Forwarder Selection in Multi-Transmitter Networks

Doug Carlson, Marcus Chang, Andreas Terzis
Department of Computer Science
Johns Hopkins University
Baltimore, Maryland 21218
Email: {carlson, mchang, terzis}@cs.jhu.edu

Yin Chen
Qualcomm Research Silicon Valley
Santa Clara, CA 95051
Email: yinc@qti.qualcomm.com

Omprakash Gnawali
Department of Computer Science
University of Houston
Houston, Texas 77004
Email: gnawali@cs.uh.edu

***Abstract*—Recent work has shown that network protocols which rely on precisely-timed concurrent transmissions can achieve reliable, energy-efficient, and conceptually simple network flooding. While these *multi-transmitter* schemes work well, they require all nodes in the network to forward every data packet, which has inherent inefficiencies for non-flooding traffic patterns (where not all nodes need to receive the data). In this work, we formalize the concept of the "useful" forwarder set for point-to-point transmissions in low power multi-transmitter networks, those nodes which help forward data to the destination. We present a mechanism for approximating membership in this set based on simple heuristics. Incorporating forwarder selection on our 66-node testbed reduced radio duty cycle by 30% and increased throughput by 49% relative to concurrent flooding while preserving a 99.4% end-to-end packet reception ratio under the collection traffic pattern. We envision forwarder selection as a fundamental task in an efficient multi-transmitter networking stack. This work demonstrates that adding forwarder selection can improve energy efficiency and throughput while providing similar end-to-end packet delivery rates to flooding.**

I. Introduction

Glossy [9], Low-power Wireless Bus (LWB) [8], Flash Flooding [13], and Insteon [15] represent examples in an emerging family of multihop wireless sensor network (WSN) communication protocols. These systems leverage a combination of non-destructive concurrent transmissions and radio capture effect [11], [22] to perform fast network floods that reach all nodes with high probability: every node receiving a packet rebroadcasts it at precisely the same time, reducing destructive interference. This approach, which we refer to as *multi-transmitter* networking, carries several key benefits. In addition to the high yield, good throughput, and low energy consumption demonstrated in LWB, these methods require little routing state to work. This makes them suitable for networks with high degrees of node mobility, and may help in challenging environments where existing routing methods struggle to find high quality links.

Despite these benefits, it is obvious that not every node needs to be involved in every non-flood data transfer (collection, point to point communication, etc). In collection, a node that is adjacent to the data sink should not need assistance from the whole network to deliver its packets, and a node which always receives packets *after* the sink cannot help other nodes, to cite two simple examples. If we can reduce the set of nodes involved in a transfer, then non-forwarders can turn their radios off to save energy. Furthermore, if the diameter of the forwarder set is smaller than that of the network, this information can be used to set inter-packet spacing and thereby increase throughput without introducing collisions. The goal of this work is to solve the problem of forwarder selection in this context and demonstrate its benefits. By introducing a better network primitive than flooding, we advance the state of the art in multi-transmitter systems.

To this end, this paper makes four contributions: (1) We formally define forwarder selection in multi-transmitter networks. (2) We propose a forwarder-selection protocol, the first of its kind of which we are aware. (3) We describe the first TinyOS implementation of a network stack based on precisely-timed non-destructive concurrent transmissions, which we call CX (Concurrent Transmissions), and (4) we evaluate how our forwarder selection mechanism, CXFS, improves performance over simple flooding in this system.

Under CX, all communication takes place in the form of multi-transmitter floods. Through the use of a hop counter in each packet, nodes learn their relative distances to each other. The hop count information allows nodes to estimate whether they are between a source and destination for a given transfer. We address the inherent unreliability of hop-count and use this as the basis for our forwarder-selection method, which keeps nodes which are between the source and destination active while allowing the rest of the network to sleep.

Results from our 66-node indoor testbed show that forwarder-selection reduces duty cycle by 30% on average over simple concurrent flooding while maintaining an average packet reception ratio (PRR) of 99.4%. In the same setting, nodes increase their throughput by 49% on average.

The paper has six more sections. We present related work in Section II and define the task of forwarder selection in Section III. Section IV describes the CX forwarder selection protocol, and section V describes the CX network stack and some key implementation details. Section VI presents results from our testbed. We conclude in Section VII with a summary.

II. Related Work

Concurrent transmissions have been studied extensively in the context of wireless sensor networks [13], [16]. However, the bulk of this work relates to the capture effect, an artifact of wireless receiver design that allows radios to lock onto and successfully decode packets in the presence of considerable interference from other packets as long as the signal-to-noise ratio is high enough. While capture effect may benefit CX, we do not rely on it for successful packet receptions.

The research community has focused less on concurrent transmissions in the form of radio signals interfering non-

destructively, mainly because the timing required to perform non-destructive interference is considered difficult to achieve on resource-constrained devices. Note that non-destructive concurrent transmissions are different from beamforming and Multiple-Input Multiple-Output (MIMO) where antenna arrays at the transmitter and/or receiver are used to generate signals that interfere constructively at some desired angles.

Early work using the principle of non-destructive interference has primarily relied on the "free" transmission synchronization provided by hardware-generated acknowledgments from IEEE 802.15.4-compliant radios. Dutta et al. [7] first used these acknowledgments to efficiently wake up a large network of nodes. The same principle has been applied to medium contention and arbitration in receiver-initiated MAC protocols such as A-MAC [6] and Flip-MAC [3]. Strawman [14] demonstrates that even destructive radio interference can be a valuable primitive in low-power networks, and works without requiring precise transmission timing.

By taking full control of the MCU on the TelosB during packet reception, Ferrari et al. succeeded in generating non-destructive concurrent transmissions using a software-based approach, and used this to build a highly efficient network flooding protocol (Glossy) with microsecond time synchronization [9] on the Contiki embedded operating system [5]. While implementation details are not widely available, Insteon [15] is a commercial product which appears to use a network protocol which is very similar to Glossy. Our CX stack enables Glossy-type flooding in TinyOS [12], though it achieves this by using our hardware platform's high-precision timer/capture module rather than deterministic execution times.

Ferrari et al. later added a scheduler on top of the Glossy flooding protocol to construct the Low-power Wireless Bus (LWB) [8], a virtual one-hop network. In LWB, a master node in the network assembles and disseminates a TDMA schedule based on bandwidth requests by the nodes in the network. Our work is focused on the performance of the network protocol (not the scheduler), though LWB demonstrates that multi-transmitter networking protocols are practical and effective building blocks for real systems.

More recently, Wang et al. studied the scalability of Glossy-like flooding protocols and found that time synchronization errors (for scheduling concurrent transmissions) could accumulate over each hop and lead to destructive interference as the network scales [21]. In response, they proposed a flooding protocol which limits the number of forwarders to reduce the timing errors. Nevertheless, this study was purely based on analytical models of baseband wireless communications, and did not consider the effects of carrier wave frequency/phase offset or model the radio hardware behaviors such as symbol clock and phase recovery. The proposed flooding protocol further assumes that the geographical locations of all nodes are known, and uses the simple unit-disk communication model. Under these assumptions, the simulation results indicate that this protocol can be more scalable than Glossy.

