One for All, All for One: Toward Efficient Many-to-Many Broadcast in Dynamic Wireless Networks

Fabian Mager*
Networked Embedded Systems Group
TU Dresden, Germany
fabian.mager@tu-dresden.de

Carsten Herrmann*
Networked Embedded Systems Group
TU Dresden, Germany
carsten.herrmann@tu-dresden.de

Marco Zimmerling
Networked Embedded Systems Group
TU Dresden, Germany
marco.zimmerling@tu-dresden.de

ABSTRACT

Many applications such as autonomous swarming drones and system services like data replication need to exchange data among many or all nodes in a network. However, wireless *many-to-many broadcast* has thus far only been studied theoretically or in simulation, and practical solutions hardly meet the requirements of emerging applications, especially in terms of latency. This paper presents Mixer, a communication primitive that provides fast and reliable many-to-many broadcast in dynamic wireless multi-hop networks. Mixer integrates random linear network coding with synchronous transmissions to simultaneously disseminate all messages in the network. To deliver the performance gains our approach enables, we design Mixer's protocol logic in response to the physical-layer characteristics and the theory of network coding. First results from testbed experiments demonstrate that, compared with the state of the art, Mixer is up to 65 % faster and reduces radio-on time by up to 50 %, while providing a message delivery rate above 99.9 %.

1 INTRODUCTION

All-to-all broadcast solves the following problem: In an N-node network, where each node has a message, disseminate all messages to all nodes. The more general case of *many-to-many broadcast* handles any initial distribution of M messages to (source) nodes and any desired final distribution of messages to (sink) nodes.

This problem is relevant because many applications and system services need to exchange information among many or all nodes in a network. Examples include data replication and leader election for fault tolerance [26], or exchange of sensor readings, local state, etc. to enable self-adaptive systems like swarming drones [3]. In fact, certain control problems are only tractable if each node can make decisions with knowledge of the whole system state [4].

Despite its relevance, wireless many-to-many broadcast has thus far only been studied theoretically [10] or in simulation [9]. Instead, we aim to solve the problem in practical wireless networks. Our work is driven by the needs of emerging cyber-physical applications [3, 11, 12] and industrial control [2, 22]:

Fast and reliable: To minimize the impact on application performance and to keep up with the dynamics of physical processes, many-to-many broadcast should be as fast as possible and message losses are only tolerable in exceptional cases.
Dynamic multi-hop topology: Rotating parts and mobile devices add to the dynamics of wireless networks, and multi-hop communication is either beneficial or a must [22].
Sizable payloads: In control, payloads are tens of bytes [11, 22]. Other applications or replication may have larger payloads.
Low power: Battery-powered and energy-harvesting devices are often desirable (*e.g.*, in process and building automation [2]). Size and weight constraints of drones call for small batteries as well as low-power wireless radios and microcontrollers [11].

Meeting these requirements is challenging. Routing, for example, becomes increasingly complex and inefficient as the number of sinks and the network dynamics increase [23]. We thus strive for a solution that operates independently of the number of sinks and the time-varying network topology. One possibility is to map the many-to-many broadcast onto a sequence of Glossy floods [8], where each source node is assigned a time slice in which it floods one packet to all other nodes in the network (see Sec. 2). For disseminating a single packet using half-duplex radios, Glossy achieves almost the theoretical minimum latency and provides a packet delivery rate (PDR) higher than 99.9 % even if nodes are moving [7, 8]. However, the overall latency increases linearly with the number of messages, because a flood can only start after the previous one is finished [7].

Contribution and road map. We present Mixer, a many-to-many broadcast primitive that breaks this linear scaling. The key idea is to *overlay* all floods by letting nodes "mix" packets using *random linear network coding (RLNC)* [1, 13]; that is, nodes send random linear combinations of received packets to simultaneously disseminate all messages in the network. Theoretical results indicate that this approach outperforms sequential flooding and achieves the best known bounds in highly dynamic networks [10]. To deliver tangible performance gains in practical wireless networks without sacrificing PDR, this paper makes the following contributions:

- Sec. 2 illustrates the intuition underlying our approach, and derives the challenges in utilizing its potential in a practical solution that meets the above application requirements.
- Sec. 3 describes how we address these challenges by designing Mixer in response to the characteristics of the wireless physical layer and the theory of RLNC.
- Sec. 4 evaluates a Mixer prototype. To compare against the state of the art, we implement Mixer on an MSP430-based platform with a 2.4 GHz IEEE 802.15.4 radio, yet our design is also applicable to other wireless technologies. Results from

*Both authors contributed equally to this work.

