Using Feedback Control within WSN's to meet Application Requirements

Mark Louis Fairbairn, Iain Bate
Department of Computer Science, University of York, York, YO10 5GH, UK
{mark.fairbairn, iain.bate}@cs.york.ac.uk

***Abstract*—Currently the main approach used to save power within WSN's is to employ reactive MAC schemes that detect when the network is becoming busy and adapt to provide an increase in available bandwidth. Once the traffic becomes lighter the available bandwidth is reduced to save power. While this works well with bursty sporadic traffic, there is a clear trade-off between latency of response and power savings. Other problems with this approach include hard to derive parameters along with their poor performance in periodic *sense-and-send* applications due to their reactive nature.**

We propose an adaptive feedback-based scheme that adjusts the duty cycle of the the motes to reduce the power consumed in typical *sense-and-send* applications by 58.4%, while still meeting a minimal level of service specified by the operator. Such an adaptive scheme is necessary to minimise power consumption as the complete network characteristics are unknown prior to deployment, and can change during runtime. Our approach is independent of node distribution and is also MAC layer agnostic, unlike the current state-of-the-art, whilst consuming less power. Our approach provides a simple method for the application designer to modify its behaviour, whilst allowing WSNs to operate longer and provide the same level of service as current approaches.

I. Background

Wireless sensor networks (WSN) are deployments of many small, cheap, battery powered devices called nodes, which use wireless radios to communicate with surrounding nodes. Nodes can utilise multi-hop communications to send messages further across the network than a single node can transmit, with intermediaries relaying messages.

WSNs are becoming more commonly used to augment traditional industry. The benefits of WSNs are clear as required modifications to the existing infrastructure are low while still providing the fine-grained information. They have been used to monitor bridges, tunnels, buildings and pipelines with the aim for large-scale, long-term operation [4] [10] [7]. These monitoring activities involve; humidity regulation to maintain the integrity of suspension bridges; deterioration in concrete structures by measuring crack sizes; measuring bridge movement over time; and the pressure within pipes. Additionally WSNs can be used to monitor natural environments, such as crop monitoring [12], monitoring habitats [8], aquatic monitoring [2]. These examples are all *sense-and-send* applications, where data from the environment is simply measured and periodically reported back to a specified base station.

One of the key issues with WSNs is power consumption, as once a node's battery is depleted, the node is effectively dead. As the main cause of power drain is communications, with the on-board radio consuming up to 30 times more power than the processor [5], we should ensure that the radio should be powered only when communications are strictly required and turned off at all other times.

Sense-and-send applications normally derive their value from the data itself, with the time it takes for the data to be received at the base-station being less critical. It can be seen in the above examples that some data could be delayed with negligible effect. This is especially true where the parameter being monitored does not vary rapidly or the information is to be used for offline analysis. One such example would be the humidity sensing within the suspension bridge, where nodes reporting a sample every 10 minutes could be replaced with a node reporting all 6 collected samples at once every hour. This does not affect the ability to estimate the rise or fall trends in the humidity, and therefore no important information is lost. Other examples include refrigerated transports, where temperature could be sampled every 10 seconds, but only sent every 50 seconds. This would allow events such as the door being left open to be detected in a reasonable amount of time.

Our solution optimises the duty cycle of a WSN network through the use of feedback, minimising the awake time and maximising the sleep time. This solution creates the maximum contiguous sleep period possible, whilst respecting the maximal message delay the application can tolerate. The effect of this behaviour is the batching of radio messages, which are all sent once the radio is active. Secondly the algorithm creates the smallest transmission window required to successfully send and relay all pending messages. This approach minimises energy waste incurred from power cycling the radio, maximising network lifetime. The advantage that our solution has over others is that the feedback approach allows it to adapt when changes in the environment occur, tolerating changes in background noise and changes in topology. Our solution also provides additional benefits, such as relaxed requirements for time synchronisation, which allows further power savings.

This work provides the following contributions:

- A MAC and Routing layer agnostic method for reducing power consumption within *sense-and-send* applications.
- Strict enforcement of operator specified maximal delay, even in the presence of external interference.
- Utilising the aforementioned delay to send readings in batches, further reducing power consumption.

- Use of industrial control theory to realise the above contributions.

The structure of the paper is as follows. Section II provides an overview of the current approaches used to obtain power savings within WSNs, finishing with an outline of the three main objectives for this work. Section III specifies the problem in detail, specifically where power is currently consumed and for what purpose. Section IV details our solution to the problem, how power is saved and how the feedback system operates. Section V describes the experimental setup and the results obtained from these. Finally section VI provides some overall conclusions.

II. Related Work

Within this section we will describe the current state-of-the art in power savings at the radio level. this includes MAC approaches to save transmission power and application based approaches to batch communications. Finally a summary will be given, with a brief outline of our proposed approach.

