End-to-end Routing for Dual-Radio Sensor Networks

Thanos Stathopoulos† Martin Lukac‡ Dustin McIntire♯
John Heidemann◇ Deborah Estrin‡ William J. Kaiser♯

† Institute of Computer Science (FORTH–ICS)
Foundation for Research and Technology–Hellas
GR-711 01, Heraklion, Crete, Greece

Center for Embedded Networked Sensing
‡ UCLA, Department of Computer Science
♯ UCLA, Department of Electrical Engineering
◇ USC, Information Sciences Institute
{thanos@forth.ics.gr, mlukac@lecs.cs.ucla.edu, dustin@seas.ucla.edu, johnh@isi.edu, destrin@cs.ucla.edu, kaiser@ee.ucla.edu}

***Abstract*— Dual-radio, dual-processor nodes are an emerging class of Wireless Sensor Network devices that provide both low-energy operation as well as substantially increased computational performance and communication bandwidth for applications. In such systems, the secondary radio and processor operates with sufficiently low power that it may remain always vigilant, while the the main processor and primary, high-bandwidth radio remain off until triggered by the application. By exploiting the high energy efficiency of the main processor and primary radio along with proper usage, net operating energy benefits are enabled for applications. The secondary radio provides a constantly available multi-hop network, while paths in the primary network exist only when required. This paper describes a topology control mechanism for establishing an end-to-end path in a network of dual-radio nodes using the secondary radios as a control channel to *selectively* wake up nodes along the required end-to-end path. Using numerical models as well as testbed experimentation, we show that our proposed mechanism provides significant energy savings of more than 60% compared to alternative approaches, and that it incurs only moderately greater application latency.**

I. Introduction

Ever-increasing application demands in conjunction with advances in low-power hardware design have resulted in an increasing use of larger, more powerful sensor nodes [1], [2]. In addition to a 32-bit CPU, those nodes include sophisticated peripherals and megabytes of RAM and flash as well as a high-bandwidth 802.11 radio. 32-bit nodes are used in standalone wireless sensor network deployments [3], [4], [5] as well as in *tiered architectures*, where they operate in conjunction with microcontroller-based WSN devices [6], [7], [8].

When the 32-bit nodes are used in a tiered architecture, they must communicate with the network of microcontroller-based nodes (typically 8- or 16-bit motes [9], [10]). For this reason a new generation of 32-bit nodes as for example the LEAP node [2] include an on-board low-power microcontroller (for constantly vigilant operation) and a second, low-bandwidth radio. As a result those nodes enable not just tiered computing but also *tiered radio networking*. When a node has multiple radios with different communication capabilities and power properties, the question becomes: How should such a multi-radio system be applied to best benefit energy and application demands? It is important to note that the high-bandwidth radio operates with much greater energy efficiency than the low-bandwidth radio, in terms of energy per bit transmitted (for example, 112 nJ/bit for 802.11g as opposed to 979 nJ/bit for 802.15.4 [2]). However, the larger radio also has a much higher state transition cost and idle energy consumption, more than 10 times that of the low bandwidth radio [1], [2], [11]. It is therefore counter-productive to use the high-bandwidth radio if there is little or no data to send or if data needs to be sent only occasionally. Instead, in order to reduce energy consumption, the high-bandwidth radio should be kept *off*, to be activated only when there is a significant amount of data that needs to be transmitted. The low-bandwidth radio, on the other hand, is less energy efficient but consumes much less energy when idle and is able to quickly transition from sleep to active state, send the necessary data, and then deactivate. It is therefore ideal for transmitting small amounts of data as well as remaining "vigilant" for long time periods, especially when techniques such as Low-Power Listening [12] are used.

In a multihop network of nodes that maintain their main CPU and high-bandwidth radio in a low operating duty cycle so as to conserve energy, end-to-end paths do not always exist. In sensor network applications where observing a phenomenon for which a well-established model exists, this problem can be solved by either a static or an adaptive scheduling algorithm, where nodes coordinate in order to guarantee that their wakeup times are synchronized and to perform other low duty cycle coordination functions. However, when such a model does not exist and may not be learned as in seismic event detection, or when *timely* notification of an event is required, as in

Contribution	Section
Demonstration of multihop path wakeup	IV , V
Use of different routing protocol for each topology	II, IV-B
Alleviates the need for matching network topologies	V
Quantification of the impact of transition constants	III
Powerdown better than suspend in low frequencies	III-B
Comparison of wakeup mechanisms	II
Wake-path better than Wake-all	III , V
Always-on preferred at very high frequencies	III

TABLE I

A SUMMARY OF THE MAIN CONTRIBUTIONS OF THIS PAPER

intrusion detection, a periodic wakeup algorithm may not always perform with sufficient low latency.

To address the needs of applications such as the ones mentioned above, we propose end-to-end routing using vigilant low-power radios that activate the mostly-off high-bandwidth radios for bulk traffic. Prior work in multi-radio systems has focused on exploiting the higher capacity of different radios [13], [8], using the low-bandwidth radio for resource discovery [14], [15], as a control channel to perform networking functions such as access point association [16] and for transmission scheduling [17]. However, to the best of our knowledge none have explored how to utilize the the low-bandwidth radios to establish an *end-to-end multihop path* for the high-bandwidth radios. In our proposed approach, each node uses its low-bandwidth radio to connect to a specific node, called the *topology controller* and request an end-to-end path to a particular destination. The controller then decides which nodes to wake up based on cached information about routing paths and sends the appropriate requests to other nodes again using the low-bandwidth radio. When nodes receive the wakeup request, they turn on their CPU and high-bandwidth radio so that the end-to-end data transfer can start.

