Lossy Links, Low Power, High Throughput

Simon Duquennoy
simonduq@sics.se

Fredrik Österlind
fros@sics.se

Adam Dunkels
adam@sics.se

Swedish Institute of Computer Science
Box 1263, SE-16429 Kista, Sweden

Abstract

As sensor networks move towards general-purpose low-power wireless networks, there is a need to support both traditional low-data rate traffic and high-throughput transfer. To attain high throughput, existing protocols monopolize the network resources and keep the radio on for all nodes involved in the transfer, leading to poor energy efficiency. This becomes progressively problematic in networks with packet loss, which inevitably occur in any real-world deployment. We present *burst forwarding*, a generic packet forwarding technique that combines low power consumption with high throughput for multi-purpose wireless networks. Burst forwarding uses radio duty cycling to maintain a low power consumption, recovers efficiently from interference, and inherently supports both single streams and cross-traffic. We experimentally evaluate our mechanism under heavy interference and compare it to PIP, a state-of-the-art sensornet bulk transfer protocol. Burst forwarding gracefully adapts radio duty cycle both to the level of interference and to traffic load, keeping a low and nearly constant energy cost per byte when carrying TCP traffic.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms

Design, Experimentation, Performance

Keywords

Sensor Network, Energy Efficiency, Interference

1 Introduction

As sensor networks move from single-purpose, homogeneous systems towards multi-purpose, heterogeneous systems, supporting multiple types of protocol is of increasing importance. Many different protocols have been developed, but have typically been considered in isolation. At one extreme, low power, low data rate data collection protocols have seen much attention [10, 23], but have a focus on low power consumption with little regard for throughput. At the other extreme, bulk transport protocols such as Flush [14] and PIP [27] focus only on throughput with no regard for low power consumption. Yet the two extremes must be combined in emerging networks.

Existing bulk transfer protocols [14, 24, 27] monopolize multi-hop path and disable radio duty cycling to achieve high throughput. We argue that multi-purpose low power networks need a generic forwarding technique able to handle cross-traffic while maintaining low power consumption.

We present *burst forwarding*, a generic forwarding technique intended to allow high throughput data transport with low power consumption in lossy wireless networks. Burst forwarding groups multiple packets into bursts that are transmitted over each hop with a stubborn link layer performing retransmissions on millisecond time scales. Bursts are coordinated across multiple hops using a two-level retransmission scheme and multi-channel operation. The nodes use radio duty cycling to keep the radio off more than 99% of the time. They wake up periodically to receive long bursts of data, store them in flash memory, and forward them to the next hop. The system adapts gracefully to interference thanks to retransmissions operating at both granularities of frames and bursts, handling both isolated losses and longer-lasting interference.

Burst forwarding is a generic mechanism that is not tied to any protocol: in this paper we used it below both bulk transfer, sensornet data collection, and IPv6. We use burst forwarding to carry TCP traffic over low-power wireless networks with high loss rates, demonstrating a nearly constant energy cost per byte, even during heavy loss.

This paper makes three contributions. First, we show that the combination of techniques in burst forwarding provides high throughput and low power consumption in multi-purpose networks even during severe radio interference. Second, we propose and evaluate a two-level retransmission mechanism addressing isolated losses, high-frequency interference as efficiently low-frequency interference. Third, we leverage burst forwarding to bring TCP to low-power wireless networks and show that a constant energy cost per byte can be reached under increasing radio interference.

SenSys'11, November 1–4, 2011, Seattle, WA, USA.

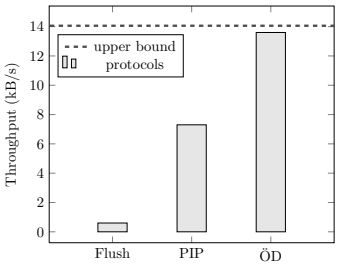


Figure 1. The throughput of existing high-throughput transport mechanisms for low-power networks: Flush [14], PIP [27], and Österlind and Dunkels (ÖD) [24]. ÖD reaches 97% of the theoretical upper bound. PIP is the fastest fully-fledged protocol with 7.3 kB/s, or 52% of the upper bound.

The paper is organized as follows: Section 2 motivates the need for general-purpose forwarding in low-power sensornets. Section 3 presents the design of burst forwarding. Section 4 discusses the application of burst forwarding to TCP transport in low-power wireless networks. Section 5 presents implementation aspects of burst forwarding. Section 6 evaluates burst forwarding in terms of energy-efficiency and throughput. It characterizes its behavior when facing interference and when handling cross-traffic. Section 7 addresses the support of TCP on top of burst forwarding, demonstrating the good interaction of rate control and radio duty cycling. We present the related work in Section 8, and conclude in Section 9.

2 A Case for General-Purpose Forwarding in Low-Power Wireless Networks

We believe that existing high-throughput mechanisms are not in line with the current trend of multi-purpose sensor networks and that low power consumption and multi-purpose applications must be seen as fundamental design aspects of future high-throughput mechanisms.

2.1 High Throughput in Sensornets

Many sensor networks can be categorized into a data logger model, where each node logs its sensed data which later is retrieved via radio. Many examples can be found in structural health and environment monitoring. In a railway monitoring deployment by Chebrolu et al. [4], every train pass triggered a download of about 7 kB of data sample from every node. In the Golden Gate deployment by Kim et al. [15], 512 kB of data were sampled twice a day from every node of a 46 hop network. The authors reported a throughput of 441 B/s. In the volcano deployment by Werner-Allen et al. [33], the design of the network operation was constrained by the low achievable throughput: as every node produced up to 1200 bytes of data per second, the authors decided not to send all samples to the base station. Instead, 1 second of samples, accounting for about 50 kB, was collected on occurrence of an event. In the latter example, the lossy environment resulted in a maximum throughput of 561 B/s.

Optimizing the throughput of bulk transfer is a natural goal. Figure 1 reports the performance of state of the art bulk transfer protocols [14, 24, 27]. The fastest results (referred to as ÖD) have been obtained by Österlind and Dunkels [24]. With the conditional immediate transmission mechanism, a raw 802.15.4 throughput of 13.6 kB/s was reached: 97% of the theoretical upper bound. Flush [14] provides reliable data transmissions with rate control, but reaches only 0.6 kB/s. PIP [27] uses TDMA, multiple-channels and conditional immediate transmission. It reaches 52% of the theoretical upper bound, because it uses the 802.15.4 header, uses link-layer acknowledgments and implements fully functional queue management.

2.2 High Throughput is not Enough

Existing solutions such as Flush and PIP monopolize the network resources in the sense that every node in the forwarding path is dedicated to a single operation, the forwarding of the current bulk transfer. The design of these protocols forces every node to keep its ratio on until the end of the transfer: Flush is based on neighbors overhearing, ÖD and PIP are sending and receiving continuously. In an ideal communication environment, with no packet losses, this is the ideal solution.

In real networks, links are lossy and losses occur on both short and long timescales [29]. If one packet is lost, chances are high that subsequent packets will be lost too. This has implications for high-throughput transport. Since existing protocols have their radios turned on for the full duration of the transfer, a loss period will cause a significant waste of energy. According to recent studies, loss periods are typically on the order of 500 ms [29, 30]. If such a loss period occurs during a transfer, all nodes participating in the transfer will needlessly spend energy since no packets can be forwarded. Since the energy consumption during such a transfer is on the order of 100 times compared to a steady-state duty-cycled network, a 500 ms loss period results in an energy expenditure equivalent to roughly one minute in steady state.

We argue that because of the lossy nature of low-power radio communication, radio duty cycling is needed even during high-throughput data transfer. With duty cycling, energy can be saved during lossy periods when no packets can be forwarded. We therefore argue that high-throughput data transfer in low-power wireless must be evaluated in terms of energy efficiency and not only in terms of throughput.

2.3 Multi-Purpose Low-Power Wireless

In the early vision of sensor networks, each network was single-purpose. But in many emerging application domains, networks are multi-purpose. This is demonstrated by the move towards IP-based sensor networks [16, 32], which by design are intended to be multi-purpose networks. In an IP-based sensor network, the network is agnostic to the applications running on top of it [8].

We argue that as sensor networks move forward, high-throughput transport mechanisms can no longer assume to monopolize the network, but must behave as good network citizens towards other protocols in the network. As with any protocol concurrency, it is not possible to avoid affecting other protocols, but the effect should be small enough for other protocols to be able to run.

3 Burst Forwarding

Burst forwarding is a forwarding layer intended to support high throughput data transport in low-power multi-purpose wireless networks with potentially high-loss links. Burst forwarding is intentionally simple and operates with mechanisms at two timescales: low-power packet bursting, which rapidly transmits a burst of packets over a single hop and over timescales of seconds, and burst coordination, which coordinate the transmission of bursts across multiple hops and over timescales of tens of seconds.

3.1 Low-Power Packet Bursting

Low-power packet bursting transmits consecutive frames in bursts in a way that is coordinated with the underlying radio duty cycling mechanisms so that both the sender and the receiver can sleep between transmissions. The duty cycling mechanism could be either sender-initiated or receiver-initiated, as long as it provides a periodic wakeup mechanism. Our current implementation runs over the sender-initiated ContikiMAC protocol [6].

