A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio

Matthew J. Miller, *Student Member, IEEE,* and Nitin H. Vaidya, *Senior Member, IEEE*

***Abstract*— For increasing the life of sensor networks, each node must conserve energy as much as possible. In this paper, we propose a protocol in which energy is conserved by amortizing the energy cost of communication over multiple packets. In addition, we allow sensors to control the amount of buffered packets since storage space is limited. To achieve this, a two-radio architecture is used which allows a sensor to "wakeup" a neighbor with a busy tone and send its packets for that destination. However, this process is expensive because all neighbors must awake and listen to the primary channel to determine who is the intended destination. Therefore, triggered wakeups on the primary channel are proposed to avoid using the more costly wakeup procedure. We present a protocol for efficiently determining how large the period for these wakeups should be such that energy consumption is reduced.**

***Index Terms*— Sensor networks, Power management.**

I. Introduction

SENSOR networks present many challenges in wireless ad hoc networks. While the exact application of sensor networks is speculative, certain properties are typically assumed. First, sensors are static after initial deployment. Second, energy is scarce and it is inconvenient or impossible to replenish the energy source frequently.

Because energy should be conserved, power save protocols are needed. This problem can be addressed at each layer of the network stack. Our specific focus is the Medium Access Control (MAC) layer since this gives a fine-grained control to switch the wireless radio on and off. The fundamental question MAC layer power save mechanisms seek to answer is: *When should a device switch to a low power mode and for how long?*

Radios typically have four power levels corresponding to the following states: transmitting, receiving, listening, and sleeping. Typically, the power required to listen is about the same as the power to transmit and receive. The sleep power is usually one to four orders of magnitude less. For Mica2 Mote sensors [1], these power levels are: 81 mW for transmit, 30 mW for receive and idle, 0.003 mW for sleep. Thus, a sensor should sleep as much as possible when it is not engaged in communication.

We present a protocol designed for a topology where all sensors are within range of each other. This protocol is then extended to multiple hop and multiple flow cases. As in previous work [2]–[4], we assume that a second radio is available to awake neighbors. This second radio uses much less power via either a low duty cycle [3] or hardware design [4]. It is assumed the second radio is capable of transmitting a busy tone, rather than actual data. This allows a simpler, more energy efficient design. However, it introduces a problem: each busy tone must wakeup a node's entire neighborhood since the intended receiver's ID is not encoded on the wakeup channel. The main contribution of this work is selectively waking up the data radio at nodes that have previously engaged in communication via rate estimation. Analytically, we derive equations to find the optimal wakeup interval to minimize the energy consumption.

In Section II, we review related work. We describe the radio energy model we use in Section III. Section IV describes and analyzes our proposed protocol. Section V presents simulation results. Finally, we conclude and discuss future work in Section VI.

II. Related Work

The IEEE 802.11 specification [5] is the standard currently used by commercial WLAN cards. It specifies a MAC protocol for wireless access in both ad hoc environments, called the Distributed Coordination Function (DCF), and centralized systems, called the Point Coordination Function (PCF). Additionally, a Power Save Mode (PSM) is also specified in the standard.

The PAMAS protocol [2] adapts basic mechanisms of IEEE 802.11 [5] to a two-radio architecture. PAMAS allows a node to sleep to avoid overhearing a packet intended for a different destination or to avoid interfering

- *M.J. Miller is with the Coordinated Science Laboratory, and the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Email: mjmille2@uiuc.edu.*
- *N.H. Vaidya is with the Coordinated Science Laboratory, and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Email: nhv@uiuc.edu.*

with another node's reception by transmitting. However, unlike our work, it ignores the idle listening problem.

The PicoRadio [4], [6] design uses a low-power wakeup channel. It uses a MAC protocol that allows nodes to wakeup a neighbor when data needs to be sent. However, the design uses a CDMA scheme that requires neighbors within a 2-hop range to be assigned a unique channel and discover and maintain the channel IDs for 1-hop neighbors. Also, the channel ID is encoded in the wakeup signal, which increases hardware complexity. Our approach can be adapted to similar hardware which uses a busy tone on the wakeup channel. A wakeup channel is also used in [7]. The protocol is implemented using off-the-shelf hardware. However, the protocol is designed for systems with centralized access points or proxies and not fully distributed networks.

A theoretical approach to multiple channel power save is investigated in [8]. This centralized protocol allows nodes to cycle through S sleep states. Each state uses less power, but requires more energy to transition to the idle state. The base station uses a RF wakeup channel to awake all nodes in a given sleep state. If a node determines the base station has no data for it, it returns to a sleep state. The protocol operates to minimize energy consumption while meeting QoS requirements.

S-MAC [9] is a protocol developed specifically to address energy issues in sensor networks. It uses a simple scheduling scheme to allow neighbors to sleep for long periods and synchronize wakeups. In S-MAC, nodes enter sleep mode when a neighbor is transmitting and fragment long packets to avoid costly retransmissions. S-MAC is designed to save energy on a single radio architecture. While this approach does allow packets to be buffered, it provides no mechanism to communicate with the receiver on-demand. Also, S-MAC uses a fixed sleep interval regardless of traffic.

STEM [3], [10] is a two-radio architecture that achieves energy savings by letting the data radio sleep until communication is desired while the wakeup radio periodically listens using a low duty cycle. A node with data to send, begins transmitting continuously on the wakeup channel long enough to guarantee that all neighbors receive the wakeup signal. Another variant of STEM [3] uses a busy tone, instead of encoded data, for the wakeup signal. Our protocol is similar to STEM, but achieves greater energy savings by periodically listening on the primary channel and buffering packets.

The STEM protocol specifies a mechanism to wakeup a host from a sleeping state. The STEM protocol is general in that it can be used in conjunction with any MAC layer transmission scheduling scheme. Also, STEM can be used in conjunction with any mechanism to determine when a host should return to sleep. For instance, a host that is awakened from a sleeping state may remain up long enough to receive a "session" of packets [11]. Alternatively, STEM can be used to wakeup a host whenever packets are pending for that host. In this case, when node **A** wakes up another node, **B**, node **A** would transmit all pending packets for **B** before **B** may return to sleep again. The latter approach is used in this paper when testing STEM.

In [12], energy is saved by adjusting to traffic. The protocol works with on-demand routing and uses 802.11's PSM when a node is not engaged in sending, receiving, or forwarding data. When a node is communicating, soft-timers are used to transition the node to an idle listening mode which reduces latency and preserves throughput better than only using 802.11's PSM. However, the timers do not adjust to the traffic rate, so if traffic is not frequent enough to refresh the timers, the benefits of the protocol are lost.

Other work chooses a subset of the nodes in a system to enter a low power state without significantly degrading the performance achievable if all nodes were to remain in high power mode. AFECA [13] and GAF [14] allow nodes to sleep based on their neighborhood size and geographic location, respectively. The basic idea is to maintain the connectivity of a network while allowing most nodes to sleep. Similarly, the goal of SPAN [15] is to save energy while not degrading the latency and throughput achievable in 802.11 without PSM. Neighborhood information helps maintain connectivity in the network with uniform energy usage among the nodes.

Another method of conserving energy in sensor networks is by doing TDMA to schedule traffic in the network and allowing nodes to sleep when they are not scheduled to send or receive. Such an approach lends itself well to sensor networks, when compared to ad hoc networks in general, because a relatively static topology is expected and traffic patterns may be more regular (e.g., periodically sending updates to a sink). The key research challenge is determining how slots can be assigned in multiple hop networks to avoid collisions [16], [17].

III. Energy Model

As mentioned in Section I, we use an energy model based on the Mica2 Motes [1]. This hardware is widely used in sensor network research. The power levels discussed in Section I are for a 3V power supply. The transmit power is for the maximum possible transmit power, so it may be less in practice depending on the desired range. According to [1], this transmit power gives an outdoor, line-of-sight range of 152.4 m (500 ft). In

addition, our energy model accounts for the time and power required for the radio to transition from the sleep state to idle and from the idle state to sleep. In practice, these values are non-negligible, but not accounted for in most previous work. We denote the time and power required for a sleep to idle transition as T_{tran_on} and P_{tran_on}, respectively. Similarly, the idle to sleep transition is characterized by T_{tran_off} and P_{tran_off}. Based on the radio used in Mica2 Motes, we use the following values: 2450 μs for T_{tran_on}, 250 μs for T_{tran_off}, 30 mW for P_{tran_on}, 30 mW for P_{tran_off}.

The time values are based on the typical transition time from the radio's datasheet [18] and correspondence with Chipcon technical support. The power values are conservative estimates which assume power consumption remains at the level of the highest power state (i.e., idle) during the entire transition. We feel this assumption is justified since the transition is when the power level has changed and the electrical components must reach an operational, steady state. We note that the sleep to idle transition, in particular, takes a relatively long time.

