the total number of ambiguity vectors. For all three experiments the number of hidden errors with $w(V_A) \leqslant 10$ was only about 2.2%.

6.1.1. False positives

An inherent problem with any error correction scheme is when the extent of the errors is such that the received packet is a valid codeword different from the original – referred to as an *undetected error*. Although packet merging makes use of the CRC in a different way to that normally used, the same mechanism makes it possible that the search for the appropriate error candidate produces a valid codeword different from the original. Rather than viewing this as an undetected error, however, it is more accurate to refer to it as a *false-positive*, although the idea is similar. A Brute-force search considers all possible error candidates, even after one has been discovered that produces a valid codeword. The only reason for doing this is to detect the presence of false-positives and while no doubt useful, this technique consumes a significant amount of energy over a node's lifetime. On the other hand, the conditioned-stop feature, used by both sPM and iPM (with only the latter being of interest now), makes the assumption that the first generated packet matching the received FCS is the correct one. The trade-off is an inability to detect false-positives compared with a significantly reduced search time and energy cost. An understanding as to the extent of the occurrence of false-positives using iPM is therefore important.

In each of the experiments described earlier the percentage of correctly decoded packets after merging was, on average, about 99.94%. The remaining 0.06% of packets were therefore incorrectly decoded due to false positives. While this may seem like a rather large value two points should be noted: Firstly, it was obtained under adverse conditions. Secondly, the figure only relates to the instances where packet merging was actually attempted. When considered against the total number of transmissions sent the fraction of false-positives was, in fact, only 2.8×10^{-5}. Again, this value is representative of a worst case scenario and is on the order of a weak upper bound

Table 2
Number of hidden errors (e_h) found.

	Ex1	Ex2	Ex3
e_h (% of $w(V_A) \leqslant 10$)	2.1	2.3	2.2
e_h (% of total)	2.8	5.5	9.8

of 2^{-16} known for undetected errors using the CRC [24]. To use iPM in its basic form there is hence a trade-off between a large reduction in the number of iterations required over a Brute-force search operation and a small probability of producing an unrecognised false-positive.

The two main techniques used in iPM play an important role in keeping the number of false-positives to a minimum. The effect of the selection policy means that the packet with the least number of errors is chosen on average. This, coupled with the improved search algorithm, minimises the number of extra errors introduced into the packet in the early stages of the search. In the incremental search method used in sPM, the number of bit changes jumps between large and small values, increasing the likelihood of false-positives in the early stages of the algorithm. Despite these effects, it should be kept in mind that a problem with the CRC in general is that the number of false-positives (or undetected errors) will tend to increase with packet size. For larger packet sizes it may be more feasible to consider using 32-bit CRCs rather than the 16-bit variants currently used in WSNs. For the same packet size a 32-bit CRC would significantly reduce the number of false positives.

7. Further iPM enhancements

This section proposes further enhancements to iPM which improve both its efficiency and robustness. These include a reduction in the number of required iterations by one-half using a technique referred to as *parity equivalence*, and the effective elimination of false-positives using two distinct generator polynomials for the original and retransmitted packets.

7.1. Use of two generator polynomials

The number of false-positives incurred using iPM is relatively small and in most cases will not be an issue. However, in more critical situations it might be desired to reduce this number further. Although not available in current hardware designs (of the type considered in this paper), future designs may provide an option for sampling the RSSI/LQI value for each symbol of the packet (as opposed to relying only on the sample taken at the beginning – as was the case with the device used to generate the results in the previous section). It is very likely that this will not only increase the likelihood of choosing the packet with the least number of errors but also, if the bits are then arranged in order of decreasing confidence, reduce the number of false-positives while at the same time increase the search efficiency. Until this option becomes available however, another very effective method, requiring a negligible increase in memory space and complexity, is to use a different FCS for the original packet and its retransmitted version. This is achieved using a second, distinct, generator polynomial for calculating the FCS of the latter. Each candidate packet produced by iPM is then checked against the *two* received FCS' using the two generator polynomials used. Although false positives are still possible, they are significantly reduced as it is less likely that an erroneous candidate will satisfy both sequences.

7.1.1. Choice of second polynomial

While it might be tempting to choose a standard polynomial (such as CRC-ANSI) as the choice of second polynomial in general they do not produce the best[3] polynomials. In fact, they perform rather poorly when compared to the best polynomials [25]. Better polynomials with 16-bit redundancy were found in [26], for example. In [25], Baicheva et al. investigate *all* polynomials of degree 16 that are suitable for use as generator polynomials of a cyclic code. It is from this source that a second polynomial was chosen for use with the packet merging scheme.

Noting the fact that a given generator polynomial does not produce codes that perform equally well over all packet sizes, it was decided to consider a source packet size of 320 bits – a typical size for mote class devices [19]. The polynomial chosen for the packet merging scheme was therefore 0x1A801 or

$$g_2(x) = x^{16} + x^{15} + x^{13} + x^{11} + 1. \quad (4)$$

This polynomial was shown in [25] to be best for $310 \leqslant n \leqslant 325$ with a minimum distance of 4 from $105 \leqslant n \leqslant 683$. For different applications it may be necessary to choose a different polynomial. However, it would be straightforward to set up a table of polynomials and switch between each of these polynomials for different size packets. The receiver would have a copy of this table and use the length of the incoming packet to determine which second polynomial was used.

