Procrastination Might Lead to a Longer and More Useful Life

Prabal Dutta, David Culler, and Scott Shenker
{prabal,culler,shenker}@cs.berkeley.edu
Computer Science Division
University of California, Berkeley
Berkeley, California 94720

1 Introduction

Energy-efficiency has pervaded nearly every aspect of wireless sensor network ("sensornet") research, including platforms [20, 16], sensor data acquisition [15], operating systems [12], media access control [32, 19], routing [21, 31] and applications [26, 27, 28]. All aspects of sensornet operations, including sleeping, sensing, storage, computation, and communication, have been closely studied and a range of system software techniques for low-power operation have emerged [5]. Despite these advances, sensornet applications report short lifetimes [26] and low data yields [28], much to the chagrin of their developers and users.

In this paper, we focus on a common class of sensor applications we call *simple data collection*, and explore how their lifetimes might be increased, and their data yields improved. These applications typically have stationary nodes with reasonably good connectivity, have relaxed latency requirements, and do not require in-network aggregation. Many simple data collection applications employ a lock-step *sense-and-send* paradigm [26, 27, 28]. Such applications periodically read sensors and almost immediately route these readings "up" a spanning tree rooted at the network base station. As a result, routing gradients and neighbor tables must be maintained continuously, which can be costly, and channel acquisition/synchronization costs can be amortized over only a single packet, which is inefficient.

A simple way to better amortize costs and improve efficiencies, it might seem, would be to decouple sensing and sending rates, batch data into bundles, and delay sending for as long as the user might allow (e.g. Nagle). Indeed, one recent paper concludes that "using local storage to perform batching in duty-cycled applications reduces communication energy costs (up to 58x) in comparison to a non-batching approach, by amortizing the per-packet transmission overhead over a large number of data bytes." [16].

Contrary to conventional wisdom and recent publications, we argue that batching alone provides no fundamental benefits for scheduled protocols. And, for current sensornet technologies and applications, batching provides only negligible benefits for polled protocols. The reason is simple: for the typically balanced send/receive workloads we see in practice, synchronization costs dominate communications overhead while radio wakeup, random backoffs, channel contention, data transfer, and retransmissions are negligible. To support our position, we carefully examine typical application workloads and identify the bounds on energy-efficiency, and to provide hope, we highlight promising research directions for improving the constant factors.

This paper assumes that efforts to reduce synchronization costs will prove fruitful and looks ahead to explore the benefits and challenges that batching creates. Procrastination, or artificial delay, often does not harm the end user and it provides a great many opportunities for optimization. If data are batched, their size can be reduced through compression, which reduces intermediate buffering requirements; their unit of reliable transfer can be bundles rather than packets, which better amortizes protocol overhead and channel acquisition costs; and their transport can occur over links whose quality does not change appreciably during the transfer, which improves reliability. While we advocate postponing communications for as long as the user will allow, this approach does raise new challenges: if routing gradients and neighbor tables are not maintained continuously, how are they established quickly and accurately when the network does finally wake up?

Local storage, which would facilitate a *batch-and-send* paradigm, historically has been avoided because of the relatively higher energy cost to write data to flash rather than send data over the radio [18]. However, modern NAND flash memory is more energy-efficient than prior generations [16] and quite affordable ($7/GB in May 2007 on the spot market) [DRAM eXchange].

2 Overhead is Harmful

Conventional wisdom holds that communications is the most costly of the energy-consuming activities that nodes undertake. Two sensornet deployments with carefully documented power profiles support this view. Table 1 shows the division of node power budget between radio operation and all other operations (e.g. sleeping, sensing, storage, and computation). The Great Duck Island (GDI) [26] figures are directly from the paper while the Wireless Soil Ecology (WiSE) [27] figures are estimated by multiplying the reported current values by 2.7 V, our estimate of the average battery voltage.

Deployment	$\mathbf{P}_{radio}$	$\mathbf{P}_{other}$	% of Budget
GDI [26]	994 μW	178 μW	84.8%
WiSe [27]	972 μW	22 μW	97.8%

Table 1: Radio operation dominates the node power budget. Data from two well-documented deployments show that even at low duty cycles of approximately 1 to 2%, radio operation is about an order of magnitude more expensive than all other operations combined. Hence, lifetime improvements will require substantial reductions in radio on-time.