III. Forwarder Selection

Our overall goal in this work is to reduce the nodes involved in forwarding data to those that do useful work, while still preserving good connectivity. In this section, we introduce the forwarder set concept and describe multi-transmitter flood, standard single-path, and our *Reduced Routeless* forwarder sets in formal terms. Section IV describes how we approximate the reduced-routeless forwarder set in real-world settings.

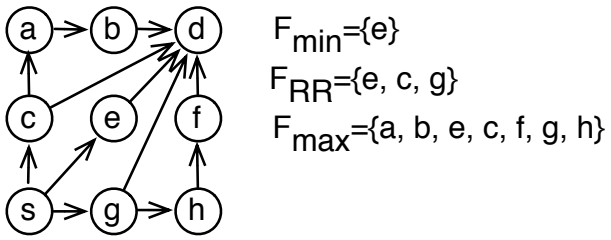


Fig. 1. Examples of minimum (F_{min}), maximum (F_{max}), and Reduced-Routeless (F_{RR}) forwarder sets for a 3×3 grid. All edges have cost 1.

A. Forwarder sets: Current Approaches

For the purposes of this section, we treat a wireless network as a directed graph G with vertices for each communication node and edges for each wireless link (having an associated packet reception ratio). We further assume that loss events are independent (e.g., no external interference). Under the concurrent communications paradigm [9], data transmissions take place in discrete communication rounds: transmissions of the same data in the same communication round are non-interfering. If a set of nodes S sends the same data packet at the same time, any node in the union of the adjacency sets of every node $s \in S$ successfully receives the packet with a probability determined by the union of packet reception ratios on the adjacent edges. In reality, synchronization errors and other effects may violate this model, though it remains useful for the purpose of explanation.

In this framework, a "valid" forwarder set from source s to destination d is any set of nodes F for which the subgraph of G consisting of nodes in F connects s to d with high probability (i.e. good end-to-end PRR).

1) Single-Path Routing: Single-path routing protocols effectively approximate some minimal forwarder set, $F_{min}(s, d)$, for each (source, destination) pair which consists only of the nodes on a single minimum cost path between source s and destination d. Path cost in this context is defined as the sum of link costs using metrics such as ETX [4]. In Figure 1, $\{e\}$, $\{c\}$, and $\{g\}$ would all fit the definition of F_{min}.

Even under perfect network conditions, single-path routing methods require coordination between nodes to estimate link metrics and calculate end-to-end paths. Nevertheless, when properly executed, such protocols have been shown to be very efficient, both in terms of energy and channel utilization [10].

On the other hand, single-path routing protocols can have difficulty adapting to variable link conditions, requiring either agile link quality estimation, reliance on conservative links, or a combination of the two. Such routing protocols can also experience periods of packet loss when previously-reliable routes break (e.g., due to node failures).

2) Multi-Transmitter Flooding: Multi-transmitter flooding selects the maximum forwarder set F_{max}: every node in the network. The validity condition is trivially satisfied for this set

(as long as some reliable end-to-end path exists between s and d anywhere in the original network). Figure 1 illustrates the difference between F_{min} and F_{max}.

In multi-transmitter floods, many more nodes are used for forwarding than is strictly necessary. Under idealized network conditions (e.g. no external interference), any node in F_{max} which does not lie on a minimum cost path between s and d is not useful: if they were removed from F_{max}, d would still receive a packet in the same number of communication rounds and the inclusion of these extra nodes is a source of unnecessary energy expenditure. For the example in Figure 1, $\{c, e, g\}$ all lie on a minimum cost path while $\{a, b, f, h\}$ are the extra nodes.

Furthermore, in order to prevent interference, a source node cannot start a new flood until the previous one has completed. When the entire network participates in flooding, this means that a node has to wait until a number of communication rounds (at least) equal to the diameter of the network have elapsed before it transmits without fear of collision.

In contrast to single-path routing, protocols that use multi-transmitter flooding have been shown to work well even in the face of link asymmetry and dynamism. Since this approach does not need to adapt routing state to link conditions, and since no bidirectional communication is required, as long as there is some reliable path from s to d at the point in time when the flood begins, d will successfully receive the packet.

B. Reduced-Routeless Forwarder Sets

Between these extremes, we define a *Reduced-Routeless* forwarder set $F_{RR}(s, d)$, which consists of all nodes f on *any* minimum cost path between s and d.

By the triangle inequality, a node f is a member of $F_{RR}(s, d)$ if and only if

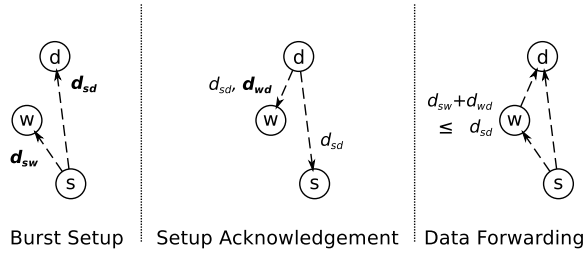
$$d_{sf} + d_{fd} = d_{sd} \tag{1}$$

where d_{ij} denotes the minimum distance, as defined by a cumulative metric (e.g., ETX, hop count), between i and j.

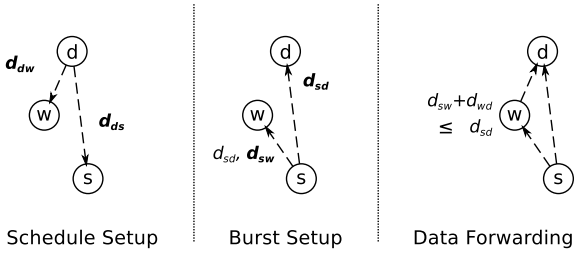
In contrast to single-path routing, each node f can test whether $f \in F_{RR}(s, d)$ without exchanging routing information with non-endpoint nodes: it only needs to have the end-to-end cost and its cost relative to each end. In other words, F_{RR} is similar to F_{max} in protocol design (thus the word "routeless") but is closer to F_{min} in its goals.

Although $F_{RR}(s, d)$ still includes more nodes than $F_{min}(s, d)$ for any network with more than one minimum cost path between s and d, it is also smaller than F_{max} as long as not all paths between s and d are minimum cost paths. Using F_{RR} instead of F_{max} should therefore reduce duty cycle (by decreasing the number of participating nodes) and increase throughput (by decreasing the diameter of the forwarder set and reducing inter-packet spacing) while retaining much of the path redundancy afforded by flooding.

However, if propagation patterns are not relatively stable (e.g. due to interference between concurrent senders or varying link conditions), then a node may remove itself from the forwarder set when it would be useful, or add itself to a forwarder set when it may be redundant. In the next section, we discuss several practical approaches to handling such variability.



(a) CX forwarder selection for point-to-point traffic. In Burst Setup, s floods a message to inform potential forwarders w and the destination d of their distance from the source (d_{sw} and d_{sd}, respectively). In Setup Acknowledgment, d floods a message which informs the network of d_{sd} and d_{wd} (assumed from d_{dw}). In Data Forwarding, w forwards messages only if $d_{sw} + d_{wd} \leq d_{sd}$.