7.1.2. Experimental results

To determine the effectiveness of using two generator polynomials a similar experiment to that described in Section 5 was conducted. This time, however, the data obtained over the course of 12 separate experiments were combined and the total number of false positives calculated. For each experiment the receiver was placed in varying positions within the same room and in adjacent rooms. As was done in previous experiments, the receiver was placed on the fringe of its radio range (from both source and relay) in order to introduce as many errors as possible. To introduce increased levels of multi-path fading the receiver was also placed on a record player turntable, which was set to a revolution speed of 33 1/3 rpm, for half of the tests (more will be said about the effectiveness of this in Section 3.2).

Over the course of the 12 experiments approximately 5 million packets were sent. Of these, approximately 216,000 combining operations were attempted. To give an example of the number of errors in each packet, the empirical cumulative distribution of the Hamming weight of the ambiguity vector for one of the experiments is shown in Fig. 11. It can be seen that almost 90% of the obtained ambiguity vectors had a weight greater than 4, 70% greater than 5, 50% greater than 6, and over 40% larger than

[3] "Best" in this sense means with the lowest probability of undetected error.

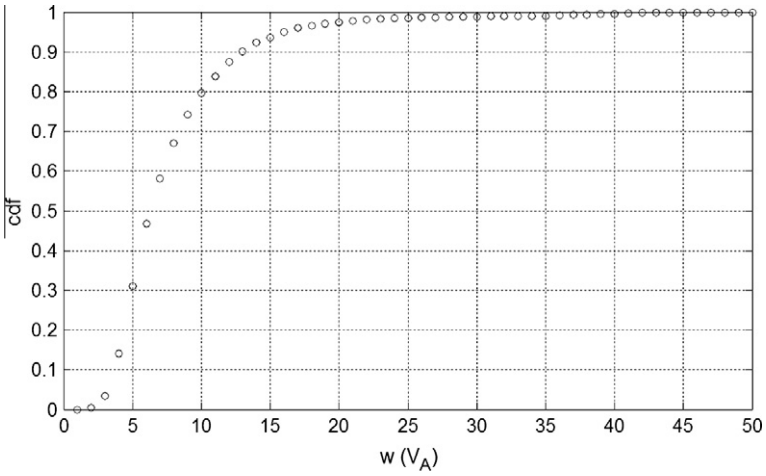

Fig. 11. Empirical cumulative distribution of $w(V_A)$ for one set of results.

7 – similar results were obtained in the other experiments. Recall that the minimum distance of the codes generated by each of the polynomials is 4. This meant that almost 90% of the combining operations had the potential to produce a false-positive, the probability increasing with larger values of $w(V_A)$.

As in the previous set of experiments, the fraction of false-positives obtained was approximately 2.8×10^{-5}. However when a second distinct polynomial generator was used by the relay node the number of false positives was reduced to 0. This suggests that the probability of a false-positive occurring, using two generator polynomials, is less than one in 5 million and is therefore *at least* an order of magnitude better than the weak upper bound of 2^{-16} mentioned earlier, which is an entirely satisfying result.

7.2. Parity equivalence

The computational complexity of iPM can be further reduced by noting that a generator polynomial with an even Hamming weight produces an FCS with the same parity as that of the message [27]. Error candidates that produce messages with the opposite parity to that of the FCS can therefore be ignored in the search. To test the effect of implementing this technique in iPM the experiments were run again with this method in place. The reduction in the number of iterations compared to a brute force exhaustive search was now found to increase to approximately 85%. This is intuitively satisfying as without the parity equivalence the value was approximately 70% [8]. A further reduction of $(100 - 70)/2 = 15\%$ thus leads to the value of 85% seen in the experiments.

The next section looks at the effects of using packet merging in a cooperative and non-cooperative ARQ scenario. It is compared with maximal-ratio combining and is shown to offer quite a significant SNR gain despite its simplicity. The methods used to show this are a combination of simulation techniques and practical experiments.

8. Energy measurements

Using iPM the overall energy consumption of the network can be reduced, not only through a decreased number of retransmissions but also through a reduction in the number of collisions and failed transmission attempts (in the case of a carrier sense protocol). To help determine the performance of iPM this section studies the energy consumption of a typical mote type architecture and discusses the energy cost of performing a merging operation; this is then compared with communication costs. The energy measurements were carried out by placing a $1\,\Omega$ resistor across the ground line of the node in order to allow measurements of the current consumed. A clean trigger was then taken from one of the I/O pins on the mote. This allowed for accurate timing measurements also. A study such as the one carried in this section is obviously not representative of all devices and scenarios. However, it is still possible to draw some general conclusions from the results obtained for the class of devices under consideration.

8.1. Energy cost of communications

The energy consumed by the node under test during a transmission, reception and merging operations were measured for the node under test. To measure the energy cost for a transmission a total of 200 synchronised voltage waveforms were collected for a particular transmission power, using the default TinyOS payload size of 28 bytes (plus 11 bytes of header). The waveforms were then averaged to reduce noise and the energy calculations carried out on the averaged waveform. This was repeated for the eight different power levels specified in the CC2420 datasheet [21]. The reception cost was also measured in a similar manner (although this was only done once as it is independent of the transmission power) and was found to be approximately 22 mA. As was expected the cost of a transmission *never* exceeded that of a reception as the maximum transmission power available was only 0 dBm.