A. MAC Based

MAC protocols have been a popular research area in which to reduce power consumption, attempting to place the radio in a lower power mode (sleep / idle) as much as possible while taking latency and bandwidth into account. MAC protocols can be summarised into two main groups, synchronous and asynchronous, however some hybrid approaches also exists. Synchronous MAC protocols such as S-MAC [14] and T-MAC [11] rely on time synchronisation between nodes to determine a pre-negotiated time and period for node communication (typically this is just after a SYNC beacon message). Asynchronous MAC protocols such as X-MAC [6] do not require time synchronisation. These protocols commonly use periodic probing of the wireless channel to signal that a node wishes to send a message. Once a neighbouring node acknowledges the request then transmission commences. Asynchronous protocols commonly have a lower latency as transmission is on-demand, however at the expense of power due to the periodic listening required by neighbouring nodes.

Integrated solutions, such as Zigbee [1], not only specify the MAC layer protocol (802.15.4) but also how the application routing is organised. Zigbee has two main modes, non-beacon and beaconed modes. Non-beacon mode requires routing nodes to be mains powered and thus does not suit our applications. Beaconed mode uses the same basic principle seen in synchronous MAC protocols whereby timed pulses are used to synchronise time and to define when nodes can transmit (superframe), however Zigbee applies this concept higher in the network stack, allowing up to 16 transmissions within a superframe from any node. The frequency of beacons is defined before the application commences, and is calculated based upon the desired network bandwidth. By default the radio is set to a 0.1% duty cycle at 250 kbit/s, equating to 15.36s between superframes.

Our approach is MAC protocol agnostic as our approach limits the periodic probing which would otherwise consume a large amount of energy. It is however ideal to pick a MAC protocol that can deliver a high number of packets within a short time period.

B. Application Based

Ramanathan et al proposed batching transmissions by adding a packet queue below a custom routing layer [9] to gain additional energy savings, with the sleep periods being fixed in the range of 20ms to 12000 seconds. The authors note that should an application only need to send data every 10 minutes, B-MAC will still turn the radio on 6000 times during that 10 minute period. Ramanathan et al's work also considers the effect that this technique has upon the routing layer, with nodes waking up slightly before the required transmission time to enable the routing layer to stabilise. We use a similar batching approach, however based above the routing layer. This approach is used to constrain the time the MAC layer is active in order to realise the power savings. Our approach removes the need to specifically specify the amount of batching at each node, instead it calculates the batching from the maximal end-to-end delay specified by the designer.

Most of the current MAC level approaches aim to increase the average throughput of the network while minimising latency, and are all operating on the scale of milliseconds. As identified within the introduction some applications are tolerant to delays on the scales of minutes. This large latency tolerance makes these MAC approaches sub-optimal due to the constant polling performed to meet unneeded sub-second latencies or tight time synchronisation, wasting energy. Work by Wang et al [13] is one of the first to consider the desired end-to-end delay, and removes the fixed polling periods by introducing a proportional feedback mechanism to vary the transmit times. These transmit times are based upon a desired end-to-end packet delay and the measured Packet Reception Rate (PRR) in the network. This work however makes a number of expensive assumptions, primarily that the routing topology is fixed providing a map of 'flows', and that the PRR is known on all links within all flows. Our approach uses the same notion of desired end-to-end packet delay, however it does not make any of the other assumptions. We only make one cheap assumption, that sense messages include the timestamp when the readings were taken, which is common in this type of application.

C. Summary

Current approaches for reducing the power consumed by the radio have typically focused on MAC layer, per packet savings. Ramanathan et al have shown that additional savings can be achieved by batching transmissions, however they have only performed basic experiments using fixed batch sizes, with no regards for application-defined specifications. Wang et al use the notion of a specified application delay, in combination with a measured PRR, to specify a fixed transmission interval that should meet the application requirements. This work however requires the application to have a large amount of knowledge about the global state, and thus would be expensive in large, or

rapidly changing environments. We aim to provide an adaptive solution that utilises both of these approaches, along with the additional consideration that the physical environment in which the WSN operates is constantly changing, and as such a given algorithm must dynamically adjust to these changes.

To address these issues with the current state-of-the-art we derive the three main objectives for this work.

Objective 1 - Batching - Readings should be sent in batches to reduce overheads, whilst also ensuring that messages are delayed no longer than a limit set by the application designer.

Objective 2 - Duty Adjustment - Awake time should be minimised, avoiding fixed sleep / awake periods, allowing for changes within the environment to be adapted to.

Objective 3 - Reaction - Network changes, such as increases in interference, should be responded to in a fast, but predictable manner.

Where appropriate we will benchmark our solution against XMAC and the system proposed by Wang et al. XMAC was chosen to compare against as it is the de-facto standard for reducing WSN energy consumption at the MAC layer, whilst requiring no information about the application. Wang et al's solution is also compared against as it is the only work identified as utilising the end-to-end allowed delay to optimise the network.