The main contribution of this paper is a new approach to end-to-end routing that enables the energy and performance optimized low-bandwidth radios to trigger high-bandwidth radios in a tiered, dual-radio network. Part of the novelty of our approach lies in utilizing different routing protocols for each radio topology, thus alleviating the need for matching topologies. We use analysis and experimentation to show the benefits of this approach, concluding that our algorithm that activates nodes along a previously established path can reduce energy consumption by more than 60% compared to alternative approaches while incurring a only a moderate increase in application latency. Table I summarizes our contributions.

II. DIFFERENT APPROACHES TO END-TO-END ROUTING FOR DUTY-CYCLED DUAL-RADIO NODES

In order to solve the problem of establishing an end-to-end path over LEAP-class nodes, we consider the following approaches:

Always-On: *A system where nodes operate all their radio and CPU resources at all times.* This system does not need a second radio and is expected to have the lowest latency in terms of data transmission but will consume the most energy (as no power-saving state is used).

Periodic-wakeup: *A system where the nodes' main processor and high-bandwidth radio are powered down and are periodically powered up to send data.* As in the previous case, this system doesn't need a second radio. This system is expected to have very low energy consumption, especially if the power-up/power-down time ratio is very low. However, timely notification will suffer since data can only be sent in predetermined periods and optimizing for energy consumption (thus powering up very infrequently) will increase latency even more. As a result, this system stresses the energy-latency tradeoff inherent in scheduling mechanisms.

Wake-all: *A system where the nodes' main processor and high-bandwidth radio are powered down and are powered up when an important event occurs.* This system uses a second radio to inform the nodes that an event has occurred and thus force them to power up. In network terms, this can be achieved by flooding (or by just turning the radio on and simply saturating the channel), i.e. no pre-established or maintained routing is necessary in the second radio. This system is expected to have low latency, as it is event-based and doesn't depend on any particular schedule. However, its energy consumption can be considerable, depending on the number of nodes in the network versus the number of nodes involved in the establishment and maintenance of the path.

Wake-path: *A system where the nodes' main processor and high-bandwidth radio are powered down but only a necessary subset of nodes are powered up when an event occurs.* This system requires a second radio to notify the appropriate nodes that are needed for the path formation. Since a specific subset of the nodes need to be reached, the system would also benefit from a unicast routing protocol on the second radio. This system is expected to have low latency, as it is event based. It is also expected to perform well in terms of energy consumption, as it attempts to wake up only the nodes that are deemed *necessary* for the path to be established. However, it depends on previous information to determine which nodes need to be woken up, and that information can potentially be invalid. As a result, its performance is more dependent on networking/link properties than that of the other systems.

The choice of the appropriate mechanism depends on the physical characteristics of the sensing phenomena as well as the networking characteristics (i.e. routing topology and bandwidth). In the following section, we will perform numerical analysis in order to discover the appropriate operating range for each mechanism.

In approaches which require the use of the low-bandwidth radio, we assume that the low-bandwidth network is *connected* and not partitioned and that the MCU which controls the low-bandwidth radio is not put to sleep, so that it can at any point in time wake up the main processor and the high-bandwidth radio. Note that this assumption does *not* mean that the two network topologies are the same. A neighboring node over the high-bandwidth radio is not necessarily a neighboring node over the low-bandwidth radio (and vice versa). The non-partition assumption implies that either the two radios have comparable ranges (i.e. within the same order of magnitude, as

is the case for 802.11g and 802.15.4) or, failing that, that the low-bandwidth network is *sufficiently augmented* with extra dual-radio nodes or even standalone mote-class nodes.

We also do not consider any *mobility patterns*, since most sensornets today are stationary. As a result the routing paths will not be invalidated due to mobility, and exploration of mobility is a potential area of future work. However, we do consider cases where routing paths become invalid when nodes *fail* and design our mechanism accordingly (Section IV-C).

III. Analysis

The primary research question that we are trying to answer is: *Under which conditions is each approach advantageous*. More specifically, we want to figure out in which case our proposed solution has clear benefits over the alternative solutions, where the benefits are measured in terms of reduced energy consumption for the duration of an entire data exchange. An "exchange" is defined as the generation of a sensor event, followed by an appropriate data transmission between the source and a destination.

The main question can be broken down into the following:

- Does powering down the nodes *always* result in reducing energy consumption? Is there a sensor event frequency for which powering down isn't beneficial anymore?
- Should nodes be always powered down or should we consider other power states such as suspend mode?
- How does the network topology affect the choice of mechanism?

As our goal is to ascertain the design space for each mechanism, in the remainder of this section we attempt to derive answers to the aforementioned questions by using simple numerical models.

A. Energy consumption for each mechanism

Using a very simple communication model where we ignore effects of channel contention, packet loss and retransmissions as well as MAC, network and transport protocol overhead, the energy required to transmit the data D_e generated by a sensor event over p hops is:

$$E_{DT,M} = \frac{D_e}{BW_M}(P_{TX,M} + P_{RX,M})\,p \tag{1}$$

where BW_r is the radio's bandwidth and $P_{TX,r}$ and $P_{RX,r}$ indicate the radio transmit and receive power respectively.

For a network of N nodes and for a total time period T, the energy consumption of the *always-on* mechanism is:

$$E_{alw\_on} = N \cdot P_{i,M} \cdot T + f_e \cdot E_{DT,M} \cdot T \tag{2}$$

where f_e is the event frequency, $P_{i,M}$ is the main processor's idle power consumption and $E_{DT,M}$ is the energy required to transmit the data generated by the event using the high-bandwidth radio, as defined in Equation 1. Note that in the above equation, we assume that in the absence of an event that requires a transmission, the 802.11 radio is *turned off* for all nodes and as a result the only idle energy cost is induced by the CPU.

In the mechanisms where nodes are turned off, we need to consider the wakeup energy cost as well as the cost of the wakeup mechanism itself.