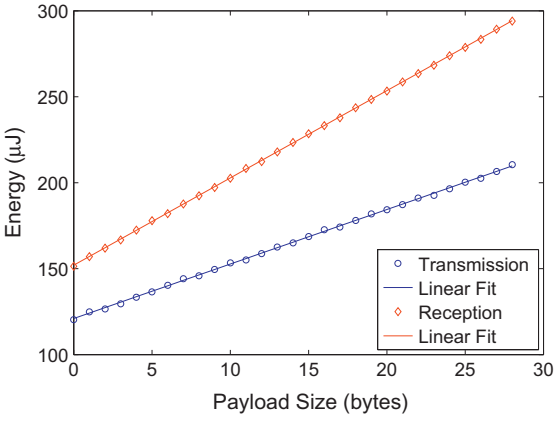

Fig. 12. The energy consumption for each extra data byte in the *payload*.

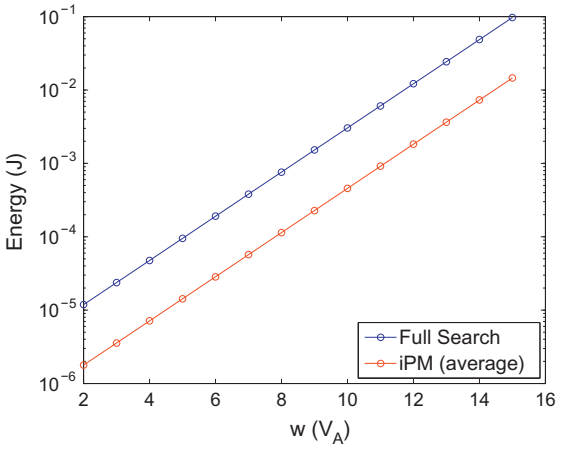

Fig. 13. Energy consumed using both a full search algorithm and iPM.

While this point may seem obvious a number of theoretical publications in the area of cooperative communications ignore the cost of a reception due to the higher power systems they consider (see, for example, [13]). For lower transmission power systems such as WSNs the cost of a reception cannot be ignored.

A comparison of the energy costs for a transmission (at 0 dBm) and a reception is shown in Fig. 12. The energy consumed for each byte transmitted (E_{txb}) is approximately 3.2 μJ. For each byte received, however, the energy cost (E_{rxb}) is 5.1 μJ. This is approximately 400 nJ per bit and 638 nJ per bit, respectively. Although not possible with the CC2420, some applications require higher transmission powers [28]. For example in [29] the NRF905 radio [20] is used at a power level of 10 dBm consuming about 693 μJ for a 32 byte packet. This is nearly 22 μJ per byte, which is about 2.5 times that of a reception on the NRF905 radio.

8.2. Energy cost of merging

Energy measurements were carried out while the node was running the merging operation using an exhaustive search. Although the methods described throughout this paper were implemented to make the search as efficient as possible, no attempts were made at optimising the source code itself. The results obtained from these measurements are therefore by no means the best obtainable. They still offer an idea of the energy savings possible using iPM however.

Fig. 13 shows an extrapolation of the energy measurements carried out for each Hamming weight. This graph was produced as follows: the nodes were instructed to perform an exhaustive search (as used in the Brute-force method) on ambiguity vectors with Hamming weights between 2 and 10 inclusive. The use of an exhaustive search ensured that a total of $2^{w(V_A)} - 2$ iterations were required for each ambiguity vector, simplifying calculations. An average of 200 voltage waveforms for each Hamming weight were then collected; this was possible as the errors in the packets were manually introduced and remained the same over the entire 200 runs of the algorithm. The time required for the algorithm to complete each run was determined by the use of an I/O pin on the mote. The energy required for each iteration of a particular run (i.e. the selection of an error candidate and the FCS comparison of the resulting packet) of the merging procedure (E_{it}) was then determined and found to be quite consistent over all tests consuming an average of 2.97 μJ with a standard deviation of about 1.7×10^{-7}J. The average value was then used to create the graph of Fig. 13 as it allowed extrapolation up to Hamming weights higher than 10 (15 in this case).

The energy consumption for iPM was then simply taken to be 15% of that of an exhaustive search as, on average, iPM reduced the number of iterations by 85%, as shown in Section 3. It should be pointed out that the energy of iPM was not directly measured as the number of iterations it requires for any one error sequence is dependent upon that particular sequence. The benefits of iPM (i.e. 85% reduction in the required number of iterations) manifest over many different error patterns of different Hamming weights as has already been shown to be the case. The actual plot of the energy consumed by iPM for each value of $w(V_A)$ would not simply be a straight line as each value of $w(V_A)$ would give a slightly different reduction in the number of required iterations. For real deployments, however, each individual value is not of interest and to simplify measurements we consider only the average reduction which is shown in the figure.

8.3. Energy savings of iPM

Using the measured results an examination of the benefits of iPM follows. Consider a simple case in which two packets are required to be sent and up to two transmissions can be attempted for each packet before it is considered to be in error. Assuming tradARQ initially, the first node transmits a packet and the energy consumed is E_{tx}. This is received by the second node and the energy consumed is E_{rx}. An acknowledgement is sent consuming E_{ackt}. The acknowledgement is then received by the transmitter consuming E_{ackr}. The total energy for one transmission is therefore:

Table 3
Energy Consumption (μJ) for the different parameters measured. Transmission power is set to 0 dBm and payload to 28 bytes.