IV. Protocol Description and Analysis

A. Triggered Wakeups with Queuing

As mentioned previously, two channels are assumed: primary and wakeup. The primary channel is used for sending data and control packets, whereas the wakeup channel is used to wakeup neighbors. For the rest of the paper, we assume that the wakeup radio achieves low power consumption via a duty cycle. Thus, the two radios have identical power characteristics (described in Section III). A node will listen for a busy tone on the wakeup channel for τ_1 time, then sleep for τ_2 time ($\tau_1 \ll \tau_2$). The sender of a wakeup signal must transmit for T_{wake_TX} time to guarantee all neighbors hear the wakeup signal, where, $T_{wake_TX} = 2\tau_1 + \tau_2 + T_{tran_on} + T_{tran_off}$. The duty cycle of the wakeup channel is defined as:

$$\frac{T_{tran_on} + \tau_1 + T_{tran_off}}{T_{tran_on} + \tau_1 + T_{tran_off} + \tau_2} \quad (1)$$

Thus, a lower duty cycle reduces idle listening energy, but increases the delay to wake a node's neighborhood. A queue threshold, L, is specified for the protocol. This threshold could be used to control delay or limit the storage usage on a sensor. For simplicity, L is expressed in packets and all data packets are the same size[1]. When the queue holds L packets, a wakeup signal must be sent so the queue size can be reduced by transmitting packets to a receiver immediately. We refer to this as a *full wakeup* because all sensors within one hop of the sender, after detecting the signal, must wakeup their primary radio. They then listen on the primary channel until a *filter packet* is sent (on the primary channel) to indicate which neighbor's radio should remain on for reception. The other neighbors then return to sleep. To avoid costly full wakeups, a sensor estimates the rate at which it is sending data and tries to schedule a *triggered wakeup* with a receiver T seconds after its previous data transmission. Figure 1 illustrates this concept with a fixed T value. The dotted arrows represent a "causes" relationship between events. At t_0, a triggered wakeup occurs T time after the last transmission, even though the sender's queue contains less than L packets. A full wakeup begins at t_2 because the sender's queue reaches size L. At t_4, all neighbors are guaranteed to have their primary radios on, so a filter packet (shown as **F** in the figure) and L data packets (shown as **D**) are sent on the primary channel. Unlike the figure, our protocol will dynamically adjust T since the rate is not known in advance and may vary with time.

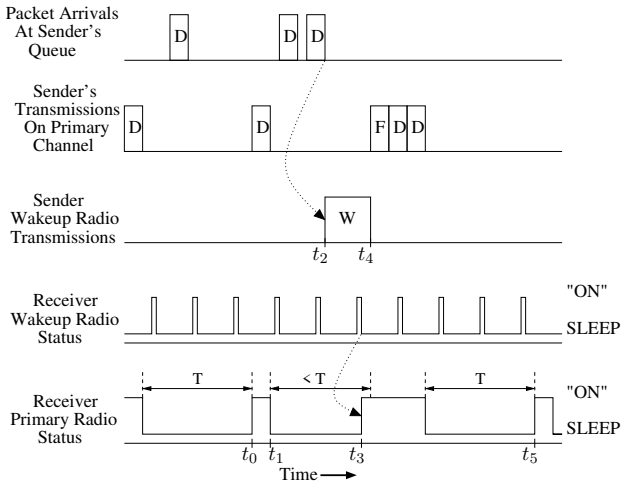


Fig. 1. Static T and $L = 2$.

If T is too small, the sender and receiver waste energy by waking up when the queue is empty. This is called an *empty triggered wakeup* and shown in Figure 1 at t_5. Such an event results in idle listening because the primary radios stay on long enough, say T_{thresh} duration, to make sure no data is available (T_{thresh} is not shown in Figure 1). Thus, they are on for T_{thresh} after doing a triggered wakeup or sending/receiving a packet. If no data is sent/received within T_{thresh} time, the primary radios return to sleep. Our goal is to find the optimal T value, T_{opt}, for a given data rate, to minimize energy consumption. We assume that both the sender and receiver sleep until a triggered or full wakeup occurs. However, the protocol can be modified for scenarios with an always awake sender (e.g., a base station).

[1] Alternatively, L could be specified in bytes.

Initially, no triggered wakeup is scheduled and a full wakeup occurs when the queue contains L packets[2]. Another timer needs to be used to make sure a packet does not remain in the queue indefinitely if the sender stops generating packets[3]. The sender will piggyback its chosen T value (in ms in our simulations) on each data packet sent. The sender and receiver will then schedule a triggered wakeup T time in the future, taking into account transmission delay. If no more data is sent or received for T_{thresh} time, the sensors will return to sleep and wakeup $T - T_{thresh}$ time later. A minimum value, T_{min}, is specified for T such that $T_{min} > T_{thresh}$. We describe how T is adjusted in Section IV-B.1.

Recall that STEM is a protocol for waking up sleeping hosts, and it may be used in conjunction with a variety of mechanisms to decide when a host may return to sleep. In this paper, we evaluate the version of STEM wherein a host may return to sleep after receiving all pending packets from the host that woke it up. This version of STEM [3] can be represented as a special case of our protocol with $T = \infty$ and $L = 1$. Unlike STEM, our protocol avoids some full wakeups by using triggered wakeups. Our protocol is different from protocols which adjust the time a radio is on once it enters the idle state (e.g., [19]). Our protocol tries to sleep as soon as possible after data communication and predict when it should next wakeup based on previous traffic patterns.

Note that our protocol is different from previous work which attempts to adjust how long nodes stay awake after a communication event before returning to sleep (i.e., adjust T_{thresh} in our protocol). This technique is popular in research to efficiently spin-down hard disks (e.g., [20] and references therein).

B. Energy Analysis of Triggered Wakeups

To find T_{opt}, we derive equations for the expected energy consumption per bit. We make some simplifying assumptions in the analysis. First, it is assumed that there is one sender transmitting to one receiver among N sensors. The remaining $N - 2$ nodes do not send or receive any data. Second, we assume that once a sensor starts sending a wakeup signal or does a triggered wakeup, only packets in the queue *at the beginning of the wakeup* are sent. Thus, exactly L packets are sent for a full wakeup and at most $L - 1$ packets are sent for a triggered wakeup. We remove this constraint in the simulations. We leave out idle energy consumed during backoff periods and due to collisions to keep the analysis tractable. These terms are relatively independent of the T value in our protocol. For ease of analysis, data packets are assumed to be transmitted instantaneously in time, though the energy cost is counted. During these instantaneous transmissions, we only count the energy for the sender and receiver since this is expected to dominate the energy of sleeping neighbors. However, the simulations model all of the energy costs, including the sleeping energy of neighbors during the transmission.

[2] L is not necessarily equal to the capacity of the queue.

[3] The simulated flows do not test this because packets never cease being generated.

TABLE I
PROTOCOL PARAMETER VALUES.

Parameter	Value
Physical Layer Header ($PLCP$)	4 bytes
Network Layer Header (IP)	20 bytes
MAC Layer Header (MAC)	32 bytes
Data Size ($DATA_{size}$)	30 bytes
Bytes in each Data Packet	$DATA_{size} + MAC + PLCP + IP$
Bytes in a Filter Packet	$33 + PLCP$
Bytes in a RTS Packet	$20 + PLCP$
Bytes in a CTS Packet	$14 + PLCP$
Bytes in a ACK Packet	$14 + PLCP$
Bitrate	40 kbps
T_{thresh}	20 ms
T_{min}	50 ms
T_{DIFS}	50 μs
T_{SIFS}	10 μs
T_{prop}	2 μs
τ_1	1 ms
τ_2	299 ms

Our parameters, shown in Table I, are based on Mica2 Motes and 802.11 (we use the power values discussed in Section I and Section III). The RTS, CTS, and ACK packet sizes and contents are unmodified from the 802.11 standard. If a smaller MAC layer byte overhead is assumed [9], the energy per packet will decrease somewhat. However, the effect of changing the MAC layer byte overhead does not change the relative performance of the protocols tested in Section V. The average power used while sleeping for the two-radio architecture, P_{sleep}, is set according to Equation 2. Let $T_{cycle} = \tau_1 + \tau_2 + T_{tran_on} + T_{tran_off}$.

When a node is sleeping, its data radio will only consume Pwr_{sleep}. The wakeup radio will consume Pwr_{sleep} for $\frac{\tau_2}{T_{cycle}}$ of the time and Pwr_{idle} for the remaining $\frac{\tau_1}{T_{cycle}}$ of the time.

$$P_{sleep} = Pwr_{sleep}\left(\frac{\tau_2}{T_{cycle}} + 1\right) + \frac{Pwr_{idle} \times \tau_1}{T_{cycle}} + \frac{Pwr_{tran_on}T_{tran_on}}{T_{cycle}} + \frac{Pwr_{tran_off}T_{tran_off}}{T_{cycle}} \quad (2)$$

We set τ_1 and τ_2 to be 1 ms and 299 ms, respectively.

These values are similar to those used in [21]. Thus, according to Equation 2, $P_{sleep} \approx 0.373$ mW.