HotWireless'17, October 16, 2017, Snowbird, UT, USA

ACM ISBN 978-1-4503-5140-9/17/10…$15.00
https://doi.org/10.1145/3127882.3127884

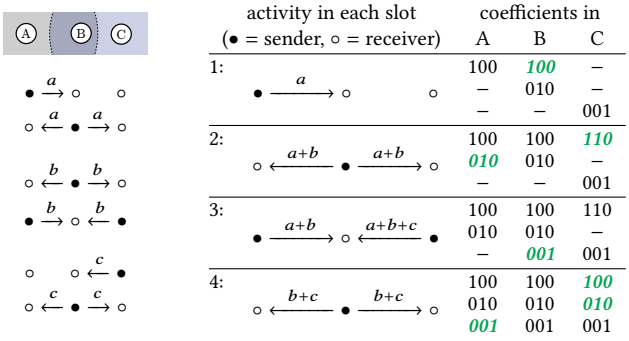


Figure 1: In a 2-hop network, nodes A, B, and C want to share 3 messages. Sequential flooding via Glossy (left) takes 6 slots. Overlaying floods via network coding (right) can achieve the same in 4 slots, assuming that B captures C's packet in slot 3.

an all-to-all scenario with 10–110-byte payloads on the FlockLab testbed [21] show that, compared with sequential Glossy floods, Mixer is on average 40-65 % faster, reduces average radio-on time by up to 50 %, and provides a PDR above 99.9 %.

To our knowledge, Mixer is the first practical solution that uses RLNC for many-to-many broadcast in dynamic wireless networks. We review related work in Sec. 5, and end the paper in Sec. 6 with concluding remarks and an outline of our current research agenda.

2 MOTIVATION AND CHALLENGES

Mixer integrates synchronous transmissions with RLNC. This section explains the intuition underlying our approach, and highlights the challenges in realizing its potential in practice.

Synchronous transmissions. Two properties of wireless networks complicate efficient and reliable communication: time-varying multi-hop topologies and the broadcast nature of the wireless medium causing packet collisions. While traditional solutions try to *avoid* collisions using, for example, carrier sensing or scheduling, recent solutions *take advantage* of collisions [8, 25]. In particular, if multiple well-synchronized senders transmit identical packets, a receiver can correctly decode the combined signal with high probability due to a combination of several physical-layer effects [27].

One solution that exploits this observation is Glossy, providing one-to-all communication in multi-hop networks of off-the-shelf IEEE 802.15.4 devices [8]. In Glossy, one node initiates the communication by sending a packet. Nodes receiving the packet retransmit it synchronously with sub-microsecond accuracy, enabling other nodes to receive and relay the packet. The transmissions are synchronized on the fly: A careful design and implementation ensure that each node transmits a minimal, constant time after reception. The result is a slotted communication scheme, where nodes operate *without* knowledge of the time-varying multi-hop topology. Glossy achieves almost the minimum latency for disseminating a single packet using half-duplex radios and provides a PDR above 99.9 %.

Many-to-many broadcast using sequential floods. The number of slots s in a Glossy flood needs to be chosen in advance based on the maximum network diameter (in hops) and the desired reliability [8]. Thus, *without* knowledge of the network topology, if we want to disseminate M messages using M back-to-back Glossy floods, we need in total $s \cdot M$ slots. For example, as shown in Fig. 1 (left), it takes 6 slots to disseminate 3 messages in a 2-hop network. We see that the number of slots and latency scale linearly in M.

This linear scaling appears suboptimal: Every message is sent multiple times, forming a simple kind of repetition coding. While we cannot do better than this if we consider each message individually, we can break the linear scaling by considering multiple messages together, that is, by *overlaying* the floods of all source nodes.

Many-to-many broadcast using overlayed floods. This is the main idea behind Mixer. As illustrated in Fig. 1 (right), rather than just retransmitting a received packet (of one flood), nodes transmit linear combinations of multiple payloads (of several floods). In slot 2, for example, node B sends the sum of A's packet a, which it received in slot 1, and its own message b. Node A can then subtract its payload a from the received sum $a + b$ to obtain b. Using its own and previously received packets, every node can reconstruct all messages by adding and subtracting the right components. This way, it takes only 4 instead of 6 slots to disseminate the 3 messages.