III. Problem Formulation

Within this section we will describe the two main problems behind objective 1 and 2. Figure 1 shows the most basic form of the two main states a WSN mote may be in. When the radio on the mote is powered off the mote is said to be in the sleep state. For our purposes the sleep state only includes the radio, as the processor consumes much less power than the radio and so does not require a strict regime. When the radio is powered the mote is said to be awake. This awake state not only includes transmit, receive and idle (active), but all high current states, such as the time in which the radio is powering up and powering down (warmup and cooldown).

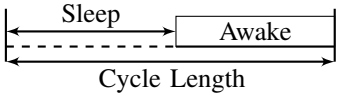

Fig. 1. The two modes of WSN node operation

A. Batching

Objective 1 states that readings should be sent in batches to reduce overheads. Initially we will analyse what these overheads are and how this approach reduces these overheads. Objective 1 also states that a maximal allowed delay must be specified by the application designer, the reasons for which are also discussed in this section.

There are four main phases that a WSN encounters during standard operation, sleep, warmup, active and cooldown. These four phases are shown in figure 2. The most common contributor to the warmup and cooldown states are the time it takes to actively change the power state of the radio from active to idle [5], which whilst taking a short amount of time, still actively consumes power whilst providing no functionality. Equation 1 shows the ratio of wasted power when in normal operation which should be minimised by our solution where possible.

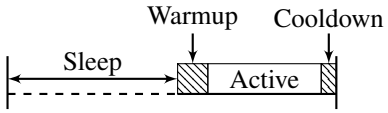

Fig. 2. The four main phases of WSN mote operation

$$\frac{radioCycles * (warmupTime + cooldownTime)}{messagesSent * activeTime} \quad (1)$$

Figure 3 shows why protocols which don't use batching cannot obtain the same low levels of power consumption. It can be seen that the warmup and cooldown penalties occur for each message, during which no transmissions occur. This figure is representative of Wang's solution, however XMAC wastes more power as it repeatedly wakes to sample the radio channel, commonly transmitting nothing.

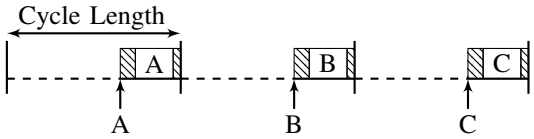

Fig. 3. A standard *sense-and-send* application

To send messages in batches it is necessary to delay the sending of messages once they have been generated. The optimal way to implement this delay is through the notion of a cycle length as shown in figure 3. If we assume that the radio will be awake long enough to transmit all buffered messages and relay others (how to meet this assumption is explained in section IV-B2), then we can increase the cycle length to capture the generation of more messages. As more messages are generated within a cycle, more are buffered to be sent in the transmission window, as seen in figure 4.

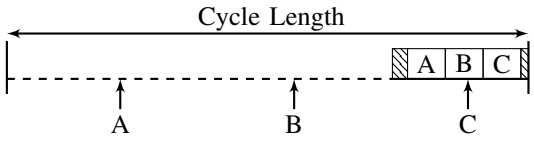

Fig. 4. Batched *sense-and-send* application

One important side-effect to consider when buffering messages, is the increase in time between the message being generated and the time at which they arrive at the base node, the message delay. The more messages are buffered, the more message delay occurs. It is important to put an upper bound on the message delay, based on equation 1 it can be clearly seen that the optimal number of message to send in a batch tends to infinity. This would in turn cause the delay to tend to infinity, rendering any application effectively useless as the messages would most likely never arrive. The application's worst case

delay can be calculated using equation 2. This assumes queued messages are sent in arbitrary order.

$$worstCaseDelay = cycleLength - cooldownTime \quad (2)$$

This delay must always be lower than the operator specified delay (as required by Objective 1), which is the maximum delay the application can tolerate. This maximum delay is typically the time after which the message data is too old to be of value, e.g. warning us of fire an hour late is probably pointless.

B. Duty Adjustment

Objective 2 is the reduction in awake time. As mentioned in section IV-B1 it was assumed that the radio remains awake long enough to transmit all buffered messages and relay others. This section will explain how we can meet this assumption whilst still ensuring that the awake time is minimised to meet the objective.

Taking the four phases as shown in figure 2 and omitting the sleep phase gives us equation 3. This is the total time, including warmup and cooldown, required for the awake period.

$$awakeTime = warmup + active + cooldown \quad (3)$$

To assess what the major contributors to the $awakeTime$ are we expand the three components in equation 4 that contribute to $awakeTime$. Each individual factor is expanded below.

$$warmup = radioOn + routing + max(clockDrift) \quad (4)$$

The warmup time for the node, as shown in equation 4, is the time it takes for the radio to transition to the awake state, the time for the routing layer to stabilise (depending on the routing algorithm), and also most importantly waiting for the maximal clock drift of all nodes. The clock drift wait is subtle, but is crucial to reliably communicate with the sink as the network needs all available routing nodes to be online. When all available routing nodes are online there is maximal chance of transmission success, as all nodes that may be required to relay the message are online.