The wakeup cost for a single node is:

$$E_{s\to w} = T_{s\to w} \cdot P_{s\to w} \tag{3}$$

where $T_{s\to w}$ and $P_{s\to w}$ are the transition time and transition power from "power down" or "suspend" to the "on" state.

In the *periodic-wakeup* mechanism nodes wake up on a timer so the cost of the wakeup mechanism can be effectively ignored if we assume that the preprocessor is also turned off and can be turned on from a hardware timer. In the event-based cases however, there is also a transmission cost associated with the wakeup that involves sending packets over the low-bandwidth radio. In the *wake-all* mechanism this can be accomplished by flooding a control packet to the entire network, so the wakeup cost is:

$$E_{W,all} = (N-1)\left[\frac{C_w}{BW_S}\left(P_{TX,S} + P_{RX,S}\right)N + E_{s\to w}\right] \tag{4}$$

where C_w is the data size of the control message and BW_S, $P_{TX,S}$ and $P_{RX,S}$ are the secondary radio's bandwidth, transmit and receive power respectively.

For *wake-path* the wakeup cost consists of sending wakeup packets to and receiving acknowledgements from all nodes involved in the high-bandwidth radio path. The actual number of transmitted packets however depends on the routing topology. To simplify our calculations, we assume a uniform path length distribution in a grid topology, so the *average* path length is $\frac{N-1}{2}$. As a result, the wakeup cost of the *wake-path* approach, assuming that the size of the control data is the same as in *wake-all* is:

$$E_{W,path} = p\left[\frac{C_w}{BW_S}\left(P_{TX,S} + P_{RX,S}\right)\cdot(N-1) + E_{s\to w}\right] \tag{5}$$

In Equations 4 and 5 we note that the per-node transmission cost (the first term inside the brackets) is almost the same. If we take into account that the size of the control data is very small (usually less than 100 bytes), the dominant term in those equations is the node wakeup cost $E_{s\to w}$ as it is about 6 orders of magnitude higher. The number of nodes N needs to be exceedingly large for the transmission cost of the control messages to be comparable to the wakeup cost therefore, for practical purposes we can ignore the transmission cost of the control messages.

Equation 5 also does not take into account the cost of *constructing* and *maintaining* the routing tree in the low-bandwidth network. One can assume that the tree formulation cost is a one-time cost and thus can be ignored. The maintenance cost however can be substantial, especially when the system needs to operate for extended periods of time. For the

	LEAP main CPU (802.11g)	LEAP MCU (802.15.4)	Stargate (802.11b)	Mica2 (CC1000)
CPU Suspend Power	25 mW	1.925 mW	300 mW	0.648 mW
CPU Idle Power	210 mW	2.095 mW	900 mW	9.6 mW
CPU Active (max load) Power	825 mW	4.31 mW	1650 mW	24 mW
CPU Off-to-On Time	30 sec	< 1 msec	30 sec	< 1 msec
CPU Suspend-to-On Time	3 sec	< 1 msec	3 sec	< 1 msec
Radio TX Power draw (max output)	1320 mW	57.42 mW	1425 mW	64.5 mW
Radio RX Power draw	924 mW	65.01 mW	925 mW	21 mW
Radio efficiency	112 nJ/bit	979 nJ/bit	427.27 nJ/bit	4453.125 nJ/bit

TABLE II

POWER CONSUMPTION, TRANSITION TIMES AND RADIO ENERGY EFFICIENCY FOR LEAP, STARGATE AND MICA2 NODES.

purposes of this analysis, we again assume that the maintenance cost is negligible. Nevertheless, the need to minimize the maintenance cost of the low-bandwidth routing protocol is one of our design decisions in creating the topology control protocol and will be discussed in more detail in Section IV-B.

The total energy consumption of *wake-all* is:

$$E_{wake\_all} = N\left(P_{i,S} + P_{sleep,M}\right)T + f_e\left(E_{DT,M} + E_{W,all} + N \cdot P_{i,M}\frac{D_e}{BW_M}\right)T \quad (6)$$

where $P_{i,S}$ and $P_{sleep,M}$ are the MCU idle and CPU sleep power consumptions respectively.

For *wake-path* the total energy consumption is:

$$E_{wake\_path} = N\left(P_{i,S} + P_{sleep,M}\right)T + f_e\left(E_{DT,M} + E_{W,path} + p \cdot P_{i,M} \cdot \frac{D_e}{BW_M}\right)T \quad (7)$$

The *periodic-wakeup* approach is similar to the *always-on* approach in that no second radio is needed. In *periodic-wakeup*, nodes wake up f_w times and in those times they transmit as much data as has been gathered since the previous wakeup, i.e. $\frac{f_e}{f_w} \cdot D_e$.

The total energy consumed for *periodic-wakeup* is:

$$E_{periodic} = N \cdot P_{sleep,M} \cdot T + f_w\left[\frac{f_e}{f_w}E_{DT,M} + N\left(E_{s\to w} + P_{i,M}\frac{f_e}{f_w}\frac{D_e}{BW_M}\right)\right]T \quad (8)$$

Using Equations 2, 6, 7 and 8 we can now compare the mechanisms with each other and determine the effects of event frequency and network topology.

B. Effects of Event Frequency and Network Topology

The event frequency f_e is the frequency at which an external sensor triggers a request for a data transfer and depending on the system used, node wakeup. Our goal is to determine the effect of this parameter on the energy consumption of each of the four mechanisms as well as determine which sleep state (power down or suspend) is more appropriate.

Our first observation is that the mechanisms that wake up nodes are *not* always more energy efficient than *always-on*. Instead, the event frequency needs to be less than:

$$f_e \leq \frac{1}{r} \cdot \frac{P_{i,M} - P_{i,S} - P_{sleep,M}}{E_{s\to w} + P_{i,M} \cdot \frac{D_e}{BW_M}} \quad (9)$$

where $r = p/N$ is the path-to-total-nodes ratio and is equal to 1 for the *wake-all* mechanism, for this to be the case. Moreover, since $r \leq 1$ we note that the *wake-path* mechanism will be always better than the *wake-all* mechanism, regardless of the value of the event frequency.