(b) CX forwarder selection for collection traffic. The Setup Acknowledgment phase of the general point-to-point process is replaced by the periodic schedule announcements from the data sink, d.

Fig. 2. CX forwarder selection process. Dotted lines indicate multi-hop paths, boldface distances are directly observed, non-boldface distances are reported in packet bodies.

IV. CX Forwarder Selection Protocol

The CX (Concurrent Transmissions) multi-transmitter network stack allows nodes to approximate their membership in F_{RR}, using these results to duty cycle their radios and set inter-packet spacing. Next, we provide details on the protocol used to select forwarders, CXFS (CX Forwarder Selection). Each exchange in the setup process uses a basic multi-transmitter flood, whose implementation is described in section V and which is evaluated separately in VI-A.

A. CXFS Operation

In order for a node w to decide whether it should forward or not, it must learn its distance from the source, its distance from the destination, and the distance from source to destination (d_{sw}, d_{wd}, and d_{sd}, respectively).

For point-to-point transmissions, Figure 2(a) shows the steps in CXFS. The source s sends a Burst Setup packet, which allows each potential forwarder w to measure d_{sw} and the destination d to measure d_{sd}. In the Setup Acknowledgment phase, d responds with a packet containing d_{sd} in its body, which allows w to obtain d_{sd} and to measure d_{wd} (under the simplifying assumption that $d_{wd} = d_{dw}$). w now has enough information to make forwarding decisions.

In the case of data collection in CX, the network root both sends out periodic TDMA schedule packets and is the destination for all data traffic. Figure 2(b) shows how this

can be leveraged. The schedule packets provide d_{dw} to each potential forwarder and d_{ds} to each potential source. The Burst Setup packet now contains d_{sd} and allows nodes to measure d_{sw} (again, under the assumption of symmetric distances). w can now make forwarding decisions for (s, d) packets while only requiring a single end-to-end communication in s's time slot.

This decreases the setup overhead at the cost of increasing the staleness of distance information. Our evaluation did not indicate problems resulting from this. In the rest of this work, we will restrict the rest of our discussion to the collection case.

In either setting, node w can compute its membership in $F_{RR}(s, d)$. In the Packet Forwarding phase, nodes not in $F_{RR}(s, d)$ turn off their radios, and s sends packets with inter-packet spacing determined by d_{sd}.

B. Hop-count as Multi-transmitter distance metric

The previous discussion has used the concept of "distance" in multi-transmitter networks without defining it. In this subsection, we describe the motivation behind our use of hop-count as the basis for our distance metric and how we make the most of it.

1) "You can't seriously be using hop count, can you?": While hop-count is a notoriously unreliable metric in the single-transmitter networking domain, it possesses several key characteristics which make it the best choice for CXFS.

First, hop-count *obeys the triangle inequality*: a node which is 4 hops from the source is on no shortest path to a node which is 3 hops from the source. Any distance metric which cannot be applied to the definition of F_{RR} in equation 1 will not work in CXFS.

Second, in order for concurrent transmissions to be reliable, *simultaneously transmitted content must be identical*. In a sense, intermediate nodes *cannot* embed information in packets that is unique to them, and forwarding decisions must be made based on information that is common to all predecessors of a forwarder. Nodes cannot tell, for instance, which sender(s) participated in transmitting a packet that they received. Hop-count can convey meaningful distance information without requiring intermediate nodes to send conflicting data.

Finally, the distance metric *must make sense in the multi-transmitter networking context*. Putting aside the first two characteristics, traditional physical-layer metrics (such as RSSI and LQI) are subject to a range of effects in the multi-transmitter domain which may limit their usefulness. For instance, phase differences between two transmitters lead to wide variability in RSSI measurements at receivers, depending on whether the transmissions interfere constructively or destructively. We view the behavior and usefulness of physical layer metrics in the multi-transmitter setting as a topic for future research.

Rather than spurn hop-count for its faults, we choose to embrace it for its virtues.

2) Hop-count variability in multi-transmitter flooding: Previous work in single-path routing protocols has repeatedly shown that link quality can vary rapidly over time [17]. Furthermore, links can be asymmetric [2]. Thus, we can expect that the distance measurements in the multi-transmitter environment are similarly ephemeral and have to be updated over time, in order to provide accurate distance estimation.

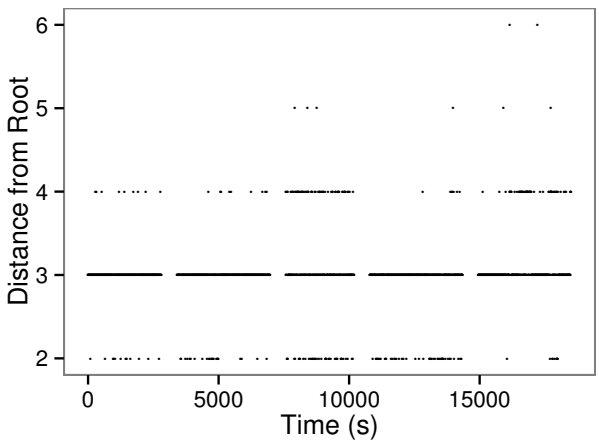


Fig. 3. Hop-count distance measurements at node d when node s sends concurrent floods back-to-back. Tests were spaced several hours apart, but concatenated to show long-term variation. Gaps separate each test run above.

Indeed, the results obtained using multi-transmitter floods on our testbed (described in Section VI) may show considerable long-term and short-term fluctuations in distance measurements for some nodes, as Figure 3 illustrates. Since F_{RR} is based on these hop-count measurements, the more accurately that we estimate distance and the better we handle its variability, the better our estimate of F_{RR} will be.

Estimating distance is analogous to link estimation and route discovery in single-path routing protocols, and faces the same challenges with regard to efficient neighborhood and routing table maintenance. We seek a distance estimation strategy that is lightweight, yet accurate enough to provide a reliable and compact forwarder set. We propose three strategies to do this: we either use the last-observed hop-count (termed "Last"), the average of observed hop-counts, or the maximum of observed hop-counts.

"Last" is easy to implement and requires basically no routing state, but may behave poorly as distances vary. "Average" may prevent nodes from responding to rapid changes, but will handle some degree of variability. "Max" will give the most inclusive forwarder sets, making it more reliable but also less power efficient. Additionally, by conservatively estimating distances between nodes, this strategy has the potential to increase inter-packet spacing by overreacting to rare bad transmissions. While "Last" is usable on networks of all sizes and for all traffic patterns, under the collection traffic pattern it is certainly feasible to maintain an average or maximum hop-count relative to the root, and in many real-world networks it is feasible to maintain this for each node in the network.

3) Distance Fluctuation Boundary: Orthogonal to the different distance estimation strategies, we add a boundary zone to the shortest-path distance in Equation (1) to account for estimation errors, short-term variability, and the assumption of distance symmetry.