Radio operation dominates the system power budget, even when the radio operates at a low duty cycle, suggesting significant lifetime improvements effectively require greater duty cycle reductions. As Table 2 shows, the radio duty cycles (DC) for many low-rate data collection applications have remained relatively constant over the years. This raises the question of whether these numbers reflect the required capacity to transfer the application data in a timely manner, or whether the numbers are an artifact of an engineering decision. The former would limit progress while the latter would provide hope.

Year	Deployment	MAC	DC	Period (s)
2003	GDI [26]	Polled	2.2%	0.54-1.085
2004	Redwoods [28]	Sched	1.3%	300
2005	FireWxNet [9]	Sched	6.7%	900
2006	WiSe [27]	Sched	1.6%	60

Table 2: Radio duty cycles (DC) for low-rate data collection applications have remained relatively constant over the years. Polled radio operation, sometimes called low-power listening [19], periodically samples the channel for radio activity and powers down the radio between successive samples. Scheduled radio operation establishes well-known and periodic time intervals when it is legal (or illegal) to transmit.

A 1.3% duty cycle, the smallest in Table 2, translates to 19 minutes of daily on-time. This suggest that the radio on-time is being used rather inefficiently. Indeed, as shown in Table 3, the data generation rate of a typical node, compared with the effective radio on-time of a node under a typical duty cycle setting, shows considerable overhead. The average data generation rate for GDI [26], our canonical data collection application, is 0.8 bits/sec including protocol overhead. The effective radio on-time of a node operating at 2.2% translates to 880 bits/sec or 1,100 times the airtime required to transfer the data for a 1-hop network, assuming no data loss. Therefore, while radio operation dominates the power budget, useless radio on-time dominates radio operation. Indeed, useful data transfer accounts for a small fraction of the time the radio is operating.

Parameter	Value
Raw data generation rate	20 bytes/5 min
Raw data generation rate	0.53 bits/sec
Packet protocol overhead	50%
Data rate with overhead	0.80 bits/sec
Full radio data rate	40 kbits/sec
2.2% radio data rate	880 bits/sec
Radio-on : Data-transfer (1-hop)	1100 : 1
Radio-on : Data-transfer (2-hop)	157 : 1 †
Radio-on : Data-transfer (3-hop)	64 : 1 †
Radio-on : Data-transfer (4-hop)	35 : 1 †

Table 3: Data from the Great Duck Island [26] deployment shows that while radio on-time dominates the power budget, useful data transfer actually accounts for a small fraction of the radio on-time. Lifetime improvements will require a significant reduction in communication overhead activities like channel acquisition, topology maintenance, and idle listening. †Figure 1 provides an explanation for these numbers.

Before exploring *why* useful data transfer accounts for such a small fraction of the radio on-time, we briefly address the issue of multi-hop data collection and the load from route-through traffic. We propose a very simple model, shown in Equation 1 and explained in Figure 1, to estimate the load on a node at depth 1-hop from nodes at depth 2-hop and greater, in an n-hop network. This model is used to generate the radio-on : data-transfer ratios shown in bottom half of Table 3.

$$load(n) = 2(n^2 - 1) + 1 \quad (1)$$

The intuition for this model is that at depth n, an area proportional to n^2 exists. We subtract an area proportional to one, corresponding to an area of depth one, leaving an area proportional to $n^2 - 1$ whose traffic must