For the *wake-path* mechanism to be more energy efficient than *periodic-wakeup*, the following needs to be true:

$$f_e \leq \frac{f_w \cdot E_{s\to w} - P_{i,S}}{r \cdot E_{s\to w} + P_{i,M}\frac{D_e}{BW_M}(r-1)} \quad (10)$$

The path-to-total-nodes ratio, r can be considered as an indication of the *network topology*. In a very sparse topology (e.g. line) and when the source and destination nodes are in the edges of the network, r will be very close to one and waking up all the nodes will have more or less same effect as waking up only the nodes along the desired. On the other hand, in very dense topologies with relatively small path lengths (in the limit, a 1-hop star topology), r will approach zero regardless of the relative position of the source and destination nodes and the *wake-path* mechanism can potentially outperform even the periodic-wakeup mechanism in terms of energy consumption.

Finally, we address the issue of selecting the appropriate low power state. For suspend mode to be more energy efficient than turning the node off, the event frequency must be:

$$f_e \geq \frac{P_{suspend,M}}{E_{off\to on} - E_{suspend\to on}} \quad (11)$$

Figure 1 shows the energy consumption as a function of event frequency for the *always-on* mechanism as well as the *wake-all* mechanism for both power-down and suspend power states, using values for the LEAP nodes provided in Table II and [2]. The time period T was set to one hour and the path length p and number of nodes N was set to 10. The thin lines in the graph represent a data size of 400 KBytes while the thicker lines represent a data size of 40 MBytes. Based on this figure as well as our equations, we can see that powering

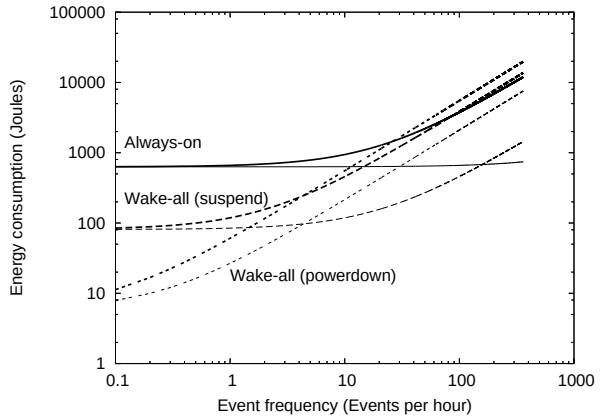

Fig. 1. Energy consumption as a function of event frequency for the *always-on* and *wake-all* mechanism using both power-down and suspend power states.

down the nodes is very energy-efficient in low frequencies. However for frequencies higher than 4.6 events per hour it is better to place the nodes on *suspend mode*. Moreover, as the event frequency reaches higher values, it actually becomes more energy efficient to keep the nodes always on. Placing nodes on suspend mode yields more benefits in this case as well. We also note that increasing the data size results in a reduction of the maximum event frequency for which the wake-all mechanism is beneficial.

IV. The topology control protocol

The purpose of the topology control protocol is to wake up all the nodes that are required in order to form an end-to-end multihop path from the initiator of the request (source) to the final destination of the data. Therefore, the protocol needs to first determine the nodes that are required for the path to form and then send control packets over the low-bandwidth radio to wake them up.

A. Distributed and Centralized approaches

One can determine the required nodes using either a distributed or a centralized approach. In the distributed approach, an any-to-any routing protocol is used over the high-bandwidth radio. Each node in the network keeps a path to all potential destinations in its routing table. In addition, the routing table is assumed to exist in non-volatile memory so that it doesn't get affected by shutdowns. When an event occurs, the node selects the candidate nodes along the path and sends wakeup packets to them along the low-bandwidth radio. Since memory restrictions on mote-class devices make any-to-any routing protocols difficult in full systems [18], the node needs to either create a low-bandwidth radio routing tree rooted at itself on the fly, or resort to flooding (or some more efficient reliable dissemination mechanism, e.g. Trickle [19]) to distribute the requests. The candidate nodes can then reply to the requesting node either by using a flooding protocol again or by unicasting their replies back, if a unicast path over the low-bandwidth radio is available. After receiving confirmation from all involved nodes, the initiator node can start the data transfer over the high-bandwidth radio.

In the centralized approach, the nodes again use an any-to-any routing protocol over the high-bandwidth radio. Each node in the network now sends its routing table information to a special node, called the *topology controller*. The topology controller is considered to be always on. In addition, the topology controller has formed a routing tree to all other nodes in the network over the low-bandwidth radio with itself as the root. When an event occurs, the initiator node sends a unicast request to the topology controller over the low-bandwidth radio. The controller then selects the candidate nodes to be woken up and sends unicast control packets to them. After the nodes wake up, they inform the controller over the low-bandwidth radio (they cannot use the high-bandwidth radio since a path to the controller doesn't necessarily exist) and the controller in turn informs the initiator node that the path exists. The initiator node can then start the data transfer.

The main advantage of the distributed approach lies in the fact that it does not need to forward data to a particular node in the system. Moreover, it does not have a single point of failure, compared to the centralized approach. On the other hand, in the centralized approach the controller has knowledge of all the nodes that are awake in the entire network. When *multiple concurrent transfers* are involved the controller can decide not to turn on any extra nodes if the network is sufficiently connected. This optimization which can lead to considerable energy savings cannot be easily done with a distributed system as each individual node does not have sufficient information about the power state of all other nodes.