3.1.1 Bursts in ContikiMAC

ContikiMAC [6] is a low-power listening protocol that borrows ideas from many sender-initiated protocols in the literature. The wakeup mechanism in ContikiMAC consists of two consecutive channel samples with an interval that guarantees that a packet transmission from a neighbor will be seen by one of the channel samples. When using a wakeup rate of 8 Hz, ContikiMAC has a steady-state duty cycle lower than 0.6% [5]. ContikiMAC uses data packets as wakeup signals. To send a packet in ContikiMAC, the sender transmits the data packet multiple times in rapid succession. When the receiver gets the packet, it responds with a link layer acknowledgment. When the sender receives an acknowledgment, it stops sending. Subsequent transmissions use the knowledge of the receiver's wakeup phase to reduce the number of wakeup transmissions.

We extend ContikiMAC so it supports low-power packet bursts, i.e. rapid transmission of successive packets after a single wakeup. Our current implementation leverages the frame pending bit in the 802.15.4 packet header to indicate to the receiver that more packets are on their way. The receiver will then keep its radio on in anticipation of the next packet. The mechanism is inspired by Hui and Culler [12] who used the frame pending bit for a similar purpose.

Combined with the ContikiMAC data-packets-as-wakeup mechanism, the bursts provide a rapid retransmission mechanism: every packet is repeatedly sent until reception of a link layer acknowledgment or the end of the wakeup transmission period, as illustrated in Figure 2.

3.1.2 Inter-packet Sleeping

Burst forwarding allows the radio of the sender and the receiver to be switched off between the transmissions in a burst. For example, the receiver may be switched off while the packet is loaded into the radio chip of the sender. On the receiving side, this requires careful timing since the radio must be turned on before the reception of the next packet in the burst. Note that this optimization does not suffer from clock-drift problems since each packet reception sets a new timer that will be triggered only a few milliseconds later.

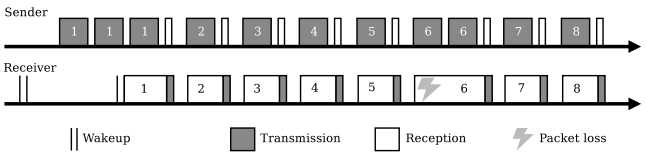
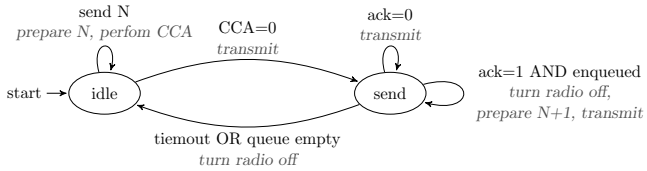
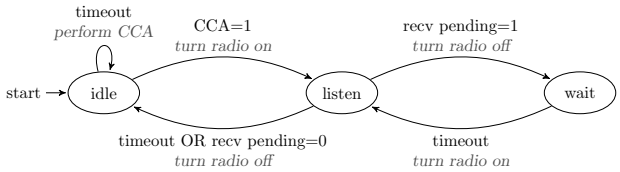


Figure 2. Packet bursting with ContikiMAC. After the wakeup phase, every packet is repeatedly sent until reception of a link-layer acknowledgment. The next consecutive packet in the burst is then transmitted.



(a) Sender



(b) Receiver

Figure 3. State diagrams of our duty cycling layer. Both sender and receiver turn their radio off between every packet of a burst.

The sender and receiver behavior is described by the state diagrams in Figure 3. We perform clear channel assessments (CCA) only before starting a burst, but not for every packet during a burst. The CCA consists of consecutive channel samples with an interval guaranteeing to find an ongoing burst. This mechanism shortens the inter-packet interval and makes long lasting bursts more likely. In the case that the last packet in a burst is lost, the receiver uses a timer to leave the *listen* state and switch off the radio after a given duration.

3.1.3 Storage Interleaving

To overcome the memory limitations of existing sensor motes, burst forwarding leverages flash memory to spill over long buffer queues. To make the critical path of forwarding as efficient as possible, we interleave flash and radio operations. When receiving a burst, burst forwarding stores the previously received packet in flash while receiving the current packet. The packet queue is kept in flash until the packets are to be forwarded. When a node sends a burst, the next packet is read from flash while the present packet is being transmitted over the radio. Note that burst forwarding uses the flash memory only for long bursts; when only a few packets are to be forwarded, packets are buffered in RAM instead.

Using the flash for packet queuing inevitably consumes power, but the power consumption is low. On a Tmote Sky, where the current draw when writing to the flash is comparable to that of the radio transceiver, flash operations are shorter than radio operations. For example the writing plus

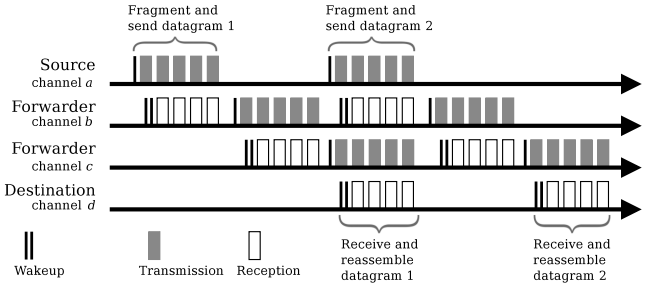


Figure 4. The bursts are coordinated with CSMA. Multi-channel operation allows transmissions pipelining over the multi-hop path. Network-layer headers are amortized with end-to-end fragmentation.

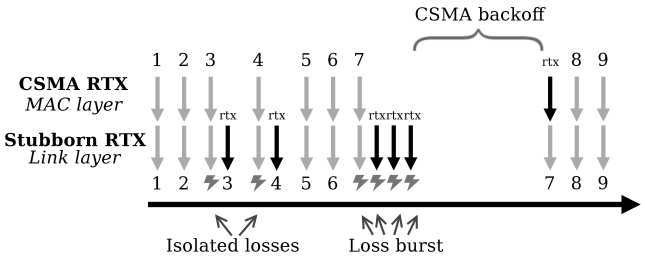


Figure 5. Retransmissions by the stubborn link layer are performed consecutively and atomically to recover from isolated losses. MAC-layer retransmissions, using increasing backoff, adapt to longer loss periods.

reading of a packet in flash has an energy cost of 0.101 mJ, whereas the reception plus transmission of a packet costs 0.523 mJ. Moreover, radio operations need to be repeated if packets are lost, making communications even more expensive. In the experiment conduced in Section 7.1, the worst-case power consumption of the flash memory we witnessed was 8% of the total power consumption.

3.2 Burst Coordination

Low-power packet bursting provides single-hop high throughput with low power consumption on timescales of seconds. To coordinate the bursts across multiple hops and on longer timescales, burst forwarding uses a two-level retransmission mechanism, network-layer fragmentation, and multi-channel operation. Burst coordination in burst forwarding is illustrated in Figure 4.

3.2.1 Two-Level Retransmissions

Wireless sensor networks are lossy and loss rates may vary over large time scales. Packet losses occur because of a number of reasons, such as interference from other radio sources, propagation loss, self-interference, cross-traffic, and packet collisions. Coping with loss is a difficult problem for any sensor network protocol. We address this problem by a stubborn link layer—rapid link-layer retransmissions—in combination with less rapid MAC-layer retransmissions, as illustrated in Figure 5.

The first level of retransmissions is the stubborn one. It works at the link layer and consists of rapid transmissions of packets, in the same way wakeup transmissions are done in ContikiMAC. The retransmission is atomic: no other packet is transmitted or received between the retransmissions. The transmissions leverage the fact that the message can be stored in the radio transceiver and sent without the need to transfer the message to the transceiver for every transmission.

The aim of the link-layer retransmissions is to recover from isolated losses and high-frequency radio interference, such as the ones generated by a microwave oven [3]. When link-layer retransmissions are successful, the current packet burst is not interrupted. When no acknowledgment is received within a given fixed period, the current burst is stopped and the second layer of retransmissions is involved.

The second level of retransmissions operates at the MAC layer, at the granularity of bursts rather than individual packets. It consists of a classical CSMA with increasing backoff, which is known to provide a good trade-off between energy-efficiency and latency. Before starting a burst, a CCA is performed. When link-layer retransmissions fail to transmit a packet, the present burst is stopped, and a loss is reported to CSMA. An attempt to resume the burst will be made after a backoff. No other packet towards the same neighbor will be sent until this backoff expires, which ensures that all packets are always transmitted in order, for a better interaction with upper layer protocols such as fragmentation and reliable streaming.

Thanks to the increasing backoff, the CSMA retransmissions adapt to long-lasting or low-frequency interference, such as the ones emitted by a WiFi station [3]. When, after a number of retransmissions, a packet eventually reaches its target, it initiates a new packet burst, leveraging the local good quality of the radio medium. After a given number of attempts, the MAC layer drops the packet, leaving to upper layers the task of performing end-to-end retransmissions in case reliability is needed.