E_{txb}	E_{rxb}	E_{tx}	E_{rx}	E_{it}	E_{ackt}	E_{ackr}	E_{tr}
3.2	5.1	210	294	2.97	127	162	793

Table 4
Comparison of energy consumption and gains for a retransmission, a Brute-force merge and iPM. The energy consumed for the merging operations are those at exactly $w(V_A)_{max}$ and hence are worst case.

	Retransmission	Standard merge	iPM	iPM
$w(V_A)_{max}$	–	8	8	10
Energy consumed (μJ)	793	760	114	456
Gain achieved (dB)	.31	1.5	1.5	1.84

$$E_{\text{tr}} = E_{tx} + E_{rx} + E_{ackt} + E_{ackr}. \quad (5)$$

Each of these values can be determined from Fig. 12. Assuming a full payload (i.e. 28 bytes) transmitting at 0 dBm the value of E_{rx} is 294 μJ and that of E_{tx}, 210 μJ. Assuming a two-byte acknowledgement the value of E_{ackt} would be 127 μJ and that of E_{ackr}, 162 μJ. The total energy for one transmission (E_{tr}) is therefore 793 μJ; this does not include the energy cost of idle listening which can potentially consume a significant amount of energy if time in this mode is not minimised. The different energy measurements are summarised in Table 3.

Assume now that two packets have already been transmitted and that both are incorrect. The decision to be made is whether to ask for a second retransmission or to perform a merging operation. The simulations show (not shown) that the reduction in the required SNR for a given PER is about .31 dB when sending three packets and about 1.84 dB when using a merging operation (for a packet size of 320-bits and $w(V_A)_{max} = 10$). These reductions are both with respect to a tradARQ system using only one retransmission. Assume for the moment, however, that the PERs are equivalent in both cases. The question then becomes: which operation consumes less energy?

If the receiver decides to ask for a retransmission the energy consumed will be equal to that shown in Eq. (5), i.e. 793 μJ. If the receiver decides to perform a merging operation it must choose an appropriate value of $w(V_A)_{max}$. When $w(V_A)_{max}$ is set to 15 then the maximum energy consumed for a packet merging operation is 97.3 mJ for an exhaustive search and 14.6 mJ using iPM (as determined from Fig. 13). It is clear therefore that with the unoptimised code considered in this case, setting $w(V_A)_{max}$ to 15 is not cost effective, in terms of energy. However, if $w(V_A)_{max}$ is set to 10 and iPM is used the *maximum* energy consumption in this case is only 456 μJ, which is more energy efficient than a retransmission. Using an exhaustive search $w(V_A)_{max}$ would need to be set to about 8, although even then it still consumes far more energy than iPM at 760 μJ compared to 114 μJ for the latter.

Further motivation to use packet merging can be obtained if the significant difference in the SNR gains between a retransmission and a merging operation is taken into account. For a reduction in the amount of energy required (i.e. from 793 μJ for a retransmission compared with 456 μJ to perform a merging using iPM) and an extra gain of (1.84 − .31 = 1.53) dB is also obtained. With regard to the different packet merging approaches it is clear that iPM outperforms a traditional brute-force approach and should therefore be used. If the value of $w(V_A)_{max}$ is set to 8 for the brute-force approach the energy cost is approximately 760 μJ with a gain of 1.5 dB (Fig. 4). Using iPM with $w(V_A)_{max} = 10$ the energy is *reduced* to 456 μJ and gain *increased* to 1.84 dB. However, it is also possible to use iPM with the *same* value of $w(V_A)_{max}$ possible with a standard merge (i.e. 8). In this case the gain is the same (1.5 dB) however the energy consumption is only 15% of 760 μJ, i.e. 114 μJ. For the cooperative scenario the extra reception required by the cooperating node makes merging even more advantageous, assuming a cooperative system is already in place. Table 4 summarises these results for the tradARQ case.

8.3.1. Potential for code optimisation

As mentioned earlier, the code used when obtaining these measurements was far from optimised. Code optimisation would likely offer significant improvements and hardware implementations even more so. It is not possible to say exactly how much improvement is possible using a hardware implementation of the algorithm but in some cases hardware implementations of algorithms have been shown to offer up to 10,000 times a reduction in energy costs over software [30]. This is highly dependent on the nature of the algorithm and whether or not it can be easily implemented in hardware. However, iPM is a simple iterative technique and might easily benefit in this respect. This would represent very interesting future work on these algorithms.

To motivate the idea of code optimisation it was noted in [11] that the cost of a merging operation with $w(V_A)_{max} = 6$ is roughly only 7% of that of a transmission with a 29 byte payload and 2% with a 128 byte payload. These measurements were carried out on a CC1000 radio operating at a frequency of 433 MHz and a power level of −15 dBm. The system therefore used less power than the CC2420 at a transmission power of 0 dBm [31,32] and the latter can therefore be used as a loose reference. Using Figs. 12 and 13, therefore, the cost of a transmission for a 29 byte payload would be 297.4 μJ. The energy cost of a merging operation using the values quoted in [11] would therefore be approximately 21 μJ. The measurements shown in Fig. 13 show the energy cost for the same operation at almost an order of magnitude larger (i.e. 192 μJ at $w(V_A)_{max} = 6$). While hardware costs may in some way be responsible for this difference, it is more likely the use of more efficient code used in [11].

8.3.2. Idle listening and collisions

If idle listening is now taken into account the situation changes somewhat. Each time the radio finishes transmitting it must wait a certain period of time for a response.