By adding the boundary zone width b and using $\widehat{d}$ to denote the estimated distance and assumption of distance symmetry,

$F_{RR}(s,d)$'s definition from Equation (1) becomes:

$$F_{RR}(s,d) = \{f : \widehat{d_{sf}} + \widehat{d_{fd}} \leq \widehat{d_{sd}} + b\} \quad (2)$$

By increasing b, we include more nodes in the forwarder set $F_{RR}(s,d)$, up to and including using the entire network. Increasing the size of the boundary zone aids end-to-end packet reception by both increasing the diversity of paths that a packet can travel (mitigating the possibility of selecting a forwarder set that lacks reliable links) and adding an error margin to the distance estimations.

As the evaluation in Section VI shows, even with these simple approximations and strategies CX offers significant improvements to the state of the art.

V. CX Software Design and Implementation

A. Software Design

We implemented our CX network stack in TinyOS 2.1 with CXFS based on the distance metrics, estimation strategies, and boundary zone described above. Figure 4 shows a high-level view of the software architecture.

The CX link layer maintains transmission scheduling information. Time is divided into fixed-length frames (on the order of a packet-length in duration), and these are grouped into fixed-size slots (on the order of 10's of frames). Each node in the network is assigned a slot, and this schedule is distributed by a single root node. Nodes turn their radios on slightly ahead of each *frame* start, and keep their radios on until either a packet is received or a fixed 1 ms timeout elapses.

When a node rebroadcasts a packet, it does so at the next frame start after incrementing its hop-count field and decrementing its time-to-live (TTL) counter. Packets with TTL of 0 are dropped.

The CX network layer provides two primitives, Simple Flood and RR Flood. Under a Simple Flood, packets have their TTL initially set to the network diameter, and a node always rebroadcasts a packet with a non-zero TTL. In an RR flood, packets have their TTL initially set to $\widehat{d_{sd}} + b$. The RR Flood component is responsible for the forwarding decisions in CXFS: nodes only rebroadcast packets for which the inequality in Equation (2) holds.

CX provides two transport layer protocols on top of these, Flood Burst and RR Burst. These control packet queuing and spacing (starting the next transmission after the current packet's TTL has expired). Nodes queue packets until some threshold is exceeded, and then they transfer as many as they can in their next slot. Broadcast packets are handled by the Flood Burst transport protocol, while unicast packets are handled by the RR Burst protocol.

RR Burst implements the CXFS protocol outlined in Figure 2(b) and described in Section IV-A. The setup phase uses Simple Floods, and the packet forwarding phase uses RR Floods. In a Flood Burst, packets are simply sent one after the other via Simple Flood, and there is no Burst Setup phase.

Nodes turn their radios off if they determine that they are not a forwarder for the current slot. If a node receives no packets in the first N frames (where N is an estimate of the network diameter), it assumes that the slot owner did not have any packets to send and turns its radio off until the next slot starts. Likewise, nodes turn their radios off until the next slot if they determine they are not in $F_{RR}(s,d)$ during the Burst Setup.

The scheduler sits above the transport protocols. We use a simple static TDMA schedule (with equal-length slots) in this work, as our focus is on the performance of the forwarder selection and network layers. The network root (also the destination for collection traffic) sends out the TDMA schedule once per cycle, dictating how long each frame is, how many frames are in each node's slot, and how many slots are in the entire cycle. The schedule layer is where much of the work reported in LWB [8] would reside in this design.

Applications use the standard `AMSend` and `AMReceive` interfaces, allowing existing systems to easily switch over to CX.

We note that if applications require mobility, multiple sink support, minimal state-maintenance, or high interference-resistance, they can set a compiler flag to use Simple Flood Burst for both unicast and broadcast data.

B. Platform-Specific Implementation details

A detailed discussion of the network stack implementation is beyond the scope of this paper, but there are a few items that may be of interest to readers.

Our platform, the "Bacon" mote, is based on TI's CC430 [19] SoC, combining a 900MHz radio core (almost equivalent to a CC1101 [20]) and an MSP430MCU.

Glossy achieved good transmission synchronization through deterministic execution and forwarding times across motes, while our system can leverage a fast and reliable 26 MHz radio crystal for capturing and scheduling events. We clock one of the timer modules from the radio crystal, and capture the preamble RX/TX interrupt from the radio core with it. If the packet is to be forwarded, the mote sets a timer compare interrupt for one frame-length from the capture, loads the packet into the radio's TX buffer, and puts the radio into the FSTXON state (frequency synthesizer running and ready to transmit). When the compare interrupt is raised, we issue the command to begin transmission. The CC430 can be run with a 16MHz main clock, which keeps the potential interrupt-handling jitter within the tolerances dictated by the radio symbol rate. By using a relatively long frame length ($\sim$40 ms), the radio stack has considerable flexibility in making forwarding decisions, interleaving non-radio operations, and logging performance data to the testbed.

We found significant differences in PRR between the various stock radio settings provided by Texas Instruments, and ultimately chose a 125Kb/s setting which provided the best balance of reliability and speed. Since the CC430 lacks the hardware support for forward error correction (FEC) found on other radios (e.g., CC2420 [18]) we implemented FEC in software to make transmissions more robust at low-to-moderate bit error rates. Combining our implementation of a Hamming(7,4) [1] encoding and a 125Kbit/s symbol rate (2-frequency-shift-keying modulation), this gives us an effective data rate of 62.5Kbit/s, one quarter that of the CC2420. In the

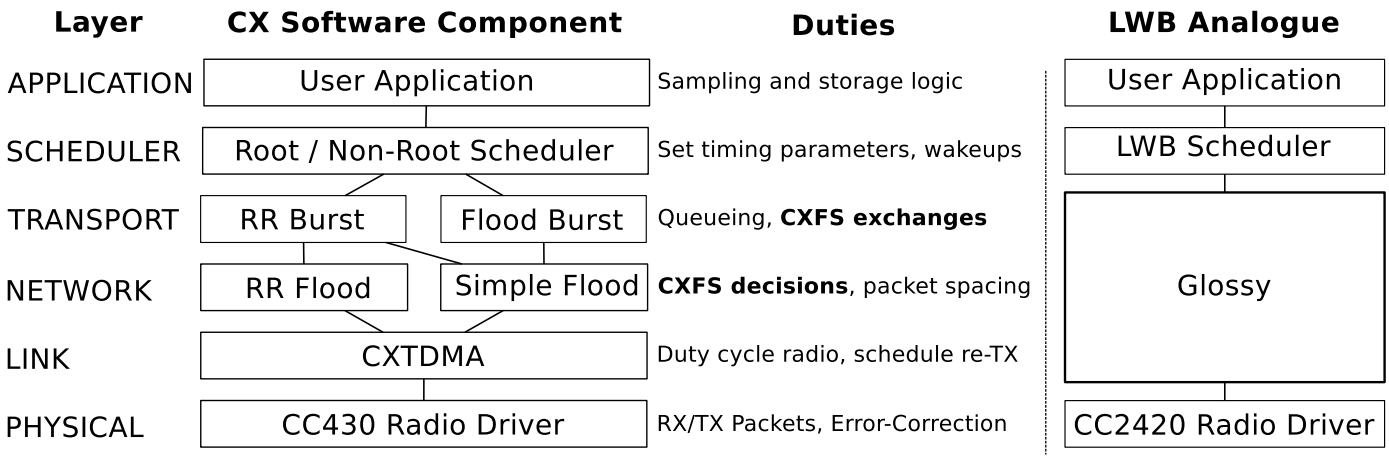


Fig. 4. CX network stack design, with rough equivalents in LWB [8]. This work focuses on the bolded duties.