3.2.2 Fragmentation

Burst forwarding uses datagram fragmentation to amortize the cost of network-layer headers across a number of link-layer frames. In the context of IPv6-based sensor networks, we use 6lowpan fragmentation [21].

To save time and energy throughout the path, we do not fragment and reassemble at every hop; instead, we fragment only at the IP source and reassemble only at the IP destination. The network layer addresses are carried only by the first fragment and are cached by forwarding nodes to route the following fragments. At the end node, the received fragments are stored in the flash while any remaining packet burst is still being received. This allows reception of long packet bursts containing several network-layer packets, thus preserving the energy-efficiency of burst forwarding. At the end node and after the reception of the full burst, packets a read from the flash, reassembled and passed to the upper layer. This end-to-end fragmentation also makes datagram sizes independent of burst duration: a datagram can span multiple bursts and a burst can contain multiple datagrams.

The use of fragmentation poses a basic trade-off between speed and reliability. Fragmentation is known to be problematic in lossy networks [13] because the loss of a single fragment may result in the loss of a larger datagram. Our hypothesis is that our two-level retransmissions mechanism will provide enough reliability for fragmentation to be ben-

eficial. We show that this hypothesis holds through experiments in Section 6.3.

3.2.3 Multi-Channel Operation

In spirit of recent work aiming at reaching high throughput over multi-hop sensornets [24, 27], burst forwarding uses multiple channels to avoid both intra-path interference between nodes involved in the same transfer and inter-path interference between nodes or networks having independent activities. This mechanism allows burst transmissions to be pipelined over multiple hops. When the density of the network allows multi-channel to avoid all intra-path interference, a full pipeline takes place involving all nodes in the forwarding path: node n receives a burst from $n-1$, then it forwards the enqueued data to $n+1$ while $n-1$ acquires new data. This case is exemplified in Figure 4.

Optimal network-wide channel allocation is a hard problem [19] and is out of scope for this paper. In our current implementation, we use a fixed reception channel for every node. When a mote needs to send a packet to a neighbor it doesn't know, it repeatedly sends it on all possible channels. After reception of the link-layer acknowledgment, it remembers the correct channel, as it does with the timing for the phase-lock technique. After this first costly transmission, all following unicast messages are sent without extra overhead, and with all the benefits of using multiple-channels.

4 TCP for Low-Power Wireless

TCP is the de facto reliable data transfer protocol for IP networks. Our aim with burst forwarding is to provide a generic mechanism for supporting transport-layer stream protocols, including TCP. Leveraging burst forwarding to bring TCP to low-power wireless networks is a challenge due to the well-known problems with TCP in wireless networks.

Using TCP with burst forwarding has numerous benefits, however. We can leverage TCP's flow control and congestion control mechanisms to avoid network overload. As a consequence, the combination of TCP with burst forwarding could provide reliable and energy-efficient transmission over varying network capacity, even in the face of the problems of TCP in low-power wireless.

TCP was designed under the loss assumptions in the traditional wired Internet and is known to be ill-suited for use over lossy links [2] and wireless networks. There are at least four reasons for this: TCP treats packet loss as congestion, which leads to suboptimal throughput over lossy links; TCP headers are large, which is a problem over low-bandwidth links; TCP uses positive acknowledgments, which may overwhelm the wireless medium; and TCP timing may interfere with link-layer recovery mechanisms.

The loss-is-congestion problem. The most important weakness of TCP over lossy links is probably the interpretations of segment losses as congestion, triggering an unnecessary significant reduction in the transmission rate. First, our two-level retransmissions system is designed to recover different types of interference. Recovering part of the losses at the link layer is a way to hide losses to TCP and avoid drastic throughput reduction. Second, we argue that adapting the output rate to link losses is not a completely wrong reaction; in fact, long-lasting interference may be the result of a congested network. In this case, it makes sense to use traditional congestion control. This observation holds as well for internetworks interference, including WiFi networks which can be interfered by 802.15.4 traffic [18]. Third, we argue that this rate adaptation, together with duty cycling, is the key to energy-efficient transmission under interference.

The TCP header overhead problem. TCP involves a 20 byte overhead for every segment, to which one can add 20 to 40 bytes for the underlying IP layer. When used over 802.15.4 links, this TCP/IP header can represent more than half of the packet payload. Using large TCP segments together with link-layer fragmentation solves this issue but is often considered as unsuitable over lossy links. Our evaluation in Section 6.3 shows that fragmentation is suitable even over extremely lossy links (link loss rate over 80%) given that enough link-layer retransmissions are performed.

The TCP ACK implosion problem. TCP provides end-to-end reliability via end-to-end acknowledgments and retransmissions. Most implementations send an acknowledgment every second received segment. The resulting extra traffic, whose direction is opposed to the main data transfer, is a source of congestion and collisions in wireless sensor networks. Most reliable high-throughput mechanisms for sensornets avoid this issue by using delayed negative acknowledgments instead of streamed acknowledgments. The use of large segments and extensively fragmented traffic reduces the impact of this problem. For instance, segments of 1.5 kB result in 16 fragments; with this setting, an acknowledgment is only sent after the reception of 32 link-layer frames.

The too-persistent link layer problem. We argue that the use of our two-level retransmissions with many tries is the enabler for TCP in wireless sensor networks. Such a persistent link layer may, however, have a negative impact on TCP, because it increases round trip time variation which may lead to redundant end-to-end retransmissions. We argue that these drawbacks are compensated by the benefits of having large TCP segments and hiding isolated link losses.

In Section 7 we use burst forwarding as the underlying forwarding layer for TCP and demonstrate that this combination provides a favorable trade-off between throughput and energy efficiency, even over high-loss wireless links.

5 Implementation

We have implemented burst forwarding in the Contiki operating system for the Tmote Sky hardware platform. We needed to instrument the CC2420 radio driver, the ContikiMAC radio duty cycling module, the CSMA MAC layer module, and the 6lowpan IPv6 forwarding module. The memory we use for packet queues is the 1 MB external Flash embedded in the Tmote Sky. Overall, the modifications amount to a few hundred lines of code, and could easily be adapted to another operating system for small devices.

5.1 Timing

Our burst forwarding implementation uses careful timing in ContikiMAC bursting, inter-packet sleeping, and storage interleaving. ContikiMAC burst timing depends on the speed of the link layer. The maximum rate of ContikiMAC for

sending full-length (127 bytes) packets on a Tmote Sky is 208 packet/s.

Storage interleaving timing is dependent on the timing of the flash memory. We measured the timing obtained when reading from flash and sending consecutive full-sized packets on a Tmote Sky. Packet transmission and acknowledgment reception take 4.7 ms, reading from flash takes 1.9 ms, packet construction and copying the packet to the radio transceiver takes 1.2 ms (on the CC2420 chip, an ongoing transmission must complete before the next packet can be loaded). By interleaving flash accesses with radio communication, we eliminate the timing overhead of using the flash. The interval between two packets is reduced from 4.9 ms to 3 ms, resulting in a rate of 130 packets/s. Our inter-packet sleeping technique is parametrized based on the measured inter-packet interval.

5.2 Memory Footprint

Our burst forwarding implementation extended the Contiki code size by 7.8 kB. Although we use external flash for packet buffer storage, we store the packets meta-data in RAM. This meta-data storage is the only significant source of RAM usage. In our implementation, every packet consumes 8 bytes in RAM for forwarding nodes, and an extra 7 bytes for end-nodes because of datagram reassembly.

In order to leave space to applications, we limit the queue size to 256 packets in all the experiments presented, thus consuming 3.8 kB of memory. Note that a 256 packet burst lasts 2.25 seconds when no losses occur. Our evaluation (Section 6.2) shows that 2 second bursts are enough to provide near-optimal throughput even with low duty cycling wakeup frequency.

5.3 Reimplementation of PIP

To provide fair comparison with the state of the art, we re-implemented the PIP bulk transfer protocol [27] for Contiki. PIP is the fastest reliable bulk transfer protocol for sensornets. It uses multiple-channels to avoid intra-path interference and conditional immediate transmission to improve throughput. Our PIP implementation is simplified in the sense that it uses static channel allocation rather than a dynamic path construction, but it includes all the functionality of the transfer phase, which is the focus of our experiments.

Our PIP implementation is 16% faster than the results published by Raman et al.: it reaches a throughput of 73 kbps whereas the original PIP paper reports a throughput of 63 kbps [27]. We believe this improvement may be due to our implementation using more accurate timing. Our implementation was developed with the help of the Contiki simulation environment [25], which provides cycle-accurate simulation of sensor motes and of the radio medium activity, making it easy to implement highly precise timing.

6 Evaluation

We conduct a series of experiments to assess burst forwarding. We investigate the impact of burst size and radio duty cycling periods, and evaluate the different mechanisms we presented in Section 3. Then, we tackle the problem of fragmented traffic forwarding over extremely lossy links (link loss rate over 80%). We show that our two-level retransmission mechanism provides enough reliability to support extensively fragmented traffic when facing intensive microwave oven and WiFi interference. We also show that burst forwarding can support cross-traffic and be used with little impact in conjunction with data collection, providing a significant advantage over the existing single-purpose high-throughput protocols, which require all other traffic to stop during the bulk transfer.