Table 5
Subset of transmission power output levels determined practically that were used in the PER experiments. The total number of packets sent for each transmission power is also shown.

Output power (dBm)	−24.73	−18.55	−15.27	−13.01	−10.22	−8.63	−6.8
Total No. of packets sent ($\times 10^6$)	1	1.6	2.2	4.6	6.4	8.0	8.5

Assuming time of flight from transmitter to receiver is negligible the length of time the receiver must wait after the NACK is sent is equal to the time spent receiving the NACK at the transmitter. For CSMA based networks there is also the likelihood of not being able to transmit immediately. A random backoff timer is then used adding another E_{rb} (μJ) of energy, the value of which increases the more transmissions are occurring in the network. This means that a larger value of $w(V_A)_{max}$ now becomes feasible with iPM and the current non-optimised code. Coupled with the fact that the reduction in PER by retransmission is very small (as verified by simulations but not shown) iPM is certainly a more beneficial option.

IPM becomes more attractive from an energy-saving and throughput perspective when one considers that, in multiple access networks interference and collisions are a major issue in many cases. By choosing to combine rather than retransmit increased levels of interference and collisions can be avoided. The less channel use taken up by retransmitted packets the more the network is free to transmit new data and the less probable collisions will be. This in itself represents interesting future work as it will save energy and increase throughput.

9. Practical results

An experiment was setup up in order to determine the error-correction performance of iPM on a real system. Each of the four situations simulated in Section 3.2.1 were considered (i.e. tradARQ, tradARQC, coopARQ, coopARQC) and the value of $w(V_A)_{max}$ set to values of 10 and 15, as was emphasised for the simulations. A number of issues existed in trying to obtain PER curves in the practical experiments. Most significantly, it was very difficult to obtain an accurate estimation of the received SNR. Instead, the PER was plotted against the transmission power rather than $\bar{\gamma}$.

In order to emulate the slow Rayleigh fading channel assumed in the simulations and analytical model the receiver was placed on a record player turntable which was set to revolve at 33 1/3 rpm. The transmit node was placed about 10 m away and remained at this distance for each different transmit power level used. The large distance between the transmitter and the centre of the record player meant that even though the receiver was moving relative to the transmitter the effect on the average received SNR was minimal. The antennas on the Tmote sky nodes are omnidirectional which allowed the node to be rotated in this manner. To compensate for the periodic nature of the rotation a random timer was used on the transmitter for each transmission.

In the simulations the PER was obtained over SNR values ranging from 0 to 20 dB, in 1 dB increments. This was not possible for the practical experiments. In order to obtain a sufficient number of incorrect packets it was necessary to set the receiver node on the fringe of its radio range for the lowest transmission power used. This allowed incorrect packets to be obtained at higher power levels with a reasonable number of transmissions. Resolution was the main issue in choosing the lowest power to use. There is almost a 20 dB increase in output power between the second and third lowest settings of the transceiver used. This is equal to the *full* SNR range used in the simulations. The difference in transmission powers was smaller at higher transmit powers. However, the highest transmission power available (0 dBm) was used for the acknowledgements to ensure they were received correctly. If the nodes were set up in such a way as to obtain errors in this power range then the acknowledgements would also contain errors. A tradeoff was therefore required as to which power levels to use. The values chosen are shown in Table 5 and offered the best trade-off between fine-grained control and correct acknowledgements. This table also shows the total number of packets sent, over multiple trials, for each transmission power.

9.1. Results

The results for the traditional ARQ case (using only a source and destination node) with $w(V_A)_{max} = 10$ and $w(V_A)_{max} = 15$ are shown in Fig. 14a and b respectively. These results were obtained using a source packet of 216 bits in length. As is generally the case, the practical results proved a lot more difficult to obtain than the simulated values. The main issue was the extremely large number of packets required at high transmission powers to produce meaningful PER values (Table 5). This meant that the experiment had to be run over the course of a number of days and required several changes of batteries during this time. Although the nodes were positioned carefully unavoidable changes occurred within the surrounding environment over this period.

Another issue with the practical setup was that the limited number of usable transmission powers meant it was difficult to obtain smooth curves. This made estimating the SNR gains rather error prone. To help quantify the increase in effectiveness for larger values of $w(V_A)_{max}$, rather than using SNR gain therefore, it is easier to consider the average ratio of the PER values at each transmission power. For $w(V_A)_{max} = 10$ the mean ratio of the PER values is approximately 3.7 whereas for $w(V_A)_{max} = 15$ it is 6.5, clearly showing the benefits of using larger values of $w(V_A)_{max}$. These experiments were carried out using only one generator polynomial for the CRC. This inevitably introduced false-positives into the results. For the larger size of $w(V_A)_{max} = 15$ the percentage of false-positives obtained was of the order of about 1.2%. Naturally, most of these occurred at the lower transmission powers with very few, if any, occurring at upper transmission powers. It