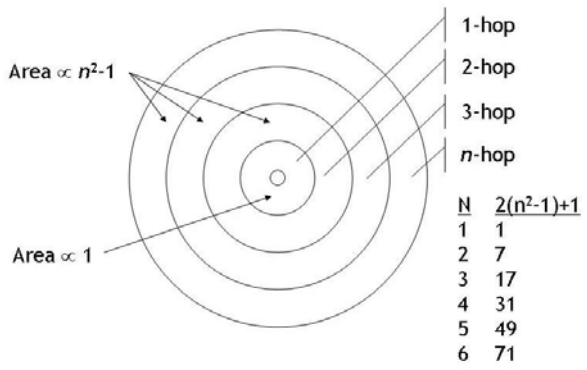

Figure 1: A simple model for estimating the total load on a 1-hop node in an n-hop network. This model assumes an admittedly unrealistic unit radius communication model and uniform node distribution, but enables a simple, conceptual analysis. One-hop nodes occupy an area of πr^2 where $r = 1$, two hop nodes occupy an area of $\pi(2r)^2 - \pi r^2$, and n-hop nodes occupy an area of $\pi(nr)^2 - \pi((n-1)r)^2$. Since we assume constant density, in an n-hop network, one-hop nodes must forward (i.e. receive and transmit) data from $(\pi(nr)^2 - \pi r^2)/(\pi r^2) = n^2 - 1$ nodes, as well as transfer their own data. Hence, the total load on a 1-hop node in an n-hop network is $2(n^2 - 1) + 1$.

be transferred through the unit area of depth one. For each node whose depth is greater than one, a node whose depth is one must first receive and then transmit the first node's data, which accounts for the factor of two. Finally each node of depth one must transmit its own data. Our model assumes unit disk radio propagation, uniform node density, a degree-constrained spanning tree for collection, and no data loss. Admittedly unrealistic, its purpose is solely to illustrate that even in multi-hop networks, the data traffic pales in comparison to the useless radio on-time. Using this simple model, we see that in even a 4-hop network, the 1-hop nodes would have an overhead of 35, and that in principal, data could be collected from a 23-hop network without increasing the radio duty cycle beyond 2.2%[1].

3 Synchronization Kills

The observation that duty cycles below 1% are seldom seen in practice raises the obvious question: *why?* The chief reasons, according to Ye et al. is that synchronization cost, either explicitly in scheduling or implicitly in long preambles, limit nearly all earlier MAC protocols to duty cycles of 1-2% [32].

[1] $1100/(2(4^2-1)+1) = 35$; $1100/(2(23^2-1)+1) = 1$

3.1 Scheduled Communications

Ye et al. demonstrate that ultra-low duty cycles of 0.1% and lower are possible using their scheduled channel polling (SCP) protocol [32]. As its name implies, SCP synchronizes polling times. An SCP MAC sends a short tone to indicate it has traffic to send, followed by the data. The length of the tone increases with the communications interval, to account for clock skew, but the tone's lower bound is in the range of 0.5-2 ms, the minimum time required to poll the channel.

While duty cycles of 0.1% do considerably lower the communications cost, a significant communications overhead still remains. For example, the theoretical channel capacity requirements of a 1-hop network can be satisfied with a 0.002% (2.2%/1100) duty cycle while a 2-hop network can be satisfied with a 0.014% (2.2%/157) duty cycle, still one to two orders of magnitude lower than what SCP currently offers. This raises the question: *what is the lowest duty cycle achievable using a scheduled communication protocol?* To explore this question, Figure 2 depicts how crystal frequency skew can affect synchronization in scheduled communication.

Typical frequency skew of $\pm$30-50 ppm due to manufacturing variations, an additional $\pm$10-20 ppm due to temperature variations, and an exponentially decreasing $\pm$3-5 ppm per year (exponent is typically 1/2) are common. These frequency skews result in relative clock drift between nodes and, as a result, nodes must include a guard time, equal to the maximum drift, which grows linearly with the interval between communications. Letting r_{skew} ($\Delta f/f$) be the frequency skew and T_{pkt} be the packet period, the minimum guard time becomes

$$t_{guard} = 2 \cdot r_{skew} \cdot T_{pkt} \tag{2}$$

If it takes t_{pkt} time to send a packet, then the total transmission time will be $t_{guard} + t_{pkt}$ and the duty cycle is

$$DC = \frac{2 \cdot r_{skew} \cdot T_{pkt} + t_{pkt}}{T_{pkt}} \tag{3}$$

which simplifies to

$$DC = 2 \cdot r_{skew} + \frac{t_{pkt}}{T_{pkt}} \tag{4}$$

This relationship establishes a fundamental lower bound of $2 \cdot r_{skew}$ on the the duty cycle and three avenues to reduce the radio on-time. One avenue is to reduce the frequency skew with higher tolerance crystals or oscillators, relatively power-hungry temperature-compensated crystal oscillators, or improved calibration. A second avenue is to reduce the packet transmission time by increasing the radio speed (reducing the radio wakeup time would also help). The final avenue is to decrease the data rate by increasing the communications period.