E_{bit} is the total energy used by all nodes (in Joules) per data bit delivered. Recall that L is the queue threshold and N is the number of sensors. Let R be the packet arrival rate. We assumed the interarrival time of packets has an exponential distribution. Later, we consider time-varying rates. First, we derive p_f, the probability a full wakeup occurs. Let X be the length of time until the L-th packet arrival and Y be the number of packets that arrive during time T (i.e., $Y \sim \text{Poisson}(\lambda = RT)$).

$$\begin{aligned} p_f &= 1 - \Pr[X \geq T] \\ &= 1 - \sum_{i=0}^{L-1} \frac{(RT)^i}{i!} e^{-RT} \end{aligned} \tag{3}$$

Equation 3 comes from the Poisson distribution [22]. Let p_e be the probability of an empty triggered wakeup and $p_{\overline{f+e}}$ be the probability of a non-empty triggered wakeup. We have $p_e = e^{-RT}$ and :

$$p_{\overline{f+e}} = \sum_{i=1}^{L-1} \frac{(RT)^i}{i!} e^{-RT} \tag{4}$$

Next, let $Q_{\overline{f+e}}$ be the expected number of packets in the queue at time T for a non-empty triggered wakeup.

$$Q_{\overline{f+e}} = \frac{\sum_{i=1}^{L-1} i \frac{(RT)^i}{i!}}{\sum_{i=1}^{L-1} \frac{(RT)^i}{i!}} \tag{5}$$

We need to find T_{sleep_full}, the expected sleep time given a full wakeup occurs. Let Z be the expected time of the L-th packet arrival. The gamma distribution models the waiting time until the L-th event occurs for events that follow a Poisson distribution [22]. Thus, $T_{sleep_full} = \text{Ex}[Z|Z \leq T]$ and $Z \sim \text{Gamma}\left(\alpha = L, \beta = \frac{1}{R}\right)$. We let $f(z)$ denote the probability density function of the gamma distribution [22]:

$$f(z) = \frac{z^{\alpha-1} e^{-z/\beta}}{\Gamma(\alpha)\beta^\alpha} \tag{6}$$

The complete gamma function, $\Gamma(L)$, is defined as [22]:

$$\Gamma(L) = \int_0^\infty x^{L-1} e^{-x}\, dx \tag{7}$$

For T_{sleep_full}, we have:

$$\begin{aligned} T_{sleep_full} &= \frac{\int_0^T z f(z)\, dz}{\int_0^T f(z)\, dz} \\ &= \frac{\int_0^T z^L e^{-Rz}\, dz}{\int_0^T z^{L-1} e^{-Rz}\, dz} \end{aligned} \tag{8}$$

Now, we can express the expected energy consumed for each type of wakeup. Let $E_{pkt} = E_{MAC_RX} + E_{data_RX} + E_{MAC_TX} + E_{data_TX}$. This is the energy required to send *and* receive one packet, where E_{data_TX} and E_{data_RX} is the energy to send and receive a data packet, respectively. E_{MAC_TX} is the energy consumed by the transmitter to send an RTS and receive a CTS and ACK. Similarly, E_{MAC_RX} is the energy consumed by the receiver to get an RTS and send a CTS and ACK. Thus, we have:

$$\begin{aligned} E_{MAC_TX} = E_{DIFS} + 3E_{SIFS} + 4E_{prop} + \\ E_{RTS_TX} + E_{CTS_RX} + E_{ACK_RX} \end{aligned} \tag{9}$$

$$\begin{aligned} E_{MAC_RX} = E_{DIFS} + 3E_{SIFS} + 4E_{prop} + \\ E_{RTS_RX} + E_{CTS_TX} + E_{ACK_TX} \end{aligned} \tag{10}$$

where E_{DIFS} and E_{SIFS} are the idle energy consumed during DIFS and SIFS durations, respectively, and E_{prop} is the idle energy during propagation delays. DIFS and SIFS are interframe spacing times in 802.11 [5].

Thus, for an empty triggered wake, the energy is:

$$\begin{aligned} E_{empty} &= 2\left(E_{thresh} + E_{tran_on} + E_{tran_off}\right) + \\ &\quad NP_{sleep}T \end{aligned} \tag{11}$$

where E_{thresh} is the energy needed to listen to the channel for T_{thresh} time. Equation 11 follows from the fact that both the sender and receiver must wakeup and keep their data radios on for T_{thresh} time and all N nodes have slept T time since the last wakeup.

For non-empty triggered wakeups:

$$\begin{aligned} E_{triggered} &= 2E_{tran_on} + Q_{\overline{f+e}} E_{pkt} + 2E_{thresh} + \\ &\quad NP_{sleep}T + 2E_{tran_off} \end{aligned} \tag{12}$$

Equation 12 is similar to Equation 11 except that extra energy is consumed to send $Q_{\overline{f+e}}$ packets.

For a full wakeup, the equation is:

$$\begin{aligned} E_{full} = E_{wake_TX} + (N-1)E_{wake_RX} + \\ NE_{tran_on} + NE_{DIFS} + E_{filter_TX} + 2NE_{prop} + \\ (N-1)E_{filter_RX} + LE_{pkt} + 2E_{thresh} + \\ NE_{tran_off} + NP_{sleep}T_{sleep_full} \end{aligned} \tag{13}$$

Equation 13 states that the sender must transmit a wakeup signal and $(N-1)$ receivers must listen to it. Then, all N nodes must turn their data radio on and wait until the filter packet is received. All the other nodes can return their data radios to a sleep state, but the sender and receiver remain on to exchange L packets. Finally, each node has slept T_{sleep_full} time since the last wakeup, unlike in Equations 11 and 12 where the nodes slept T time since the last wakeup.

Thus, the expected energy consumed per bit is:

$$E_{bit} = \frac{p_f E_{full} + p_{\overline{f+e}} E_{triggered} + p_e E_{empty}}{Data_{size} \times 8 \times (p_f L + p_{\overline{f+e}} Q_{\overline{f+e}})} \tag{14}$$

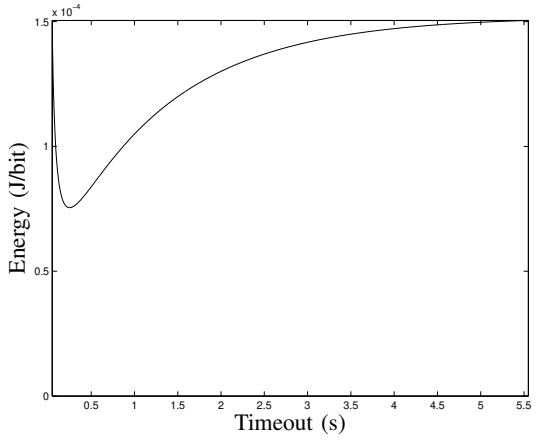
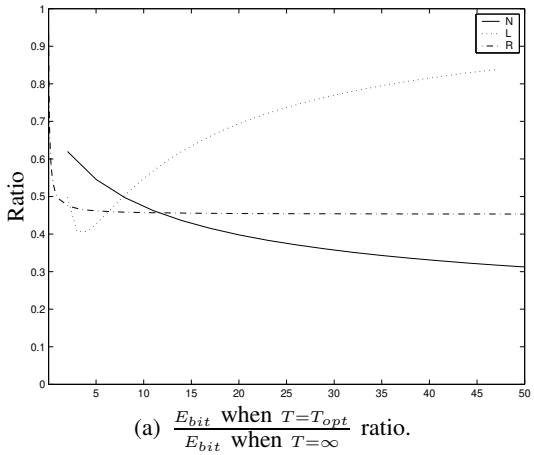


Fig. 2. Energy versus timeout.

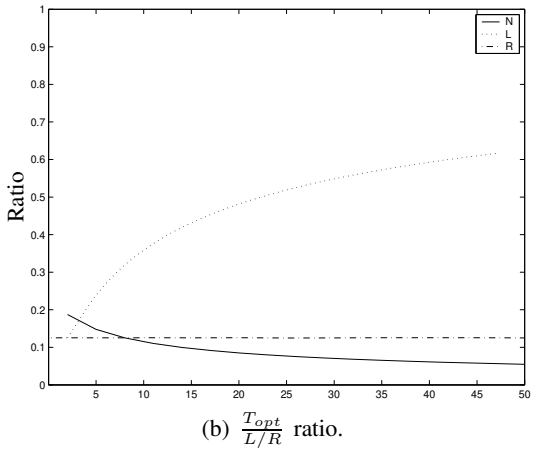
Using Equation 14, Figure 2 shows E_{bit} as a function of T for $R = 1.0$, $L = 2$, and $N = 8$. Recall that R, L, and N are the sending rate, queue threshold, and number of nodes, respectively. Note that E_{bit} is minimized at $T = T_{opt} \approx 0.251$ s. Clearly, choosing $T = T_{opt}$ should minimize energy consumption.

Figure 3(a) shows energy savings we can expect when T_{opt} is used compared to setting $T = \infty$ (i.e., a full wakeup occurs every time the queue fills up). The graph shows how the energy savings changes with R, L, and N, based on our analysis. The horizontal axis is the value of the changing parameter (i.e., R, L, or N) while the other two parameters stay fixed. The fixed values are: $R = 1.0$, $L = 2$, $N = 8$. For example, when R and L stay fixed and $N = 40$, T_{opt} gives about a 67% improvement (i.e., it uses only 33% of the energy per bit that $T = \infty$ uses). As another example, when R and N stay fixed and $L = 40$, there is only about a 20% improvement for reasons discussed below.

