Network On or Off? Instant Global Binary Decisions over UWB with Flick

Enrico Soprana, Matteo Trobinger, Davide Vecchia, Gian Pietro Picco
{enrico.soprana,matteo.trobinger,davide.vecchia,gianpietro.picco}@unitn.it
University of Trento, Italy

ABSTRACT

In many low-power wireless systems, a condition occurring at some nodes (e.g., an anomalous sensor sample, an aperiodic packet to transmit, a new joining node) determines whether the entire network should be awake (e.g., to react to the anomaly, deliver the packet, update the node group) or enter sleep. State-of-the-art protocols exploit periodic network-wide floods based on concurrent transmissions (e.g., via Glossy) to establish the global decision quickly, reliably, and efficiently. Still, time is of the essence: the faster the network agrees, the faster it either reacts or enters sleep.

Flick achieves this global decision with order-of-magnitude latency improvements and 5-nines reliability by detecting and disseminating a binary ON/OFF vote via the preamble of a packet instead of its full content. The actual realization of this simple idea entails several techniques on the ultra-wideband (UWB) radios we use. We evaluate Flick over a 78-node network in the Cloves testbed showing that, e.g., a single node can globally flick the switch to ON across a 10-hop diameter in <500μs, i.e., roughly the time for a Glossy packet to go across a *single* hop, and with 4.4× less energy. We demonstrate the potential of Flick by integrating it into staple protocols and evaluating the performance improvements it enables.

CCS CONCEPTS

• **Networks → Network protocol design**.

KEYWORDS

Low-power wireless, concurrent transmissions, ultra-wideband.

ACM Reference Format:
Enrico Soprana, Matteo Trobinger, Davide Vecchia, Gian Pietro Picco. 2023. Network On or Off? Instant Global Binary Decisions over UWB with Flick. In *The 22nd International Conference on Information Processing in Sensor Networks (IPSN '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3583120.3586967

1 INTRODUCTION

Low-power wireless communications have become an integral part of many modern systems and applications, where they bridge the digital world with the physical one. Due to the unpredictability of the latter and the energy concerns of the former, several systems exhibit a pattern of operation where the network becomes active at designated times to check whether a given condition holds at some node. If so, the network remains awake to perform a required action; otherwise, it returns to sleep. Prominent systems exhibiting this pattern include event-triggered ones, aperiodic data collection stacks, and distributed schedule updates. The review of state-of-the-art examples serves as motivation for our work (§2).

The role of concurrent transmissions. These systems resolve the tension between quickly reacting to condition occurrence and minimizing energy consumption by relying on concurrent transmissions (CTX). Tightly-synchronized transmissions do not necessarily result in a collision; instead, one of the concurrent packets is received with very high probability, under some PHY-level constraints. Glossy [6] popularized CTX-based network-wide flooding on IEEE 802.15.4 radios by achieving unprecedented improvements in reliability, latency, and energy consumption. Since then, a large body of literature [25] has shown that CTX achieve similarly remarkable benefits for different radios and several traffic patterns, including those exploiting the condition pattern above.

Flick in a nutshell. Nevertheless, in the systems mentioned above (§2), time is of the essence: when a check is scheduled, the faster the network agrees on the binary ON/OFF condition the faster it can either take action, ensuring reactivity, or enter sleep, saving energy. This is where Flick comes into play, enabling order-of-magnitude faster global decisions w.r.t. conventional Glossy-like floods.

Figure 1 visually illustrates the concept: *the dissemination of a binary condition in Flick completes across the entire network in roughly the same time it takes Glossy to complete a single hop.* At the core of this remarkable result is a simple observation. Glossy, like other CTX-based approaches, is designed to transmit packets; at each hop, the *entire* packet must be received before its (concurrent) transmission across the next hop can begin. However, this is wasteful when only a binary condition must be disseminated. In this case, it is sufficient to detect the *intent* of other nodes to communicate, achieved in Flick by detecting the *preamble* preceding a packet. The radio needs not listen to the whole preamble; once its presence is detected, the transmission of a new one can immediately start, effectively repropagating the binary condition. This process requires only a few μs, instead of the hundreds of μs necessary to packet reception and subsequent retransmission, explaining the order-of-magnitude latency improvement hinted at in Figure 1.

Goals, methodology, and contributions. We aim to design a novel network primitive acting as a global network switch of sorts that, when used at designated times, can be flicked to ON by nodes where a given condition occurs, or left OFF otherwise. Realizing this goal via the simple preamble-based scheme above, requires a combination of techniques entailing low-level aspects of the radio, e.g., the ability to precisely control timeouts and interrupts and the

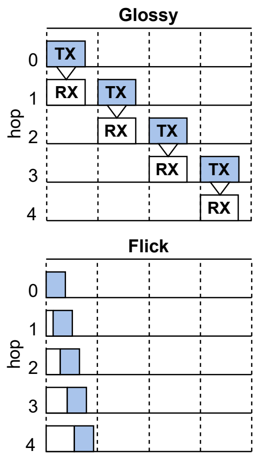


Figure 1: Latency: Glossy vs. Flick.

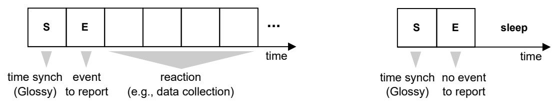
mechanics of preamble hunting. These are offered by the popular Qorvo DW1000 UWB radio we target, although the concepts are in principle applicable to others. We concisely summarize the relevant UWB features (§3) before illustrating the design of Flick (§4).