Network coding. The concept of (re-)combining multiple packets before or during transmission is known as *network coding (NC)* [1]. We use *digital network coding*, where a sender computes the combination before transmission, as this does not require any changes to the physical layer. Instead, any flavor of *physical-layer network coding* [14, 24], where the combination is built by mixing signals over the wireless channel, requires significant extensions that are challenging and not implemented in off-the-shelf devices. Moreover, we realize that this feature is not necessarily needed; for example, in slot 3 (see Fig. 1), B only needs the packet from C since it can build the other packet by itself from previously received packets.

To be able to reconstruct all messages $m_1, \ldots, m_M$, a receiver needs to know which messages are combined in each packet p_i. This information is described by the corresponding *coding vector* $c_i = (c_{i1}, \ldots, c_{iM})$ such that p_i's payload equals $\sum_k c_{ik} m_k$. In a fully deterministic system, all coding vectors can be determined beforehand and hardcoded into each node. However, in the face of unpredictable packet losses in real wireless networks, the only safe way is to embed the coding vector into each packet. This approach also has the advantage that a sender can pick the coefficients randomly, known as RLNC [13], without any additional information (*e.g.*, about the network topology). This property makes RLNC very appealing for dynamic wireless networks, which we target.

It remains to discuss how a node can decode the original messages from a set of received packets. Since the packets contain linear combinations with known coefficients, the decoding problem can be stated as a linear system of equations, where the coding vectors form the rows of the coefficient matrix (see Fig. 1). If this matrix has full rank, then the system has a unique solution, which can be found by known algorithms like Gaussian elimination.

Challenges. To obtain the above benefits in a practical many-to-many solution, we must address at least the following challenges:

When should a node send or listen? Since nodes randomly combine messages, they typically send different packets. Thus, a common receiver can successfully receive one of the packets only if the *capture effect* occurs [19]. In IEEE 802.15.4 networks, capture happens if the SINR exceeds 3 dB and the strongest packet arrives no later than 128 μs after the first packet [28], which corresponds to the air time of the synchronization header. However, due to the SINR condition,

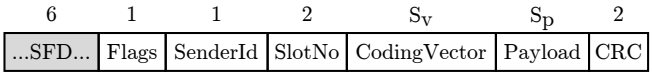


Figure 2: Mixer packet format. *Parts in gray are defined by the IEEE 802.15.4 standard; CRC is generated by the radio. Sizes in bytes.*

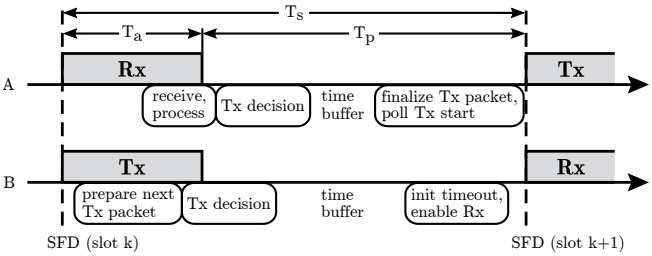


Figure 3: Tasks within a Mixer slot shown for two nodes A and B in receive (Rx) and transmit (Tx) mode, respectively.

the probability of capture drops rapidly as the number of senders increases. How can a node locally decide whether to send or listen in a slot, maximizing spatial reuse without destroying capture?

How to ensure synchronous transmissions without a global clock? To meet the timing condition of capture, nodes must communicate in a globally slotted fashion. Unlike Glossy, however, nodes need to spend more and varying amounts of time on processing in each slot, which impairs synchronization in the face of clock drift.

What should a node send? A packet is *innovative* for a node if it is linearly independent from previously received packets. To achieve low latency, we need to find a policy that allows senders to randomly build packets that are likely innovative for their neighbors.

How to achieve an efficient runtime operation? Network coding improves the utilization of the wireless medium (*i.e.*, reduces the number of slots), but requires nodes to store and process the coding vector and payload bytes of the packets. Limited memory and compute power on embedded devices may hinder harnessing the benefits (*e.g.*, by blowing up the length of the slots), so we need to design efficient coding and decoding strategies. In addition, nodes should locally detect when they are no longer needed for the dissemination process, so they can turn off their radio to save energy.

