A Practical Implementation of an Improved Packet Combining Scheme for Wireless Sensor Networks

Damien O'Rourke and Conor Brennan
Research Institute for Networks and Communications Engineering (RINCE)
Dublin City University, Dublin 9, Ireland
Email: {orourked, brennanc}@eeng.dcu.ie

***Abstract*—An improvement to a recently proposed packet combining procedure based on the use of a CRC is developed. Upon receiving two packets, sent over independent channels, a large error burst may appear in one; however, it is unlikely that the same situation will have occurred for its pair. By making an intelligent choice as to which packet the burst is most likely to be in, and choosing the other packet for correction, the decoding time can be reduced significantly. A practical implementation is provided with real results that show not only can this decision be made correctly on the majority of occasions but, once the correct packet is chosen, an intelligent search operation can be performed further reducing overhead.**

I. Introduction

Packet combining can be used in simple, low-power Wireless Sensor Networks (WSNs) 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 WSNs [1]. Cooperative Communications is mainly seen as a viable alternative to MIMO techniques which are unrealisable on current WSNs due to their size, cost and hardware limitations. Although generally aimed at more complex transceiver designs, it is quite reasonable to assume that, with some modifications, gains can also be realised on simple, low-power WSNs.

Some of the combining techniques used, such as maximal-ratio combining or the use of Space Time Block Codes (STBCs) [2], are not viable options for commercially available architectures such as the popular Tmote Sky [3], MicaZ [4], and other IEEE 802.15.4 [5] compliant devices due to their simplified hardware. In addition, Channel State Information (CSI) at both the transmitter and receiver is not available and only partial information can be obtained at the receiver (in the form of LQI and RSSI[1]).

From a more practical perspective, the Simple Packet Combining (SPaC) protocol was introduced in [6]. This protocol is designed with low-rate, low duty cycle sensor networks in mind and seems to offer a lot of promise for these type of devices. In the SPaC protocol, when a node receives two or more corrupt packets, a packet combining procedure is attempted. Either a *packet merging* or *decoding* operation is performed; the latter being a form of coded cooperation [7]. The 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.

The weakness of SPaC stems mainly from its packet merging procedure which was shown to be inferior to decoding. Due to the relative simplicity and ease of implementation of merging however, it is reasonable to ask whether or not any improvements can be afforded. The main contribution of this paper is to answer this question in the affirmative and to describe a practical implementation of this improvement.

[1]Link Quality Indicator (LQI) and Received Signal Strength (RSSI).

II. Packet Merging

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):

$$m_S \oplus m_R = e_S \oplus e_R = V_A. \tag{1}$$

Here m_S and m_R are the two received, corrupt data vectors from the Source (S) and Relay (R) respectively, and e_S, e_R are, respectively, the error vectors corresponding to these packets. 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 one or other of the data packets, and a 0 where either no error occurred or where one occurred in the same position in both packets. This latter case is not correctable using packet merging and is referred to as a *hidden error* (see [6] and [8] for more details).

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.

III. Practical Setup

In order to try to gain a deeper understanding of the merging process a practical setup was devised consisting of a three node architecture: a Source (S), a Relay (R) and a Destination (D). The particular devices used were the popular Tmote Sky motes [3] which incorporate a 2.4GHz spread-spectrum radio compliant with the IEEE 802.15.4 standard [5]. 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

978-1-4244-2052-0/08/$25.00 ©2008 IEEE

to produce packets with large numbers of errors in order to test packet merging in these scenarios. To this end, the nodes were setup 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[2].

As the power levels increased it was necessary to increase the separation distance between the nodes in order to maintain a situation where the nodes operated at the fringe of their radio range. This involved using node separations between 1 to 30 meters. At the smaller distances the communication would have been dominated by a large LOS component and little or no multi-path interference while at large separations (involving nodes being placed in separate rooms), there would have been no LOS but increased levels of multi-path. 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 protocol used was essentially the simple adaptive decode-and-forward protocol (simple AdDF) described in [9] (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 the CRC check passed. Upon reception, the destination carried out another CRC check on both 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:

$$(m_S \oplus m_R \neq 0) \text{ AND } (\text{FCS}_S \oplus \text{FCS}_R = 0), \qquad (2)$$

where FCS_S and FCS_R represent the frame check sequence of the source and relay messages respectively[3]. Corrections were not attempted either if the Hamming weight (denoted by $w(.)$) of the ambiguity vector was greater than 10; i.e. $w(V_A)_{max} = 10$. The size of $w(V_A)_{max}$ 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 $w(V_A)$ the number of hidden errors began to increase significantly. Also, as the computational complexity of the algorithms developed in this paper are less than that used in SPaC (where $w(V_A)_{max} = 6$) it is possible to allow for a larger value of $w(VA)_{max}$.