For this network switch to bring practical benefits, it must be not only very fast but also extremely reliable, to guarantee that *all* nodes take the same decision. We perform an extensive evaluation (§5) in a 90-node portion of the Cloves [13] testbed, where we deactivate 12 nodes to obtain a challenging 10-hop network diameter. Our results, based on ≈25 million attempts, show that Flick correctly detects the corresponding on condition in 99.99976% of cases, therefore approaching 6-nines reliability. As expected, these detections are extremely fast, as Flick propagates information across one hop in only ≈47µs on average vs. the ≈472µs needed by Glossy.

From a configuration standpoint, Flick depends on a single parameter, the maximum listening duration. This inherent simplicity enables us to contribute a simple analytical model, validated in our experimental setup, to properly configure this parameter as a function of the network diameter, whose knowledge is generally required by CTX-based approaches. The low latency translates in significant energy savings, which we amplify by exploiting the energy vs. latency tradeoffs enabled by an intermittent preamble hunting mode (Sniff) offered by the radio, achieving up to a 4.4× reduction in energy consumption w.r.t. Glossy.

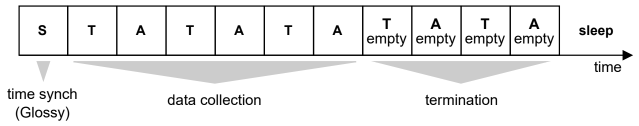
Interestingly, not only the benefits above hold regardless of the number of nodes triggering the on condition but, contrary to what happens in packet-based CTX approaches, its increase significantly decreases both latency and energy consumption without detriment to reliability.

Despite the remarkable performance of Flick, a question remains about its impact once integrated in a complete system. We first answer in general terms (§6) by connecting the results from the evaluation above (§5) to the representative classes of systems we considered (§2). We then focus on the most complex among them, aperiodic data collection, and evaluate Flick-enabled variants of Crystal and Weaver against the original ones (§7), offering at once a concrete example of how to integrate Flick in state-of-the-art

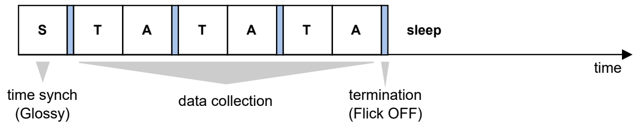


(a) Event detected. **(b) No event detected.**

Figure 2: Event-triggered operation in eLWB and WCB.



(a) Original.



(b) With Flick.

Figure 3: Collection and termination in Crystal.

protocols and a quantitative assessment of the performance improvements it enables. We end the paper (§8) with brief concluding remarks including opportunities for future work.

2 RELATED WORK, MOTIVATION, AND APPLICABILITY

Event-triggered systems are a first prominent example among those potentially benefiting from Flick. For instance, the system in [17] is designed to monitor the environment and, only if and when acoustic emissions are detected, report the corresponding data. A similar scheme is at the core of event-triggered control (ETC) [19] that, in contrast to classic periodic sensor sampling, unlocks communication savings by reporting samples only when these deviate from a model ensuring certain control convergence properties. In these systems, when multiple sensor nodes trigger, possibly simultaneously, their data must be reported quickly and reliably; otherwise, the entire network should be asleep, to preserve energy. Figure 2 depicts at a high level how these conflicting requirements are tackled by eLWB [17] and WCB [21]. Both exploit a periodic schedule starting with a timeslot (S) for the sink-initiated Glossy flood time-synchronizing the network, common in CTX-based protocols, followed by a timeslot (E) dedicated to an "event" flood. This can be initiated by one or more nodes where the triggering condition occurred, signaling the need for the entire network to remain awake to react, e.g., by collecting data from some [17] or all [21] nodes; conversely, the absence of transmission implies the absence of triggering, enabling the network to re-enter sleep.

Aperiodic data collection offers a more complex example where packet transmissions occur unpredictably based on *local* decisions of the sending nodes. In contrast, in event-triggered systems, data transmission by one node is required only if and when an (aperiodic) event occurs, possibly at a *different* node. The network-wide event flood (Figure 2) is therefore key to trigger data transmissions at nodes *other than* those detecting the event, yet becomes wasteful when a node can transmit of its own volition as in aperiodic data collection; a different strategy is required.

Figure 3a offers a high-level view of the Crystal [8] protocol. After the initial synchronization flood (S) each node has the opportunity to flood a packet (T). If multiple nodes decide to do so, CTX properties guarantee reception of one of their packets at the sink, whose subsequent flood (A) acknowledges the successful sender, enabling the others (if any) to re-attempt their flood while catering also for the occasional packet loss from the sender. This interleaving of (nodes) transmission and (sink) acknowledgment floods continues until no node attempts the former and therefore the latter is empty. However, to ensure correct termination, a single empty flood may not be enough as it may be caused by packet loss rather than absence of senders. Multiple T–A iterations may be required in noisy environments [9], making termination the crucial factor in the reliability vs. energy tradeoff. Notably, this termination phase is required because the number of senders is unknown—another difference w.r.t. the event-triggered systems above where the nodes from which to collect reports is known.