The active time shown in equation 5 is an interleaving between sending the messages that this node has buffered, and forwarding messages for others. This active time allows for all messages to traverse as many hops as required to reach the sink node, in the worst case this may be the entire width of the network.

$$active = sendMessges + relayMessages \quad (5)$$

These timings are difficult to derive as equation 5 can be further decomposed into equation 6 where tx and rx are the times to send and receive messages, c is the computation time and PRR is the packet reception ratio.

$$transmit = \frac{\#send(tx + c)}{PRR} + \frac{\#relay(rx + tx + c)}{PRR} \quad (6)$$

It can be seen that $transmit$ relies heavily on the PRR, which is a factor of the environmental noise, which changes constantly. $Transmit$ also relies heavily on $\#relay$, which is a product of the routing algorithm, with dynamic routing algorithms like AODV having a non-static $\#relay$.

The final factor is shown in equation 7.

$$cooldown = radioOff \quad (7)$$

C. Reaction

Objective 3 states that we need to react to changes in the environment. This requires monitoring of external variables and feeding these back into the system so that it can react appropriately. For our scenario there are a small number of additional requirements that must also be met.

Firstly we wish to minimise the number of times the target value is exceeded. Exceeding the target value correlates with exceeding the target delay, which in our scenario is the application invalidating its specification. This requirement rules out a typical conditional statement which just consists of two rules, one for below and one for above the target value. From this we can deduce that the difference between the current value and the target value must be used to control the algorithm.

Secondly we wish for the algorithm to react in a fast but stable manner. Fast reaction is needed so that changes in the environment can be adapted to in the shortest time possible. Stability is required as fast changes commonly lead to oscillations in output value as oscillations may lead to exceeding the target value. Ideally any parameters used to control the algorithm should work in a wide range of scenarios, removing the need for the algorithm to be trained for each scenario.

The most common form of feedback controller which meets these criteria are PID controllers, which are used in more than 95% of all industrial control problems [3]. PID controllers are advantageous due to their simple operation and the lack of prior information required about the problem. PID controllers focus around the notion of distance (the error) to the target value (the setpoint) and using these values to calculate its output.

IV. Proposed Solution

A. Basic Strategy

Our solution can be summarised by figure 5. Two PID controllers modify the $awakeLength$ and the $cycleLength$ of the motes, with the reference value being the slack target and the maximum delay accordingly. The defintions and advantages for using these values and not the ones from section III are given in sections IV-B1 and IV-B2.

B. Realisation

The most basic form of PID controller is a simple proportional controller where the error is directly multiplied by a constant factor (P value), with the result being set as the output value. An integral component can be used which looks at the cumulative error over time, adjusting the output accordingly, and a derivative component may also be used, which looks at

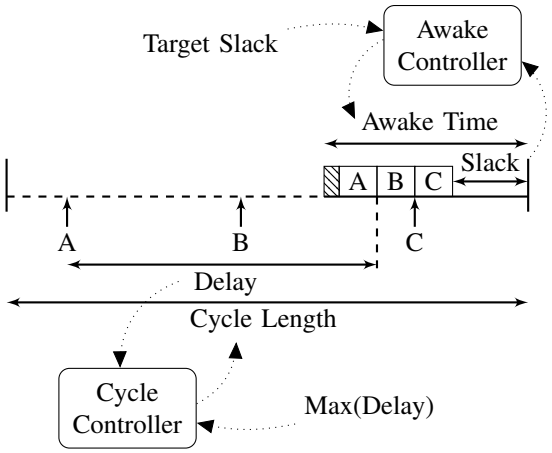

Fig. 5. The standard WSN node cycle and the PID controllers

the the rate of change in the error over time. The full formula is given in equation 9, where K_p, K_i and K_d are the P, I and D values respectively. How these controllers are used within our algorithm will be described in the next section.

$$e(t) = setpoint - value(t) \tag{8}$$

$$output = K_p e(t) + K_i \int_0^t e(t)\,\mathrm{d}t + K_d \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{9}$$

1) Batching: One solution could be to use the notion of worst case delay as mentioned previously, however getting an accurate reading for the $cooldownTime$ to calculate the delay would require constant monitoring of all the nodes. An alternative approach is to directly measure the delay experienced by the packets arriving at the operator node. This requires the message generation time to be included within the message and comparing these to the sink node on arrival. Fortunately most *sense-and-send* applications already include the time at which the reading was taken, requiring no additional information to be included within the message. This gives equation 10 as the applications message delay.

$$applicationDelay = \max_{\forall messages} delay(message) \tag{10}$$

An advantage our batching approach has is the need to only $warmup$ once per batch, allowing this wasted overhead to affect our solution less than others (we only incur the $warmup$ penalty once per X messages). This can be further reduced through the use of basic time synchronisation, as clock drift is a contributing factor to the one remaining $warmup$ time.