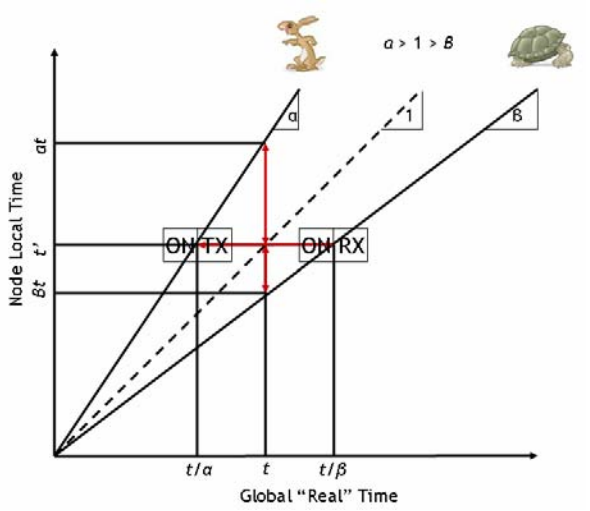

Figure 2: Channel synchronization is affected by crystal frequency skew in scheduled communications. A typical crystal oscillator exhibits a frequency skew from manufacturing variations, temperature drift, and aging. Collectively, these skews may add up to variations of $\pm$50 ppm or more, and affect synchronization. Consider nodes A and B, with frequency skew $\Delta f/f$ given by $\alpha - 1$ and $\beta - 1$, respectively. Assume that $\alpha > 1 > \beta$. Then, node A will expect to communicate at real time t/α while node B will expect to communicate at real time t/β. The difference between t/α and t/β may grow unbounded over time, which is why scheduled protocols must resynchronize periodically, and usually include a guard time that is at least twice the maximum drift.

Some observations follow from these relationships. First, simply batching packets and transmitting them less frequently provides little or no benefit, assuming that the data rate stays constant, clocks are resynchronized during each communication, and the radio wakeup time is a small fraction of a packet time, all of which are usually true. This is because the total overhead from clock drift is the same in both cases, but that it is paid in pieces in the streaming case and paid at once in the batching case.

Second, for a typical frequency skew of $\pm$50 ppm (or rather variations in skew between nodes), the lower bound on the duty cycle is 0.01%. This bound is two orders of magnitude better than current deployments, and one order of magnitude better than state-of-the-art [3, 32]. Finally, note that t_{pkt}/T_{pkt} is negligible for the applications from Table 2. In GDI for example, $t_{pkt} = 6$ ms (30 bytes/40 kbps), and $T_{pkt} = 300$ s, making $t_{pkt}/T_{pkt} = 0.002\%$. These results suggest another order of magnitude improvement in duty cycles is within our grasp but that further reductions will require compensating for drift [1] rather than increasing the radio speed.

3.2 Polled Communication

A low-power listening (LPL) receiver is one that duty cycles its radio. The radio is turned on for time t_{poll} every T_{poll} period, giving an effective duty cycle

$$DC_{listen} = \frac{t_{poll}}{T_{poll}} \tag{5}$$

To communicate, an LPL transmitter would have to send a preamble of length in time $T_{poll} + t_{poll}$ prior to sending a data packet. Assuming the length in time of a data packet is t_{pkt}, and a data packet is sent with a period of T_{pkt}, the effective transmit duty cycle is

$$DC_{tx} = \frac{T_{poll} + t_{poll} + t_{pkt}}{T_{pkt}} \approx \frac{T_{poll}}{T_{pkt}} \tag{6}$$

The approximation follows from the fact that for low duty cycles, $T_{poll} \gg t_{pkt} \gg t_{poll}$.