From Figure 3(a), we can observe how each of the parameters affects the energy savings. The energy savings is almost constant as R changes. At very low rates, the ratio asymptotically approaches one because the sleep time between packets is so large that energy spend sleeping between packets dominates the difference in energy to do a full wakeup versus a triggered wakeup. Otherwise, the ratio is constant as R changes primarily because the rate functions as a scaling factor that does not change the relative energy difference. As N increases, T_{opt} results in more energy savings because the entire neighborhood will only awake with probability p_f when packets are sent. When $T = \infty$, the neighborhood will awake every time packets are sent, resulting in increased relative energy usage when N is large. Generally, increasing L results in less energy savings because the full wakeup costs are amortized over



(a) $\frac{E_{bit} \text{ when } T=T_{opt}}{E_{bit} \text{ when } T=\infty}$ ratio.



(b) $\frac{T_{opt}}{L/R}$ ratio.

Fig. 3. Effects of N, L, and R

more packets. Note, however, this trend is reversed when L is between about 2 and 5. This is attributed to the fact that, despite the increased amortization of full wakeup costs, there is a large variance in when the L-th packet arrival occurs when L is small. Thus, there are more full wakeups and empty triggered wakeups (i.e., $p_{\overline{f+e}}$, from Equation 4, will be small). For example, when $L = 2$, $p_{\overline{f+e}}$ is about 0.2, but when L increases to 5, $p_{\overline{f+e}}$ increases to about 0.7. Therefore, when L is small, the $p_{\overline{f+e}}$ factor dominates the relative energy savings. As L grows larger, the amortization factor begins to dominate.

1) Adjusting T: The sender estimates its sending rate via a weighted average of the interarrival time of packets. The estimate, $R_{est} = \frac{1}{t_{est}}$, is computed by the equation:

$$t_{est} = \rho t_{est} + (1 - \rho) t_{diff} \tag{15}$$

where t_{diff} is the most recent sample of interarrival time.

Figure 3(b) shows how the ratio $\frac{T_{opt}}{L/R}$ (where L/R is the expected time for the queue to reach L packets) changes with R, L, and N, based on our analysis. The horizontal axis is the value of the changing parameter (i.e., R, L, or N) while the other two parameters stay fixed. The fixed values are: $R = 1.0$, $L = 2$, $N = 8$.

From Figure 3(b), when L and N are fixed, we observe that for some constant γ,

$$T_{opt} = \gamma \frac{L}{R} \tag{16}$$

where γ is independent of R. Figure 3(b) also shows that γ is not constant if L and N are dynamic[4]. In our evaluation, we assume L and N are known in advance. Thus, γ is calculated as a function of L and N.

2) Communicating with Multiple Neighbors: The protocol, as described, is designed for a sending node transmitting to one neighbor and a receiving node getting packets from one neighbor. However, to perform in a more general setting, the protocol must be modified to handle communication with multiple neighbors.

The first situation to consider is when a receiving node is getting data from multiple senders. This type of communication is prevalent in sensor network with routing protocols that form tree structures rooted at data sinks [23]. Our protocol is not affected at the sender, because each sender has only one flow. However, the receiver must adopt multiple schedules and respond to wakeups from multiple senders. This does not change our protocol. If a receiver has scheduled overlapping triggered wakeups, then the senders have to compete according to the MAC protocol. In the future, we plan to explore how the triggered wakeups can be scheduled more efficiently to reduce overlapped triggered wakeups and reduce channel contention.

The second situation is when a sending node is transmitting data to multiple receivers. This scenario is more complicated than the first because the sender's queue is essentially shared among flows intended for different destinations. The filter packets are capable of advertising up to M addresses of receivers that should remain awake. Choosing M represents a tradeoff between the byte size of the filter packet and number of distinct destinations that can be kept awake after the wakeup procedure. In our implementation, we set $M = 4$. Therefore, if the sender has packets intended for different destinations when it does a full wakeup, it will use the filter packet to tell all of the destinations to remain on to receive data packets. With multiple receivers, we still update each T according to Equation 16, but now R refers to the cumulative rate of all flows. Thus, each destination may receive the same T value, but at different times. Also, at different times, T may be different since the rate estimate changes. Note that this requires receivers to do triggered wakeups more frequently than if all packets in the queue were destined for only one destination. This is necessary because the queue is shared and therefore the frequency of full wakeups is based on cumulative rate rather than the rate to each destination independently.

The intuition behind this scheme is as follows. Assume that there are n flows at a sender. Let L_{v_i} refer to the *virtual* queue threshold for flow i, and R_i denote the rate of the flow. Thus, $L_{v_i} = L\left(R_i / \sum_{j=1}^{n} R_j\right)$, where $R = \sum_{j=1}^{n} R_j$ is the cumulative rate of all flows. Therefore, $T_i = \gamma_i \frac{L_{v_i}}{R_i} = \gamma_i \frac{L}{R}$. However, γ_i cannot be calculated online since it changes with respect to L_{v_i}. Also, for $L = 2$, at least $n-1$ of the L_{v_i} values are less than one. This cannot be calculated because our analysis requires $L \geq 2$. Therefore, for γ_i, we just use the γ value for L. This does have a small effect on the protocol since the chosen value of T_i is larger than if γ_i had been used. When the number of receivers is small, the average number of full wakeups per receiver increases as the number of receivers grows instead of remaining constant.

In general, a node can be both a sender and receiver. For example, a node may do a triggered wakeup to receive data at the same time that it does a full wakeup to send data. If a node is awake as a sender and receiver at the same time, the MAC protocol allows a node multiplex its sending and receiving during the wakeup.

V. Experimental Results

We implemented our protocol from Section IV-B in *ns-2* [24] by modifying the 802.11 MAC and physical layer code in *ns-2*. Eight sensor nodes were placed within range of each other and a random sender and receiver were chosen to begin communicating with Poisson traffic at rate R. The remaining six nodes did not send or receive any data. We tested R values of 0.2, 0.5, 1.0, 1.5, and 2.0 packets per second. The resulting T_{opt} values were always greater than T_{min}. Unlike the analysis, packets were sent if they arrived after a wakeup occurred. We set $L = 2$ for all simulations in this section to demonstrate the simplest case of our protocol: rather than sending a packet immediately, we try to delay it until a triggered wakeup occurs. Each data point is averaged over 50 runs and error bars show standard deviation. The simulation time was such that the expected number of packets sent was the same regardless of rate (unless otherwise noted, this value was set to 200). Thus, if the desired expected number of packets is P, the test for rate R_i would be run for $\frac{P}{R_i}$ time. The values in Table I and Sections I and III were used when applicable.

A. Effects of ρ

First, we investigate how ρ, from Equation 15, affects energy consumption. Recall that ρ is the weight given to

[4]Because we require L to be constant for Equation 16 to hold, L could be included in the γ term and we would have $T_{opt} = \gamma \frac{1}{R}$. We separate L from γ for ease of explanation.

the previous, cumulative interarrival estimate when a new arrival occurs to obtain a new interarrival time estimate. Intuitively, if ρ is large, the rate estimate is slow to adjust to rate variations, but more robust to occasional outliers. If ρ is small, T will adjust quickly to rate changes, but occasionally adjust too much in response to outliers.

Our simulations show that the energy consumption is fairly constant regardless of the ρ value for relatively low ($R = 0.2$) and relatively high ($R = 2.0$) rates. The only exception is when ρ is very close to 1.0. In this case, the rate estimate is based almost completely on the interarrival time of the first two packets and does not adjust to subsequent packet interarrival times. In this situation, we see a slightly higher average energy consumption and much larger variance. Based on our results, we use $\rho = 0.9$ for subsequent tests, unless stated otherwise.

B. Comparison of Different Protocols

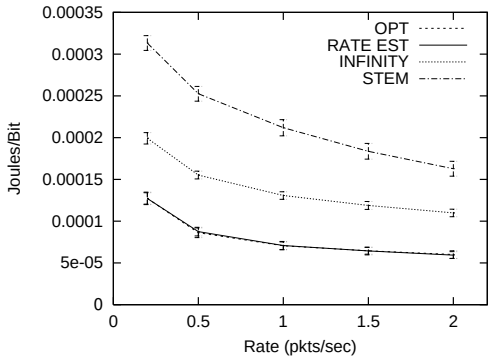
For comparison, we evaluated several protocols. *Rate estimation* (RATE EST) is our proposed protocol; γ is analytically calculated to be 0.1253. For the *static optimal* (OPT), T is statically set to be T_{opt}, calculated analytically using the given rate. Thus, RATE EST estimates rate dynamically, whereas OPT "magically" knows the rate. *STEM* is the version with a busy tone [3] evaluated in this paper. Finally, $\mathbf{T} = \infty$ (INFINITY) only sends packets via full wakeups (i.e., no triggered wakeups). Essentially, this is a STEM variant that buffers up to L packets before doing the wakeup procedure.

1) Energy Comparison: Figure 4(a) plots the energy consumption of the protocols with rate on the horizontal axis. We see that regardless of rate, our protocol and the static optimal result in comparable energy consumption (the two curves almost overlap), which is significantly lower than the other protocols.