6.1 Methodology

We primarily use testbed experiments to evaluate burst forwarding. We compare burst forwarding to the state-of-the-art PIP protocol using our re-implementation. We measure energy-efficiency with Contiki's integrated power profiler [5]. To create losses and interference, we use the JamLab tool that can generate and replay interference from WiFi sources and microwave ovens [3]. In situations where our testbed setup is too coarse-grained, we use simulation with the Contiki simulation environment.

6.1.1 Testbed Setup

We use a small testbed with 11 Tmote Sky motes to obtain our experimental results. The Tmote Sky is based on the MSP430 16 bit CPU running at 3.9 MHz and a CC2420 radio chip. It provides 10 kB of RAM, 48 kB of on-chip ROM and as a 1 MB external Flash.

To measure the energy consumed by the sensor motes during our experiments, we use the Contiki Powertrace built-in power profiler [5, 7]. Contiki Powertrace uses power state tracking to estimate the energy consumed by the hardware components of the mote. In order to provide a fair comparison of different protocols independently of specific battery properties and environmental setting, we consider a constant voltage of 3 V. Note that in the specific case of the Tmote Sky, writing on the external flash requires a minimum voltage of 2.7 V. A real deployment would either use hardware that does not suffer from this limitation, or would stop queuing packets in flash as soon as the battery voltage becomes too low.

In a few cases hardware limitations require us to use simulation to obtain our results. We then use the Contiki simulation environment, which provides a cycle-accurate emulation of the Tmote Sky mote with bit-accurate network simulation. In the simulator, we run the exact same binary image as in the testbed.

Unless explicitly mentioned, we always use the following settings for burst forwarding: the burst duration is 2 seconds, the low-power listening is performed 8 times per second and the queue has a capacity of 256 packets. We show in Section 6.2 that these settings provide an arguably good trade-off between energy and throughput. We use a fixed topology providing fine control of the number of hops and removing the overhead of routing algorithms. All results presented in the graphs are averaged over 10 runs, error bars show the standard deviation.

6.1.2 Controlled Interference Generation

In order to evaluate the impact of interference in a reliable fashion, we use JamLab controlled interference generation by Boano et al. [3] for both experiments on motes and cycle-accurate bit-accurate simulations. In the latter case, interference is replayed from trace files containing RSSI recorded by

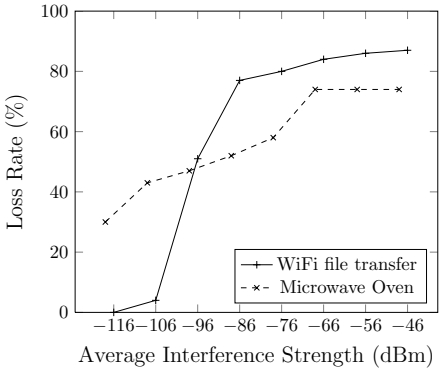
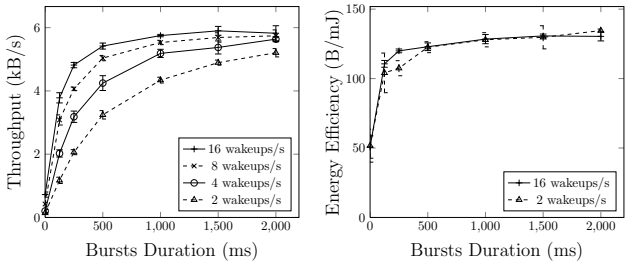


Figure 6. The measured loss rate of full 802.15.4 frames as a function of interference generation strength both with the microwave oven and the WiFi transfer models.



(a) Throughput (b) Energy-efficiency

Figure 7. The length of the burst period is essential to provide a high throughput (left) as the throughput increases even with a high wakeup period. The impact of burst length on energy efficiency (right) is not as evident.

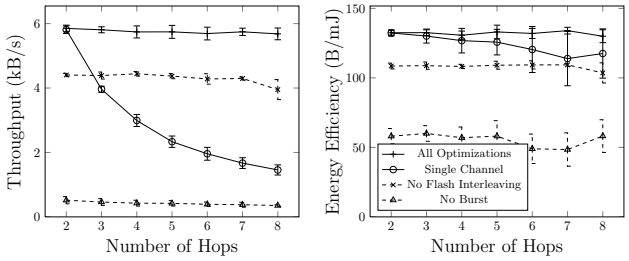
a mote located close to an interference source. We use two different traces provided by Boano et al.. The first one contains the noise level recorded close to an active microwave oven, the second one has been recorded at a WiFi station while performing a file transfer. Microwave ovens generate regular low-frequency interference (about 50 Hz), in an all-or-none manner. WiFi generates a more irregular pattern, alternating variable length loss periods and non-loss periods.

To control the strength of the interference, we generate new trace files by adding different gains to the recorded signal strength. The resulting trace files have an average RSSI on the interfered node ranging from -118 to -46 dBm. Note that a trace with an average strength of -118 dBm – less than the sensitivity of the CC2420 radio chip – may still cause losses because it contains spikes. Figure 6 shows the loss rate depending on the average interference strength, for both WiFi and microwave oven, when using full-size 802.15.4 packets. The loss rate ranges from 0% to 87%.

We also use JamLab interference generation from Tmote Sky nodes, allowing the control of noise levels in testbed experiments. The WiFi emulator uses the Garetto model to emulate interference generated by a router with 25 hosts, resulting in a loss rate of 81%. For more details about the interference pattern or about the regeneration technique, we refer the reader to the original JamLab paper [3].

6.2 Packet Burst Efficiency

We first evaluate the efficiency of low-power packet bursting over a duty-cycled radio. Figure 7 shows the throughput (at the MAC layer) and energy efficiency depending on the burst duration, ranging from no burst to 2 second bursts, and for various wakeup periods. The measurements were done on a 4-hops path in our testbed. This experiment shows that high-throughput requires very long bursts, on the order of seconds. The longer the wakeup period, the longer the bursts needed to amortize the time spent synchronizing with the receiver. In terms of energy efficiency, bursts of a few hundreds of milliseconds are enough to amortize wakeup costs. The wakeup period, which impacts the steady-state power consumption, has little effect on energy efficiency during a transmission because fewer wakeups are needed during a transmission.

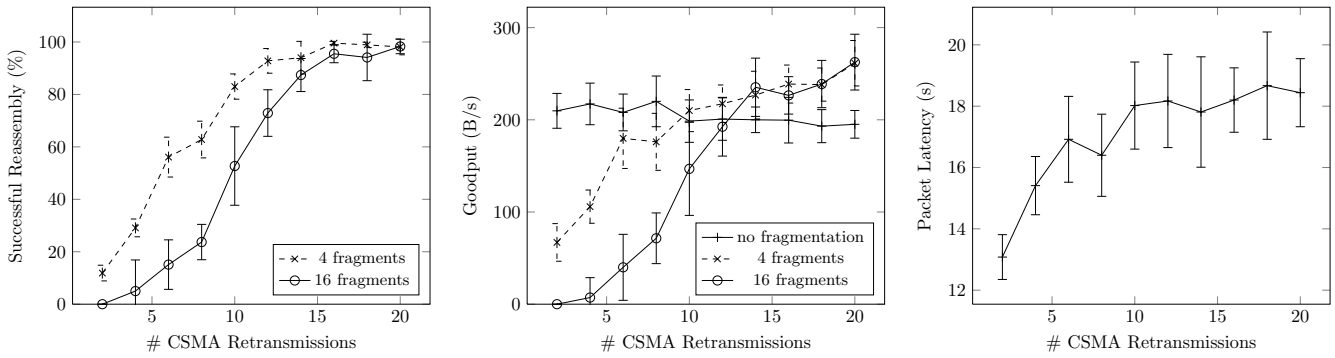


(a) Throughput (b) Energy-efficiency

Figure 8. The relative contribution to throughput and energy efficiency from the individual mechanisms.

In absolute terms, the maximum throughput reached by burst forwarding is 5.8 kB/s, i.e. about 20% slower than PIP. The reason for this difference is that burst forwarding forwards data with the granularity of bursts rather than packets. Consequently, it cannot use the conditional immediate transmissions mechanism [24] that PIP uses [27]. Rather, it has to copy the packet to the radio transceiver between every transmission, because the CC2420 chip does not allow a packet to be loaded while sending another one.

We conduct an experiment to measure the relative contribution to throughput and energy efficiency of multiple channels, storage interleaving and the use of bursting. We run a file transfer in our testbed over 2 to 8 hops. In this experiment, all nodes are in range of each other. In the general case, such a setup would have the drawback of artificially eliminating the hidden-terminal problem; in our case, this problem does not occur since nodes are forwarding data over multiple channels. The results, in Figure 8, show that multi-channel operation improves both throughput and energy efficiency. The improvement becomes more important as the number of hops increases, because of the pipelining allowed throughout the network path. Storage interleaving provides a constant improvement in throughput (about 32%), and in energy (about 22%). Overall, the use of packet bursts has the highest impact on both throughput and energy.