In applications that send data to a particular node in the system (as is the case for several data collecting applications) one can use the centralized approach and co-locate the controller with the destination node. As each node needs to send data to that single destination, the controller can collect the routing topology without any extra transmissions as in the general case. One extra advantage of co-locating the controller with a well-known data destination is that the low-bandwidth network can now be efficiently used (compared to flooding) for transmitting low-rate sensor data, e.g. temperature, humidity etc. For these reasons, our current implementation is based on the centralized approach. A further study of the distributed approach and a comparative analysis with the centralized approach is part of our future work.

B. Routing considerations for both radio topologies

In choosing the routing protocols for the high-bandwidth and low-bandwidth radios, we need to consider the usage and power patterns of those two networks. The high-bandwidth network will be disconnected for most of the time as nodes are placed to sleep. Once a path needs to be established however, routing formulation should take as little time as possible, in order to minimize idle energy consumption which is considerable in the high-bandwidth radio and main CPU. In addition, since the high-bandwidth path will in general be

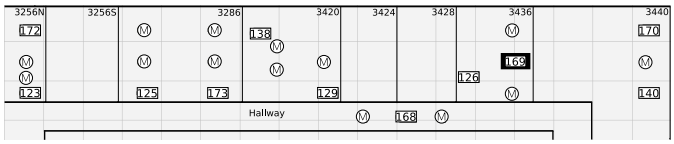

Fig. 2. The testbed used in our experiments, where Stargates are represented by their ID and standalone motes are indicated by a circle.

short-lived, the control overhead of the routing protocol and consequently its *quiescent cost* is not of primary concern.

In the low-bandwidth network however, the quiescent cost and control overhead are points of concern. Since the low-bandwidth network is considered to be always connected, even a small but *periodic* routing control overhead can amount to considerable energy consumption over a large time period without any tangible gains, as data is rarely being transmitted. As a result an *on-demand* routing protocol is more appropriate for the low-bandwidth network, in terms of long-term energy consumption.

C. Implementation

Based on the above, we used a custom implementation of DSR [20] that uses the ETX metric [21] as our 802.11 routing protocol and CentRoute [22] as our low-bandwidth routing protocol. CentRoute is a centralized on-demand unicast mote routing protocol which provides high degrees of network connectivity in high-density networks. The topology controller receives routing input from DSR in order to build up its cache of paths to all other nodes in the network. When a path request arrives over CentRoute the controller consults its paths cache and selects the appropriate nodes for the wakeup based on the most recent valid 802.11 path that existed before the nodes were put to sleep. It then dispatches wakeup control packets to those nodes over CentRoute and waits for those nodes to confirm that they have turned on their main CPU.

Since the candidate path is selected based on past information and since we consider node failures, it is possible that the candidate path will be invalid. Therefore, the topology controller needs to be able to discover such cases and select alternative paths, if possible. In our current implementation we only consider node failures that affect *both* the main CPU and the MCU and both radios. If a confirmation packet is not received within a specific time period, the controller retransmits the request. After a number of repeated requests the controller considers that node to be *invalid*. All the paths containing the invalid node are subsequently invalidated as well and the controller attempts to select another valid path in order to satisfy the request. As a fallback measure, if no such path is known to exist (from the controller's perspective) the controller floods a wakeup request to *all* nodes in the network.

V. Experimental Evaluation

In this section, we use experiments on real hardware in order to ascertain the validity of our numeric models as well as measure the performance of our topology control protocol.

Since we only had a very limited number of LEAP nodes, we conducted all our experiments using our testbed consisting of 11 Stargates, each with with a Mica-2 mote attached. Radios are 802.11b on the Stargates and CC1000 running B-MAC [12] on the motes. During our initial experiments, we discovered that even though the 802.11b link topology was adequately connected, the CC1000 topology was partitioned. In particular, the correlation coefficient between the two link topologies had an r^2 value of 0.19, with several 802.11 links lacking an equivalent in the CC1000 topology. Our topology control mechanism requires the low-bandwidth network to be connected; therefore we used 14 extra standalone motes from our testbed to augment the CC1000 topology. The experimental topology is shown in Figure 2 where the stargates are shown by rectangular numbered boxes.

All of our experiments were run using the EmStar framework [23]. The mote-specific code for both Stargate-connected and standalone motes was run using the EmTOS [24] emulation module of EmStar, which allows development of fully functional NesC applications in the resource-rich environment of a 32-bit platform. All the motes ran the MoteNIC software [24] which is similar in functionality to the TinyOS *Serial Forwarder* and enables the host device (PC or Stargate) to use the mote as a *network interface card*.

The sleep and wakeup process on the Stargate hardware was simulated by sending specific signals to the Stargate-specific communication stack (which included the DSR routing module and a neighbor discovery module) from the emulated mote code. For our experiments, we chose *suspend* mode as our low-power state. The suspend-to-on transition time was simulated by using a delay timer of 3 seconds, based on the values reported in Table II and [2]. The timeout value for the topology controller's reliability mechanism was set to 5 seconds and the number of retries was also set to 5. Throughout our experiments we report mean values. Error bars indicate 95% confidence intervals.

A. Energy Consumption

Our first experiments involved measuring the energy consumption of our *wake-path* mechanism and comparing its performance with that of *wake-all* mechanism.