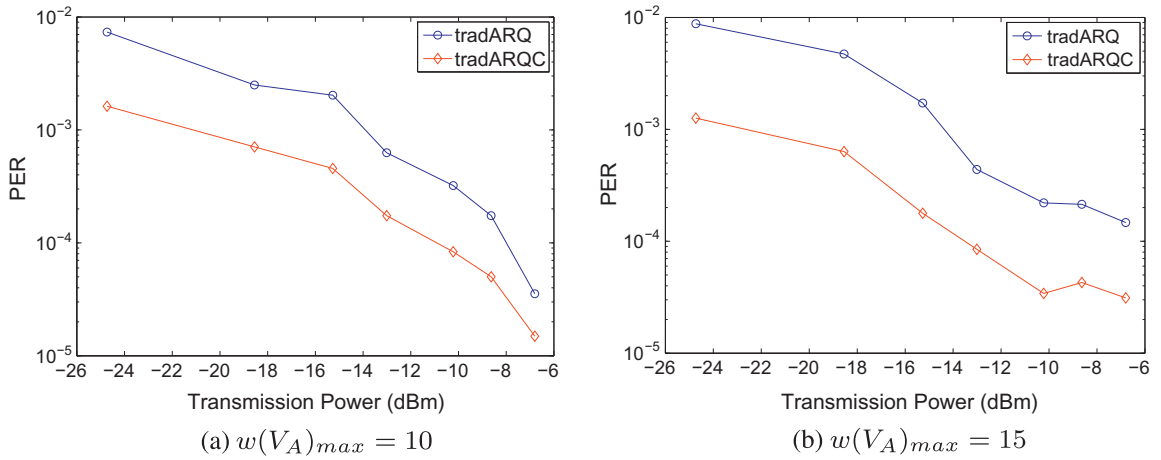

Fig. 14. Practical results. These results were taken with two nodes only, i.e. tradARQ (C), and a source packet size of 216 bits.

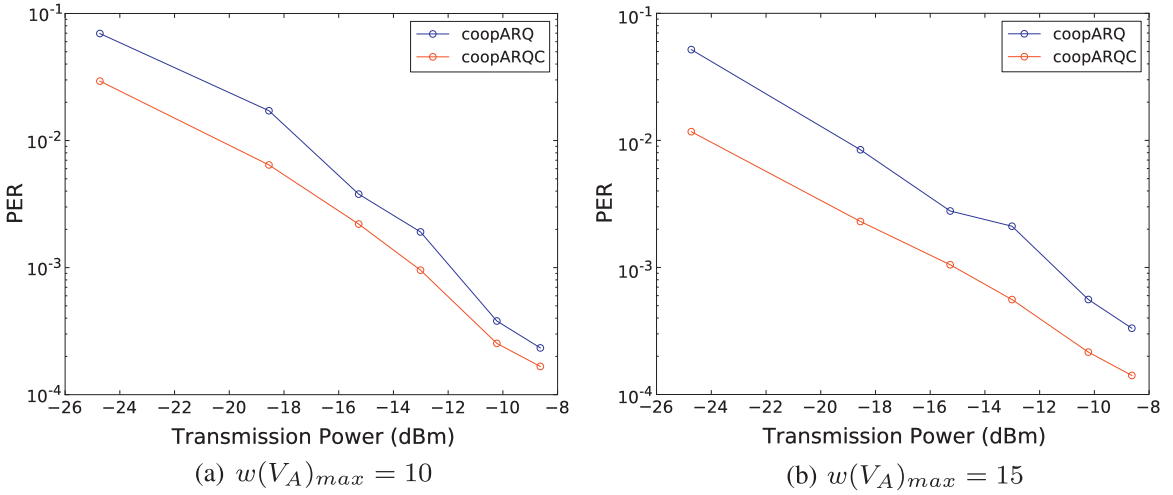

Fig. 15. Practical results. These results were taken using three nodes, i.e. coopARQ (C), with $w(V_A)_{max} = 10$ and a source packet size of 320 bits.

should be kept in mind, however, that in a well designed network the average SNRs at the receiving node will, more often than not, be closer to those produced by the higher transmission power levels in this experiment. Regardless, using two generator polynomials would effectively eliminate the false-positive problem, as was shown in Section 7.1.

The results for the cooperative ARQ case (using a source, relay and destination node) with $w(V_A)_{max} = 10$ and $w(V_A)_{max} = 15$ are shown in Fig. 15a and b respectively.[4] These results were taken with the source packet set to 320 bits in length. Again the ratio of PERs for each transmission power was used as a metric of success. For the first case, packet combining reduced the PER by an average factor of 1.94. For the second case packet combining reduced it by an average factor of 3.2. The lower relative values to the traditional case are to be expected due, to some extent, to the longer packet size. More significant, however, is that the setup emulated a cooperative system with a very good source–relay channel. This type of system was shown to produce a smaller combining gain as diversity becomes more significant. The values produced are still quite significant and clearly show the benefits of using packet combining in both cooperative and non-cooperative scenarios.

10. Discussion

A highly improved implementation of a simple packet combining scheme referred to as *improved Packet Merging (iPM)* has been developed in this paper. It has been shown that even though complex combining schemes such as maximal-ratio combining are currently not possible on resource-constrained wireless sensor networks, it is still possible to achieve quite significant performance gains using the simple techniques proposed. This section concludes

[4] Due to the difficulty in obtaining these results the highest power level was omitted as this did not significantly affect the results.

the work carried out in this paper and provides some discussion and potential future work.