(a) Reassembly (b) Goodput (c) Latency

Figure 9. The performance of fragmentation under heavy WiFi interference with 81% loss rate. Fragmentation can be used even under interference given enough retransmissions at the CSMA level. The use of many fragments and many retransmissions nevertheless impacts message latency.

6.3 Fragmentation under Interference

In order to achieve high goodput, it is tempting to use large network-layer packets together with fragmentation because fragmentation amortizes the cost of packet headers across multiple frames. Fragmentation is, however, often recommended not to be used in lossy environments because every single frame loss involves full network-layer packet retransmission, either in a hop-by-hop or end-to-end scope [13]. In light of this, our hypothesis is that the two-level retransmissions in burst forwarding may reduce losses to a level where fragmentation is nevertheless useful.

We investigate the behavior of fragmentation in a highly interfered environment by extending our testbed with a WiFi interference generator. We run a 4-hops bulk transfer in which one mote on the path is exposed to the interferer, having an average loss rate of 81% on full-size frames. We use IP payloads ranging from 76 to 1512 bytes resulting in 1 to 16 fragments par datagram. Figure 9 shows the results obtained as a function of the number of CSMA retransmissions. The underlying stubborn link layer always used 4 transmissions; we leave to Section 6.4 the investigation of interactions between MAC and link-layer retransmissions.

As expected, the rate of successful reassembly grows with the number of CSMA retransmissions. With few retransmissions the successful reception of consecutive fragments is unlikely, while infinite retransmissions guarantee that all datagrams are completely forwarded. As a direct consequence, we reach the highest goodput (throughput at the IP layer) by using extensively fragmented datagrams (large payload) with many retransmissions (successful reassembly).

In the non-fragmented case, we observe a decrease in goodput as we use more retransmissions. This is a consequence of CSMA's increasing backoff: with more retransmissions, the queue at intermediate nodes overflows, making forwarding less successful, increasing the transmission backoff, and eventually decreasing the reactivity at every hop. Another side-effect of having a persistent CSMA is an increase in latency. Our conclusion is that we would not recommend to systematically use many CSMA retransmissions, but rather to adapt the MAC layer to the expected network needs. While some applications need to carry small data with low latency, others would benefit from larger payloads and increased goodput.

6.4 Retransmissions under Interference

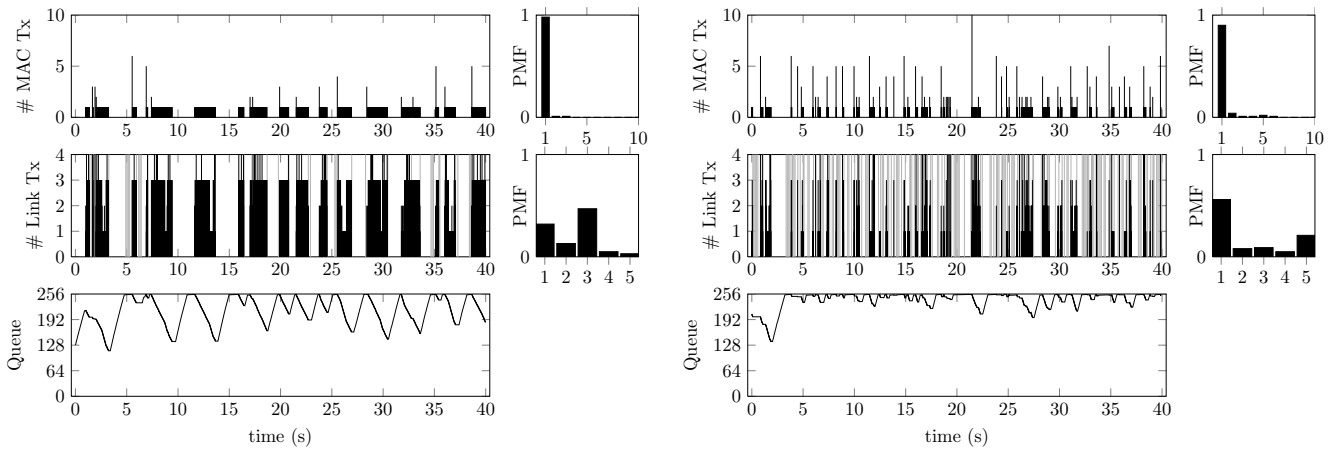
We next investigate how our two-level retransmissions mechanism handles different types of interference. We use interference replay from trace files and run a bulk transfer over burst forwarding in the Contiki simulation environment. We use two interference generators, as shown in Figure 6, with different patterns but similar signal strength and loss rate:

Microwave oven the average RSSI at the receiver is -98 dBm, resulting in 52% link losses. The interference pattern is regular, resulting in 50 Hz spikes;

WiFi file transfer the average RSSI at the receiver is 96 dBm, resulting in 51% link losses. The interference has periods with high loss and periods with low loss, with each period having varying duration and strength.

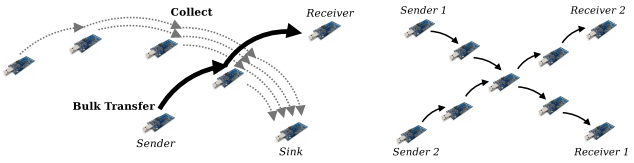
Figure 10(a) shows the behavior of the node sending to a mote in range of the microwave oven. At the MAC layer, we observe the pattern of packet bursts, resulting in consecutive successful CSMA transmissions. At the link layer, the bursts can again be identified, although only 68% of the packets need fast retransmission (see the Probability Mass Function, PMF, next to the timeline). Only 3% of the packets fail at this layer, stopping the current burst and requiring a CSMA retransmission. This is due to the regular interference pattern, triggering many isolated losses but rarely triggering 4 consecutive retransmissions. We also plot the queue occupancy at the sender. It is highly varying: it fills during reception of a burst, and empties during transmission. The overall throughput for the transfer was 2.9 kB/s.

Figure 10(b) shows the result of the same experiment in the case of the WiFi interferer. This time, packet bursts are hardly identifiable. At the link layer, 56% of the packets do not need retransmission, but 21% the packets are dropped. Packet bursts are more often interrupted, and more CSMA retransmissions occur (maximum of 10 against 6 in the microwave experiment). This is due to the irregular nature of



(a) Microwave oven (b) WiFi file transfer

Figure 10. Adaptation of the stubborn link layer and MAC-layer retransmissions to microwave oven and WiFi interferer triggering more than 50% packet losses. The timeline and distribution of link-layer and MAC-layer retransmission are shown, as well as the queue occupancy at the node whose transmissions are interfered. High-frequency interference is mainly handled by the fast strobes. Low-frequency interference disrupts the bursts and requires more MAC-layer retransmissions.



(a) Concurrent Applications (b) Cross-Traffic

Figure 11. Testbed setups used to evaluate concurrent applications and cross-traffic.

WiFi interference. Note that a few consecutive packets are sometime sent as a burst, when the link quality is locally good. The queue at the sender is almost always full, because successful bursts are rare. Although the loss rate was the same as in the microwave case, the resulting throughput, 1.6 kB/s, is not as high, because of the lack of packet bursts.

These two experiments show that the two level of retransmission are complementary and allow burst forwarding to adapt to different interference patterns. The stubborn link layer allows the packets burst to keep working in case of isolated losses of high-frequency interference, while the MAC layer retransmissions cope with periodic loss patterns.

6.5 Cross-Traffic: Data Collection and Bulk Transfer

In a multi-purpose network, a bulk transfer protocol cannot monopolize all network resources. Burst forwarding has been explicitly designed to allow cross traffic and concurrent applications, while being power efficient. To evaluate the performance of burst forwarding with cross traffic and concurrent applications, we run two testbed experiments, with the configuration shown in Figure 11.