From this section the following requirements can be deduced. Minimisation of wasted power (equation 1) is achieved by increasing the number of messages sent per batch, which in turn is done by increasing the cycle length. This must be done whilst ensuring that the delay (which will be directly measured at run-time) will not exceed the operator specified limit.

2) Duty Adjustment: Instead of attempting to compute the factors in the equations previously presented we use a similar direct measurement technique as in section IV-B1, and alternatively measure the amount of free time (slack) in the awake period. This changes equation 3 to the following.

$$awakeTime = busy + slack + radioOff \tag{11}$$

Obtaining a measurement for the slack is simple once a small observation has been made, the end of the busy period can be determined by the arrival time of the last message. As we already define the start of the awake time, the length of the busy period can then be calculated. If we assume the $radioOff$ time is negligible then we can also easily compute the length of the slack as shown in equation 12.

$$slack = awakeEnd - time(lastReceivedMessage) \tag{12}$$

This is more clearly demonstrated in figure 6. As long as this slack is sufficiently large to allow for variations in the number of packets being sent, but reasonably small such that little time is wasted, we can reduce the awake time to another optimisation problem. We wish to minimise the awake time whilst allowing all packets to be sent. This is done by monitoring the slack and ensuring that it never reaches zero.

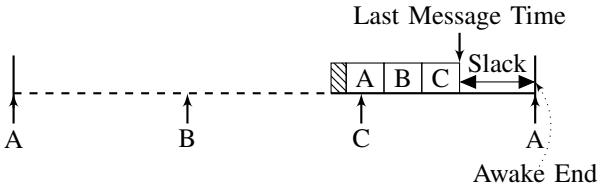

Fig. 6. Awake, sleep and slack timings

C. Summary

Taking the goals from the end of sections IV-B1 and IV-B2 we can derive the setup for the two PID controllers, shown in table I. These controllers allow us to meet objectives 1 and 2 respectively. If both the cycle length and the awake length were independent of each other then all that needs to be done is to tune the PID values, however these two factors are not independent.

TABLE I
PID LOOPS

Output	Setpoint	Input
Cycle Length	Target Delay	Current Application Delay
Awake Length	Target Slack	Current Slack

An issue arises as the output of the $cycleLength$ affects the result of the $awakeLength$, as shown in equation 13. This can more clearly be seen in figure 5, as an increase/decrease in the $cycleLength$ must either cause the $awakeLength$ to increase/decrease accordingly or the $sleepLength$ to increase/decrease. It is important to consider the interaction between these variables, as section IV-B1 shows an increase in $sleepLength$ creates an increase in batch size, which in turn reduces slack.

$$cycleLength = sleepLength + awakeLength \tag{13}$$

TABLE II
EXAMPLE APPLICATION PARAMETERS

Simulation Time	1 hour
Data Generation Rate	1 every 10 seconds
Maximal Data Delay	50 seconds
Time To Transmit	2 milliseconds
X-MAC Sleep Time	800 milliseconds
X-MAC Awake Time	6 milliseconds

To negate this issue as much as possible, we want changes to the cycle length to have minimum impact on the amount of slack. Therefore the $awakeLength$, not the $sleepLength$, must be controlled by the secondary PID loop as shown in table I. This ensures that the $awakeLength$ remains stationary as the $cycleLength$ increases, ensuring the slack also remains stationary. This has the side-effect that the $sleepLength$ now increases with the $cycleLength$. One final consideration remains however, the $cycleLength$ controller must not overpower the $awakeLength$ controller. If this is not the case, and the $awakeLength$ stays fixed whilst $cycleLength$ increases quickly, the slack will rapidly be reduced to zero. Once slack reaches zero the packet delay incurs a penalty of at least one whole $sleepLength$, causing sudden variations in the $cycleLength$ controller due to the vast change in input.

V. EXPERIMENTS AND RESULTS

In this section, we present an analysis of our approach compared against a standard X-MAC implementation and the system proposed by Wang et al. As part of the analysis we consider how well our approach meets the three objectives set out in section II-C.

To perform the analysis we developed a typical sense-and send application, with each node periodically sending sensor readings to a singular sink node. For the simulated deployments these readings are simply random as they do not impact on the results of this work. Table II gives the default parameters of the experimental application and the underlying communication parameters. For X-MAC the optimal values identified within the literature for a data rate of 0.1 packets per second were chosen [6], with Wang's approach needing no additional configuration. As with any simulators it is important to accurately model all behaviour that may affect the results, in this case the simulator needs to accurately represent the time the radio is alive. This is done by using measurements from real motes [9] for the radio transition times and the time it takes for a packet to be successfully transmitted. Brownfield et al [5] note that radio transitions take a non-negligible amount of time (4.47ms on average) and power (3.38mA) which are also modelled within the simulator.