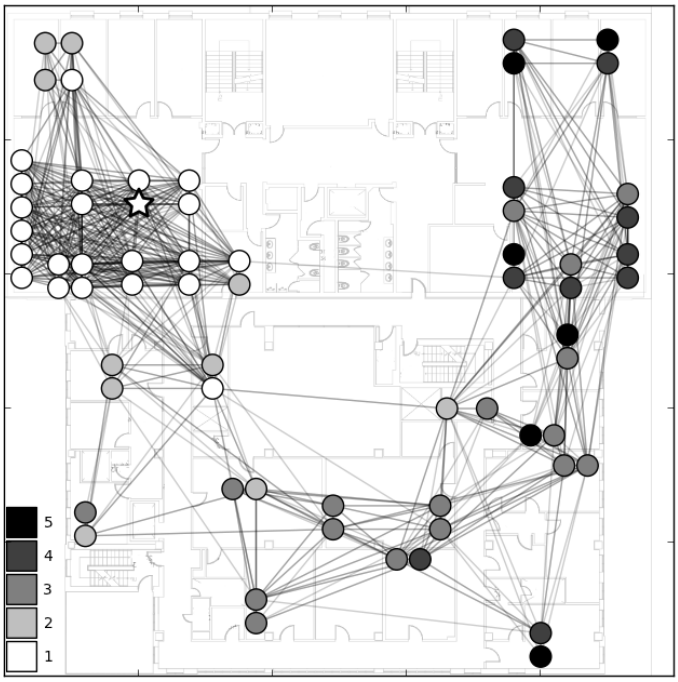


Fig. 5. Single-transmitter connectivity graph: links shown have PRR $> 95\%$ at -6 dBm output power, with one node transmitting at a time. Labels indicate distance from root node (marked with a star). Darker colors represent farther distances, with black being 5 hops and white being 1 hop. The area shown is roughly 50 m x 50 m.

future, we would like to find 250K (or higher) settings which work well, and we would like to use a more efficient coding scheme.

VI. Evaluation

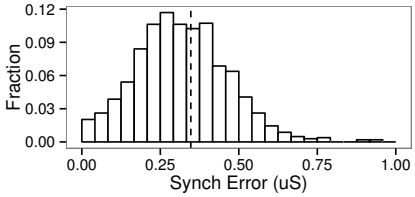
We evaluate CXFS on our wireless testbed consisting of 66 Bacon motes connected to a testbed server via TMote Connect NSLUs. Figure 5 shows a map of this testbed. The network is physically spread over a roughly 50 m x 50 m office area, though the large open space at the top of the map forces the network to be roughly 6 hops in diameter at an output power of -6 dBm (in the single-transmitter connectivity graph, based on offline measurements of PRR when nodes take turns transmitting non-concurrently).

Unless otherwise noted, all data presented is derived from aggregating over at least 3 test runs, and each test was run for one hour. Nodes generate one packet per minute and use a slot length of 40 frames for all tests. Each frame is roughly 40 ms in length, and nodes switch from RX to Idle if no packet is received within the first 1 ms of a frame start. Radio duty-cycles were obtained by recording the time of each radio state-change interrupt with a 6.5MHz timer driven by the radio crystal. The radio was considered to be active when it was in any of the RX, TX, or FSTXON states. Nodes used -6 dBm output power for all tests unless otherwise noted, though we do validate performance at several TX powers to confirm some degree of topology independence.

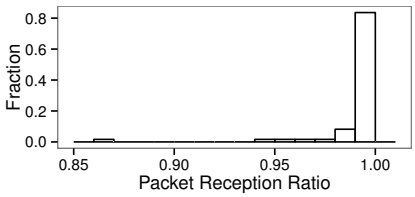
A. Baseline Performance

In this work, our goal is to evaluate the benefits that forwarder selection provides over simple flooding. Rather than port the state-of-the-art to our platform and operating system, we will compare against the CX implementation of simple flooding. In this section, we justify our implementation as a fair baseline by showing that it achieves reasonable timing precision, reliability, and duty cycle, not by showing that it is categorically better than Glossy.

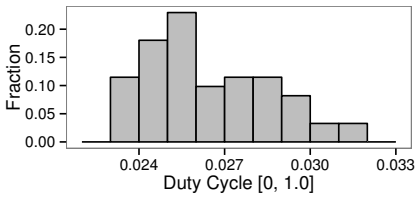
1) TX Timing Microbenchmark: Figure 6(a) shows the degree of timing precision we are able to achieve between two forwarders (as measured by recording the difference between the forwarders' start-frame-delimiter indicator pins on a 24 MHz logic analyer). The 91st percentile of the difference between the two transmitters' SFD signals is 0.5416 μs, somewhat worse than the 0.5μs reported in Glossy [9] with 30 transmitters. We note that the driver of multi-transmitter timing requirements is the radio's chip rate: our radio's 125 KB/s *symbol* rate should be compared against the CC2420's 2 MChip/s *chip* rate due to the CC430's lack of DSSS. In a separate experiment, we measured a bit error rate of 1.18% when



(a) Distribution of synchronization error between two concurrent senders. Mean error (dotted line) is 0.35 μs, roughly two ticks of the 6.5 MHz timer used for transmission scheduling.



(b) Histogram of packet reception ratios for each (src, dest) pair when using simple flooding. The average PRR was 99.5% and the median was 100%.



(c) Distribution of duty cycles when using Flood Burst transport on our testbed.

Fig. 6. Baseline measurements of CX Flooding performance.

two nodes transmit 1 μs apart with no capture effect.[1] Even the worst synchronization errors we measured are within the FEC's operating limits, and we expect to see lower bit error rates when capture effect is present. This level of precision is adequate to support multi-transmitter flooding.

2) Simple Flood Performance: To demonstrate that the above results lead to reliable flooding, we perform a series of tests to evaluate Flood Burst on its own. In these tests, nodes generate a packet once every 60 seconds, and initiate a Flood Burst when they have queued 10 packets. Figure 6(b) shows the distribution of packet reception ratios between pairs of nodes in the network: the mean PRR was 99.5%. While Glossy used parameter "N," for the number of rebroadcasts that each node performed upon receiving a packet, we found that our implementation provided high PRR with a single broadcast.