The expected time to receive using this scheme is approximately $T_{poll}/2 + t_{pkt}$, since we would expect that on average a receive would wake up halfway through the preamble transmission, stay awake during the second half of the preamble, and remain awake to receive the packet. If, instead of using a long preamble, a packet is transmitted repeatedly, the expected receive cost can be reduced significantly without changing the transmit cost. The expected receive cost becomes $1.5 \cdot t_{pkt}$ since we would expect on average that a receiver would wake up halfway through a packet transmission, stay awake until the end of that packet, and then receive the packet the next time it is sent.

To estimate the effective receive duty cycle, we subtract the t_{poll} time from $1.5 \cdot t_{pkt}$, since the poll time has already been accounted for in the listen duty cycle.

$$DC_{rx} = \frac{1.5 \cdot t_{pkt} - t_{poll}}{T_{pkt}} = \frac{1.5 \cdot t_{pkt}}{T_{pkt}} - \frac{t_{poll}}{T_{pkt}} \tag{7}$$

Several important observations follow from these relationships. First, for an approximately balanced receive/transmit workload, DC_{listen} and DC_{tx} are inversely proportional, so any duty cycle benefits from changing T_{poll} are lost. Second, reducing DC_{listen}, and to a lesser extent DC_{rx}, by reducing the poll time, t_{poll}, is one path to a lower average duty cycle. The t_{poll} parameter typically falls in the range of 0.5-2 ms [32] and is usually the sum of two different things: the startup time of the radio's crystal oscillator and the time to sample the RF channel power. The oscillator startup time can reduced by lowering its series resistance [20] or using a quick start oscillator [2]. These optimization can result in startup times of 580 μs and 10-15 μs, respectively, but the gains are tempered by the 128μs or typically required for channel power sampling [4], limiting the DC_{listen} improvements to a factor of four or so.

Batching provides a second route to lower duty cycles. Increasing the packet period, T_{pkt}, decreases the effective transmit duty cycle, DC_{tx}, by the same factor and does not adversely affect the listen duty cycle, DC_{listen}. Batching and sending more data less frequently marginally decreases the receive duty cycle, DC_{rx}. For GDI, without batching, we have $DC_{listen} \approx 0.8\%$ (8 ms/1.085 s), $DC_{tx} \approx 0.4\%$ (1.085 s/300 s), and $DC_{rx} = 0.003\%$ (9 ms/300 s).

These numbers, coupled with the GDI data, suggest that GDI was a well-engineered system operating close to its theoretical capacity. However, they also underscore a different concern: that today's purely polled communications cannot achieve duty cycles much below about 1% for balanced receive/transmit workloads. Of course, batching can help greatly: for $T_{poll} = 80$ s and $T_{pkt} = 86400$ s, we have $DC_{listen} \approx 0.01\%$ (8 ms/80 s), $DC_{tx} \approx 0.1\%$ (80 s/86400 s), and $DC_{rx} = 0.003\%$ (2.5 s/86400 s). To achieve a $DC_{tx} = 0.01\%$, the transmitter would have to batch data for ten days.

Finally, note that increases in radio speed are negligible, dual-radio wake-on-wireless can provide constant factor improvements [22], and radio-triggered wake-up can eliminate synchronization cost [8].

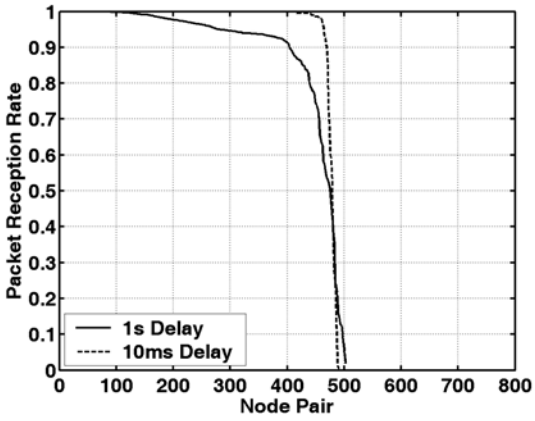

Figure 3: Delivery probability of packets sent over IEEE 802.15.4 wireless links with different inter-packet delays. Packets were broadcast by 28 nodes in a static network. Each node broadcast 200 packets with inter-packet delays of 10 ms and 1 s. Over short periods of time, links are bimodal but over longer periods of time, existing links deteriorate and new ones emerge.