A. Batching

The first experiment provides validation that the batched approach, as identified by Objective 1, reduces the number radio cycles and thus power. Our approach was tested with operator specified application delays in the range 0-100 seconds, in increments of 1 second, with the resultant awake time being recorded. This same procedure was also applied to Wangs solution. As the operator specified delay does not affect the behaviour of the X-MAC algorithm it was only run once.

Experiments were conducted using the custom simulator so that a large range of delays could be evaluated. The simulator was set to count the number of radio transitions that occurred, along with the cumulative time spent with the radio active within the simulated hour. These counts were then combined with the energy consumptions mentioned previously to obtain a realistic estimate of the power consumption.

All three systems were run with the standard application parameters, with the PRR at 1.0. Both X-MAC and Wang's approach needed no configuration, however in our system the PID values for the cycle and the awake period need to be set. From preliminary experiments these were set to 0.06 across the board as this provides satisfactory behaviour.

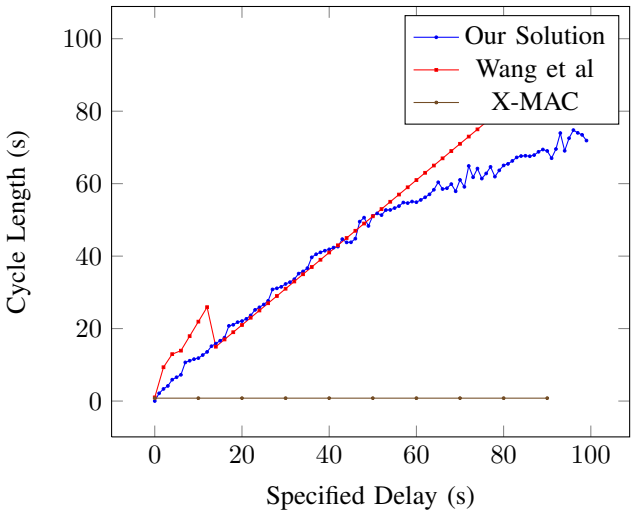

Fig. 7. The cycle length of the mote as the allowed delay increases

The results from the experiment are shown in figure 7. This figure shows the resulting cycle length that each of the three algorithms had for each of the 100 experiments. For the results each algorithm was given time for the cycle length to settle before the value was recorded. This settle time will be looked at in more detail within section V-B. In figure 7 it can be seen that our solution is near the optimal value for lower cycle lengths (cycle length == maximal delay), however at larger values the cycle length needs to reduce to accommodate the larger batches of messages. Wang et al's solution has near optimal cycle length for the entire range of results, which if ignoring batching would be the best solution. X-MAC does not adapt to the application specified delay and periodically probes the radio at a constant 800ms for all delay values.

Without batching, solutions with their cycle length closest to the maximal delay would be optimal, however further savings can be realised by batching transmissions. Figure 8 shows the proportion of cycle length that each of the three algorithms

TABLE III
RESULTS OF TYPICAL RUN, PRR 1.0

	X-MAC	Wang	Feedback Loop
Radio Transitions	8932	720	144
Total Time In Transition	39.30s	3.17s	0.63s
Time Per Tx Period	6ms	3ms	11ms
Total Tx Time	26.80s	1.08s	0.864s
Total Expense	66.10s	4.25s	1.494s
Total Power	0.291mAh	0.124mAh	0.121mAh

spends with the radio active. As our solution has a fixed amount of waste slack it is inefficient for small batch sizes (<4 messages per batch). However it can be seen that as the specified delay increases, the waste slack becomes insignificant compared to the savings obtained from not cycling the radio. This is due to the increase in batch size as the delay has increased. This is clearly shown when the delay is set to 98 seconds, as our solution is asleep for 77.9% longer than Wangs approach which remains fairly linear at high delay values. X-MAC remains linear across the board due to its fixed cycle length.