Figure 6(c) shows the distribution of duty cycles across the network, with mean/median of roughly 2.8%. To put this in perspective, the closest comparison we can make is against the FlockLab results reported in LWB [8]: their test used 54 nodes, a 120 second inter-packet interval, and a 15 byte payload vs. our 66 nodes, 60 second IPI, and 12 byte payload. They report a 0.43% duty cycle for the resulting 6.75 B/s aggregate load, while our 2.8% figure corresponds to a 12.1 B/s aggregate load. When one considers that our slower radio and software encoding cuts our effective transmission speed to 1/4 of the CC2420's, this indicates that our implementation is roughly on par with the state of the art in terms of energy-efficiency.

We acknowledge that this comparison elides some of the complicating factors, but remind the readers that our goal is not to show that our implementation of flooding is better than Glossy, but to show that it works correctly on our platform and is a fair baseline for demonstrating forwarder selection.

B. RR Burst vs. Flood Burst

The goals of this work are to reduce network duty cycle for data transmissions which need only travel through part of the network, and to use distance estimates to improve inter-packet spacing. This section focuses on this impact and keeps a simple fixed TDMA schedule as described above.

As a quick sanity-check, Figure 7 shows how often nodes join $F_{RR}(s, d)$ for an example pair of s and d nodes and we

[1]Two transmitters were connected to the receiver via a wired connection. Variable attenuators equalized their RSSI at the receiver, and their transmissions are triggered with GPIO edge interrupts of controlled delay for this experiment.

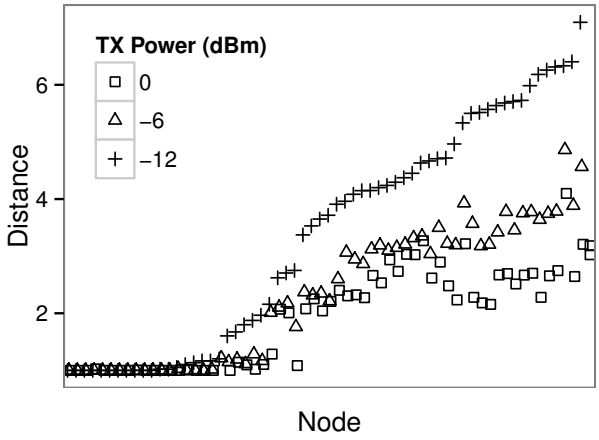


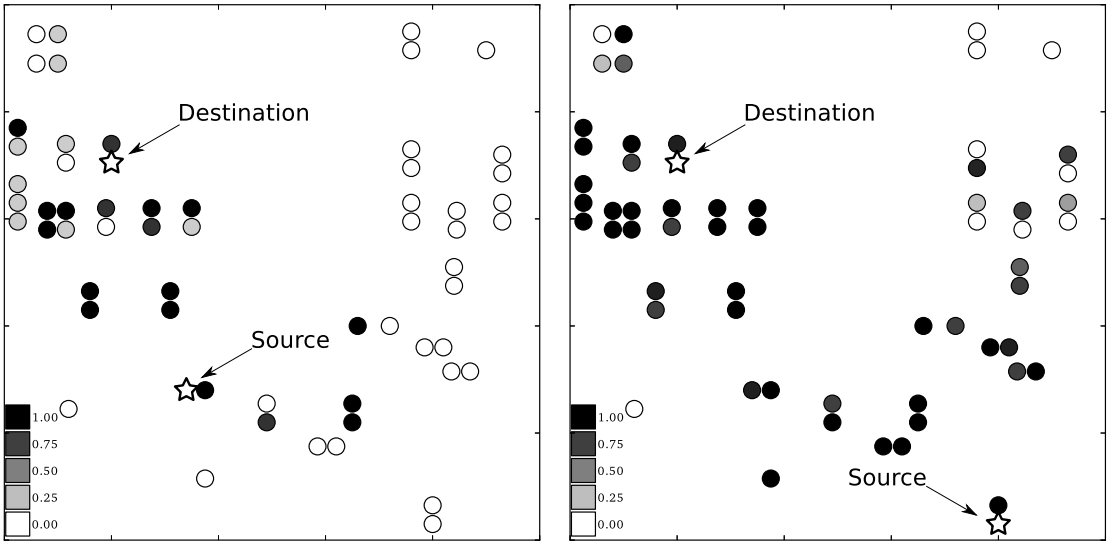
Fig. 8. Mean of observed distances from the sink for each node at several TX power settings: each column shows results for the same node.

indeed observe that nodes physically between the source and destination are the most likely to participate. However, one can also clearly see that some nodes outside of the geometrically shortest path participate with high probability, and others clearly on the shortest path participate with low probability (due to non-heterogeneity of links, variability, etc.).

To quantify this result, we use the RR Burst protocol (with the "Average" metric, boundary width 2) to transfer data from all nodes to the sink and adjust the radio transmission power to vary the network depth and topology. Figure 8 shows how node distances change under different TX powers to give a sense of our testbed's size.

In order to find the relation between where traffic originates and what impact this has on the duty cycle across the network, we first calculate the average duty cycle in each slot. Since each slot belongs to a unique node, this duty cycle is an indicator for a single node's impact on the rest of the network. In figure 9 we show the average network duty cycle for each node's slot as a function of the slot owner's distance to the root at three different transmission powers.

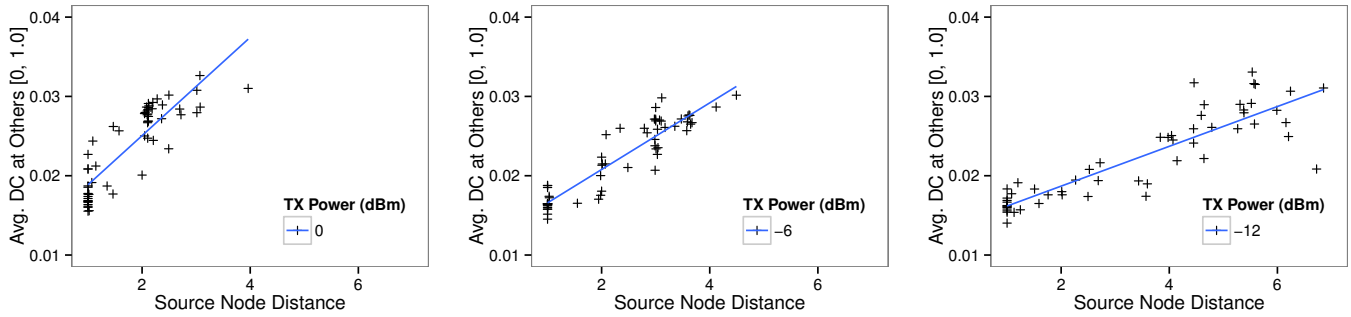
The same trend is visible at all tested transmission powers: slots belonging to nodes close to the sink (having low depth) contribute less to network duty cycle than nodes far from the sink. Additionally, the sparser the network is, the less impact a node at a given depth has on the rest of the network. This makes sense: all other things being equal, a node at depth n will have fewer nodes between itself and the root in a sparse



(a) Participation frequency for a node at moderate distance.

(b) Participation frequency for a more distant node.

Fig. 7. Heatmaps showing frequency of joining $F_{RR}(s, d)$ for source nodes at different distances from root. Dark colors represent higher frequencies, with black being 100% and white being 0%. Nodes used average distance with $b = 0$ to determine membership in F_{RR}.