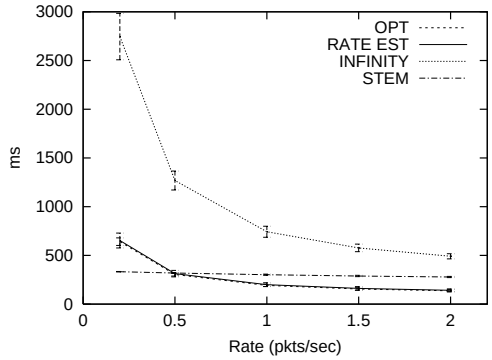
Figure 4(a) shows that rate estimation represents about 65% improvement over STEM regardless of rate. When compared to $T = \infty$, the rate estimation shows about 45% improvement. This shows the need to schedule triggered wakeups even if the full wakeup cost is amortized over multiple packets. In general, all the protocols improve as the rate increases. This is due to the decrease in sleep time between packets at higher rates.

Figure 4(a) also compares favorably to the analytical expectation shown in Figure 2. Both show the energy per bit to be about 70 μJ when $R = 1.0$. The 45% improvement over $T = \infty$ is also close to what is predicted in Figure 3(a).

2) Latency Comparison: Our protocol's performance is even better when the average packet latency is considered in Figure 4(b). Again, the rate estimation and



(a) Energy consumption of protocols.



(b) Latency of protocols.

Fig. 4. Comparison of protocols.

the static optimal performance overlap. Rate estimation shows more than 70% improvment compared to $T = \infty$. Thus, we can see our protocol performs much better than setting $T = \infty$ for both energy consumption *and* average packet latency. As expected, STEM's latency is nearly constant at each rate. This latency corresponds to the time to do a full wakeup and shows virtually no variance. At higher rates, our protocol performs better than STEM since T_{opt} is less than the time required to do a full wakeup. At low rates, T_{opt} is larger than the time to send a wakeup signal, which is why STEM has a lower latency at low rates. Note that $T = \infty$ will asymptotically approach STEM's latency, but never do better. On the other hand, because our protocol can avoid full wakeups, the latency will continue to be reduced until T reaches T_{min}. Thus, the theoretical bound, as R increases, of the ratio of the latency of rate estimation to $T = \infty$ and STEM is $\frac{T_{min}}{T_{wake_TX}}$. which is about 0.165 with our experimental setup (i.e., an 83.5% improvement).

We can easily check to see that the experimental results for the latency at $T = \infty$ are close to their expected value. In this case, we expect the average per packet latency for $L = 2$ to be:

$$\frac{(1/R + T_{wake_TX}) + T_{wake_TX}}{2} \tag{17}$$

TABLE II
MEASURED LATENCY FOR $T = \infty$ VERSUS ANALYSIS (IN MS).

Rate	Analytical Expectation	Measured Latency	
		Avg.	Std. Dev.
0.2	2803.7	2746	239
0.5	1303.7	1269	97
1.0	803.7	743	57
1.5	637	577	39
2.0	553.7	491	26

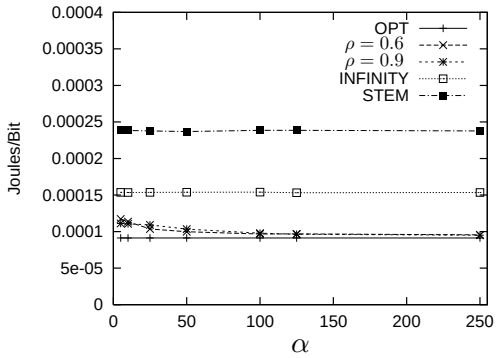
because the first packet waits, on average, the expected interarrival time between packets ($1/R$) plus the time to send a wakeup signal (T_{wake_TX}), whereas the second packet only has to wait long enough to send a wakeup signal (T_{wake_TX}). We then divide the total latency by 2 since $L = 2$ to obtain Equation 17. Table II shows the analytical values for latency with $T = \infty$ using Equation 17 match well with the observed experimental values. At a higher rate, Equation 17 is not as accurate since it does not account for packets which arrive during the wakeup and are sent without incurring the wakeup delay. At a higher rate, this happens more frequently, thereby reducing the average packet latency.

C. Effects of Traffic with a Time-Variant Rate

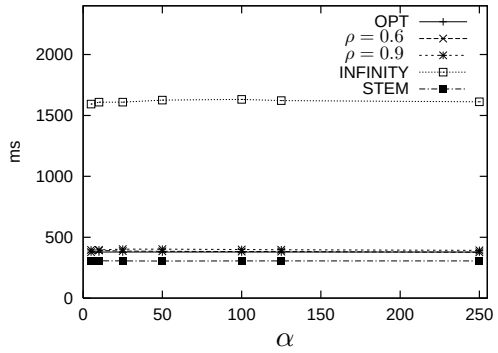
A strength of our protocol is that it to adjusts to traffic on-demand as the sending rate changes. To test this dynamic adaptation, R was switched from 0.2 to 2.0 packets per second periodically. We let α be the frequency with which the rate changes. Specifically, α is the expected number of packets generated at the current rate before switching to the other rate. For example, if $\alpha = 10$, packets are generated at $R = 0.2$ for $\frac{10}{0.2} = 50$ seconds, then at $R = 2.0$ for $\frac{10}{2.0} = 5$ seconds. This behavior was repeated for several cycles until the total expected number of packets sent per rate was 500. For the static optimal, we ran two separate scenarios at the different rates and averaged the results. This represents the best energy consumption possible if the protocol adjusted to rate changes immediately.

1) Energy Comparison: Figure 5(a) plots energy consumption with α on the horizontal axis. As expected, rate estimation does better when the sender spends a long time at a fixed rate before switching rates. When rate change is infrequent, rate estimation uses only about 5% more energy than the static optimal. When $\rho = 0.6$, rate estimation converges more quickly toward the static optimal since it is more responsive to rate change. STEM and $T = \infty$ stay relatively constant and use significantly more energy than our protocol.

Even in the worse case, where the rate is changing very frequently, our protocol only uses about 22-29% more energy than the static optimal (depending on the ρ value). By comparison, $T = \infty$ always uses about 68% more energy than the static optimal and STEM always uses over 2.5 times as much energy as the static optimal.



(a) Energy consumption for time-variant traffic.



(b) Latency for time-variant traffic.

Fig. 5. Traffic with time-variant rates.

2) Latency Comparison: The trends for the latency with time-variant traffic, shown in Figure 5(b), are similar to those in Figure 4(b). STEM stays near constant at the time to do a wakeup. The rate estimation protocol remains almost constant with the static optimal regardless of α. The latency of the static optimal protocol remains constant since α does not affect the static optimal in our experiments. The latency of the rate estimation and static optimal protocols is slightly above STEM because if we were to average the latencies of $R = 0.2$ and $R = 2.0$ from Figure 4(b), they are slightly higher than STEM for reasons discussed in Section V-B.2.

Again, we can verify that the experimental results for $T = \infty$ match the expected average per packet latency. From Equation 17, we can average the expected latency at $R = 0.2$ (2803.7 ms) and $R = 2.0$ (553.7 ms) to get 1678.7 ms. This is close the experimental average values, which are observed to be between 1593 and 1632 ms.

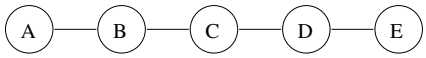


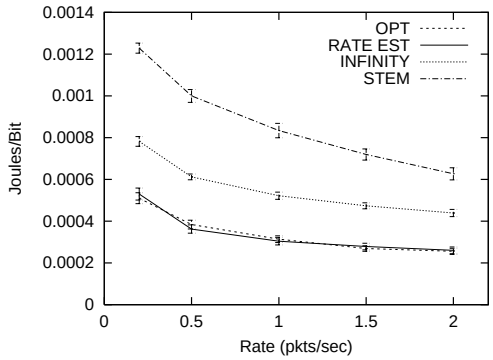
Fig. 6. Topology for testing multiple hop performance.

D. Multiple Hop Performance

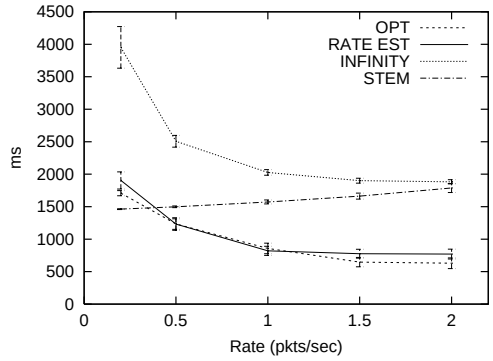
After examining the performance of our protocol in a single-hop, single-flow scenario, we wanted to see how it would perform in more complex scenarios. The first we tested was the multiple hop scenario in Figure 6. Traffic was sent from **A** to **E**, the rate was varied, and the simulation time was varied inversely with the rate such that the expected number of packets generated during a simulation run (set to be 200) remained constant.

Each node in the topology had seven neighbors, to allow comparison with Section V-B results. Note that neighbors not on the data path are not shown in Figure 6. Thus, the two end nodes in the topology have six neighbors and the intermediate nodes have five neighbors. The neighbors not on the data path were placed such that they overhear exactly one node on the data path (i.e., nodes **A**, **B**, **C**, **D**, **E** do not share any neighbors other than those shown in Figure 6). For example, **A** has six neighbors not shown in Figure 6 which do not send or receive data and receive **A**'s wakeup signal, but are out-of-range of **B**'s wakeup signal. In addition to these six neighbors, **A** has **B** as a neighbor, as shown by the link in Figure 6, giving **A** a total of seven neighbors. In the discussion, the term *downstream neighbor* refers to a node that is closer to the destination than the current node (e.g., **C** is a downstream neighbor of **B**). The term *upstream neighbor* refers to a node that is closer to the source than the current node (e.g., **C** is an upstream neighbor of **D**).