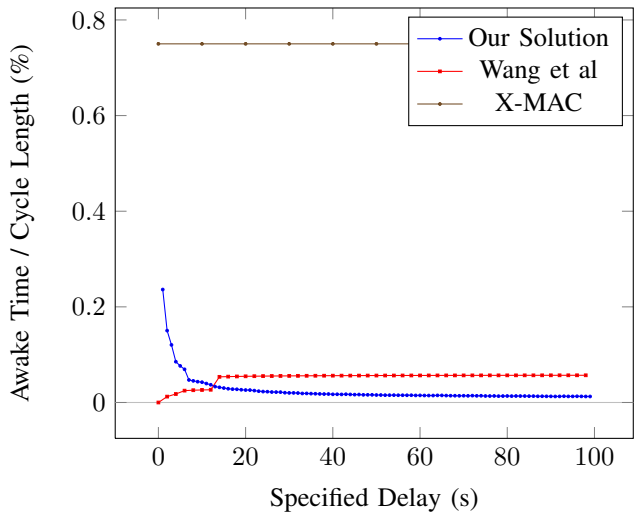

Fig. 8. Proportion of time consumed for radio communications

Table III shows the detailed results of this experiment when the maximal allowed delay is 50s. It is clear that X-MAC, even when tuned to the stated optimal values, still undergoes a large number of transitions. X-MAC also spends a large amount of time sampling the channel to check if other nodes wish to transmit, leading to a total of 66.1 seconds in high-current mode (radio active or in transition). Wang's solution performs much better due to its carefully scheduled transmissions, however transitions still dominate the majority of the high-current consumption. Our feedback approach only performs a small number of transitions, spending the majority of high-current consumption transmitting messages, clearly demonstrating that it meets Objective 1.

B. Duty Adjustment and Reaction Rate

The second experiment demonstrates how all three algorithms cope with variations within the environment, more specifically variations in the PRR. Applications were tested within the simulator due to the accurate control over the PRR that this allows. The application setup is the same as outlined in table II, with the target delay set to 50 seconds. The PRR changes numerous times during each trial, once at $t = 15$ to a PRR of 50%, at $t = 30$ to a PRR of 33%, at $t = 45$ to a PRR of 100% and at $t = 60$ to a PRR of 33%.

The message delay will be measured instead of the number of lost packets, due to active acknowledgement of received readings. As messages are actively acknowledged upon receipt, retransmission is repeated until all messages are received by the sink node. This retransmission of packets causes delay to occur, which should force the network to react in order to keep this delay below the application specified limit. This adaptive behaviour should satisfy Objective 2.

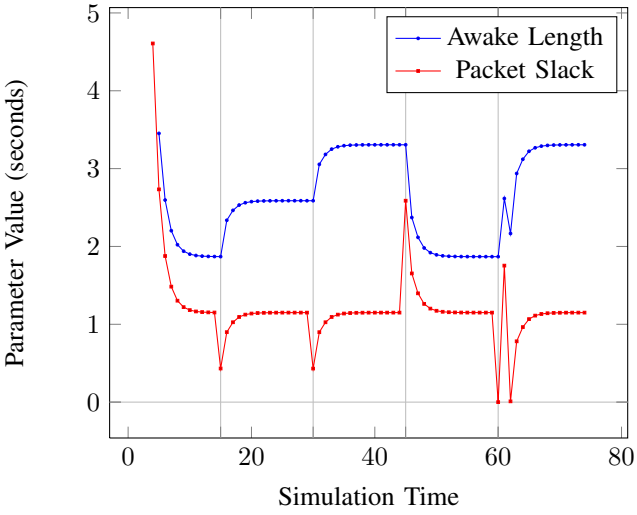

Fig. 9. Variations in cycle length due to changes in PRR

Figure 9 shows the response that the awake controller from our solution has when small and large changes in the PRR occur. It can be seen that sudden decreases in the PRR causes a drop in packet slack, which causes the awake length to be increased. Only when very large PRR changes occur does the slack reach zero, at which point the packets miss the transmission window. When packets miss the transmission window this causes large packet delays to occur. This can be more easily seen in Figure 10. Figure 9 is not directly comparable to Wang's solution as their awake length is fixed which in turn restricts the packet slack.

Figure 10 shows how this dynamic behaviour effects Objective 3, the ability for the system to react to changes in the environment. Figure 10 demonstrates that in the majority of scenarios the cycle length remains constant, meeting the objective. However in extreme circumstances, should packets miss their transmission window, the packet delay incurs a penalty. This sudden increase in packet delay correctly causes

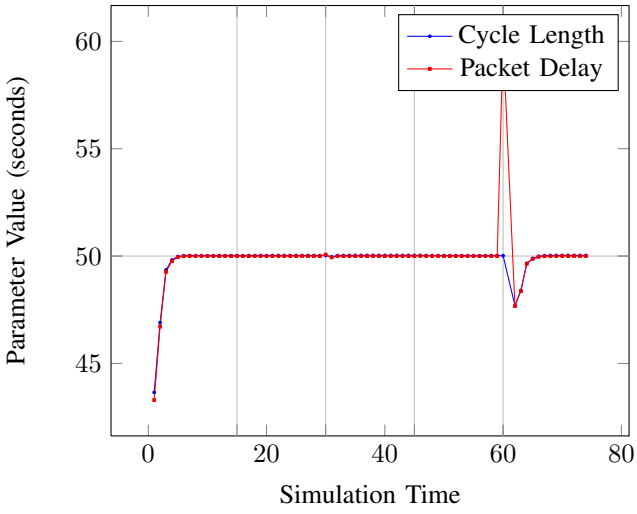

Fig. 10. Variations in cycle length due to changes in PRR

the cycle length to decrease, compensating until the slack can be restored.