(a) 0 dBm TX Power

(b) -6 dBm TX Power

(c) -12 dBm TX Power

Fig. 9. Duty cycle as a function of source-node distance. Each point corresponds to a single node's allocated slot. The X-value indicates the distance of the source node from the sink, the Y-value indicates the average duty cycle across the network during that node's slot.

network than in a comparable network of higher density.

We note that the duty cycles in Figure 9 are for each individual slot and do not include inactive slots where the whole network is asleep. Since the duty cycles in Figure 6(c) also include inactive slots, a direct comparison between the two figures is not possible. However, when we calculate the overall duty cycles in these tests, we find that at -6dBm, we see a 30% improvement in the average network duty cycle (25% at 0 dBm, 20% at -12 dBm).

Figure 10 shows each node's normalized throughput as a function of distance to the root at the three different transmission powers. This flood throughput is computed offline by assuming the maximum observed hop-count is used for flood packet spacing: in practice, most network designers will not know this ahead of time and will conservatively set the flood TTL to some estimate of the network diameter (further hurting flood throughput). Under CX, only the setup packets of RR Burst transmissions are affected by a conservative diameter estimate. The figure shows that nodes close to the sink can indeed use their estimate of d_{sd} to increase their throughput. On our testbed, this results in an average throughput increase

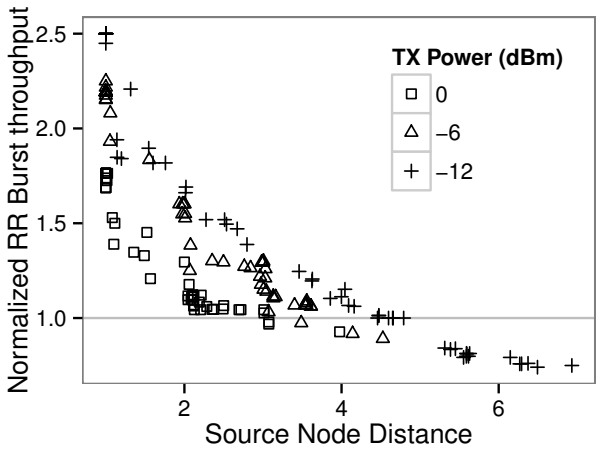
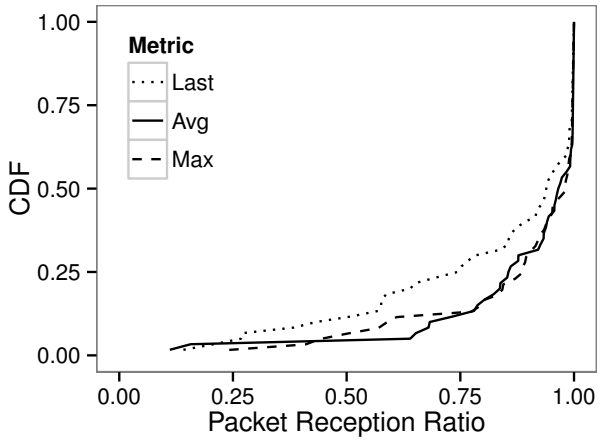
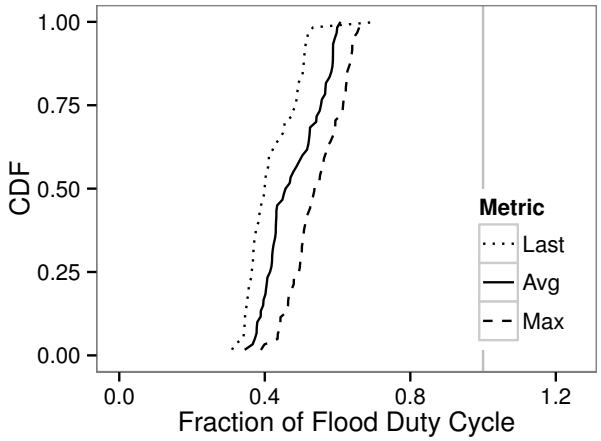


Fig. 10. Per-node throughput as a function of source node distance. These results are normalized to flood throughput (1.0 = same as flood, $>$ 1.0 = higher throughput than flood).



(a) CDF of PRR by distance estimation strategy.



(b) CDF of duty cycle improvement by distance estimation strategy.

Fig. 11. Impact of distance estimation strategy on PRR and duty cycle. All tests conducted using boundary width of 0.

of 49% when nodes transmit at -6 dBm (31% at 0 dBm, and 28% at -12 dBm).

Nodes at the edge of the network will always see worse throughput under RR Burst than Flood Burst: they have to pay the cost of the setup phase for each burst (whereas if they used Flood Burst, they could use that time to send data). In these tests, slots were 40 frames in length (roughly five Flood Burst rounds), so the setup phase has a relatively high cost (roughly 20% of available throughput). Increasing the length of a slot in the schedule would help to further amortize this at the cost of increased latency. Such scheduling optimizations and trade-offs are an important candidate for run-time tuning in future work. That being said, when averaged across the network, throughput increases over Flood Burst for all of these tests.

1) Distance Estimation: Next, we evaluate the effects of several distance estimation strategies from Section IV-B.

Figure 11(a) shows the impact of each strategy on end-to-end packet reception ratio, while figure 11(b) shows their impact on duty cycle. These were conducted with boundary width $b = 0$ to isolate the effect of the metric.

As expected, "Last" achieves the best duty cycle improvement over flooding (with per-node savings of nearly 60%), while it also shows the worst PRR. For some applications, this may be an acceptable tradeoff.

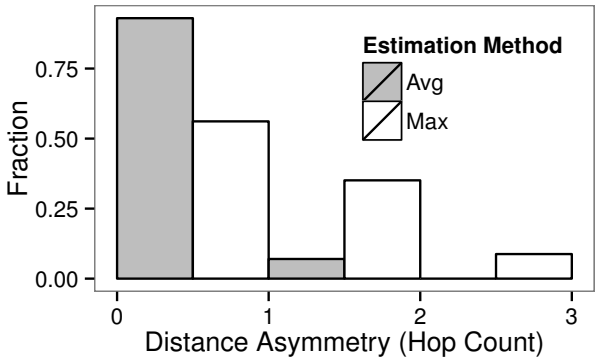


Fig. 12. Distribution of path length asymmetry from a 1-hour test. The average root-to-leaf distance was less than 2 hops different from the average leaf-to-root distance for all nodes. The worst asymmetry was 2 hops when considering maximum root-to-leaf and leaf-to-root distances.

"Max" and "Average" show relatively similar performance in PRR, but the latter achieves lower duty cycles. This suggests that "Max" can be overly conservative while "Average" can strike a balance between efficiency and reliability. On our testbed, it is reasonable to set aside the 70 bytes of RAM necessary to track the average or maximum distance of each node, though in much larger networks "Last" would be the only viable option.