3 DESIGN AND IMPLEMENTATION

Mixer is a many-to-many broadcast primitive for dynamic wireless multi-hop networks, such as swarming drones [11] and wireless control networks [2, 22]. Nodes in these networks are often embedded devices with limited memory, compute power, and energy. We design Mixer for IEEE 802.15.4, but in principle our approach works with any physical layer featuring capture (*e.g.*, Wi-Fi).

Mixer sits between the physical layer and a high-layer protocol like LWB [7] that informs all nodes about the initial and final distributions of the M messages to the N nodes before Mixer starts.

Distributed operation and synchronization. Using Mixer, each many-to-many broadcast, called *round*, is divided into adjacent *slots*. As shown in Fig. 3, each slot has the same fixed length T_s, which accounts for the air time T_a of one packet and processing time T_p.

One node starts the round by transmitting its message. Nodes receiving the packet take timestamps when they receive the start-of-frame delimiter (SFD) field, which is part of the synchronization header (see Fig. 2) and defines the beginning of a slot. As detailed

Algorithm 1 semi-coordinated transmission

```
d = 1 + num_neighbors                    ▷ local density (from history)
owner = (slot_no + 1) mod N              ▷ determine owner of next slot
if owner = my_node_id then               ▷ my slot
    p_t = 1
else if Tx in current slot then          ▷ don't transmit twice
    p_t = 0
else if owner is neighbor then           ▷ foreign slot
    p_t = 0
else                                     ▷ shared slot
    p_t = 1 / (d + 1)
end if
```

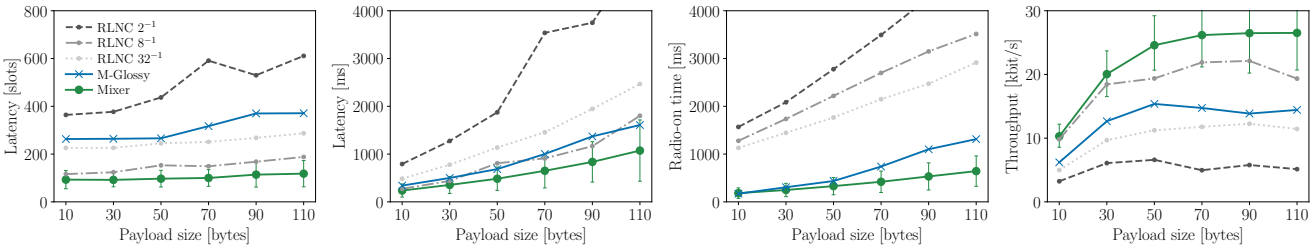
below, some nodes will decide to send in the next slot, allowing other nodes to capture SFD events. To align their transmissions, they perform meticulous timer polling. This way, nodes progressively establish a slotted communication. Nodes detect missing SFD events through timeouts, and estimate the missing timestamps based on previous events. If a node receives no packet for several consecutive slots (around ten), the estimations become inaccurate. In this case, a node drops out and resynchronizes with the next received packet.

As shown in Fig. 3, a node performs three main tasks in each slot: packet processing, transmit decision, and preparation for the next slot. We next discuss Mixer's transmission policy and then packet processing for innovative coding and efficient decoding.

Adaptive transmission policy. Our transmission policy aims to maximize the number of received packets per slot. To this end, the number (and selection) of transmitters should be well balanced—high enough (and spatially distributed) to reach many nodes, but not too high so capture can occur. Since real-world and especially dynamic networks have a varying node density, a fixed transmit probability performs poorly, as we demonstrate in Sec. 4. Therefore, each Mixer node maintains a list of received SenderIds (see Fig. 2) within the last H slots. Using this sliding-window history information, which is discarded at the end of a round, nodes monitor their current neighborhood to drive an *adaptive transmission policy* whose pseudocode is given in Algorithm 1. This policy updates the transmit probability p_t of a node and also incorporates a kind of local round-robin scheduling of selected slots. The level of determinism incurred by this policy increases with node density: In high-density regions nodes use stronger coordination than in sparsely populated areas. This *semi-coordinated transmission* scheme outperforms any fixed transmit probability we tested (see Sec. 4).

If a node decides not to transmit, it enters receive mode and sets a timeout to stop listening if no valid packet arrives within the slot.