4 Procrastination Causes New Problems

This paper advocates postponing communications for as long as the user will allow, but this approach raises new challenges: if routing gradients and neighbor tables are not maintained continuously, how are they established quickly and accurately when the network does finally wake up?

4.1 Periodic Data Collection

The wisdom of constructing a routing tree on-demand rather than maintaining one continuously is debatable. Our proposal is based on the observation that over short periods of time, low-power wireless links are largely bimodal with either high or zero delivery probability but over longer periods of time, links exhibit a wider distribution of delivery probabilities [24]. This suggests that routing metrics are usually accurate soon after tree construction but borderline links can deteriorate quickly, become stale, and cause packet loss. Figure 3 illustrates that the fraction of links with intermediate delivery probabilities grows with time.

The challenge in this environment, then, is constructing a tree quickly. This problem can be divided into two subproblems: "flooding" a beacon and selecting a parent. The problem with naïve flooding is the high-degree of contention, while epidemic dissemination protocols have slower convergence times and long tails [13]. To quickly flood beacons in a network, we prototyped *Ripple*, a conceptually simple adjustment to flooding that significantly reduces the probability of collisions. Ripple delays retransmission of a received flood packet by a time proportional to the received signal strength indicator (RSSI) and hop count of the packet. Nodes that receive packets with the smallest RSSI values are usually the ones furthest away in each hop. Since nodes retransmit with a delay proportional to their RSSI, the nodes furthest away transmit first, allowing the flood to quickly reach the edge of the network in concentric rings, while the nodes in each ring transmit with successively smaller radii.

The second subproblem is picking good links rapidly. The missing piece here is an agile link estimator. Current estimators for low-power radios focus on stability over agility and have convergence times on the order of tens of beacons. For example, the version of MintRoute [30] used in the Redwoods deployment required about ten beacons to converge – beacons which were received over the course of an hour. The system designers assumed that links churned more slowly but Figure 3 suggests that this may not be the case. Figure 4 shows an easier way on modern radios.

A recent study has suggested that RSSI on modern low-power radios might be under appreciated [25] while another study found a very sharp and predictable transition region from good to bad links based on RSSI [23]. RSSI is a measure of RF energy in a radio packet as observed by the receiver. Figure 4 shows that over short

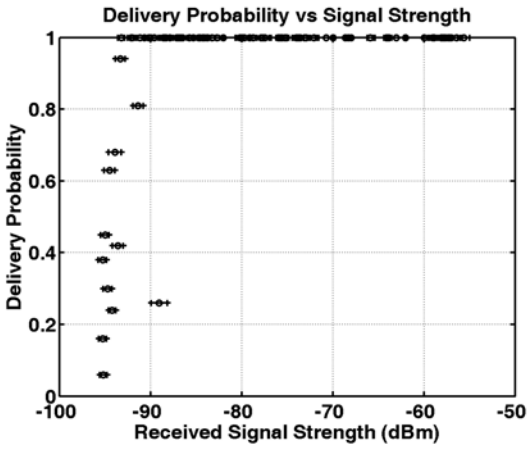

Figure 4: Packet delivery probability versus the received signal strength indicator (RSSI). Each of ten nodes arranged in a line broadcast 100 packets. The error bars show one standard deviation in RSSI values. The experiments were carried out using Telos B motes [20].

packet bursts, RSSI values are quite stable with typical deviations less than 1 dBm. In addition, there is a sharp cliff near -90 dBm, suggesting that neighbors with RSSI values above this level are good candidates for routing. In practice, the RSSI curves are receiver-specific and would have to be learned over time by each receiver. However, armed with this information, even a single packet might exceeding this threshold might suggest a promising link.