In the same class of systems, Weaver [20] improves performance by relying directly on individual CTX, weaving multiple transmissions into a single flood as they occur. Nevertheless, it also relies on sink-based global acknowledgments and a termination phase, although both with different mechanics w.r.t. Crystal.

Distributed schedule updates offer another example that, on the contrary, is very simple in its mechanics. Classic CTX-based stacks, e.g., LWB [7], allow nodes to request the possibility to transmit in the next communication period (epoch) by performing a Glossy flood in a dedicated contention-based slot at the end of the current epoch. The LWB controller receives the packet from one of the competing nodes (if any) and updates the schedule disseminated at the beginning of the next epoch. However, this scheme is wasteful in the (common) situation where the system is performing mostly periodic collection at pre-defined intervals and only occasionally additional traffic needs to be provisioned; all nodes are awake to help potentially relay a request that may never occur.

A similar scheme, and related concerns, are found in completely different settings. For instance, the SociTrack [2] system provides continuous tracking of interaction among people by exploiting a combination of BLE, for neighbor discovery, and UWB, for ranging. When a group of devices is joined by a new one, it is discovered via BLE by one or more devices and its presence announced via UWB to the entire group, to enable the global update of the schedule of ranging exchanges among group members. This join announcement occurs by disseminating a packet with information about the new node via a group-wide Glossy flood, performed in a dedicated slot periodically repeating in the group communication schedule over UWB. This latter aspect is crucial. A short period ensures fast inclusion of joining nodes but also high energy consumption, especially considering that UWB is roughly 10× more expensive than BLE in this respect. A large period mitigates this problem at the cost of delaying joining nodes, possibly affecting the accuracy of interaction tracking. Moreover, during periods in which the group membership is stable, both solutions incur the "idle listening" mentioned above for LWB, with nodes wasting precious energy listening for joining announcements that are not happening.

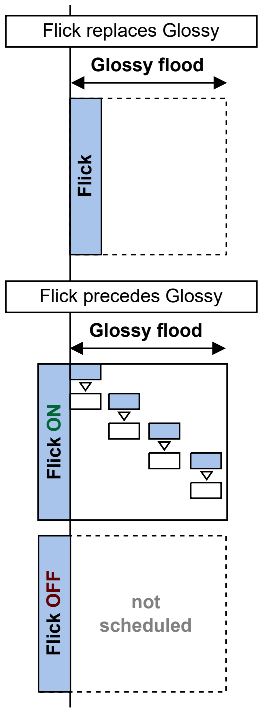


Figure 4: Using Flick instead of, and along with, Glossy floods.

How does Flick help? In the contexts above, the ability to quickly and globally establish a binary condition offered by Flick can be exploited in at least two ways (Figure 4).

In the case of event-triggered systems, Flick can *directly* represent the presence or absence of an event, therefore *completely replacing* the corresponding Glossy flood and significantly reducing both latency and energy (§5).

At the other extreme, in the case of distributed schedule updates, Flick can be exploited to minimize the network-wide idle listening by Flick *preceding*, rather than replacing, the Glossy flood. Nodes with information to transmit (e.g., a schedule request in LWB, a joining announcement in SociTrack) perform first an on round with Flick, followed by the intended Glossy flood. This allows the other nodes to check whether this flood is actually expected. If so, they remain active to participate in it; otherwise, an off round is detected and nodes can safely go immediately to sleep, saving precious energy. In a sense, this is reminiscent of the classic argument about *"the most consequential decision that a low-power link makes: whether to stay awake or go to sleep after probing the channel"* [5], but applied to the entire network rather than a single link. Along the same lines, Flick can be seen as the (very fast and reliable) network-wide equivalent of a Clear Channel Assessment (CCA) in MAC protocols [15].

In aperiodic data collection, this use of Flick before each transmission (T) flood makes the termination phase superfluous, therefore removing the main source of energy consumption (Figure 3b). In the original Crystal, when no node intends to transmit, energy consumption is determined by the execution of no less than three

Glossy floods: the initial synchronization (S) plus data transmission (T) and acknowlegment (A) without packets exchanged. This is actually the common case in the sparse, aperiodic traffic targeted by this class of systems. With Flick, the same case is handled by the initial S flood plus an off round, achieving nearly a threefold reduction in energy consumption. On the other hand, this comes at a slight increase in latency when packets are indeed present (Figure 3). However, this is very small due to the short duration of Flick (§5); further, it is also compensated by the removal of termination. We later show quantitative results corroborating these claims (§7), for both the Crystal and Weaver systems.

Flick vs. wake-up radios. The notion of a "radio switch" is also reminiscent of wake-up radios (WuR) [14], which have been applied also in the CTX context. Zippy [16] targets event-triggered systems with specialized WuR hardware offering very low-power consumption, used to disseminate via a CTX flood the detection of events, assumed to occur rarely. In turn, BLITZ [18] exploits Zippy to trigger the execution of an LWB-like communication round. In a sense, Zippy plays the role of the event flood in eLWB and WCB, pursuing the same "network switch" functionality we achieve in Flick. The added advantage is that the very low power consumption enables the WuR to remain always on and ready to trigger at any time rather than in designated timeslots. On the other hand, it requires additional, custom WuR hardware, while we demonstrate the use of Flick on off-the-shelf hardware. Moreover, WuR generally have a shorter communication range than the main radio, effectively limiting the range of the latter to the former [14]; the range reported in [16] is up to 15 m indoor and 30 m outdoor vs. the ≈80 m range achieved by our UWB radios. Finally, Flick demonstrates superior performance. In [16, 18], the end-to-end latency for wake-up events is up to 100 ms with a reliability ≥90% in a 4-hop, 13-node network, a far cry from the <500µs latency and 5-nines reliability we report for Flick in a 10-hop, 78-node network (§5).