Building innovative packets. A sender prepares a packet using RLNC: it adds every row (*i.e.*, packet) from its matrix to the packet with a probability of 1/2. In Mixer we add two features somewhat restricting the randomness to improve average performance. The first one is based on the insight that with high probability an innovative packet is also innovative for a node's neighborhood. Thus, a node adds every innovative packet immediately to the prepared transmit packet such that the innovation gets relayed with the next transmission for sure. Further, a node always adds its own message until it has received three packets that contain it. In this way, every



(a) Number of slots. **(b) End-to-end latency.** **(c) Radio-on time.** **(d) Throughput.**

Figure 4: Performance of Mixer, M-Glossy, and naïve RLNC configurations on the FlockLab testbed for different payload sizes. *Bars show the 25th and 75th percentiles. Mixer trades communication efficiency for compute power, outperforming M-Glossy across all metrics.*

node publishes its own message as soon as possible, which leads to a fast-growing coding potential at the beginning of a round.

Efficient runtime operation. We make four key design and implementation decisions to achieve an efficient operation of Mixer.

First, we choose to perform all computations on finite field GF(2), allowing for an efficient implementation of all operations on standard hardware. As a result, the coding vector contains one bit for each of the M messages to be exchanged. Hence, its size is M bits, which translates into $S_v = \lceil M/8 \rceil$ bytes in practice (see Fig. 2).

Second, we use incremental Gaussian elimination to process received packets and decode the original messages. Every node keeps a matrix of coding vectors and stores the corresponding payloads separately. After receiving a packet, a node uses the packet's coding vector to check whether it is innovative. To amortize the cost over all received packets, nodes keep the matrix of coding vectors in row echelon form, and incorporate the new coding vector into the matrix through step-wise row reduction. If the packet is innovative, then the number of rows of the matrix, which equals its rank due to its particular form, increases. Once the matrix reaches full rank, the original messages can be obtained in a final elimination step. This approach leads to low processing times and bounds the number of packets a node needs to store by the number of messages M.

Third, we try to parallelize radio and processing activities. As shown in Fig. 3, in a slot where a node receives a packet, processing can only commence after reception is completed. Instead, if a slot is used for transmission, a node can prepare the next packet while the current one is still being sent by the radio. Using the time budget of transmit slots in this way saves remarkable processing time in receive slots where a node decides to transmit in the next slot.

Fourth, we use the history information mentioned above to detect if all nodes in an area have full rank. A node indicates that it has full rank in the Flags field of the packets it transmits (see Fig. 2). If all neighbors have full rank, a node concludes that it is no longer helpful for the communication and turns itself off to save energy.

4 EVALUATION

Settings and metrics. We prototype Mixer on the TelosB platform, which features an MSP430 microcontroller (MCU) running at 4 MHz and an IEEE 802.15.4 radio operating in the 2.4 GHz band. We run experiments on $N = 27$ nodes of the FlockLab testbed [21], which form a 5-hop network using the maximum transmit power of 0 dBm.

We consider an all-to-all scenario, where each node starts with one message (*i.e.*, $M = N$). We test different payload sizes between 10 and 110 bytes in different runs. Each run lasts for 10 minutes.

We compare Mixer with the state-of-the-art solution that uses sequential Glossy floods, called M-Glossy. We also test three naïve RLNC configurations where each node adds any of its previously received packets with a probability of 1/2 and sends with a fixed probability of 1/2, 1/8, or 1/32. We carefully configure every protocol so that they all achieve a PDR of about 99.9 %.

We consider three performance metrics. *Latency* L is the time or the number of slots from the beginning of a round until all messages are received (or decoded in case of Mixer). *Throughput* is computed as $T = M \cdot S_p \,/\, L$ where S_p is the payload size. *Radio-on time* is the accumulated time the radio is turned on during a round. We report averages over all nodes and rounds, and 25th and 75th percentiles.

Results. Fig. 4 plots the performance of Mixer, M-Glossy, and the three naïve RLNC configurations for different payload sizes. We find that Mixer achieves the best results in all metrics.

To understand why, we first have a look at latency in terms of number of slots, which is a measure of communication efficiency. We see from Fig. 4a that Mixer needs significantly fewer slots than M-Glossy and is nearly unaffected by the payload size. Larger packets are more susceptible to transmission failures due to a longer air time. To counteract this effect, we had to increase the number of slots per flood in M-Glossy, from 10 slots for 10–50 bytes to 12 slots for 70 bytes and finally to 14 slots for 70–110 bytes. The fact that Mixer contains random processes leads to some scattering around the average values while M-Glossy is very predictable.