Wang's approach can be seen in Figure 11. This shows that this solution also manages to adjust the message delay to meet the application requirements, however it always allows the packet delay to exceed the application limit should the PRR reduce. Even with relatively small changes in PRR, such as at $t = 15$, Wang's solution allows the application limit to be exceeded by 15 seconds, whereas our solution exceeds it by 1.45ms. Even in the case where the variation in PRR is too extreme for our solution to keep the delay under control it returns to acceptable levels in 75% less time than the alternative.

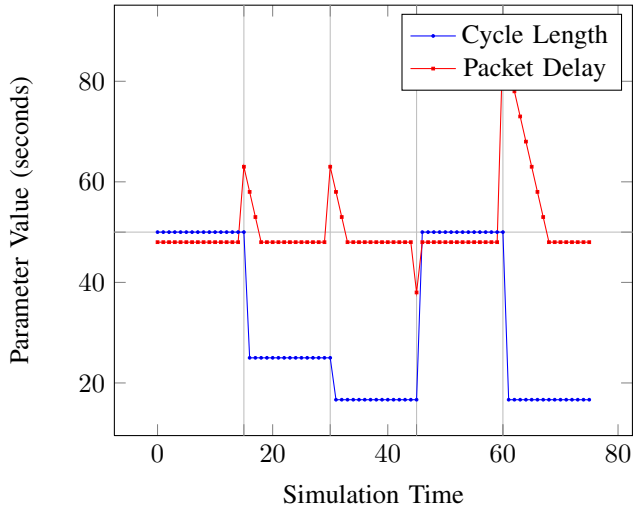

Fig. 11. Variations in cycle length of Wang's solution due to changes in PRR

X-MAC performed well, with the maximum packet delay being maintained in the sub-second range in all PRR scenarios. This is mainly due to the rapid probing that X-MAC performs, which consumes much power as shown in the previous section.

VI. Conclusions

From section V-A it can be seen that in all scenarios our solution performs with much less power than typical MAC schemes, and with less power consumption than that proposed by Wang et al when the batch sizes are 4 or greater. Our solution also relaxes a number of restrictions that Wang's approach requires, primarily we do not impose a restriction over which MAC layer and Routing layer protocols may be used with our solution. It has been demonstrated in Section V-B that in addition to these benefits our protocol provides strict control over the specified packet delay, ensuring that under normal circumstances it does not exceeded the specified limit. It has also been shown that should extreme changes in the environment occur that cause the limit to be exceeded, our solution returns to acceptable levels of delay faster than the other state-of-the-art approaches.

References

[1] Zigbee specification. Technical Report 053474r06, Zigbee Alliance, 2006.

[2] C. Alippi, R. Camplani, C. Galperti, and M. Roveri. A robust, adaptive, solar-powered WSN framework for aquatic environmental monitoring. *Sensors Journal*, 11(1):45–55, 2011.

[3] K. Astrom and R. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2008.

[4] P. Bennett, I. Stoianov, P. Fidler, C. Maksimovic, C. Middleton, N. Graham, K. Soga, and N. Hoult. Wireless sensor networks: creating'smart infrastructure'. In *Institution of Civil Engineers. Civil engineering*, volume 162, pages 136–143, 2009.

[5] M. Brownfield, A. Fayez, T. Nelson, and N. Davis. Cross-layer wireless sensor network radio power management. In *Wireless Communications and Networking Conference*, volume 2, pages 1160–1165, 2006.

[6] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *4th international conference on Embedded networked sensor systems*, pages 307–320, 2006.

[7] M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The Torre Aquila deployment. In *2009 International Conference on Information Processing in Sensor Networks*, pages 277–288, 2009.

[8] J. Polastre, R. Szewczyk, A. Mainwaring, D. Culler, and J. Anderson. Analysis of wireless sensor networks for habitat monitoring. *Wireless sensor networks*, pages 399–423, 2004.

[9] N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy, and D. Estrin. A stream-oriented power management protocol for low duty cycle sensor network applications. In *2nd IEEE Workshop on Embedded Networked Sensors*, pages 53–61, 2005.

[10] F. Stajano, N. Hoult, I. Wassell, P. Bennett, C. Middleton, and K. Soga. Smart bridges, smart tunnels: Transforming wireless sensor networks from research prototypes into robust engineering infrastructure. *Ad Hoc Networks*, 8(8):872–888, 2010.

[11] T. Van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *1st International Conference on Embedded networked sensor systems*, pages 171–180, 2003.

[12] S. Vijayakumar and J. Rosario. Preliminary design for crop monitoring involving water and fertilizer conservation using wireless sensor networks. In *3rd International Conference on Communication Software and Networks*, pages 662–666, 2011.

[13] X. Wang, X. Wang, G. Xing, and Y. Yao. Dynamic duty cycle control for end-to-end delay guarantees in wireless sensor networks. In *18th International Workshop on Quality of Service*, pages 1–9, 2010.

[14] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *21st Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1567–1576, 2002.