Computing network-wide OR/AND conditions. Finally, we observe that globally establishing a binary on/off value can be straightforwardly generalized to computing a boolean value via network-wide OR/AND operators. An OR is obtained in Flick by mapping on ≡ true and off ≡ false. All nodes are initially false (off); if one or more nodes initiate the dissemination of a true value (on), the global result becomes true at all nodes. In this respect, the mechanics of Flick is reminiscent of a wired-OR logic connection in electronics, where all circuit inputs are by default low, and a single high input is enough to drive the output to high (true). Interestingly, an AND is obtained with the reverse mapping (on ≡ false, off ≡ true), again, similar to the implementation of a wired AND obtained by reversing the active levels (low vs. high).

These simple observations extend the potential of Flick beyond the protocol-level examples discussed so far and enable its use directly at the application layer, e.g., to globally set the boolean values of shared state variables. However, hereafter we focus on the former on/off case as it allows us to *quantitatively* ascertain the impact of Flick on the performance of existing systems.

3 ULTRA-WIDEBAND

The basic principles of Flick are supported by several radios, but we identify a strong compatibility with ultra-wideband (UWB), for which we provide a brief overview of the characteristics we exploit.

UWB is a low-power radio technology renowned for ranging and localization applications. Its decimeter-level error is unmatched by other popular RF-based technologies in these domains, e.g., Bluetooth and WiFi. In contrast, UWB is seldom considered for communication systems in favor of technologies such as Bluetooth and IEEE 802.15.4 narrowband, due to their lower energy consumption. However, the relatively high consumption of UWB is deceptive, because packets take much less time to transmit and receive. This is due to a short preamble sequence and a high data rate, both by virtue of UWB physical encoding, based on short pulses (≈ 2 ns).

In more detail, an UWB packet (Figure 5) starts with a preamble of repeated symbols, each composed of a standard-defined sequence of pulses of ≈ 1 μs duration. These sequences can be detected by a transceiver in *preamble hunting* mode by correlating the expected signal with the incoming one in windows of few μs, called preamble acquisition chunks (PAC). To detect the incoming signal, it is sufficient for the radio to hunt for only one or few PAC, without the need to receive all symbols. After preamble detection, the radio begins searching for the start-of-frame delimiter (SFD). The SFD indicates the beginning of the physical header (PHR) and the payload, both encoded via BPM-BPSK. The PHR is limited to 850 kbps, but the payload enjoys a 6.8 Mbps data rate, already accounting for forward error correction.

If properly harnessed, these characteristics can unlock reliable, fast, and energy-efficient communication. In this respect, recent works [12, 22] have shown that UWB radios support concurrent transmissions (CTX), further amplifying the potential of UWB in the low-power communications landscape. Our proposed primitive, Flick, while also relying on CTX, offers a complementary technique to these mainstream packet-based approaches.

4 DESIGN

Many low-power wireless protocols depend on deceptively simple, yet crucial binary decisions, e.g., whether nodes should stay awake or enter sleep mode, which should be taken timely and reliably by the whole network. With Flick, we tackle this problem at the root by providing an unprecedentedly fast, reliable, and energy-efficient communication primitive for the dissemination of a binary decision across multi-hop wireless networks.

Basic principles. The key insight behind Flick operation is that a full packet reception is not needed to disseminate a binary decision network-wide; it is sufficient to detect and synchronously repropagate the preamble. In principle, this approach can considerably reduce latency w.r.t. staple packet-based solutions; preamble detection takes few μs, compared to hundreds of μs for full message reception and retransmission (Figure 6).

In each Flick execution (or, hereafter, round), a node is termed *initiator* if it transmits to start a preamble flood. For example, it may be a sensor encountering an event-triggering condition and signaling the network to take some action by "flicking the network switch". If there is at least one initiator, the Flick round is on, otherwise it remains off, the default state. The proposed approach

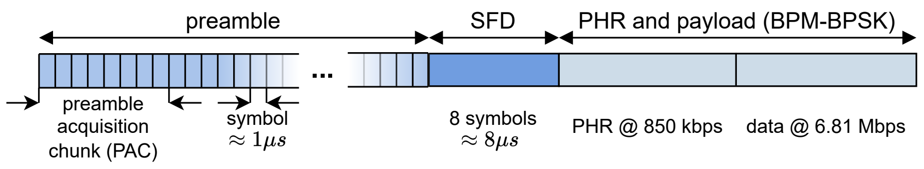
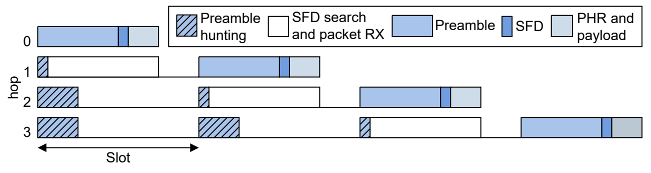
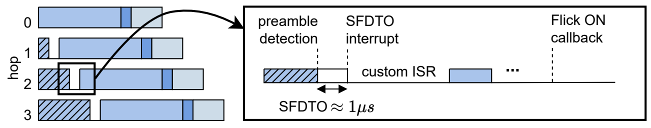


Figure 5: UWB packet.