Fig. 4b plots latency in milliseconds. This is essentially the number of slots multiplied by their length, which accounts for the processing overhead. Because M-Glossy's processing overhead is less than 100 MCU cycles, its latency is dominated by the packet air time, which increases with the payload size. By contrast, Mixer has a larger processing overhead that grows with the payload size, but needs considerably fewer slots. Hence, Mixer trades communication efficiency for compute power, which is an important property. Overall, Mixer reduces average latency by 30–40 % compared with M-Glossy, which translates into a 40–65 % average speedup.

The energy consumption of a node is determined by its MCU and radio activities. Both Mixer and M-Glossy keep the MCU on during a slot, which allows to derive the MCU time directly from the latency in Fig. 4b. As Fig. 4c shows, Mixer outperforms M-Glossy in terms of radio-on time with increasing payload size. This result follows from the fact that Mixer needs fewer slots than M-Glossy and that nodes turn themselves off when they are no longer needed.

Looking at Fig. 4d, we see that Mixer provides the highest throughput, outperforming M-Glossy by 58–91 %. Throughput with Mixer

increases from 10.2 kbit/s with 10 bytes to 26.5 kbit/s with 110 bytes. M-Glossy's throughput peaks at 15.3 kbit/s with 50-byte payloads.

Finally, we learn from the results of the naïve RLNC configurations that choosing the right transmit probability is important. Transmitting with a probability of 1/2 is too aggressive: the number of transmitters per slot is too high for capture to work. As a result, packets are received with low probability such that rounds require significantly more slots. On the other hand, a probability of 1/32 is too low: many slots are unused in this case. A probability of 1/8 performs somewhat better and provides a performance comparable to M-Glossy. We suppose there is a sweet spot for this parameter that depends on the average node density. However, nodes in dense areas should be more reluctant than nodes in sparsely populated areas. Mixer's semi-coordinated transmission policy automatically adapts to different local node densities and provides the best results.

5 RELATED WORK

Ahlswede et al. introduced network coding, showing that it achieves the multicast capacity of wireline networks [1]. It was found that these bounds can be reached using linear codes, and that encoding and decoding can be done in polynomial time [16, 20]. This also holds if nodes pick random coding coefficients [13]. These works form the theoretical foundation of RLNC, which we exploit in Mixer.

Network coding has been extensively used in wireless networks, albeit for different purposes than in Mixer. Among the practical works, COPE [15] and MORE [5] are the first implementations of network coding focusing on one-to-one and one-to-many traffic, respectively. Other recent multicast protocols like Pacifier [17] improve on MORE in Wi-Fi networks, and Pando [6] uses fountain codes to provide fast one-to-all data dissemination in IEEE 802.15.4 networks. Compared with Mixer, these works target stationary wireless networks and focus on different communication patterns.

Chaos [18] is a primitive for computing network-wide aggregates in practical wireless networks. By contrast, many-to-many broadcast has only been studied in theory [10] or in simulation [9], presenting encouraging optimality results and performance gains using RLNC. Mixer bridges the gap between theory and practice.

6 CONCLUSIONS AND NEXT STEPS

A new breed of cyber-physical applications is currently emerging that either requires or would greatly benefit from an efficient and reliable many-to-many broadcast primitive to unfold its full potential. In this paper, we have reported on our ongoing efforts to design such a communication primitive, called Mixer, for dynamic embedded wireless networks. Our preliminary results are promising, demonstrating that our approach of integrating synchronous transmissions with random linear network coding provides significant performance gains over the state of the art and high reliability.

Nevertheless, we still see a huge potential for even higher performance gains by improving Mixer's design and implementation along the following dimensions. First, we are conceiving mechanisms to accelerate both the initial and the final phase of a Mixer round. Initial results from simulations are promising, showing that our mechanisms do reduce the number of slots. Second, additional code optimizations to, for example, further parallelize the radio and processing activities, will make each slot shorter. Third, we intend to exploit an important property of Mixer, its ability to translate higher compute power into shorter communication latency and higher throughput, which is not possible with Glossy. To this end, we are porting Mixer to state-of-the-art ARM Cortex-M based platforms, using packet traces collected from testbeds to extrapolate the performance one would get if the outdated TelosB was replaced.