10.1. Conclusions and future work

In Section 3 Improved Packet Merging (iPM) developed by the authors [8] was reviewed and further improved upon. This algorithm not only intelligently chooses, in real time, which packet is likely to have less errors, but also carries out a more efficient search for the correct packet. The improvements made included using the parity equivalence technique where it was observed that the parity of the Hamming weight of the message should be equivalent to that of the Frame Check Sequence (FCS) accompanying the message. Use of this particular technique was shown to reduce the required number of searches by one-half. Experiments then showed that by using iPM more than an 85% reduction in the required number of iterations compared with a standard brute-force search are obtainable. The trade-off using this method, however, was a slightly increased probability of false-positives. For the packet size used, the number of false positives occurring was shown to be quite small when using iPM in its basic form. Even still, to satisfy the needs of more data critical applications a technique was developed whereby a second generator polynomial was used with the original packet's pair. This technique was shown to effectively eliminate false positives. In iPM a packet selection metric such as RSSI/LQI is used for packet selection. It was noted that future chip designs may provide a means of sampling the RSSI/LQI for each symbol of the incoming packet. It is very likely that this would significantly improve the selection policy of iPM as the current algorithm relies on one sample only (an average RSSI value taken over the first 8 symbol periods of the packet) to provide accurate information about the rest of the packet. This should not only increase the likelihood of choosing the packet with the least number of errors but also, if the bits are then arranged in order of decreasing confidence, reduce the number of false-positives and increase the search efficiency. This also represents interesting future work.

It will be highly beneficial to use iPM to enhance protocols such as SPaC. In these cases iPM will not be used as a stand-alone device but in conjunction with other forms of combining such as packet decoding. As packet merging has already been implemented alongside packet decoding [11] it is relatively straight forward to retrofit the existing architecture with iPM. Protocols such as this will then be more energy efficient (Section 8) and will lead to larger gains than with iPM alone. To keep things in perspective however, it is interesting to note that the coding gain obtainable by the (7,4) Hamming error correction code is approximately .45 dB at an SNR of 10 dB [33].[5] The (7,4) Hamming code requires 3 redundant bits for every 4 information bits. For a packet size of 300-bits this is an extra 225-bits of redundancy which adds significantly to the cost of a transmission and reception (Section 8).

[5] Of course this coding gain is calculated in terms of Bit Error Rate (BER) and not Packet Error Rate (PER) but it is still informative to keep it in mind.

In Section 3.2 an analysis of packet merging was carried out using simulations and practical experiments. Comparisons were conducted against the gains possible using MRC, a technique which represents an upper-bound on the gains obtainable for pre-detection combining, which is not currently possible on simple resource-constrained devices. As a standalone device iPM was shown to offer quite significant gains considering its simplicity. It was suggested, however, that for maximum benefit iPM could be used in conjunction with other forms of combining, such as the packet decoding technique described in [11].

The practical experiments carried out used a specific type of architecture. However, it was shown that, for the specific device under test, packet merging was a beneficial combining technique to use. This was done through energy measurements of the node whilst running the algorithm developed and comparing the energy cost to that of a communication. The practical experiments also consisted of a real implementation of the algorithm. It was shown that significant reduction in PERs are possible when using packet merging. Although it was difficult to measure the actual gains obtainable in the practical experiments iPM was shown to reduce the PER by a factor of 3.7 when $w(V_A)_{max} = 10$ and 6.5 when $w(V_A)_{max} = 15$. This was an average value over different transmission powers. The suggestion was made that highly optimised code or hardware implementations may allow these gains to be obtained and represents interesting future work in this area.

Each experimental setup used was representative of a worst case scenario in order to stress-test iPM: the receiving node was always placed on the fringe of its radio range, in a highly cluttered environment. To ensure sufficient fading the nodes were placed on a record player rotating at 33 1/3 rpm with randomly timed packet transmissions for maximum effect. Although these experiments do not cover all possible scenarios, such as a very dense network with many interferers, for situations where the distribution of errors is similar, the combining gains will be approximately the same – independent of the system used. For more benign environments than that considered combining will either not be used (due to the adaptive nature of the approach) or, if used, the search time and energy will be significantly reduced due to the smaller number of errors.

References

[1] S. Wicker, Error Control Systems for Digital Communication and Storage, Prentice Hall, 1994.
[2] A. Nosratinia, T.E. Hunter, A. Hedayat, Cooperative communication in wireless networks, IEEE Communications Magazine 42 (10) (2004) 74–80.
[3] A. Goldsmith, Wireless Communications, Cambridge University Press, 2005.
[4] D. Varshney, C. Arumugam, V. Vijayaraghavan, N. Vijay, S. Srikanth, Space-time codes in wireless communications, IEEE Potentials 22 (3) (2003) 36–38.
[5] Moteiv Corporation, Tmote Sky Datasheet. <http://www.moteiv.com> (accessed 27.09.05).
[6] Crossbow Technology Inc. <http://www.xbow.com/> (accessed 08.04.05).
[7] J. Kahn, R. Katz, K. Pister, Next century challenges: mobile networking for smart dust, in: MobiCom '99: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999, pp. 483–492.