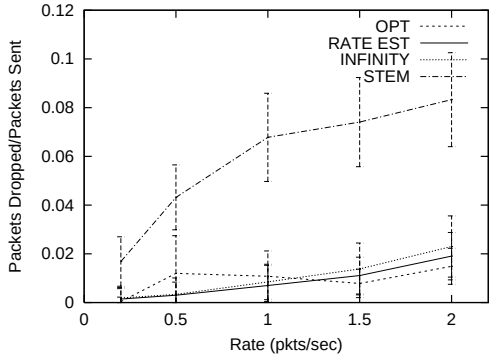
Figure 7(a) shows that energy consumption follows a similar trend to that of Figure 4(a). It is reasonable to expect that the results from Figure 4(a) would be scaled by a factor of about four since there are now four links per packet delivery rather than one. However, there are some effects due to the dependence of packet arrivals on a link. For example, in STEM, if **A** is able to send two packets during a wakeup instead of just one, then each node along the path will also send two packets during the wakeup of their downstream neighbor instead of one. Thus, STEM and $T = \infty$ only increase energy consumption by a factor of about 3.9. This factor is obtained by taking the ratio of the results in Figure 7(a) to those in Figure 4(a). However, rate estimation and the static optimal are negatively affected by the link dependence. If **A** causes a full wakeup, then the full wakeup will cascade down the entire path instead of being independent at each hop. This effect



(a) Energy consumption for multiple hop scenario.



(b) Latency for multiple hop scenario.



(c) Packet drops for multiple hop scenario.

Fig. 7. Multiple hop scenario.

does not cause major degradation, however, with the energy consumption increased by a factor of 4 to 4.5 over the single hop case depending on the rate.

We note an issue with our protocol when tested on the multiple hop topology. When multiple packets are sent to the next hop, they come in a burst. Thus, the receiver's rate estimation calculates a very short interarrival time. This problem can be addressed by ignoring interarrival times that are smaller than a fixed threshold. This modification is not implemented in the simulations.

Figure 7(b) shows the latency of the protocols. This shows a similar trend to Figure 4(b). However, like the energy consumption results, these results do not always scale by a factor of four. At a low rate, STEM

does show a 4.4 times increase (when compared to the single hop case) because at each hop, the packet must wait for the wakeup to occur and a small amount of contention is now induced (e.g., when **A** tries to send to **B** and **C** tries to send to **D**, one must defer their transmission to avoid a collision). However, the protocols which use $L = 2$ show less of an increase because the intermediate hops can immediately send packets to their downstream neighbor when they receive L packets from their upstream neighbor. For example, at $R = 0.2$ for $T = \infty$, the latency increases by a factor of only 1.44. In the single hop case, the first packet to arrive in the empty queue for $T = \infty$ has to wait, on average, 5 seconds plus the wakeup time. However, now both packets only have to wait about a wakeup time at each intermediate node. Similarly, rate estimation and the static optimal have their latency increase by a factor of only about 2.9 due to the decreased latency on full wakeups.

If we use Equation 17 to verify the latency for $T = \infty$ at $R = 0.2$, the first hop still has an expected latency of 2803.7 ms (as shown in Section V-B.2). However, the next three links have an average latency of only 302.7 ms. Thus, the overall expected latency is $(2803.7 + 3 \times 302.7) = 3711.8$ ms, which is within the deviation shown in Figure 7(b).

At a higher rate, the burstiness at downstream links does not help the $L = 2$ protocols as much because the interarrival time of packets is smaller. Also, at a higher rate, all the protocols show an increase in latency, relative to the single hop case, significantly greater than at a low rate. This is due to the increased contention on the links as the rate increases. The effects of increased contention are also seen in the data packet drop rate, shown in Figure 7(c). As the rate increases, more drops occur due to the medium being increasingly busy. Packet drops generally occur because a sender miscalculates when the receiver will be up and does not receive a response to its RTS. In *ns-2*, a data packet is dropped after an RTS for the packet has been retransmitted seven times without receiving a CTS. For the protocols with triggered wakeups, this can occur when a packet is lost and hence the nodes believe they should wakeup at different times. For example, a sender could transmit a data packet telling the receiver to wakeup T_{new} seconds after reception. However, if the receiver does not receive the packet due to a collision and the sender is not able to retransmit the packet before the pair's scheduled sleep times, the receiver may believe it is supposed to wakeup T_{old} time after it estimates the sender has turned off.

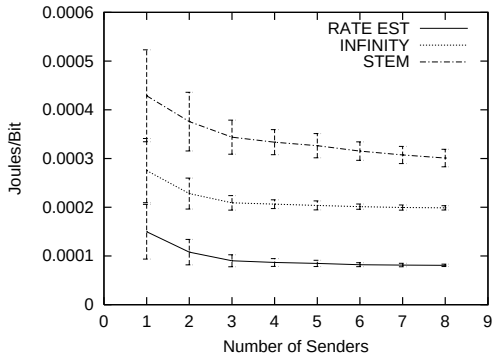
Packet drops can also occur during a full wakeup due to excessive retransmissions. For example, **B** could begin transmitting a wakeup signal for **C** and shortly thereafter, **A** begins transmitting a wakeup signal for **B**. Thus, **B** may not receive **A**'s wakeup signal because it never listened on the wakeup channel during that time. Thus, **A** could believe **B** to be awake after transmitting the wakeup signal even though **B** has already finished its data transmission to **C** and returned to sleep without ever hearing **A**'s wakeup signal. STEM is most affected by this since it does about twice as many wakeups as $T = \infty$ for $L = 2$. This is because STEM does a wakeup for every packet whereas $T = \infty$ does a wakeup for every other packet. In the rate estimation protocol, if a sender is not able to successfully transmit an RTS or data packet within the number of retransmissions specified in 802.11, T will be set to ∞. This forces a full wakeup the next time L packets arrive for that destination. This assures that two nodes will not become persistently unsynchronized in their triggered wakeups.

E. Multiple Flow Performance

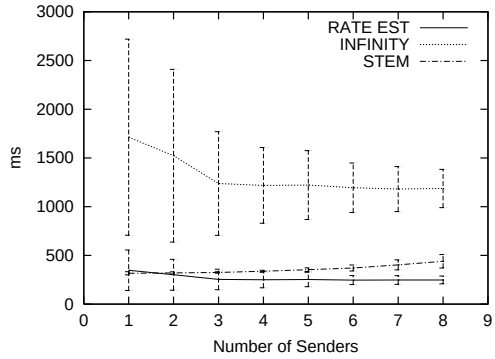
In this section, we look at the effects of having multiple flows on the protocols. Section IV-B.2 describes how the rate estimation protocol works in multiple flow settings. We do not consider the static optimal in these scenarios since it only represents the static optimal in the single flow case. In these scenarios, we doubled the number of nodes to be 16 (compared to tests in Section V-B that had $N = 8$). This is because we needed to increase the number of flows beyond eight to show interesting behavior. This is also the reason the energy values are higher in these scenarios when there is only one flow than in previous sections. For each scenario, connections between nodes were chosen to have a rate probabilistically. More specifically, each link has a rate of $R = 0.2$ or $R = 1.0$ with a probability of 0.5. Thus, the average rate per link is $R = 0.6$, or one packet every 1.67 seconds (even though this specific rate is never chosen for a link). This was done to show how the rate estimation performs when nodes are sending or receiving at multiple rates to or from different neighbors. Each simulation run lasts 500 seconds. Because each scenario could have links with two different rates, there was no way to normalize the simulation time such that the expected number of packets generated per link remained constant and each link contended for medium access throughout the entire simulation run.

For testing multiple sender flows, we used a topology with one node specified as a receiver and varied the number of nodes sending data to it. The overall number of nodes in the scenario remains constant at 16 and all nodes are within range of each other.

Figure 8(a) shows that the energy consumption of each protocol remains relatively constant as the number



(a) Energy consumption for multiple sender scenario.



(b) Latency for multiple sender scenario.

Fig. 8. Multiple sender scenario.

of senders increases. This implies that the extra energy incurred by an additional flow is compensated by the extra packets that are delivered. The rate estimation protocol represents about a 50% improvement over $T = \infty$ and a 65% improvement over STEM. This is consistent with the results discussed in Section V-B.1. The rate estimation protocol does slightly better in the multiple sender scenario because $N = 16$ instead of 8. As explained in reference to Figure 3(a), the larger N makes full wakeups more expensive.

Figure 8(b) shows the latency for the protocols in the multiple sender scenario. Latency is relatively constant for the rate estimation protocol and $T = \infty$ regardless of the number of senders. The latency for these protocols is the same as the interpolated latency for $R = 0.6$ in Figure 4(b). STEM shows a slight increase as the number of senders grows due to the increased contention caused by more wakeups occurring for the receiver. All protocols show a larger variance when the number of senders is low. This is due to the large difference in latency according to which rates are probabilistically chosen on the links. For example, when there is only one sender, about half the scenarios have $R = 0.2$ for the connection, while the other half have $R = 1.0$, which results in drastically different average latencies.