4.2 Interactive Data Collection

We briefly sketch an alternate approach that may be better suited to applications where data collection is required interactively. Every node transmits a beacon in time t_{beacon} and repeats this process with a period T_{beacon}. After each beacon, a node listens briefly for time t_{listen} to check for a channel activity, and stays awake if it detects any, and goes back to sleep otherwise. When the user wishes to extract the data stored in the network, the system "walks the tree" from the root to the leaves. The process would work as follows. First, the root listens for T_{beacon} to identify all 1-hop nodes with strong RSSI values. Then, during the next T_{beacon} period, the root contacts each of the 1-hop nodes in turn to initiate data collection. This communication synchronizes their clocks and allows efficient communications during the remainder of the data collection phase. The 1-hop nodes repeat this process to contact the 2-hop nodes, and so on. If two or more nodes attempt to contact the same node in a lower tier, their concurrent transmissions may collide but since the receiver will detect channel activity and remain awake, this will give the contenders an opportunity to enter backoff and compete. Once data has been requested from a node, it is that node's responsibility to collect data from its children and deliver all of this data back to the node that originally requested this data.

5 Related Work

We essentially advocate a delay-tolerant networking (DTN) approach to data transfer [7], but not for the usual reasons, since sensornets for simple data collection can provide a contemporaneous path from the data source to the data sink; can offer end-to-end round-trip times of a few seconds or less; are embedded in the physical world and therefore exhibit almost no mobility over short time scales; and offer links with low loss rates over short periods of time. Rather, our motivation comes from better amortizing overhead costs, much like Nagle, cache coherence, and disk buffering algorithms.

Others have explored DTN for sensornets but their focus has been for the usual DTN motivations [17, 10, 14]. These ideas have also been explored in the context of low-power MAC protocols [19, 3, 32]. Estrin has suggested that reliability can be improved by employing hop-by-hop storage in the presence of network disconnections [6]. Mathur et al. have suggested using large flash memories for archival purposes, to overcome sensor platform memory constraints, and to reduce communications overhead [16].

Of course, not all sensornet deployments avoid local storage, but historically, local storage has been limited in scope to logging for backup [28], as a small circular buffer for high-rate data streams [29], and as a queue during network disruptions [11]. While others have suggested DTN be used by necessity, we suggest DTN be used by choice. We also explore the other challenges involved in improving energy-efficiency and data yield.

6 Closing Thoughts

We set out to explore a sensornet data collection architecture built around the rather obvious concept of increasing network longevity and data reliability through batching and delayed communication. We discover, instead, that delay provides little to no practical benefits for polled and scheduled protocols, respectively. We identify radio wakeup latency and clock skew as the fundamental constraints limiting communications efficiency and highlight promising research efforts in these areas. We also show that artificially introducing delay raises new research challenges like quickly establishing routing gradients and neighbor tables, and we sketch some possible avenues to address these challenges.

References

[1] ARFVIDSSON, J., PARK, E., AND LEVIS, P. Lowering radio duty cycle through temperature compensated timing. In *Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys'06)* (New York, NY, USA, 2006), ACM Press, pp. 409–410.

[2] BLANCHARD, S. A. Quick start crystal oscillator circuit. In *Proceedings of the 15th Biennial University/Government/Industry Microelectronics Symposium* (2003), pp. 78–81.

[3] BURRI, N., VON RICKENBACH, P., AND WATTENHOFER, R. Dozer: ultra-low power data gathering in sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks* (New York, NY, USA, 2007), ACM Press, pp. 450–459.

[4] CHIPCON AS. *CC2420: 2.4 GHz IEEE 802.15.4 /ZigBee-ready RF Transceiver*, June 2004.

[5] DUTTA, P. K., AND CULLER, D. E. System software techniques for low-power operation in wireless sensor networks. In *Proceedings of the 2005 International Conference on Computer-Aided Design (ICCAD'05)* (2005), pp. 925–932.

[6] ESTRIN, D. Reliability and storage in sensor networks. Tech. Rep. CENS Technical Report 59, 2005.

[7] FALL, K. A delay-tolerant network architecture for challenged internets. In *Proceedings of ACM SIGCOMM 2003 (SIGCOMM'03)* (August 2003).

[8] GU, L., AND STANKOVIC, J. Radio triggered wake-up capability for sensor networks. In *Real-Time Applications Symposium (RTAS'04)* (May 2004).

[9] HARTUNG, C., SEIELSTAD, C., HOLBROOK, S., AND HAN, R. Firewxnet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the Fourth International Conference on Mobile Systems, Applications, and Services (MobiSys'06)* (2006).