The first experiment involves a 4-hop data collection tree crossed by a bulk transfer of 240 kB. The collection has a

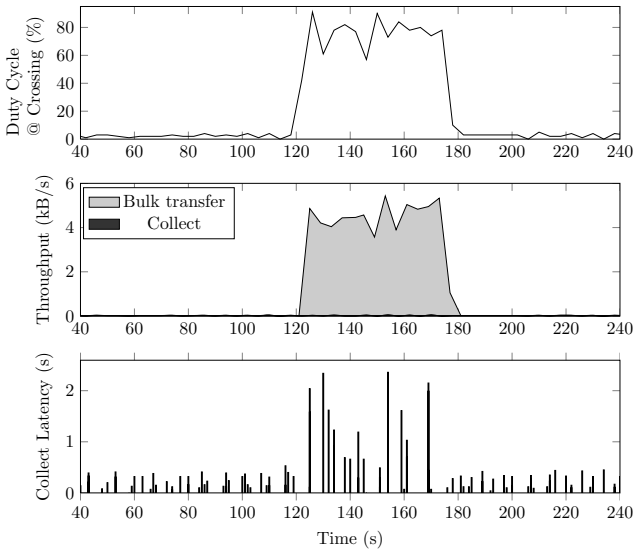


Figure 12. The radio duty cycle at the crossing node quickly adapts from less than 4% to about 80% during the transfer. The throughput of the transfer is 82% of the isolated stream case, and the latency of the data collection is increased by about 2 seconds during the transfer.

random period between 5 and 7.5 seconds. Figure 12 shows the interaction between the collection and the bulk transfer. The radio duty cycle at the crossing node is less than 4% during the collection, and rises to reach about 80% for the duration of the bulk transfer. This shows that the duty cycle follows the activity of the radio medium, which is intensively used when both protocols are simultaneously active. The average throughput of the transfer is 4.6 kB/s, 82% of the throughput obtained with a single stream. The col-

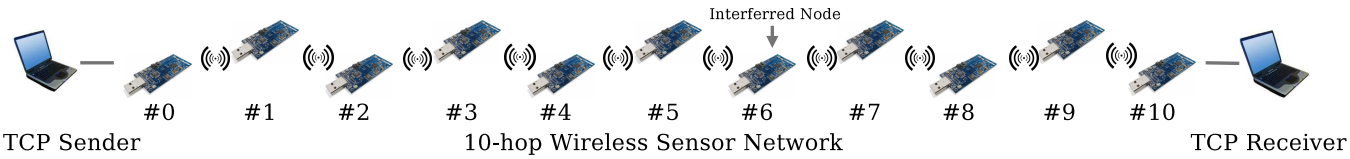


Figure 14. Our tunnel setup allows us to run TCP traffic from a full-scale TCP/IP implementation over burst forwarding. Both the client and the server run Linux and transfer their data across the multi-hop sensor network.

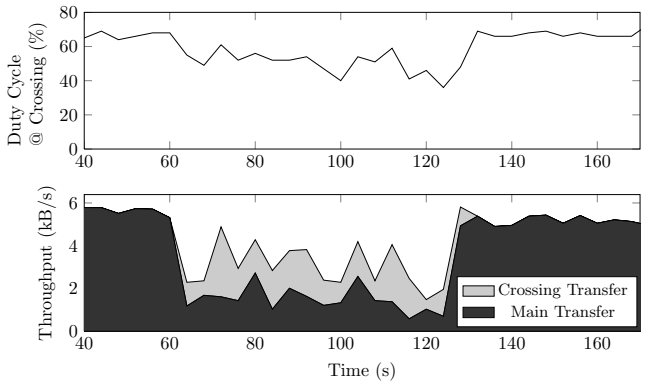


Figure 13. Burst forwarding allows multiple bulk transfers to co-exist. Although the throughput of both transfers decreases, so does the radio duty cycle of the intersecting node.

lection does not suffer from any loss and its throughput is not affected by the burst. The only noticeable impact on the data collection is a latency increase about 2 seconds, which is negligible for most applications.

This testbed experiment shows that burst forwarding allows bulk transfer protocols to co-exist with non-bulk transfer protocols, with little negative impact on either protocol. This is unlike existing state of the art high-throughput protocols, which dedicate a path for a single stream, stopping all other applications in the network.

The second experiment involves two crossing bulk transfers over 4-hop paths. The first bulk transfer is continuous and the second finite, with 100 kB of data. We measure the throughput for both streams as well as the radio duty cycle of the intersection node. The experiment runs for 170 seconds and the second stream starts after 60 seconds. Figure 13 shows the result. Before the start of the second stream, the first transfer has a mean throughput of 5.6 kB/s and the intersection node has a radio duty cycle of 67%. When the second stream starts, the throughput of both streams drops down, reaching respectively 1.5 kB/s and 1.6 kB/s. Meanwhile, the duty cycle of the intersection node also drops, reaching 50% in average. When the second transfer terminates, the first transfer quickly reaches its full speed again.

7 TCP over Burst Forwarding

Burst forwarding is intended to be a generic forwarding layer for high-throughput and low-power traffic in lossy networks. We use TCP, as the most commonly used transport protocol in the IP family, to see how well burst forwarding can sustain high throughput and low power consumption in the face of radio interference for TCP flows. We study the effect of TCP congestion control and compare the results with the PIP state-of-the-art bulk transfer protocol for wireless sensor networks [27].

We evaluate TCP over burst forwarding with the experimental setup shown in Figure 14, in which a wireless sensor network forwards TCP traffic produced by two computers running Linux and its full-scale TCP/IP implementation. A WiFi interference generator implemented with JamLab is set to disturb one specific node in the path, node #6. The mean noise is -76 dBm, giving a mean link loss rate of 80%.

We first analyze the system behavior and its capability to adapt to interference. Figure 15 shows different timelines monitoring the TCP connection at the sender (computer), the transmissions in the sensor network and queue size of different motes. We use a cycle-accurate simulation of 500 seconds. We enable the WiFi interference generation from $t_1 = 60$ s to $t_2 = 300$ s. The average RSSI at the interfered mote is -76 dBm, resulting in a link loss rate of 80%. For this specific experiment, we reduced the motes queue capacity from 256 to 64 packets, resulting in a faster adaptation of the end-to-end mechanisms, but negatively impacting the throughput of burst forwarding. The resulting average TCP goodput before and after interference was 1488 B/s, and 151 B/s during the interference.

The topmost graph in Figure 15 shows the TCP congestion window at the sender. In the beginning of the experiment, the window grows and stabilizes between 20 and 40 segments. When the interference starts, the window decreases and stabilizes at about 30 segments. When we stop the interference, the window grows quickly, generates some congestion, and stabilizes a few dozen of seconds later. The second graph shows SRTT, the Round-Trip Time estimation made by the TCP sender. The losses trigger a slow growth of SRTT, which quickly decreases at the end of the interference, at it is supposed to do.

The third graph shows the transmissions at the TCP level. End-to-end retransmissions are slightly increased by the interference, due to variations in the round trip caused by link losses. Thanks to the reliability of our two-level retransmissions, we never observed more than 3 TCP transmissions of the same segment. The fourth graph shows link-layer transmissions at the CSMA level. Before and after interference, the transmissions are dense and most packets do not need retransmissions. During the interference period, the transmissions are quite sparse, and most packets need retransmission.

The two last graphs show the queue occupancy at mote #5 (sending to #6) and #6 (the mote under interference). During the interference period, the queue at #5 saturates (because #6 is hard to reach), triggering a slow-down in the TCP out-

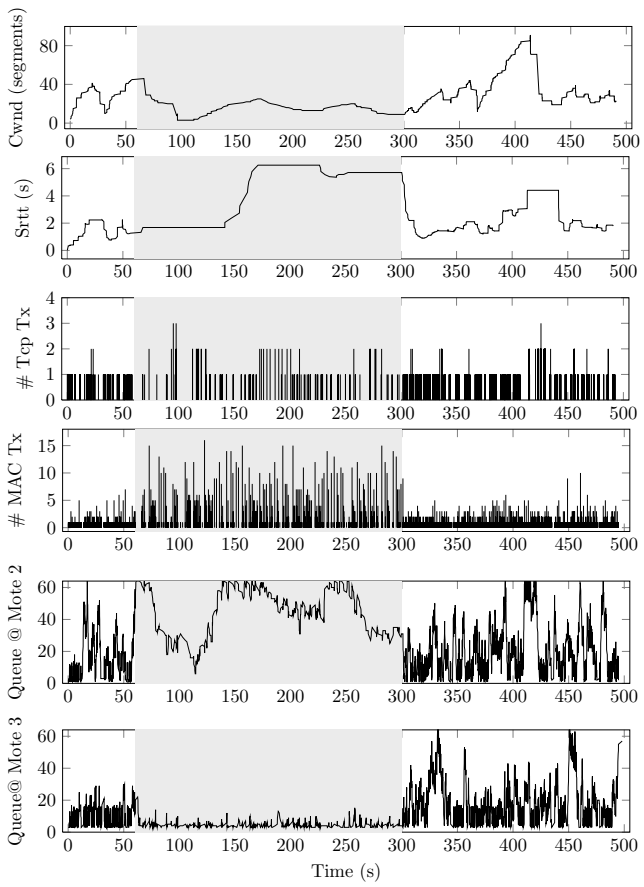


Figure 15. Behavior of the system with WiFi interference occurring from second 60 to 300 (link loss rate of 80%). Even a standard TCP implementation adapts accordingly to temporary interference over a multi-hop duty-cycled network using burst forwarding, resulting in a reasonable amount of end-to-end retransmissions.

put rate (see the congestion control graph). The queue occupancy keeps increasing and decreasing as the TCP sending rate adapts. Out of interference periods, we can observe the same adaptation at a higher rate. Finally, the queue of node #6 is almost empty during the interference, because the link rate between #5 and #6 is lowered.

Our experiment shows that even in an intensively interfered environment, TCP behaves properly, adapting its sending rate to the level of interference. During the interference, link-layer retransmissions are extensively used, triggering only a low amount of end-to-end retransmissions: with 80% link loss, only 8% of the TCP segments needed end-to-end retransmissions.

7.1 TCP Congestion Control and Energy Efficiency

We now take a closer look at the energy consumed during a TCP transmission over burst forwarding in a lossy environment. We use our 11-mote testbed to get a 10 hop path with WiFi interference at node #6 (Figure 14). Figure 16 shows the average power consumption for all hardware components of every node in the path, except for the two end motes, which are powered by the computers.