We chose Stargate 169 as the node that ran the topology controller which consequently was the root of the CentRoute tree and also the destination node for all other Stargates. As an initial bootstrap process, all Stargates attempted to establish valid paths over 802.11 to node 169. All the nodes (besides 169) were subsequently put to (simulated) sleep as described in the beginning of this Section. We then sent a "wakeup" command to a Stargate which initiated the topology control protocol. Once a path over 802.11 had been established, we initiated a 4 MByte data transfer over 802.11 to node 169, again using TCP as our transport protocol. After the completion of the transfer, we again put all Stargates to sleep. The above process was repeated until we had at least 10 sample points for path lengths ranging from 2 to 5. Our measurements involved the number of nodes woken up, the

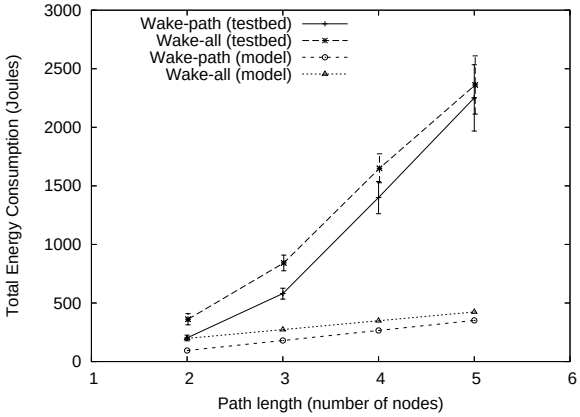

Fig. 3. Total energy consumption for all nodes in the network as a function path length for a 4MByte data transfer, for the *wake-path* and *wake-all* mechanisms.

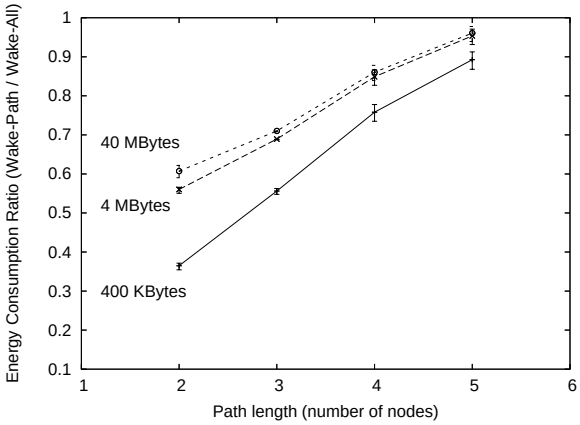

Fig. 4. Energy consumption ratio of the *wake-path* mechanism versus the *wake-all* mechanism as a function of path length for 400KByte, 4MByte and 40MByte data transfers.

duration of the wakeup process as well as the duration of the data transfer and the total number of bytes transmitted and received. Those values were then multiplied with the energy/power values of Table II.

Figure 3 shows the total energy consumption for all nodes in the network as a function of path length, using the "suspend-to-on" power state transition mode. Our expectation, based on the numerical models that the *wake-path* mechanism is more energy efficient than the *wake-all* mechanism is confirmed by the testbed results. However, we also note that there is an order-of-magnitude difference between the numerical results and the testbed ones which becomes especially pronounced as the path length increases. The reason for this large deviation is due to the *effective bandwidth* which is much smaller than the nominal bandwidth of 802.11b (1 Mbit/sec in ad-hoc mode for our Stargate testbed) used in the numerical model. As a result of the reduced bandwidth, the latency of the entire operation increases and this has a direct effect on energy consumption as the nodes need to keep their main CPU and 802.11b radio on for significantly longer time periods.

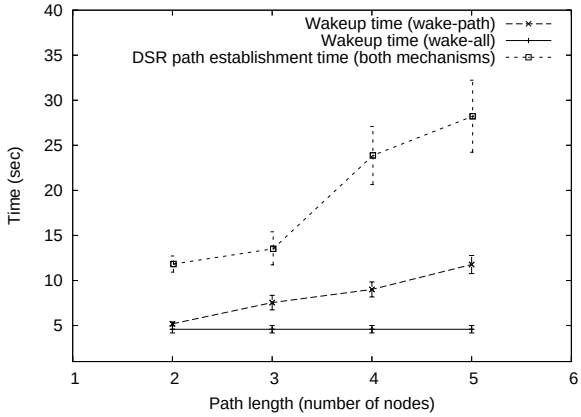

Fig. 5. Time required for the node wakeup process using the the *wake-path* and wake-all mechanisms as well the DSR path establishment time for both mechanisms as a function of path length.

Based on Equations 6 and 7, the energy gains of *wake-path* compared to *wake-all* are higher for smaller data sizes. Figure 4 shows this to be indeed the case for our testbed experiments. The *wake-path* mechanism is up to 60% more energy efficient than the *wake-all* mechanism when transferring 400 KBytes of data. The differences become significantly less pronounced as the data size increases and the data transfer energy cost becomes the most dominant energy consumption factor. Figures 3 and 4 also showcase the effect of the *path-to-total-nodes* ratio, r. As the path length increases, so does r and consequently the energy consumption of the *wake-path* mechanism also increases.

B. Latency

Our next experiments focus on characterizing the timing properties of the *wake-path* and wake-all mechanisms. In particular, we are interested in discovering the time required for the wakeup operation, the DSR path establishment time (once the wakeup operation has completed) and the time required for TCP to transfer the data.

The extra time required for the topology controller to wake up the required nodes as well as the DSR path establishment time is shown in Figure 5. As expected, the latency cost of the wake-all mechanism is independent of the path length, since all nodes are being woken up. When using the *wake-path* mechanism, the topology controller needs to contact the nodes as well as wait for replies from them. Moreover, even though CentRoute employs link-layer retransmissions, there is always a probability of a packet loss and the probability of *at least one* control packet being lost increases as the number of control packets increases, i.e. when the path length increases. The controller (or the node that requested the path) will timeout if a reply has not been received after 5 seconds as mentioned in Section IV; therefore the increased probability of a packet loss also has an effect on the latency of the path establishment mechanism. Nevertheless, we note that the latency penalty of the wakeup mechanism itself is fairly low. Considering that the simulated suspend-to-on time is 3 seconds, the actual delay

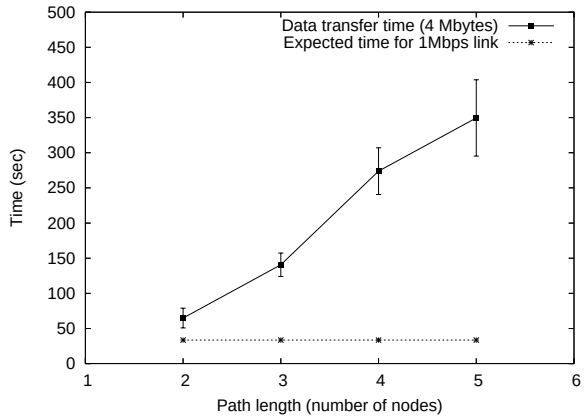

Fig. 6. Transfer time for a 4 MByte TCP data transfer as a function of path length.

induced by the protocol is approximately 9 seconds at higher path lengths.