(a) Glossy.



(b) Flick.

Figure 6: Glossy vs. Flick operation: A detailed view.

propagates the signal extremely quickly in ON rounds and enables a very short listening time in OFF ones, therefore reducing energy consumption even when there is no decision to disseminate.

How does it work? When a Flick ON round begins, initiators transmit a legit UWB packet composed of a 64-symbol preamble (the shortest allowed, ≈64 μs) followed by the start-of-frame delimiter (SFD), the physical header (PHR), and the smallest payload allowed by the radio (1B data plus 2B for CRC). Non-initiators begin in RX mode, scanning the medium for a preamble with a PAC size of 8 preamble symbols, again the shortest allowed. The rationale behind these choices is that the shorter the PAC, the faster a node can detect a preamble, and therefore the smaller is the flood latency; similarly, the shorter the preamble and the payload, the lower the energy consumption due to their transmission.

Upon preamble detection, which at the first hop typically occurs in a few PAC, nodes immediately switch to TX, effectively leading to a CTX flood (Figure 6b). They do so by exploiting the SFD timeout (SFDTO) of the UWB radio, but in an unconventional way. Normally, the SFDTO is meant to keep the radio from lingering in RX mode for too long in case of false preamble detection, generating an interrupt if the SFD signal is not received in time. In Flick, instead, we *i)* configure the SFDTO to trigger 1 μs after a preamble detection, and *ii)* leverage a custom interrupt service routine (ISR) to switch the radio to TX as quickly as possible upon this event, thereby reducing the overall flood duration. After transmitting the packet, nodes switch the radio off and report an ON Flick round to the higher layers.

Improving latency and reliability. To fully unleash the potential of Flick, we integrate the basic procedure above with few key enhancements. We further optimise the RX-TX turnaround time by *i)* postponing all non-critical (yet time-consuming) radio operations in the ISR, e.g., radio resets upon RX errors and timeouts, till after the end of the Flick round, and *ii)* avoiding to load in the TX buffer of the radio the data portion to be transmitted. This latter optimisation causes a corrupted frame (wrong CRC) to be sent, which notably is not an issue in Flick as operations are SFDTO-driven and therefore the received payload is ignored. Nonetheless, we observe that, in rare cases, the UWB radio may miss the preamble and yet directly detect the SFD; if this happens during a Flick round, nodes attempt packet decoding and raise an RX error, potentially blocking the flood. To prevent this from hampering reliability, we configure nodes to react to any RX error (e.g., due to a wrong CRC) in the same way as when a preamble is detected, and immediately start to transmit. After all, SFD detection still signals with high probability the intent of a neighbor to transmit, in line with the basic principle behind Flick.

These enhancements, together, make Flick extremely fast in addition to reliable, directly benefiting energy consumption. Nonetheless, further improvements are possible on this front.

Reducing energy consumption. Preamble hunting is a complex and power-consuming operation of the DW1000 [3]. Keeping nodes in this radio state until a preamble is detected exposes Flick to significant energy costs, yet it is often unnecessary; until neighbors begin their transmissions, remaining in RX yields no benefit. Alternatively, probing the channel with a lower periodicity, up to the preamble duration, should suffice for detecting an incoming flood and ensuring correct operation, with clear energy improvements.

In Flick, we follow exactly this approach, exploring increasingly aggressive radio duty-cycling strategies to replace continuous listening. We accomplish this by leveraging a special, low-power, preamble hunting variant offered by the DW1000, called *Sniff mode* ([3], p. 35). In brief, Sniff enables receivers to rapidly cycle between preamble hunting and idle states, based on a configurable period T_{idle}. If during the radio-on time a preamble is detected, the node automatically switches to TX and repropagates the Flick flood one step further.

5 EVALUATION

We investigate the reliability, latency and energy consumption of Flick, comparing it to mainstream packet-based dissemination.

5.1 Experimental Setup

Hardware and testbed. We evaluate Flick in Cloves [13], a large-scale, public UWB testbed at our premises. We focus on the largest 90-node portion, deployed in two adjacent buildings connected by a suspended metallic passageway (Figure 7). Each node hosts a Qorvo EVB1000 board equipped with a DW1000 UWB radio controlled by a STM32F105 ARM Cortex M3 MCU, a ST-LINK V2