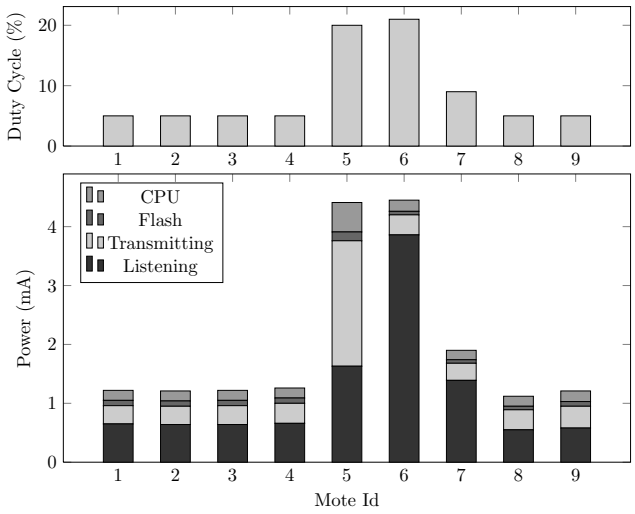


Figure 16. Average duty cycle and current draw per component of every mote in the path, where mote #6 is subject to WiFi interference (link loss rate of 80%). Thanks to TCP rate control, nodes at two hops or more of the interference are spending most of their time in low-power listening, thus saving a significant amount of energy.

The node under interference and the one preceding it consume more energy than the other nodes in the path. Node #6 spends most of its energy in listening, because ContikiMAC triggers redundant wakeups due to the ambient noise and because received packets mostly are corrupt, which requires them to be retransmitted. Node #5 spends most of its energy in transmission, because it keeps retransmitting packets to #6. The time spent in active waiting in radio duty cycling layer also involves a rise of CPU activity at node #5. Node #7 is slightly impacted by the interference, because the link-layer acknowledgments it sends to #6 are sometimes lost, triggering (unnecessary) retransmissions by #6.

All other nodes in the path have a low power consumption, with a radio duty cycle of about 4%, against 21% at node #6. This is the result of TCP rate control, which avoids forwarding of data that will eventually be lost because of queue overflows. Note that node #8 and #9 have a slightly increased energy consumption; this is because some packet bursts are broken by the interference, leading to a slightly less energy-efficient forwarding after the lossy zone. Another result from this experiment is the characterization of the power penalty involved by queuing in flash: the external flash is responsible for 1% to 8% of the overall consumption.

7.2 Comparing Reliable Transport Protocols

To demonstrate the energy-efficiency of rate controlled traffic over a duty-cycled network, we compare TCP over burst forwarding to the PIP reliable bulk transfer protocol for sensornets and to *Snack*, a simple PIP-like protocol running over UDP. PIP does not use rate control, but always transmits at full rate. Also, PIP nodes never turn off their radios during a transfer. Instead, the protocol attempts to complete

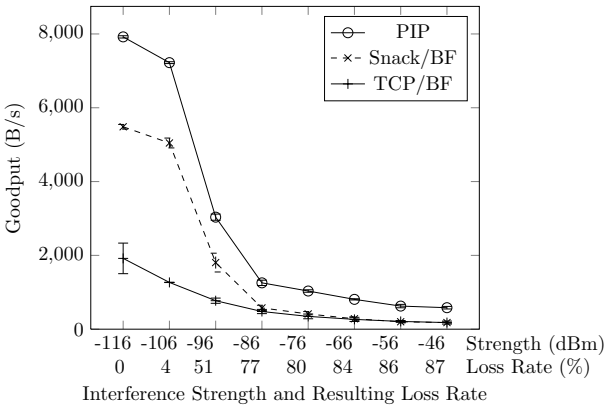


Figure 17. PIP uses an always-on radio layer and achieves a higher throughput than Snack or TCP over burst forwarding.

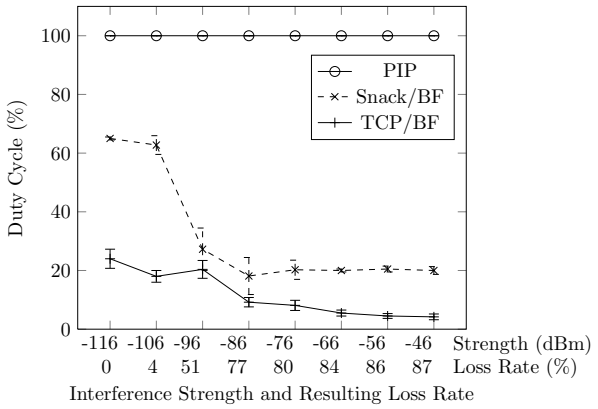


Figure 18. The radio duty cycle of PIP is constant and significantly higher than that of PIP and Snack over burst forwarding, as PIP uses an always-on radio. As interference increases, Snack's duty cycle decreases until it reaches a lower bound. Due to TCP's congestion control, TCP/BF keeps reducing its duty cycle even under extreme levels of interference.

the transfer as quickly as possible so that the nodes can go back to sleep when the transfer is complete.

To isolate the effects of burst forwarding and of TCP, we have designed a simple transport protocol – called Snack – running over UDP/IP that provides end-to-end recovery similar to that of PIP. Like PIP, Snack provides end-to-end reliability via negative acknowledgments (NACKs), thus implying less traffic overhead than the continuous positive acknowledgment mechanism of TCP. Like PIP, Snack does not use rate control, meaning that the traffic source keeps sending at maximum rate until all data are received.

To have a fine-grained control of the interference strength, we use the Contiki simulation environment and replay WiFi traces with strength ranging from -116 to -46 dBm, resulting in loss rates between 0% and 87% (Figure 6). To allow for a fair comparison, we also use the tunnel setting for PIP, meaning that the end nodes need to read and write all data from and to the motes serial line, resulting in a slightly reduced performance for PIP: 64 kbps instead of 73 kbps, which is still faster than the original PIP implementation by Raman et al. [27].

Figure 17 shows that for all observed levels of interference, PIP achieves a higher goodput (measured at the application layer) than Snack and TCP over burst forwarding (referred to as BF in the figures). This is because PIP keeps its radio on all time and always sends at maximum rate at every hop. Over good quality channels, TCP is substantially slower than Snack because its rate control mechanism makes suboptimal use of the available link capacity.

Figure 18 shows the average duty cycle at a node before the interference zone. Unlike PIP, the use of burst forwarding for Snack and TCP allow them to adapt their radio duty cycle to the interference strength. For Snack, this adaptation is due to network congestion caused by the saturation of buffer queues. As the congested mote has no more memory available, it stops receiving bursts from its previous hop. This behavior triggers a back pressure reaching the sender, which ultimately slows down its transmission rate. However, because Snack does not include rate control, all motes in the path keep trying to transmit. For this reason, as the interference reaches a given threshold (about -86 dBm in our experiment), the duty cycle of Snack stagnates. In contrast to PIP and Snack, TCP generates rate-controlled traffic. As a reaction to losses and increased round-trip, caused by congestion in the network, the output rate at the transmitter is decreased. Fewer packets are to be forwarded, thus reducing the radio duty cycle of the nodes in the path. With the strongest interference (-46 dBm, link loss rate of 87%), the duty cycle drops to 4%.

7.3 Energy Per Byte

Figure 19 shows the energy cost per byte for a forwarding node before the interference. In a lossless environment, PIP is 19% more energy-efficient than TCP, because it makes near-optimal use of the channel and does not carry TCP headers and ACKs. As the strength of interference increases, PIP's energy cost grows, reaching 99 mJ/kB in the worst case. This is a consequence of a reduced throughput together with an always-on radio. Snack is more energy efficient than PIP over lossy links, thanks to burst forwarding. This is because the congested node creates a back pressure that affects nodes throughout the path.

Unlike PIP and Snack, TCP/BF exhibits a constant energy-cost at all our observed levels of interference, because the radio activity of the forwarding nodes drops as the path capacity decreases. The increasing backoff of CSMA allow the nodes to sleep during a lossy period, and the packet bursts leverage interference-free periods with a locally high throughput. With a cost of 16 mJ/kB, TCP/BF is up to 6 times more energy-efficient than PIP, leading to an expected lifespan expansion in the same order of magnitude. This experiment demonstrates the good interaction between TCP's congestion control and burst forwarding's low-power packet bursting.

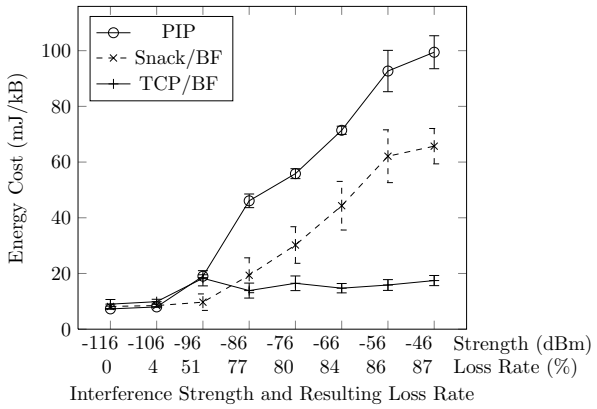


Figure 19. The energy efficiency of TCP is better than that of PIP and Snack with increasing levels of interference. As fewer packets get through, the energy efficiency of TCP/BF remains almost constant because of the rate adaptation caused by TCP's congestion control.