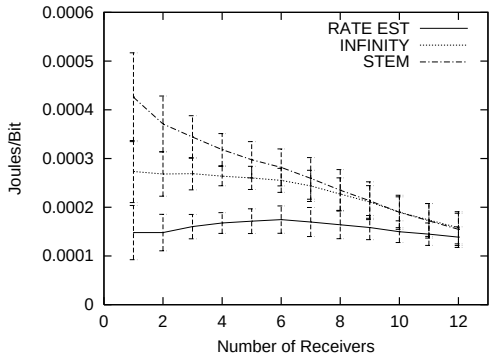
Few packet drops occurred in this scenario. The packet loss was less than 0.2% regardless of the protocol or number of senders. Because there is only one receiver, even if the sender begins sending RTS packets when it incorrectly predicts the receiver will be on, they may still get a response if the receiver is up for a different sender.

To test flows with multiple receivers, we used a topology with one node specified as a sender and varied the number of nodes receiving data from it. The overall number of nodes in the scenario remains constant at 16 and all nodes are within range of each other.

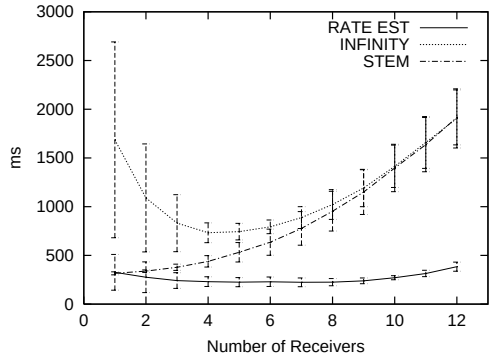
The energy consumption of the protocols is shown in Figure 9(a). STEM's energy begins to drop when the number of receivers is increased because the overall rate is increasing and hence the sleep time per packet is decreased. For $T = \infty$ and the rate estimation protocol, the energy starts to increase with the number of receivers because, when multiple destinations are awakened, all but one of the receivers must idly listen to the transmission. Also, as discussed previously, the rate estimation protocol results in more full wakeups per receiver as the number of receivers increases.

When the number of receivers reaches about 6 or 7, a couple of interesting events occur. First, the overall rate becomes high enough that the expected interarrival time of packets is less than the time to do a full wakeup. Thus, the service rate is less than the arrival rate for STEM and its queue begins building up. As the queue length increases, it actually decreases the energy consumption because the probability that multiple packets can be sent from the queue on a wakeup is increased. When the number of receivers is about 9 or 10, the overall rate becomes high enough that the expected interarrival time of packets is less than half the time to perform a full wakeup. Thus, the queue begins building up for $T = \infty$ and its energy consumption converges with STEM. When this situation occurs, the queue will eventually start dropping packets due to finite storage space. Also at about 6 or 7 receivers, the overall rate is high enough that T_{opt} drops below T_{min}. Thus, the rate estimation protocol begins to gradually decrease at this point as the time between triggered wakeups cannot decrease further. This decrease is because periodic wakes cannot occur more frequently, yet the queue fills up faster. Thus, more packets in the queue increases the probability multiple packets can be sent when a wakeup occurs.

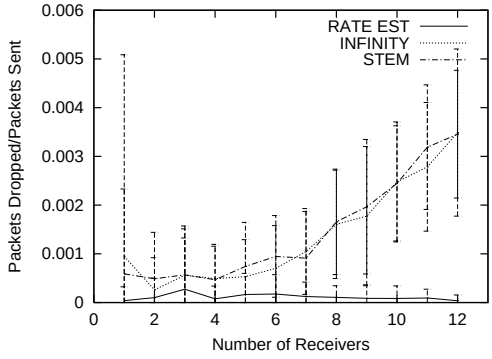
We learn more about the protocols' behavior by looking at the latency in Figure 9(b). When the number of receivers is small, as the number of receivers increases, the latency curves show a decrease for our protocol and $T = \infty$. However, STEM and $T = \infty$ begin showing increased latency when the number of receivers gets



(a) Energy consumption for multiple receiver scenario.



(b) Latency for multiple receiver scenario.



(c) Packet drops for multiple receiver scenario.

Fig. 9. Multiple receiver scenario.

larger and packets must spend more time in the queue. Again, STEM and $T = \infty$ converge in this metric when the number of receivers is about 10, which is the point that both protocols have a service rate smaller than the arrival rate. The rate estimation protocol is able to do better in terms of latency because it can service packets faster than the other protocols. Specifically, since $T = T_{min}$ at a high rate and T_{min} is much less than the time it takes to do a full wakeup (a ratio of about $\frac{1}{6}$ in our tests), packets wait less time in the queue.

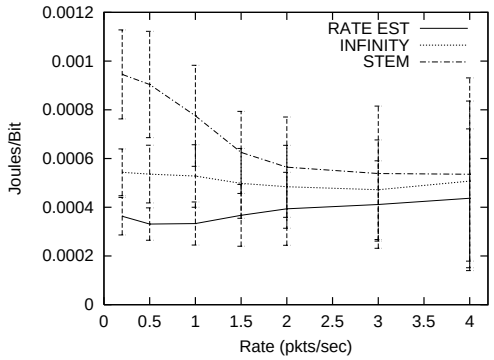
Because there is only one sender and no contention for transmissions, the amount of packet drops remains low (i.e., less than 0.4% on average). The packet drops for STEM and $T = \infty$ are predominantly from packets in the queue when the simulation ends (packets queued at the end of the simulation are counted as dropped). Thus, this gives a rough estimate of how the sender's queue size increases with more receivers. Because the rate estimation protocol has frequent triggered wakeups, the queue does not build up in our tests. Therefore, virtually no packets are in the queue at the end of the simulation. Figure 9(c) shows the packet drop percentage for the multiple receiver scenario.

We note a source of MAC layer retransmissions with the rate estimation protocol. In *ns-2*, if the length of time between when a receiver sends a CTS and when the receiver gets the data packet is too long, the data packet is dropped by the receiver. Occasionally, the sender, $\mathbf{S}$, initiates a full wakeup for $\mathbf{R_1}$. Just before $\mathbf{S}$ sends its filter packet for $\mathbf{R_1}$, $\mathbf{S}$ does a periodic wake with $\mathbf{R_2}$ and exchanges an RTS and CTS. However, before the data packet is sent to $\mathbf{R_2}$, the filter packet is sent for $\mathbf{R_1}$. This delays the data packet longer than expected and when it is finally sent to $\mathbf{R_2}$, it is dropped.

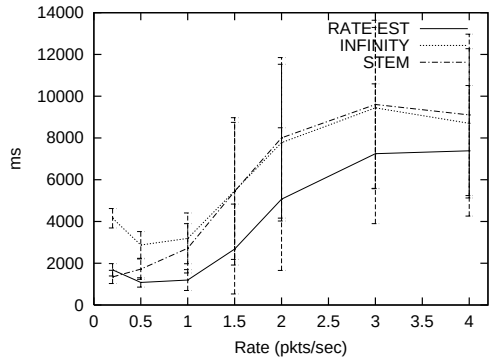
Giving the filter packet priority over the data packet is a design decision in our protocol. The intuition is as follows. If the filter packet gets delayed long enough, the intended receiver could return to sleep while the sender is transmitting a data packet to another destination. The energy cost of doing another full wakeup in this situation is greater than just retransmitting the data packet. As an alternative, when the sender begins a full wakeup, it could block the transmission of all data packets until the filter packet is sent. Another possibility is the sender could delay sending a packet for a triggered wakeup if the packet transmission would not finish before the filter packet will be sent. These two modifications are not implemented in the simulations.

F. Random Network Topologies

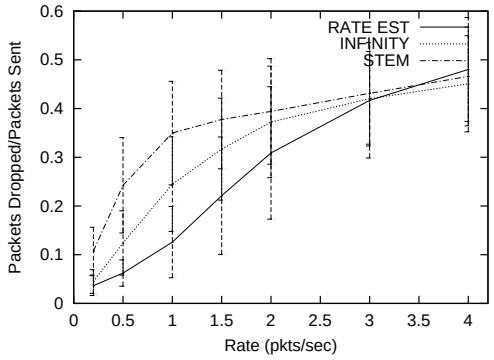
In this section, we tested the protocols in random topologies. Thus, the protocols had to operate in a multiple hop *and* multiple flow environment with potentially a large amount of channel contention. For these experiments, we randomly placed 50 nodes in a 1000 m $\times$1000 m area. Each scenario had 10 flows, where the source and destination pairs are chosen at random. Each data point is averaged over the results for 50 topologies where every node could reach every other node. The shortest path routing table for each node is calculated offline to avoid the effects of routing overhead. Also offline, the size of each node's neighborhood, N, is computed and the appropriate γ value is input to each node. The simulation times for each scenario are inversely proportional to the sending rate. Thus, for each run, the expected number



(a) Energy consumption for random network topologies.



(b) Latency for random network topologies.



(c) Packet drops for random network topologies.