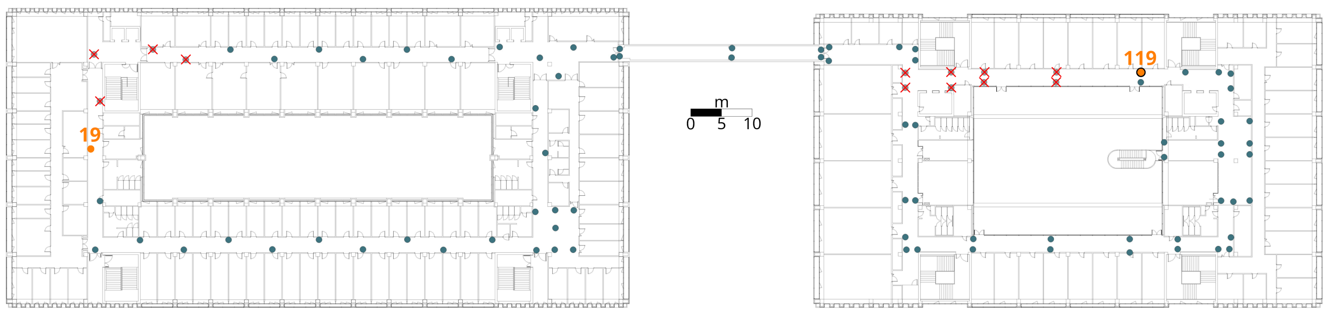


Figure 7: The 90-node portion of the CLOVES [13] testbed used. The 12 nodes we disabled are marked with a cross. Node 19 and 119 are at the extremes of the 10-hop network diameter; the latter node also serves as the sink for the experiments in §7.

programmer, and a Raspberry Pi 3 (RPi). The latter is connected to a dedicated Ethernet infrastructure, enabling the automation and remote execution of tests and the collection of logs.

Network characteristics. Nodes are installed on the ceiling above narrow and long office corridors [13], yielding a mostly linear topology where strong signal reflections are likely, especially within the metallic passageway. We disable 12 nodes on the top left corner of both buildings, to prevent communication across these edges. This reduces the actual size of the network to 78 nodes, still larger than many testbeds available, but *i)* increases the network diameter to a rather challenging value of 10 hops, and *ii)* severely limits spatial diversity, often forcing floods to proceed along a single path. This is exacerbated in the left building where receiver redundancy is brought to the minimum, with a single node deployed on the bottom left corner connecting the left corridor with the rest of the network. Overall, this setup provides a realistic, yet particularly challenging, indoor testing environment, ideal to evaluate communication stacks and push their performance to the limit.

Radio configuration. We use a 64-MHz pulse repetition frequency (PRF) on channel 4, and the TX power recommended by the manufacturer [4] for this combination of settings. As discussed in §4, we exploit the shortest UWB preamble length and PAC size (≈64 μs and ≈8 μs, respectively) along with the highest 6.8 Mbps DW1000 data rate commonly used in the literature.

Packet-based flood baseline. State-of-the-art approaches typically rely on Glossy for disseminating binary decisions (§2), making it the natural packet-based flood baseline to compare FLICK against.

Glossy is governed by two main parameters, N and H. The former is the number of retransmissions from each node; $N > 1$ is typically used to improve dissemination reliability. H is the maximum number of hops a packet could traverse; together with N, it is used to allocate the total time for the flood.

To implement Glossy, we take advantage of Time Slot Manager (TSM) [20], a publicly available layer providing core functionality supporting CTX-based protocols. Indeed, programming atop TSM allows us to tightly synchronize nodes and control re-transmission times, key to reduce Glossy energy consumption and optimize the behavior of our baseline, enabling a fair comparison. Specifically, instead of keeping the radio on for the entire flood duration, we limit listening to half the preamble duration (≈32 μs) at the beginning of each RX slot; we verified that reducing listening further impairs reliability. The slot duration is set to 472 μs to accommodate the TX and RX of the 7-byte TSM header, a 1-byte payload, and all radio operations including reading and writing into the radio buffer. Finally, we employ the "TX-only" Glossy variant first introduced by the system [11] that won the EWSN'17 competition. Unlike the original Glossy [6], which alternates between TX and RX, this variant groups all TX slots back-to-back after the initial (and only) RX, yielding a more energy-efficient alternative when $N > 1$.

5.2 Reliability

The question whether FLICK can be exploited in practice hinges on its reliability, whose evaluation is the first target of our experimental campaign. In an ON round, non-initiators should detect the UWB preamble (i.e., false negatives should be infrequent). Conversely, in an OFF round, nodes should not mistakenly detect a preamble (i.e., false positives should be infrequent).

We start by analyzing reliability in the first case, false negatives in ON rounds, and determine the average network-wide ON condition detection rate (*CDR*), i.e., the sum of ON rounds correctly detected at each node over the total number of detection attempts across all nodes in the network. A single test begins by re-synchronizing network nodes with a Glossy flood, followed by 100 FLICK ON rounds. This test is repeated every 0.5 s for half an hour, amounting to ≈325200 ON rounds, i.e., more than 25 million detection attempts across all nodes. Each FLICK round is performed by a randomly-selected initiator, effectively sweeping all network nodes. False negatives are extremely rare: *CDR* = 99.99976%. This approaches 6-nines reliability, one order of magnitude above the 5-nines commonly considered the desirable threshold for, e.g., industrial control applications.

What remains to be assessed is whether this high reliability in detection comes at the cost of false positives in OFF rounds, which could affect the energy consumption of FLICK in real systems by keeping nodes awake when they should instead enter sleep. To study this aspect, we follow the same methodology above, this time scheduling OFF rounds (no initiator) and logging any preamble detection. False positives occur very rarely; the fraction of incorrect ON decisions at each node over all detection attempts across the network is as low as 0.0032%, demonstrating that our approach is indeed applicable in practice even in large networks.