A. Search algorithms

Two algorithms were developed for the above scheme: the second resulting from the first. The first algorithm, which will be referred to as Simple Packet Merging (sPM), chooses m_{ch}, *a priori*, and adheres to that choice throughout the experiment. As there are a total of $2^{w(V_A)} - 2$ possible error candidates [8] 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; see figure 1) modifying m_{ch} and determining whether the following is true:

$$\text{CRC}(m_{ch} \oplus e_c) \oplus \text{FCS} = 0, \qquad (3)$$

where $\text{CRC}(x)$ calculates the FCS for the value x. Once equation 3 is satisfied the algorithm returns the value found.

It makes more sense, however, to step through the error candidates in a more structured manner. The second algorithm, which will be referred to as Improved Packet Merging (iPM), attempts to do this. Instead of 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 until either a packet is found which produces the correct FCS or all candidates have been exhausted (cf. figure 1). The motivation for using this particular search method (which will be referred to as an "improved" search) can be seen from the graph of figure 2 (showing actual data). While this represents an extreme example, it shows that it is possible for a large error burst to only be contained in one packet. It can be seen from the figure that m_R contains many more errors than m_S. Although $w(e_R)$ has a value of 27, $w(e_S)$ is only 3 - a significant difference. Therefore, by choosing the packet with the smaller number of bits and using the improved search method a significant energy saving can be made. In SPaC, the method used is a brute force search of every possible error candidate. The purpose of doing this is to identify when false positives occur. This is discussed further in section V.

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...1111	1111...1111 } $w(e_c) = w(V_A)_{max}$

Fig. 1. Comparison of the search methods for sPM and iPM.

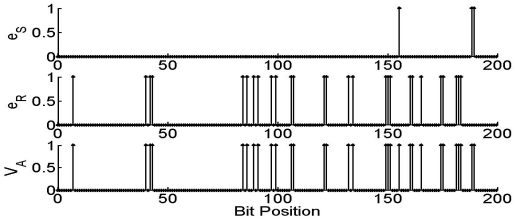

Fig. 2. Plot showing actual error vectors (e_S and e_R) and ambiguity vector (V_A) obtained from experimental setup 3.

[2]This is not an unrealistic situation for WSNs and, in fact, some of the motivation given for the use of cooperative communications is that in a normal relaying scenario the receiver discards as noise what might be useful information from the transmitter. In cases like this the SNR at the receiver due to the transmitter would be bordering on the sensitivity levels of the receiver.

[3]Henceforth referred to simply as FCS due to their required equality.

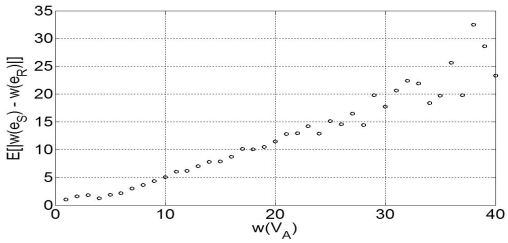

Fig. 3. Average difference in Hamming weights between the source payload (e_S) and the relay payload (e_R) compared with the Hamming weight of the ambiguity vector ($w(V_A)$).

B. Packet Selection

Figure 2 clearly shows that in some cases one of the packets may contain many more errors than its pair. While this is an extreme case, the experimental results showed that, on average, the errors were not equally distributed between the two packets. To see this, consider figure 3 (obtained from the results of experiment 3). This figure consists of a plot of the average difference in Hamming weights between the source and relay packets (E[$|w(e_S) - w(e_R)|$]) against the Hamming weight of the ambiguity vector ($w(V_A)$). Had an equal number of errors been contained in each packet, for each value of $w(V_A)$, then E[$|w(e_S) - w(e_R)|$]) would have remained at zero and the graph would simply have been a horizontal line. If, however, all the errors had been contained in only one of the packets then the graph would have appeared with a slope of 1. As can be seen, the graph in figure 3 has a slope of approximately 0.5 implying that, on average, one packet contained 3 times as many errors as its pair. The reason for this is likely due to the independent channels the data were sent over[4] implying that a deep fade corrupting one packet is unlikely to occur for its pair. Similar graphs were seen for the other setups.