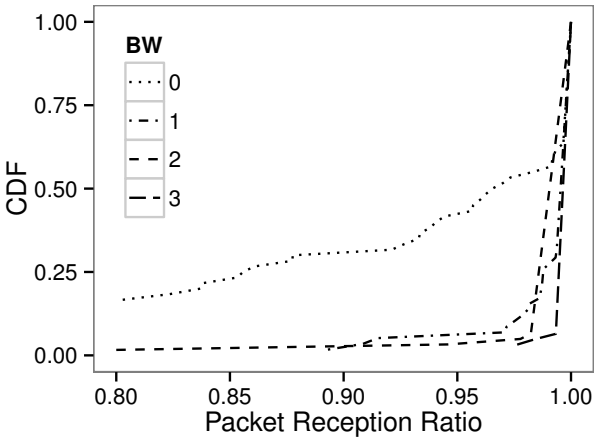
2) Boundary Width: Recall that we added boundary zone b to the shortest-path distance to account for the distance asymmetry shown in figure 12 and the variability shown in figure 3. As we are currently restricted to a single testbed, we don't yet know how dependent this asymmetry is on physical topology, but we can see that distances are generally not *too* asymmetric.

The boundary width essentially reflects a trade-off between efficiency (duty cycle) and reliability (PRR) as shown in Figure 13. Specifically, Figure 13(a) shows how PRR increases with boundary width, while Figure 13(b) shows how the duty cycle becomes less efficient. When the boundary width is set to 0, we see many nodes with poor PRR due to routing failures: distance variations can easily generate cases where some nodes that are on the shortest path decide not to join F_{RR} and participate in forwarding packets. On the other hand, as boundary width increases, more and more nodes that are not on the shortest path decide to participate and waste energy.

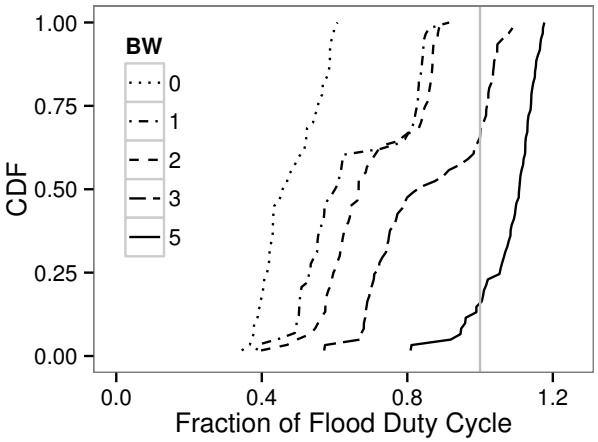
A boundary width of 2 achieves a good balance with an average PRR of 99.4% and the average node's duty cycle is only 70% of what it experiences when using Flood Burst.

VII. Conclusion

Recent work from industry and academia has shown the feasibility and benefits of using multi-hop concurrent transmissions for point-to-point and convergecast traffic in low-power wireless networks. Approaches such as Low-Power Wireless Bus and Insteon offer high yield and throughput, low duty cycles, and simple operations. Despite these benefits, flood-based approaches are intuitively wasteful since they involve



(a) CDF of PRR by boundary width. X-axis truncated and $b = 5$ omitted for clarity.



(b) CDF of duty cycle improvement by boundary width. With a 3-hop boundary zone, some nodes see higher duty cycle due to the increased setup cost of transmissions in RR Burst, and this effect is even more pronounced with a 5-hop boundary zone.

Fig. 13. Impact of boundary width (BW) on PRR and duty cycle. All tests conducted using "Average" distance metric with large enough routing table to track all nodes.

every node in the network for every data transfer. This work proposes adding forwarder selection to the emerging concept of a multi-transmitter network stack.

We formally defined the forwarder selection process for multi-transmitter protocols and provide simple mechanisms that nodes in a network can use to determine whether they should participate in a transmission or turn off their radio to conserve power. Further, we present the CXFS protocol that shows how the forwarder selection process can be adapted for real networks with asymmetric and time varying links.

Results from our 66-node testbed show that CX reduces average duty cycle by 30% and increase average throughput by 49% over simple flooding while preserving a 99.4% average end-to-end packet reception ratio.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1111507.

REFERENCES

[1] Hamming(7,4). From http://en.wikipedia.org/wiki/Hamming(7,4).

[2] N. Baccour, A. Koubâa, L. Mottola, M. Zuniga, H. Youssef, C. Boano, and M. Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Trans. Sen. Netw.*, 8(4):34:1–34:33, Sept. 2012.

[3] D. Carlson and A. Terzis. Flip-mac: A density-adaptive contention-reduction protocol for efficient any-to-one communication. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1 –8, June 2011.

[4] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proceedings of the 9^{th} ACM International Conference on Mobile Computing and Networking (MobiCom 2003)*, Sept. 2003.

[5] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Nov. 2004.

[6] 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 8^{th} ACM Conference on Embedded Networked Sensor Systems (Sensys)*, pages 1–14, Nov 2010.

[7] P. Dutta, R. Musăloiu-E., I. Stoica, and A. Terzis. Wireless ACK collisions not considered harmful. In *Proceedings of the 7^{th} Workshop on Hot Topics in Networks (HotNets-VII)*, pages 19–24, Oct 2008.

[8] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-power wireless bus. In *10th ACM Conf. on Embedded Networked Sensor Systems (SenSys 12)*, 2012.

[9] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, Chicago, IL, USA, 2011. ACM/IEEE.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7^{th} ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 1–14, Nov 2009.

[11] J. F. K. Leentvaar. The Capture Effect in FM Receivers. *IEEE Transactions on Communications*, 24(5):531–539, 1976.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. Tinyos: An operating system for sensor networks. *Ambient intelligence*, 35, 2005.

[13] J. Lu and K. Whitehouse. Exploiting the capture effect for low-latency flooding in wireless sensor networks. In *INFOCOM*, 2009.

[14] F. Österlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels. Strawman: resolving collisions in bursty low-power wireless networks. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 161–172. ACM, 2012.

[15] SmartLabs, Inc. Insteon: The details. Available from: http://www.insteon.net/pdf/insteondetails.pdf, 2005.

[16] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of concurrent transmission in wireless sensor networks. In *Proceedings of the ACM Sensys*, Nov. 2006.

[17] K. Srinivasan, M. Kazandijeva, S. Agarwal, and P. Levis. The β-factor: Measuring Wireless Link Burstiness. In *Proceedings of the 6^{th} ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2008.

[18] Texas Instruments. CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Available at http://www.ti.com/lit/gpn/cc2420, 2006.

[19] Texas Instruments. MSP430 SoC with RF Core. Available at http://www.ti.com/product/cc430f5137/, 2010.

[20] Texas Instruments. CC1101 Low-Power Sub-1 GHz RF Transceiver (Rev. H). Available at http://www.ti.com/product/cc1101, 2012.

[21] Y. Wang, Y. He, X. Mao, Y. Liu, Z. Huang, and X. Li. Exploiting constructive interference for scalable flooding in wireless networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 2104–2112, march 2012.

[22] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of the 2^{nd} IEEE workshop on Embedded Networked Sensors (EmNets)*, 2005.