Fig. 10. Random network topologies (absolute values).

of packets during the simulation is identical regardless of sending rate (set to be 200 packets).

To test performance when contention is high, the maximum per-flow sending rate for these experiments is 4 packets per second. For the data packet, RTS, CTS, and ACK, this is a per-flow rate of about 4.7 kbps. For the 40 kbps channel that we use, this is about 12% of the channel bitrate per flow. As stated previously, each scenario consists of 10 multi-hop flows accessing the channel. In addition, there is extra overhead for filter packets, backoff slots, etc.

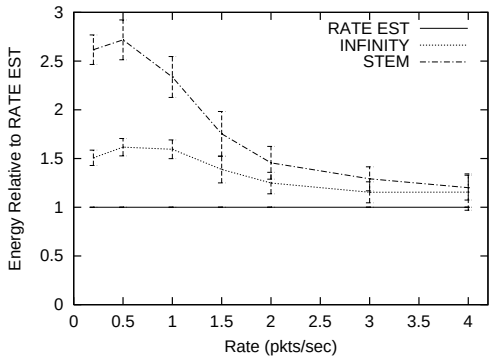
Figure 10 shows the energy, latency, and packet drop percentage values that are obtained for the protocols in this setting. As Figure 10 shows, there is a rather larger variance in the data because of the randomness in topologies and connection patterns for each scenario. To gain a better understanding of how the protocols perform relative to each other, without the inherent randomness of the topologies, Figure 11 shows each protocol's performance normalized to the performance of rate estimation for each individual scenario. We can see that the rate estimation protocol always does better in terms of energy, especially at relatively low sending rates. At a relatively high sending rate, there is lots of contention on the medium and the protocols' performances begin to converge.

VI. Conclusions and Future Work

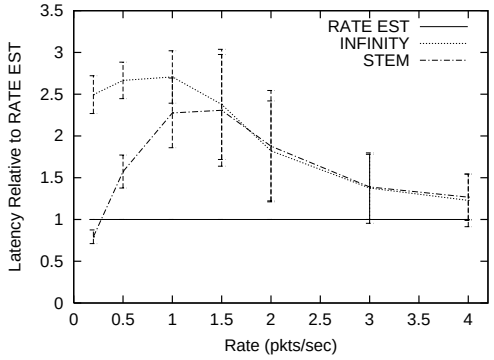
We have analyzed a protocol for sensor networks that improves energy efficiency by buffering packets, thereby amortizing the energy cost of communication over multiple packets. Because storage space may be a scarce resource in sensors, we propose adding a second, low-power radio to allow senders to force receivers to wakeup when a specified number of packets are buffered. Our analysis reveals an optimal timeout value for periodically waking up to send and receive packets that minimizes energy consumption. Our protocol uses rate estimation to achieve results close to the static optimal. In addition, we show significant energy savings over other, similar protocols. The protocol seems to behave well when multiple hop and multiple flow scenarios are introduced as well. In such situations, it almost always outperforms other protocols in energy and latency. For future work, we outline a few directions that could be pursued.

First, we would like to adapt our protocol into a more realistic environment. For example, every sensor node could report periodically at a low rate in steady state. Then, when an event occurs, the affected sensors begin transmitting at a much higher rate for the duration of the event. Also, we plan to investigate how to efficiently stagger multiple schedules so that interfering nodes minimize the time they are both on.

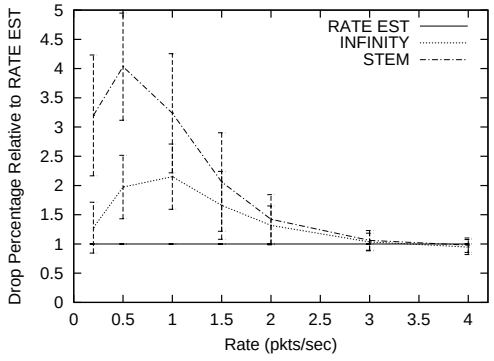
Currently, we assume all nodes must share the same wakeup channel. However, interesting problems arise if we consider the case in which a few bits can be encoded in the wakeup signal. Then, nodes only wakeup if their assigned ID is in the wakeup signal. For example, if we know the rates at which nodes are sending data and have k wakeup channels to use, we would like to assign channels to the nodes in such a way that reduces the energy consumption caused by full wakeups. There are also other ways to partition the wakeup channel, other than encoding bits in the wakeup signal, which can be explored. For example, a different frequency band could



(a) Relative energy consumption for random network topologies.



(b) Relative latency for random network topologies.



(c) Relative packet drop percentage for random network topologies.

Fig. 11. Random network topologies (relative values).

be used for each wakeup signal and nodes could be assigned a frequency on which to listen for wakeups.

Another improvement to our protocol is to utilize the hard disk spin-down techniques mentioned in Section IV-A. Thus, T_{thresh} would be adjusted dynamically. When traffic is heavy, T_{thresh} could be larger since it may take more time for the sender to access the medium to transmit a packet. The disadvantage of this approach is that it requires more synchronization to make sure neighboring nodes agree on a T_{thresh} value. Finally, the overhead of filter packets can be reduced if their information is piggybacked onto RTS or DATA packets.

ACKNOWLEDGMENTS

Thanks to Rex Min and Chipcon for answering hardware questions. This work is funded in part by NSF grant ANI-0125859 and a NDSEG fellowship.

REFERENCES

[1] MICA2 Mote Datasheet, `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-05_A_MICA2.pdf`.

[2] C. S. Raghavendra and S. Singh, "PAMAS – Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks," *ACM Computer Communications Review*, July 1998.

[3] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 70–80, January-March 2002.

[4] C. Guo, L. C. Zhong, and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks," in *IEEE GlobeCom 2001*, November 2001.

[5] IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[6] J. M. Rabaey, M. J. Ammer, J. L. da Silva Jr., D. Patel, and S. Roundy, "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking," *IEEE Computer*, July 2000.

[7] E. Shih, P. Bahl, and M. J. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," in *ACM MobiCom 2002*, September 2002.

[8] C. F. Chiasserini and R. R. Rao, "Combining Paging with Dynamic Power Management," in *IEEE Infocom 2001*, April 2001.

[9] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *IEEE Infocom 2002*, June 2002.

[10] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Topology Management for Sensor Networks: Exploiting Latency and Density," in *ACM MobiHoc 2002*, June 2002.

[11] C. Schurgers, V. Tsiatsis, and M. Srivastava, "STEM: Topology Management for Energy Efficient Sensor Networks," in *IEEE Aerospace Conference 2002*, March 2002.

[12] R. Zheng and R. Kravets, "On-demand Power Management for Ad Hoc Networks," in *IEEE Infocom 2003*, April 2003.

[13] Y. Xu, J. Heidemann, and D. Estrin, "Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks," USC/Information Sciences Institute, Tech. Rep. 527, 2000.

[14] ——, "Geography-informed Energy Conservation for Ad Hoc Routing," in *ACM MobiCom 2001*, July 2001.

[15] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *ACM MobiCom 2001*, July 2001.

[16] M. L. Sichitiu, "Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks," in *IEEE Infocom 2004*, March 2004.

[17] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[18] Chipcon CC1000 Datasheet, `http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf`.

[19] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[20] D. P. Helmbold, D. D. E. Long, and B. Sherrod, "A Dynamic Disk Spin-down Technique for Mobile Computing," in *ACM MobiCom 1996*, November 1996.

[21] X. Yang and N. H. Vaidya, "A Wakeup Scheme for Sensor Networks: Achieving balance between energy saving and end-to-end delay," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) 2004*, May 2004.

[22] E. W. Weisstein, "MathWorld," `http://mathworld.wolfram.com`.

[23] A. Woo and D. E. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," in *ACM MobiCom 2001*, July 2001.

[24] `ns-2` – The Network Simulator, `http://www.isi.edu/nsnam/ns`.

Matthew J. Miller received the BS degree, summa cum laude, from Clemson University in computer engineering and the MS degree from the University of Illinois at Urbana-Champaign (UIUC) in computer science. Currently, he is pursuing a PhD degree at UIUC, focusing on energy efficient protocols for ad hoc networks. In 2001, he received a US National Science Foundation Fellowship and an ASEE National Defense Science and Engineering Graduate Fellowship. He is a student member of IEEE. For more information, please visit `https://netfiles.uiuc.edu/mjmille2/www/`.

Nitin H. Vaidya received the PhD degree from the University of Massachusetts at Amherst. He is presently an Associate Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign (UIUC). He has held visiting positions at Microsoft Research, Sun Microsystems and the Indian Institute of Technology-Bombay. His current research is in the areas of wireless networking and mobile computing. His research has been funded by various agencies, including the National Science Foundation, DARPA, BBN Technologies, Microsoft Research, and Sun Microsystems. Nitin Vaidya is a recipient of a CAREER award from the National Science Foundation. Nitin has served on the program committees of several conferences and workshops, and served as program co-chair for the 2003 ACM MobiCom. He has served as editor for several journals, and presently serves as Editor-in-Chief of the *IEEE Transactions on Mobile Computing*. He is a senior member of IEEE and a member of the ACM. For more information, please visit `http://www.crhc.uiuc.edu/~nhv/`.