[10] HO, M., AND FALL, K. Poster: Delay tolerant networking for sensor networks. In *The First IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004) (invited)* (Oct. 2004).

[11] JUANG, P., OKI, H., WANG, Y., MARTONOSI, M., PEH, L., AND RUBENSTEIN, D. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceedings of the ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (oct 2002).

[12] LEVIS, P., MADDEN, S., POLASTRE, J., SZEWCZYK, R., WHITEHOUSE, K., WOO, A., GAY, D., HILL, J., WELSH, M., BREWER, E., AND CULLER, D. TinyOS: An operating system for wireless sensor networks. In *Ambient Intelligence* (New York, NY, 2004), Springer-Verlag.

[13] LEVIS, P., PATEL, N., CULLER, D., AND SHENKER, S. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks. In *Proceedings of the First USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)* (2004).

[14] LOUBSER, M. Delay tolerant networking for sensor networks. Tech. Rep. SICS Technical Report T2006:01, 2006.

[15] MADDEN, S., FRANKLIN, M., HELLERSTEIN, J., AND HONG, W. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS) 30*, 1 (Mar. 2005), 122–173.

[16] MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. Ultra-low power data storage for sensor networks. In *Proceedings of IEEE/ACM Conference on Information Processing in Sensor Networks - Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems* (Apr. 2006).

[17] PATRA, R., AND NEDEVSCHI, S. Dtnlite: Delay tolerant networking on constrained devices, 2003.

[18] POLASTRE, J. Design and implementation of wireless sensor networks for habitat monitoring. Master's thesis, University of California at Berkeley, 2003.

[19] POLASTRE, J., HILL, J., AND CULLER, D. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM Conference on Embedded Network Sensor Systems* (2004).

[20] POLASTRE, J., SZEWCZYK, R., AND CULLER, D. Telos: Enabling ultra-low power wireless research. *The Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)* (Apr. 2005).

[21] SEADA, K., ZUNIGA, M., HELMY, A., AND HARI, B. K. Energy-efficient forwarding strategies for geographic routing in los sy wireless sensor networks. In *Conference On Embedded Networked Sensor Systems* (Nov. 2004).

[22] SHIH, E., BAHL, P., AND SINCLAIR, M. J. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the Eighth Annual ACM Conference on Mobile Computing and Networking* (Altanta, Georgia, USA).

[23] SON, D., KRISHNAMACHARI, B., AND HEIDEMANN, J. Experimental analysis of concurrent packet transmissions in low-power wireless networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys'06)* (2006).

[24] SRINIVASAN, K., DUTTA, P., TAVAKOLI, A., AND LEVIS, P. Understanding the causes of packet delivery success and failure in dense wireless sensor networks. Tech. Rep. Technical Report SING-06-00, 2006.

[25] SRINIVASAN, K., AND LEVIS, P. RSSI is under appreciated. In *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets 2006)* (2006).

[26] SZEWCZYK, R., MAINWARING, A., POLASTRE, J., AND CULLER, D. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)* (Nov. 2004).

[27] SZLAVECZ, K., TERZIS, A., OZER, S., MUSALOIU, R., COGAN, J., SMALL, S., BURNS, R., GRAY, J., AND SZALAY, A. Life under your feet: An end-to-end soil ecology sensor network, database, web server, and analysis service. Tech. Rep. Microsoft Technical Report MSR TR 2006 90, 2006.

[28] TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D. E., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. A macroscope in the redwoods. In *Proceedings of the Second ACM Conferences on Embedded Networked Sensor Systems (SenSys)* (2005).

[29] WERNER-ALLEN, G., LORINCZ, K., JOHNSON, J., LEES, J., AND WELSH, M. Fidelity and yield in a volcano monitoring sensor network.

[30] WOO, A., TONG, T., AND CULLER, D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys)* (2003).

[31] XU, Y., BIEN, S., MORI, Y., HEIDEMANN, J., AND AH ESTRIN, D. Topology control protocols to conserve energy in wireless ad hoc net works. In *Technical Report 6* (Apr. 2003).

[32] YE, W., AND HEIDEMANN, J. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the Fourth Internation Conferences on Embedded Networked Sensor Systems (SenSys'06)* (2006), pp. 321–334.