Acknowledgments. We thank our anonymous reviewers for their useful feedback. This work was supported by the German Research Foundation (DFG) within cfaed and SPP 1914, project EcoCPS.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. 2000. Network Information Flow. *IEEE Trans. Inf. Theory* 46, 4 (2000).
[2] J. Åkerberg, M. Gidlund, and M. Björkman. 2011. Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation. In *INDIN*.
[3] BBC. 2017. US military tests swarm of mini-drones launched from jets. http://www.bbc.com/news/technology-38569027. (2017).
[4] V. D. Blondel and J. N. Tsitsiklis. 2000. A Survey of Computational Complexity Results in Systems and Control. *Automatica* 36, 9 (2000).
[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. 2007. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM*.
[6] W. Du, J. C. Liando, H. Zhang, and M. Li. 2015. When Pipelines Meet Fountain: Fast Data Dissemination in Wireless Sensor Networks. In *SenSys*.
[7] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. 2012. Low-power Wireless Bus. In *SenSys*.
[8] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. 2011. Efficient Network Flooding and Time Synchronization with Glossy. In *IPSN*.
[9] C. Fragouli, J. Widmer, and J. Le Boudec. 2008. Efficient Broadcasting Using Network Coding. *IEEE/ACM Trans. Netw.* 16, 2 (2008).
[10] B. Haeupler and F. Kuhn. 2012. Lower Bounds on Information Dissemination in Dynamic Networks. In *DISC*.
[11] S. Hayat, E. Yanmaz, and R. Muzaffar. 2016. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Commun. Surveys Tuts.* 18, 4 (2016).
[12] H. Hellwagner and C. Bettstetter. 2016. Networking research challenges in multi-UAV systems. https://bettstetter.com/uav-networking-challenges/. (2016).
[13] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. 2006. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Inf. Theory* 52, 10 (2006).
[14] S. Katti, S. Gollakota, and D. Katabi. 2007. Embracing Wireless Interference: Analog Network Coding. In *SIGCOMM*.
[15] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. 2006. XORs in the Air: Practical Wireless Network Coding. In *SIGCOMM*.
[16] R. Koetter and M. Medard. 2003. An Algebraic Approach to Network Coding. *IEEE/ACM Trans. Netw.* 11, 5 (2003).
[17] D. Koutsonikolas, Y. C. Hu, and C. Wang. 2012. Pacifier: High-throughput, Reliable Multicast Without "Crying Babies" in Wireless Mesh Networks. *IEEE/ACM Trans. Netw.* 20, 5 (2012).
[18] O. Landsiedel, F. Ferrari, and M. Zimmerling. 2013. Chaos: Versatile and Efficient All-to-all Data Sharing and In-network Processing at Scale. In *SenSys*.
[19] K. Leentvaar and J. Flint. 1976. The Capture Effect in FM Receivers. *IEEE Trans. Commun.* 24, 5 (1976).
[20] S. R. Li, R. W. Yeung, and N. Cai. 2003. Linear Network Coding. *IEEE Trans. Inf. Theory* 49, 2 (2003).
[21] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. 2013. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. In *IPSN*.
[22] M. Luvisotto, Z. Pang, and D. Dzung. 2016. Ultra High Performance Wireless Control for Critical Applications: Challenges and Directions. *IEEE Trans. Ind. Informat.* 13, 3 (2016).
[23] L. Mottola and G. P. Picco. 2011. MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks. *IEEE Trans. Mobile Comput.* 10, 12 (2011).
[24] B. Nazer and M. Gastpar. 2011. Compute-and-Forward: Harnessing Interference Through Structured Codes. *IEEE Trans. Inf. Theory* 57, 10 (2011).
[25] H. Rahul, H. Hassanieh, and D. Katabi. 2010. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *SIGCOMM*.
[26] F. B. Schneider. 1990. Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.* 22, 4 (1990).
[27] M. Wilhelm, V. Lenders, and J. B. Schmitt. 2014. On the Reception of Concurrent Transmissions in Wireless Sensor Networks. *IEEE Trans. Wireless Commun.* 13, 12 (2014).
[28] D. Yuan and M. Hollick. 2013. Let's Talk Together: Understanding Concurrent Transmission in Wireless Sensor Networks. In *LCN*.