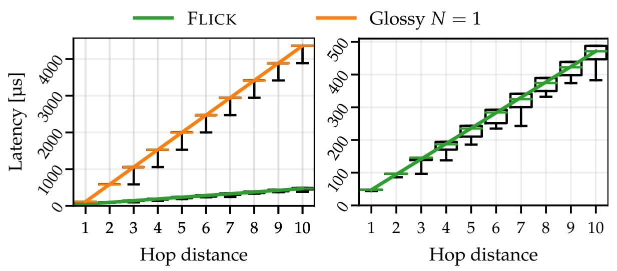


Figure 8: Latency per hop: Flick vs. Glossy ($N = 1$). The right chart offers a zoomed-in view for Flick.

For comparison, we report the reliability of Glossy in the same experimental setup and using the same *CDR* metric. Interestingly, Glossy is not affected by false positives, as detection occurs only when the radio reports a correct packet RX. Nevertheless, when configured with $N = 1$, Glossy achieves $CDR = 99.9983\%$, which increases to $CDR = 99.99969\%$ with $N = 2$. The latter, more expensive, configuration is necessary to achieve a reliability comparable to Flick while retaining the advantages of Glossy, i.e., the absence of false positives and the ability to propagate a full packet instead of a binary decision. On the other hand, Flick greatly outperforms Glossy on latency and energy consumption, as we show next.

5.3 Latency

When studying the latency of Flick we design experiments differently, as the main focus is the time is takes for the preamble-based flood to propagate through a multi-hop wireless network. Crucially, we inspect the detection time separately for each network hop, as this allows us to generalize our findings to networks of different size and density. As in the reliability experiments, we exploit a periodic Glossy flood to re-synchronize the network every 100 Flick rounds, and repeat the procedure to accrue 325200 on rounds. Upon the first Glossy RX, nodes log their hop count. This is done to assign each node to a given hop, and aggregate latency results accordingly. Different from reliability experiments, however, we exploit a single node at one end of the network (node 19 in Figure 7) as the sole initiator for both Glossy and Flick, enabling *each* flood to propagate across *all* the 10 hops of our topology, which is key to investigate the per-hop latency.

Figure 8 compares the median detection time of Flick and Glossy at different hops. Results showcase the superiority of our primitive: Flick unlocks *order-of-magnitude latency improvements* w.r.t. Glossy. In the time a Glossy flood propagates across a *single* hop, Flick disseminates the on decision *across the entire 10-hop network.* Also, receivers can more easily detect a preamble (Flick) than receive a full packet (Glossy). This fact occasionally yields a longer communication range in Flick, "skipping" hops and reducing the minimum latency (lower bars in the right chart of Figure 8).

Notably, Flick is not only fast, but also consistently so; its per-hop latency is very stable around the average $T_{hop} = 47.08\,\mu s$, with a minimum of 40.83 µs and a maximum of $49.23\,\mu s$. We exploit this predictable per-hop latency next.

5.4 Configuring Flick

The stability of the per-hop latency can be directly exploited to dimension the duration T_F of a Flick round. The goal is to provide a simple configuration method based only on the diameter of the network, yet able to determine a value for T_F ensuring high *CDR* without unnecessary energy consumption. Our guidelines follow a simple observation: for a node at the farthest hop H from the initiator, the last chance to trigger an SFDTO is upon the retransmission of the preamble from its neighbours at the same hop H. Therefore, in an H-hop network, we configure T_F to be equal to the time this last preamble ends, i.e., $T_F = T_{hop}(H+1) + T_{preamble}$, where T_{hop} is the average per-hop latency of Flick (§5.3) and $T_{preamble}$ the duration of an UWB preamble. In our 10-hop network topology, and with our radio configuration (§5.1), this translates to $T_F = 47.08\,\mu s \times 11 + 65.13\,\mu s \approx 583\,\mu s$. We verified empirically via dedicated experiments that limiting the maximum listening duration T_F to this value does not affect the 5-nines reliability of Flick on rounds, corroborating the validity of the proposed approach.

5.5 Energy Consumption

We analyze on and off rounds separately, as their mechanics are quite different and therefore also their energy consumption.

off rounds. In this case, Flick incurs the highest energy cost: no communication occurs and the network remains in preamble hunting for the maximum listening duration T_F. Therefore, all nodes consume the same energy $E_{off} = I_{listen} \times V \times T_F$. For the DW1000, $I_{listen} = 113\,\text{mA}$ and $V = 3.3\text{V}$, yielding $E_{off} = 217\,\mu J$ in our 10-hop network. In the same conditions, the energy expenditure of an empty Glossy flood without packet receptions is $421\mu J$, nearly double w.r.t. Flick. This result is actually achieved with the most energy-efficient $N = 1$ configuration; with $N = 2$, the energy consumption of Glossy increases to $459\mu J$, 2.1× the one of Flick.

on rounds. Energy consumption varies depending on the hop distance from the initiator. To analyze it, we carry out dedicated experiments in the same network topology used in §5.3, and evaluate the per-hop energy cost of Flick and Glossy via StateTime [20], a publicly available software module for energy estimation on the DW1000 radio. Figure 9 summarizes the results. At the first hop, Flick consumes 43% less than Glossy with $N = 1$, and 63% less with the more reliable $N = 2$. Unlike Glossy, in Flick nodes do not remain in RX for a complete packet RX but immediately switch to TX upon preamble detection, abating both reception and processing time. The energy gap between the two network primitives increases with the number of hops. Thanks to its very fast propagation, Flick halves the average per-hop energy cost of Glossy, from $37.1\,\mu J$ to $17.4\,\mu J$. At hop 10, the energy consumption of Glossy with $N = 1$ is 2.05× that of Flick, 2.28× for $N = 2$.