The second innovation of iPM attempts to correctly choose the packet with the least number of errors and use it as m_{ch} (cf. equation 3). The capabilities of the physical devices being worked with will obviously have an influence on this selection policy. Although the current setup was carried out on the Tmote Sky mote it can easily be extended to many other similar devices (such as those supported by the TinyOS operating system [10]).

With the Tmote Sky mote (and others of its class) there are two obvious methods for choosing the best packet: the Received Signal Strength Indicator (RSSI) or the correlation value (CORR)[5]. Both of these values are returned by the CC2420 (the transceiver used on the Tmote) for each incoming packet and are averages over the first eight symbol periods after the Start of Frame Delimiter (SFD).

The use of one of these metrics to determine the best packet is similar to the idea of selection combining, where the signal with the largest instantaneous power is chosen for processing. In general, the main drawback of selection combining is that the SNR on each branch has to be monitored separately: leading to either an increase in hardware complexity or reduced spectral efficiency. In the cooperative relaying situation the relay channel has to comply with the orthogonality constraint, and this is therefore not an issue. Of course, the advantage of the scheme described in this paper is that it doesn't discard the other "branch", (as selection combining does) but makes use of it to create the ambiguity vector.

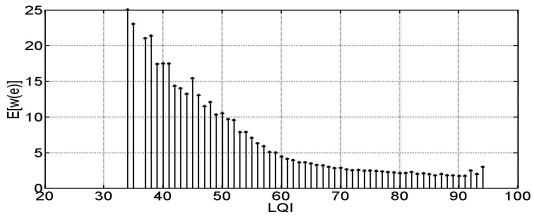

Fig. 4. Example of how the Hamming weight of the error vector changes with LQI (taken over a static link with a mean of about 67).

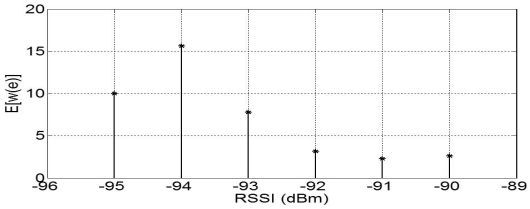

Fig. 5. Example of how the Hamming weight of the error vector changes with RSSI (taken over a static link with a mean of about -92 dBm).

Figure 4 (resp. 5) shows the relationship between the Hamming weight of an error vector and its LQI (resp. RSSI) value; both of these graphs were taken from the experimental results. At first it would appear that the LQI is a much better packet selection metric than the RSSI; however, these are plotted against the average values of the Hamming weight of the error vectors (E[w(e)]) and the distributions are in fact quite different. The RSSI value was quite consistent throughout the entire experiment with a mean of about -92 dBm and a standard deviation of only 0.63 dB;

It can be seen that, the decrease in the number of errors as the LQI increases, on average, is quite consistent; however, the RSSI value doesn't produce such fine grained results. Interestingly, a number of RSSI values above -90 dBm (not shown) were obtained where the number of errors jumped dramatically: an apparent contradiction. This is likely due to in-band interference increasing the RSSI value and causing a weak link to appear to be of good quality.

Considering these results, whether or not to use RSSI for the selection of the appropriate packet will, in general, be determined by the particular environment. If noise is the main problem with the system then RSSI driven diversity

[4]Due to the orthogonality (or half-duplex) constraint, these channels are separated in both time and space.

[5]Although not the same, CORR will henceforth be referred to as LQI as it is more common; the former can be used as an estimation of the latter.

could be used (perhaps in conjunction with CORR). However, in environments with large levels of co-channel interference the RSSI can overestimate the SNR and should therefore be avoided (unless the level of interference can be determined and compensated for).

	Ex1 (%)	Ex2 (%)	Ex3 (%)	Average (%)
LQI	61	73	79	71
RSSI	53	69	81	68

TABLE I
THE PERCENTAGE OF CHOSEN PACKETS, USING EITHER LQI OR RSSI, CONTAINING THE LESSER NUMBER OF ERRORS.

Over the course of quite a number of experiments (including others not described in the current paper), the LQI did in fact prove to be a slightly better choice for determining m_{ch}: but not by much. An example of this can be seen in table I where, averaging over the three experiments, on 71% of occasions the correct value was chosen (i.e. the value with the smallest Hamming weight). This can be compared to the RSSI value (also shown) which averaged about a 68% success rate - still very accurate, relatively speaking. The closeness of these results implies that in some cases the RSSI may prove a better metric and indeed this can be seen to be the case for experiment 3. One possible way of choosing between the use of RSSI and LQI in real-time may be to initially alternate between both metrics for each successive merging operation. The number of iterations required compared with a complete search can then be determined for each. The metric leading to the least number of iterations on average would then be chosen.