The DSR path establishment time on the other hand requires considerably more time than the wakeup mechanism itself and is also dependent on the path length. This is to a large extent due to our particular implementation of DSR, where, in order to deal with unreliable links as well as limit the overhead of path discovery, the time between consecutive route requests was set to 10 seconds. Setting the timer to 5 seconds resulted in an average of 2–5 seconds reduction in the delay. As an further optimization, we have considered disabling DSR route discovery after a wakeup operation and instead piggyback the required routing information on the wakeup packets sent by the topology controller. We also note that DSR path establishment time is *independent* of the choice of a wakeup mechanism, as path establishment happens after the wakeup process has completed.

The dominant latency factor in the entire system is the data transfer time which can be almost an order magnitude more than the expected time based on the capacity of the link, as shown in Figure 6. Even in the one-hop case (path length of 2) the average data transfer time is 64.87 seconds, corresponding to a bandwidth of 559 Kbits/sec. Moreover, there is a significant increase in latency (which translates directly to an increase in energy consumption as shown in Figure 3) when the path length is 4 nodes. This is due to our experimental topology, where the central nodes have highly varying links and often unreliable links to their neighbors, with link qualities ranging from 40% to 95%. Those central nodes (in particular nodes 168 and 129) are *always* found in path lengths of 4 nodes or more to destination 169. This is also the reason why the time for DSR path establishment exhibits a considerably higher delay when the path length is 4 nodes or more (Figure 5). Consequently the end-to-end path reliability of long paths can be as low as 40% or less at times.

TCP is known to perform poorly when the end-to-end path exhibits such high losses (in addition to consecutive packet losses). As a result we are considering using alternative transport protocols like DTN [25] or loss- or wireless-optimized TCPs for our future implementations and deployments.

Failure case	Normal	Single-Node
Success Rate (%)	100%	96.7%
Wakeup Time (sec)	9.3	42.6
Nodes active	4	6
Path bandwidth (Kbps)	134.4	92.8

TABLE III
SUCCESS RATE, WAKEUP TIME, ACTIVE NODES AND PATH BANDWIDTH FOR NORMAL AND SINGLE-NODE-FAILURE CASES.

C. Reliability

Our final experiment focused on the reliability aspects of the wake-path mechanism. For this, we used a simple *single-node failure* model, where we turned off a node along the expected path *before* sending a wakeup signal. For this experiment, we chose a single Stargate (123) as our sender and again 169 as the receiver and the host of the topology controller. After the bootstrapping process was completed we again placed all Stargates to sleep. We then queried the topology controller to discover the path that would be selected should a wakeup request arrive from 123 and randomly turned off one of the nodes in the path (excluding the endpoints). We measured the wakeup delay, the number of nodes turned on by the *wake-path* mechanism and the end-to-end bandwidth reported by TCP for our single-node-failure scenario as well as a normal scenario without any node failures. We ran each scenario 30 times and report average values.

Table III shows the results of our reliability experiment. The reliability mechanism of *wake-path* allowed it to establish successful paths in 29 out of 30 runs yielding a reliability percentage of 96.7% for the single node failure case. However, the reliability mechanism induces significant delay in terms of This is due to the current implementation of the topology controller which attempts to wake up a candidate Stargate up to 5 times in a row, with 5 second timeout intervals in between successive attempts. Moreover the effective bandwidth of the end-to-end path is reduced in the single-node-failure scenario. This is expected since the *wake-path* mechanism first attempts to bring up the *best path*, based on the ETX metric which is the path with the highest throughput.

Based on our encouraging initial testbed results, in the future we plan to test our topology controller protocol in the actual LEAP platform as well as evaluate its performance in outdoors and real-world deployment scenarios.

VI. RELATED WORK

Prior work on multi-radio systems has focused on hierarchical power management [26], exploiting the capacity of the higher-bandwidth radio in a tiered architecture [8], using the second radio as a paging and control channel for resource and neighbor discovery and mobility support [14], [16], [15] and for transmission scheduling [17].

Turducken [26] is a multi-tiered power management architecture for mobile systems. The mobile devices are comprised of three individual nodes: a mote, a PDA and a laptop. The mote remains always on while the PDA is duty-cycled and used in system tasks and application tasks that do not require user input and finally the laptop is used only whenever user input is required. Turducken is similar to our system in that it uses both low and high-bandwidth radios and multiple tiers as well as being able to activate higher-power tiers like the PDA and the laptop from the lower-power tier. However, the use of the low-bandwidth radio is limited to time synchronization. In addition the high-bandwidth radio is only used in single-hop infrastructure mode so there is no need for multihop path establishment.

Yarvis et al. [8] propose using selected line-powered nodes with high-bandwidth radios as backhaul links to deliver data with the goal of increasing the lifetime of the entire network. This work is similar in that it consists of a tiered networked system with both high and low-bandwidth nodes. However, its goal is to use the longer range and increased bandwidth of the high-bandwidth radio to reduce the total number of transmissions and thus the energy consumption of the mote-class devices. In contrast, our system uses the high-bandwidth radio as its primary data transfer device and focuses on reducing the energy consumption of the high-power nodes.