[8] D. O'Rourke, C. Brennan, A practical implementation of an improved packet combining scheme for wireless sensor networks, in: IEEE International Conference on Communications: Workshop on Cooperative Communications and Networking: Theory, Practice, and Applications, 2008, pp. 332–336.
[9] P. Sindhu, Retransmission error control with memory, IEEE Transactions on Communications 25 (5) (1977) 473–479.
[10] S.S. Chakraborty, E. Yli-Juuti, M. Liinaharja, An ARQ scheme with packet combining, IEEE Communications Letters 2 (7) (1998) 200–202.
[11] H. Dubois-Ferrière, D. Estrin, M. Vetterli, Packet combining in sensor networks, in: SenSys '05, 2005.
[12] D. Chase, Code combining – a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets, IEEE Transactions on Communications 33 (5) (1985) 385–393.
[13] S. Yi, B. Azimi-Sadjadit, S. Kalyanaraman, V. Subramanian, Error control code combining techniques in cluster-based cooperative wireless networks, in: IEEE International Conference on Communications, 2005, pp. 3510–3514.
[14] T.E. Hunter, A. Nosratinia, Cooperative diversity through coding, in: Proc. IEEE ISIT, Laussane, Switzerland, 2002, p. 220.
[15] S. Chakraborty, M. Liinaharja, E. Yli-Juuti, An adaptive ARQ scheme with packet combining for time varying channels, IEEE Communication Letters 3 (2) (1999) 52–54.
[16] S. Chakraborty, M. Liinaharja, K. Ruttik, Selection diversity based ARQ schemes and their enhancements using packet combining in a slow rayleigh fading channel, 2002 IEEE International Conference on Personal Wireless Communications, 2002, pp. 192–195, doi: doi:10.1109/ICPWC.2002.1177275.
[17] S. Chakraborty, M. Liinaharja, K. Ruttik, Diversity and packet combining in rayleigh fading channels, IEE Proceedings on Communication 152 (3) (2005) 353–356.
[18] G. Yu, Z. Zhang, P. Qiu, Efficient ARQ protocols for exploiting cooperative relaying in wireless sensor networks, Computer Communications 30 (14-15) (2007) 2765–2773.
[19] TinyOS Website. <http://www.tinyos.net> (accessed 20.09.07).
[20] Nordic Semiconductor, NRF905 Datasheet.<http://www.nordicsemi.com/files/Product/data_sheet/nRF905_rev1_3.pdf> (accessed 15.06.08).
[21] Texas Instruments, CC2420 Datasheet. <http://focus.ti.com/docs/prod/folders/print/cc2420.html> (accessed 02.03.07).
[22] Institute of Electrical and Electronic Engineers, IEEE 802.15.4 Std., 2006.
[23] P. Herhold, E. Zimmermann, G. Fettweiss, A simple cooperative extension to wireless relaying, in: International Zurich Seminar on Communications, 2004.
[24] T. Fujiwara, T. Kasami, A. Kitai, S. Lin, On the undetected error probability for shortened hamming codes, IEEE Transactions on Communications 33 (6) (1985) 570–574.
[25] T. Baicheva, S. Dodunekov, P. Kazakov, Undetected error probability performance of cyclic redundancy-check codes of 16-bit redundancy, IEE Proceedings on Communications 147 (5) (2000) 253–256.
[26] G. Castagnoli, J. Ganz, P. Graber, Optimum cyclic redundancy-check codes with 16-bit redundancy, IEEE Transactions on Communications 38 (1) (1990) 111–114.
[27] D. O'Rourke, Practical Packet Combining for Use with Cooperative and Non-cooperative ARQ Schemes in Wireless Sensor Networks, PhD, Dublin City University, September 2009.
[28] L. Selavo, A. Wood, Q. Cao, T. Sookoor, H. Liu, A. Srinivasan, Y. Wu, W. Kang, J. Stankovic, D. Young, J. Porter, Luster: wireless sensor network for environmental research, in: SenSys '07: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, 2007, pp. 103–116.
[29] T. Wark, P. Corke, Springbrook: Challenges in developing a long-term rainforest wireless sensor network, in: ISSNIP '08: Proceedings of the Fourth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2008.
[30] W. Hu, P. Corke, secfleck: A public key technology platform for wireless sensor networks, in: EWSN '08: Proceedings of the 6th European Conference on Wireless Sensor Networks, 2008.
[31] Texas Instruments, CC1000 Datasheet. <http://www.focus.ti.com/docs/prod/folders/print/cc1000.html> (accessed 08.07.08).
[32] K. Schwieger, A. Kumar, G. Fettweis, On the impact of the physical layer on energy consumption in sensor networks, in: IEEE Proceedings of the Second European Workshop on Wireless Sensor Networks, 2005, pp. 13–14.
[33] T. Moon, Error Correction Coding: Mathematical Methods and Algorithms, Wiley-Interscience, 2005.

Damien O'Rourke graduated from Dublin City University (DCU) in 2000. Following a short period as a network design engineer he went on to pursue a research masters degree in attacking embedded cryptosystems. Upon graduating in 2003 he took on a full-time lecturing position which later led him to pursue his PhD in Wireless Sensor Networks which he obtained in 2009. He has been working as a post-doctoral research fellow for CSIRO since October 2008 and his current research interests include the development of efficient event detection and signal processing algorithms for Wireless Multimedia Sensor Networks, as well as diversity techniques for Wireless Sensor Networks in general.

Conor Brennan graduated from Trinity College Dublin (TCD) with a BA (Mod) in Mathematics in 1994 and a PhD in 1998. He spent several years as a post-doctoral researcher in Dr. Peter Cullen's wave scattering group in TCD before being appointed lecturer in the School of Electronic Engineering, Dublin City University in 2003. His research interests are in numerical methods for wave propagation modelling as well as energy-efficient communications. He has written over 50 peer-reviewed papers in these areas. He is a member of the IEEE and the IET as well as the Royal Irish Academy Committee on Communications and Radio Science.