For the experiments described in this paper it was simply decided to use LQI for packet selection due to its higher success rate in general. 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 situation as the current algorithms rely on one sample only, taken over the first 8 symbols of the packet, to provide accurate information about the rest of the packet.

IV. EXPERIMENTAL RESULTS

The two algorithms described in section III-A (sPM and iPM) were used in each experiment. Recall that the search algorithm used by the first was the simple incremental search. The selection criterion was to choose either m_S or m_R at the beginning of each experiment and adhere to this choice throughout. Both options were evaluated for each experiment and will be referred to as Case 1 and Case 2, respectively. For iPM the LQI was used to choose, in real time, the best packet to use as m_{ch}. The search algorithm used was the improved one described in section III-A. In summary:

1) sPM
   a) Selection Criteria:
      - Case 1: Always choose m_S
      - Case 2: Always choose m_R
   b) Search Method: Incremental
2) iPM:
   a) Selection Criteria:
      - If $\text{LQI}_{m_R} > \text{LQI}_{m_S}$, choose m_R: else, m_S
   b) Search Method: Improved

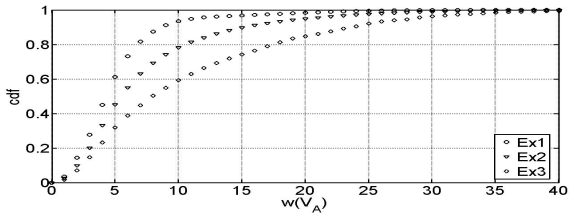

Fig. 6. Empirical cumulative distribution of $w(V_A)$.

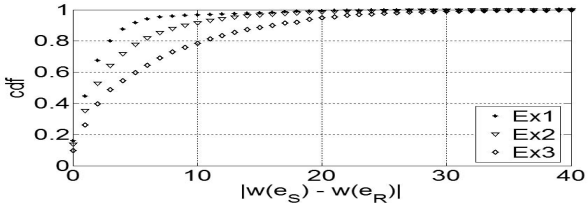

Fig. 7. Empirical cumulative distribution of the difference in Hamming weights between m_S and m_R for each experiment.

Figure 6 shows the empirical cumulative distribution (marked as cdf for short) of the Hamming weight of the ambiguity vectors received. It can be seen that there is a clear increase in reception of larger weights from experiments 1 to 3 possibly due to the increased multipath component (cf. section III). In the first and second experiments, respectively, 93% and 78% of the ambiguity vectors had a Hamming weight less than $w(V_A)_{max} = 10$, while for the third case this was reduced to 59%.

Consider now figure 7. It can be seen that the difference between the error vectors gets increasingly large for each experiment. For the third experiment, 40% of the vectors are separated by a difference greater than 5, whereas this value is only 22% and 8% for experiments 2 and 1, respectively. By using iPM, it should be possible to cope more efficiently with the larger $w(V_A)$.

	sPM (%)			iPM (%)	Difference (%)
	Case 1	Case 2	Average		
Ex1	57	44	51	47	4
Ex2	57	46	52	36	16
Ex3	42	65	54	33	21

TABLE II
COMPARISON OF THE PERCENTAGE OF REQUIRED ITERATIONS FOR EACH SETUP.

Table II shows a comparison of the results between the sPM and iPM for the three different setups. These values represent the number of iterations the algorithms had to go through

before finding the correct packet (expressed as a percentage of the total possible number of iterations). For experiment 1, it can seen that sPM, in fact, performed slightly better than iPM when constantly choosing m_R (case 2). The problem in this case is most likely the fact that the LQI only achieved a 61% success rate (cf. table I) which itself was most likely due to the very similar channels from S to D and R to D.

In a realistic scenario, however, it may not be possible to choose, *a priori*, which is always going to be the best packet and a real time evaluation is needed; iPM does exactly this and it is seen that, even with only a 61% correct-choice success-rate, it comes closer to the better of the two cases for sPM. On the other hand, experiments 2 and 3 exemplify the benefits of iPM even more so due to the increase in $|w(e_S) - w(e_R)|$. For experiment 2, iPM improves upon the best sPM case (which there is no way of guaranteeing in practice) by 10% and for experiment 3 by 9%. In the latter case, a further improvement would have been made had the RSSI been used in choosing the optimal packet (cf. table I).