Using the low-power radio as a paging and control channel and as a means to wake up a higher tier has been explored in [14] and [16]. Wake-on-Wireless [14] is a two-tier system comprised of a PDA and a mote-class device. The system is used in a Voice over IP scenario where the mote-class device is acting as a paging channel by announcing its presence to a nearby server and waiting for a possible incoming call. If a reply is received from the server, the main device (PDA) and 802.11 radio are turned on. Again as in our system, the high-power tier (PDA and 802.11 radio) is placed in a low-power state and can be woken up from the low-power tier. However the operation of both radios is limited to single-hop infrastructure mode without any requirements for multihop routing. In [16] the low-power radio is used to discover access points, and configure and activate the high-power 802.11 radio accordingly. Using the low-power radio as opposed to 802.11 to discover nearby access points results in significant power savings. This system however operates exclusively in infrastructure mode.

Jun et al. [15] explore using the low-power radio to discover neighboring nodes in order to transmit data in the context of mobile delay tolerant networking. Contrary to our system, both radios are duty cycled, routing is DTN-like and the low-power radio is exclusively used for neighbor discovery. Moreover, this work only deals with radio duty cycling and does not consider turning off any other parts of the system like the main CPU.

STEM [17] focuses on using the second radio as a paging channel that only transmits activation signals. The duty-cycling is in the order of hundreds of milliseconds and as a result STEM is similar to a TDMA scheduling mechanism. We instead consider the case of sometimes using the second radio to transmit data, a multi-hop low-power radio network and considerably larger duty-cycling times.

VII. Conclusion

In this paper we presented a topology control mechanism for establishing end-to-end paths in networks of dual-radio nodes, where the secondary radios are used as a multihop control channel. The topology control mechanism *selectively* wakes up only the nodes required for the path to form, by sending control messages to them through the low-bandwidth radio. We considered alternative approaches such as waking up all the nodes in the network or keeping all the nodes active at all times. We then used numerical models to determine the best parameters where our proposed approach is beneficial, as well as ascertaining the appropriate low-power state. Our models indicate that the mechanisms that place nodes to sleep are most beneficial when the sensor event frequency is low but they can be counterproductive when events are very frequent.

We evaluated the performance of our proposed mechanism in terms of energy consumption and latency using a testbed comprised of Stargate and mica2 nodes. Our results show that our mechanism provides energy savings of more than 60% compared to a mechanism that wakes up all the nodes while incurring up to 12 seconds of additional delay. Finally, our topology control mechanism was able to deal with single-node failures and establish alternative paths 96.7% of the time.

Acknowledgements

This work was made possible with support from The Center for Embedded Networked Sensing (CENS) under the NSF Cooperative Agreement CCR-0120778 and by a Marie Curie Transfer of Knowledge (ToK- DEV) Grant ASPIRE within the 6th European Community Framework Program

References

[1] Intel, "Intel stargate, http://platformx.sourceforge.net," 2002.

[2] D. McIntire, K. H. Hing, B. Yip, A. Singh, W. Wu, and W. Kaiser, "The low power energy aware processing (leap) system," in *IPSN SPOTS*, April 2006.

[3] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating distributed acoustic sensing platform," in *ACM SenSys 2006 (to appear)*, 2006.

[4] M. Lukac, L. Girod, and D. Estrin, "Disruption tolerant shell," in *ACM SIGCOMM CHANTS (to appear)*, Pisa, IT, 2006.

[5] W. Merrill, L. Girod, B. Schiffer, D. McIntire, G. Rava, K. Sohrabi, F. Newberg, J. Elson, and W. Kaiser, "Dynamic networking and smart sensing enable next-generation landmines," *IEEE Pervasive Computing Magazine*, Oct-Dec 2004.

[6] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *SIGCOMM Workshop on Communications in Latin America and the Carribean*, 2001.

[7] M. Batalin, G. S. Sukhatme, Y. Yu, M. H. Rahimi, G. Pottie, W. Kaiser, and D. Estrin, "Call and response: Experiments in sampling the environment," in *ACM SenSys*, 2004.

[8] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *IEEE Infocom 2005*, 2005.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," in *ASPLOS-IX*, 2000.

[10] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *IPSN SPOTS*, 2005.
[11] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *SenSys '04*.
[12] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM Sensys*, 2004.
[13] E. Brewer et al, "A network architecture for heterogeneous mobile computing," 1998.
[14] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on wireless: An event driven energy saving strategy for battery operated devices," in *Mobicom*, 2002.
[15] H. Jun, M. Ammar, M. Corner, and E. Zegura, "Hierarchical power management in disruption tolerant networks with traffic-aware optimization," in *ACM SIGCOMM CHANTS (to appear)*, 2006.
[16] T. Pering, V. Raghunathan, and R. Want, "Exploiting radio hierarchies for power-efficient wireless discovery and connection setup," in *18th International Conference on VLSI Design*, 2005.
[17] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. B. Srivastava, "Topology management for sensor networks: Exploiting latency and density," in *MobiHoc*, 2002.
[18] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM Sensys*, 2003.
[19] P. Levis, N. Patel, D. E. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks.," in *NSDI*, 2004.
[20] David B Johnson and David A Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. 1996.
[21] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Mobicom*. ACM, 2003.
[22] T. Stathopoulos, L. Girod, J. Heidemann, and D. Estrin, "Mote herding for tiered wireless sensor networks," Tech. Rep. CENS-TR-59, 2005.
[23] L. Girod., J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin, "Emstar: a software environment for developing and deploying wireless sensor networks," in *USENIX*, 2004.
[24] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "Tools for deployment and simulation of heterogeneous sensor networks," in *SenSys*, 2004.
[25] K. Fall, "A delay tolerant network architecture for challenged internets," in *SIGCOMM*, 2003.
[26] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins, "Turducken: Hierarchical power management for mobile devices," in *MobiSys*, 2005.