8 Related Work

Although wireless sensor networks are typically focused on low data rate communication, there are situations in which bulk data transfer is used and many protocols dedicated to bulk data transfer have been developed. Kim et al. [14] present Flush, a bulk data transfer protocol for multi-hop wireless networks based on hop-by-hop rate control [20], a mechanism that adapts quickly and precisely to the available link capacity. Flush uses a small packet size and single-channel operation, which results in data rates below 1 kB/s. Österlind and Dunkels [24] demonstrated that by using larger packets and multi-channel operation, the throughput of Flush-like protocols could be significantly improved. Moreover, Österlind and Dunkels highlighted the importance of optimizing the critical forwarding path and presented a packet forwarding technique called conditional immediate transmission. With this technique, a multi-hop throughput that reaches 13.6 kB/s, or 97% of the theoretical maximum IEEE 802.15.4 throughput was experimentally demonstrated. Raman et al. present PIP [27], a multi-hop bulk transfer protocol based on TDMA, multi-channel operation, and the conditional immediate transmission technique by Österlind and Dunkels. PIP reaches a throughput of 7.3 kB/s over a multi-hop path. Common to all above protocols is that they monopolize the network: the radio is turned on for all nodes on the path and no other protocols are allowed while the bulk transfer is in place. If packet losses or interference occur during the transfer, the network becomes progressively less energy efficient as radios are kept on while the throughput reduces. By contrast, our work focuses on providing a forwarding mechanism that is general enough to support different protocols, that provides low-power operation through radio duty cycling, and that does not monopolize the network resources.

Low-power communication in wireless sensor networks is an active topic through the study of radio duty cycling protocols [6, 9, 22, 23, 31]. Most work related to radio duty cycling protocols focus on reducing energy requirements when performing periodic data collection, and do not explore mechanisms to provide high-throughput, low-power forwarding. Gu and He explicitly address low-power forwarding in sensornets [11], but address individual packets routing rather than bulk data transfer. Hui and Culler investigate the performance and energy of IP-based duty cycled networks [12]. We were inspired by their use of the 802.15.4 Frame Pending bit to send bursts of packets.

The characterization of radio interference is an active topic of research [29, 30]. This understanding has previously been used in protocol conception [1], and has motivated our design of the two-level retransmission mechanism.

The lossy nature of wireless mediums has been known for a long time to be an issue for bulk data transmission. Li et al. present Hop, a bulk transfer protocol for multi-hop 802.11 networks [17]. Hop and burst forwarding have similar design points in common: they send bursts of fragmented data and use link-layer retransmissions to improve hop-by-hop reliability, aiming for high-throughput over lossy multi-hop links. Unlike Hop, burst forwarding targets low power in addition to pure efficiency and its design takes into consideration the resource constraints of sensor motes. Balakrishnan et al. improved the performance of TCP with the help of a dedicated gateway between a wired and a wireless network [2]. Providing efficient and accurate rate control in wireless sensor networks is also a difficult issue by itself. RCRT performs rate control and allocation among several contenders towards a sink [26], Sridharan and Krishnamachari [28] provide efficient rate control for data collection. We argue that the issue tackled by this work is orthogonal to the one addressed by this paper; a protocol running on top of burst forwarding could leverage these results to provide even more energy-efficient rate control.

9 Conclusions and Future Work

Low power and high throughput are often opposing goals in wireless sensor networks. We see the packet forwarding as the critical operation to attain fast and low power multi-hop communication. Burst forwarding shows that high throughput can be obtained with a high energy efficiency without monopolizing network resources. Our two-level retransmissions survive heavy interference with both short-lived and periodic patterns. We also demonstrate that with burst forwarding, TCP can be efficiently adapted to lossy low-power wireless networks, and that rate controlled transport together with burst forwarding can provide a nearly constant energy cost per byte, even under interference.

We believe burst forwarding is generic enough to be adapted to other radio duty cycling protocols with different energy/latency trade-offs. We also believe burst forwarding could leverage different rate-controlled transport protocols to push further energy efficiency under interference. In particular, a transport layer using hop-by-hop rate control could be particularly efficient on top of burst forwarding, providing fast reaction to variations in link quality. Finally, we plan to investigate dynamic channel allocation with automatic channel switching to make burst forwarding even more resilient to interference.

Acknowledgments

This work was financed by the SSF through the Promos project, ERCIM through the Alain Bensoussan postdoc fellowship program, and CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053. Special thanks to Thiemo Voigt and Carlo Alberto Boano for providing the JamLab software [3], and to Jonathan W. Hui, our shepherd, for his useful feedback.

10 References

[1] M. Alizai, O. Landsiedel, J. Link, S. Götz, and K. Wehrle. Bursty traffic over bursty links. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, California, Nov. 2009.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the International Conference on Mobile Computing and Networking (ACM MobiCom)*, Berkeley, California, Nov. 1995.

[3] C. Boano, T. Voigt, C. Noda, K. Römer, and M. Zúñiga. JamLab: Augmenting Sensornet Testbeds with Realistic and Controlled Interference Generation. In *Proceedings of the 10th international conference on information processing in sensor networks (IPSN)*, 2011.

[4] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar. Brimon: a sensor network system for railway bridge monitoring. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys), Breckenridge, CO, USA, 2008.

[5] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. Technical Report T2011:05, Swedish Institute of Computer Science, Mar. 2011.

[6] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne. The announcement layer: Beacon coordination for the sensornet stack. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 2011.

[7] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based online energy estimation for sensor nodes. In *Proceedings of the IEEE Workshop on Embedded Networked Sensor Systems (IEEE Emnets)*, Cork, Ireland, June 2007.

[8] S. Duquennoy, N. Wirström, N. Tsiftes, and A. Dunkels. Leveraging IP for Sensor Network Deployment. In *Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011)*, Chicago, IL, USA, Apr. 2011.

[9] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, Nov. 2010.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, CA, USA, 2009.

[11] Y. Gu and T. He. Dynamic Switching-Based Data Forwarding for Low-Duty-Cycle Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, pages 272–285, 2010.

[12] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, North Carolina, USA, Nov. 2008.

[13] P. Karn. Advice for Internet Subnetwork Designers. RFC 3819, July 2004.

[14] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica. Flush: A reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Sydney, Australia, Nov. 2007.

[15] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, pages 254–263, 2007.

[16] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, M. Durvy, J. Vasseur, A. Terzis, A. Dunkels, and D. Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, Nov. 2011.

[17] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani. Block-switched networks: a new paradigm for wireless transport. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Boston, Massachusetts, Apr. 2009.

[18] C. Liang, N. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power zigbee networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, Nov. 2010.

[19] C.-J. M. Liang and A. Terzis. Rethinking Multi-Channel Protocols in Wireless Sensor Networks. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, Killarney, United Kingdom, June 2010.

[20] P. Mishra, H. Kanakia, and S. Tripathi. On hop-by-hop rate-based congestion control. *IEEE/ACM Trans. Netw.*, 4:224–239, April 1996.

[21] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet proposed standard RFC 4944, Sept. 2007.

[22] D. Moss and P. Levis. BoX-MACs: Exploiting Physical and Link Layer Boundaries in Low-Power Networking. Technical Report SING-08-00, Stanford University, 2008.

[23] R. Musaloiu-E., C.-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, St. Louis, Missouri, USA, 2008.

[24] F. Österlind and A. Dunkels. Approaching the maximum 802.15.4 multi-hop throughput. In *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensor Systems (HotEmnets)*, June 2008.

[25] F. Österlind, J. Eriksson, and A. Dunkels. Cooja timeline: a power visualizer for sensor network simulation. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zurich, Switzerland, 2010.

[26] J. Paek and R. Govindan. RCRT : Rate-Controlled Reliable Transport Protocol for Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 7(3), September 2010.

[27] B. Raman, K. Chebrolu, S. Bijwe, and V. Gabale. PIP: A Connection-Oriented, Multi-Hop, Multi-Channel TDMA-based MAC for High Throughput Bulk Transfer. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Zürich, Switzerland, 2010.

[28] A. Sridharan and B. Krishnamachari. Explicit and precise rate control for wireless sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, California, Nov. 2009.

[29] K. Srinivasan, M. Jain, J. Choi, T. Azim, E. Kim, P. Levis, and B. Krishnamachari. The kappa factor: inferring protocol performance using inter-link reception correlation. In *Proceedings of the International Conference on Mobile Computing and Networking (ACM MobiCom)*, pages 317–328, Chicago, Illinois, USA, 2010.

[30] K. Srinivasan, M. Kazandjieva, S. Agarwal, and P. Levis. The β-factor: measuring wireless link burstiness. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[31] Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Raleigh, NC, USA, 2008.

[32] J. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[33] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, USA, November 2006.