Due to the unrealistic selection procedure for sPM, a more reasonable comparison might be to compare iPM with the average of the two cases for each experiment. This is also shown in table II and it can be seen that there is a definite improvement in each case. As much as a 21% improvement in the number of iterations required can be seen - a significant savings in terms of energy consumption over the lifetime of a node.

V. Final Considerations

In each of these experiments the percentage of correctly decoded packets after merging was on average about 99.9%. The other 0.1% uncorrectable packets were due to false positives. There is hence a trade-off between about a 50 - 70% reduction in the number of iterations required over a brute-force merging operation, and a small probability of producing an unrecognised false-positive. It should be noted however that the value 0.1% was determined under extreme conditions and also the probability of a merging actually occurring was not taken into account. As mentioned earlier, in some cases it might be possible to sample the RSSI/LQI value for each symbol of the packet (as opposed to relying only on the sample taken at the beginning). 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 and increase the search efficiency. A more thorough analysis of these issues will be carried out in future work.

In [6] it was shown that the cost of a merging operation is roughly only 7% of that of a transmission with a 29 byte payload and 2% with a 128 byte payload. The purpose of the current work is to demonstrate how aspects of those results can be improved upon using readily available metrics such as LQI and RSSI. Using sPM or iPM these figures can be significantly improved upon increasing the lifetime of each node in a network. A quantification of this reduction in energy consumption will be carried out in future work.

In certain circumstances the computational complexity of both sPM and iPM can be improved further by noting that, for generator polynomials with an even Hamming weight (such as that used in IEEE 802.15.4) the parity of the message will be the same as that of the FCS. This can be seen because calculation of the FCS is simply a repeated modulo-2 sum of a multiple of the generator polynomial with the message. It can therefore be said that:

$$w(ps_{i+1}) = w(ps_i) + w(G) - 2n \tag{4}$$

where ps_i is the partial sum at the i^{th} stage of the FCS calculation and n is the number of ones in the same bit position for both ps_i and G. As $2n$ will always be even and assuming $w(G)$ is even, then $w(ps_{i+1})$ will be even if $w(ps_i)$ is even and odd if $w(ps_i)$ is odd. The parity of the FCS will therefore always equal that of the message for an even parity generator polynomial. This can be used to reduce the search by one-half as error candidates that produce messages with odd parity can be eliminated.

VI. Conclusions

An improved type of packet merging that is practical for use on simple, low-power motes has been developed. Two algorithms were implemented both of which use a pair of packets received over independent channels for error correction. The first algorithm, referred to as Simple Packet Merging (sPM), chooses one channel, *a priori*, and always uses the packets from that channel in the error correction process. It also implements a simplistic search procedure for the correct packet by running through all possible error candidates in an incremental fashion. The second algorithm, referred to as Improved Packet Merging (iPM), not only intelligently chooses, in real time, which packet has less errors, but also does a more efficient search for the correct packet. It was shown that iPM improves upon the average number of iterations required by sPM by up to 21% - a significant saving over the lifetime of a node.

References

[1] A. Nosratinia, T. E. Hunter, and A. Hedayat, "Cooperative communication in wireless networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 74–80, October 2004.

[2] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[3] *Tmote Sky Datasheet*, Moteiv Corporation, available from http://www.moteiv.com [Accessed 27 September 2005].

[4] Crossbow Technology Inc. Available from http://www.xbow.com/ [Accessed 08 April 2005].

[5] *IEEE 802.15.4 Std.*, Institute of Electrical and Electronic Engineers, 2006.

[6] H. Dubois-Ferrière, D. Estrin, and M. Vetterli, "Packet combining in sensor networks," in *SenSys'05*, November 2005.

[7] T. E. Hunter and A. Nosratinia, "Cooperative diversity through coding," in *Proc. IEEE ISIT*, Laussane, Switzerland, July 2002, p. 220.

[8] S. S. Chakraborty, E. Yli-Juuti, and M. Liinaharja, "An ARQ scheme with packet combining," *IEEE Communications Letters*, vol. 2, no. 7, pp. 200–202, July 1998.

[9] P. Herhold, E. Zimmermann, and G. Fettweiss, "A simple cooperative extension to wireless relaying," in *International Zurich Seminar on Communications*, February 2004.

[10] *TinyOS Website*, available from http://www.tinyos.net [Accessed 20 September 2007].