5.6 Energy-efficient Preamble Detection

While remarkable, the energy consumption of Flick does not match the order-of-magnitude improvements w.r.t. Glossy we observed for latency (§5.3). This is to be ascribed to the inner mechanics of Flick, which force nodes to long preamble hunting periods. Until detecting a preamble, nodes keep listening to the channel, draining a significant amount of energy, especially in off rounds or when they are far from the flood initiator. Reducing the maximum Flick

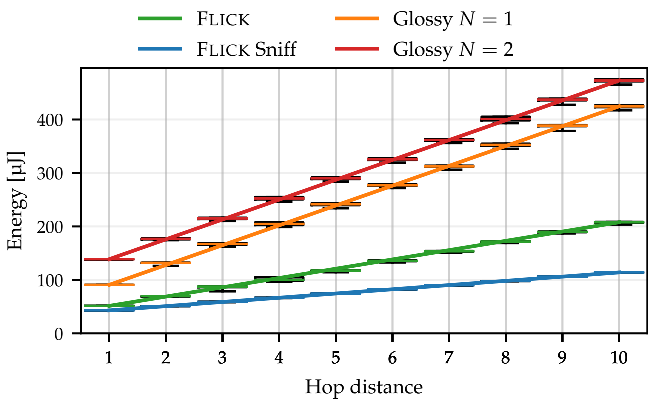
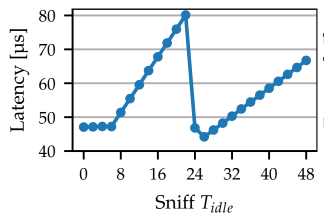
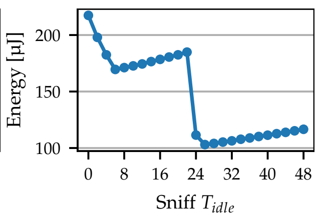


Figure 9: Per-hop energy consumption of FLICK and Glossy.



(a) Average per-hop latency.



(b) Energy (OFF round).

Figure 10: Sniff-based FLICK performance with varying T_{idle}.

duration T_F is a possibility to spare energy, but comes at the risk of reducing reliability as well. A more effective alternative exists: to replace continuous listening with periodic channel sampling by leveraging the Sniff mode of the DW1000 radio. To investigate this option and identify the best Sniff configuration, we carry out dedicated experiments by varying the idle time T_{idle} in between preamble hunting phases. We configure the latter to be 2 PAC long, the minimum allowed by the radio, and explore values of T_{idle} ranging from 0 to ≈48 μs in steps of ≈2 μs, effectively ensuring at least one full detection attempt per preamble. Note that these are nominal values; the actual ones are slightly higher, as 1 μs is derived from 512 counts of the 499.2 MHz UWB clock.

Does Sniff affect reliability? The first question is whether this intermittent preamble hunting can achieve the same reliability as the base variant with Sniff disabled ($T_{idle} = 0\,\mu$s). We replicate the same methodology employed in §5.2 and verify that *CDR* generally remains above 5 nines. The few exceptions occur with $32\,\mu s \leq T_{idle} \leq 40\,\mu s$, nevertheless achieving $CDR \geq 99.998\%$. Given that reliability is largely unaffected by Sniff, the main factors for the choice of T_{idle} become latency and energy consumption, analyzed in the rest of the section.

What is the impact on latency? The average per-hop latency follows a clear trend (Figure 10a), explained by how the preamble TX overlaps with the Sniff period. Indeed, the per-hop latency remains stable for low values of the idle time, until T_{idle} grows enough to cause a misalignment between the preamble TX and the periodic detection attempts. When $T_{idle} < 24\,\mu$s, one attempt is wasted, and the system incurs a linear delay w.r.t. T_{idle}, as exemplified in Figure 11a. Around $T_{idle} = 24\,\mu$s, the idle gaps between attemps become sufficiently large for the (formerly wasted) detection attempt to overlap with a preamble, reducing the chance of no detection (Figure 11b). Latency is minimal at $T_{idle} = 26\,\mu$s, meaning the ISR delay closely matches the time to complete a Sniff period and no detection attempt is wasted. On the other hand, when $T_{idle} \geq 28\,\mu$s, the Sniff period is simply not as short as it could be, and any increase of T_{idle} directly translates into an increase in latency.

Interestingly, the per-hop latency with $T_{idle} = 26\,\mu$s is even lower than without Sniff, $T_{idle} = 0\,\mu$s. This is due to the greater variability observed for the latter, as nodes at some hops detect the preamble later and drive the average up, which does not occur with Sniff.

Configuring FLICK for Sniff. The final step of this evaluation is the estimation of energy consumption. This is related to latency but also depends on the FLICK duration T_F we computed in §5.4. In principle, this value should not change, as it depends on the overall duration of $T_{preamble}$ which one would expect to remain unaltered despite the change from continuous to intermittent preamble detection attempts. Nevertheless, when configured in Sniff mode, the radio begins its periodic detection attempts by starting in idle mode. To account for this difference, we adapt our dimensioning guidelines, and redefine $T'_F = T_{idle} + T_F$. Failing to do so would result in an unintended reduction of the preamble hunting time when using Sniff, potentially affecting reliability.

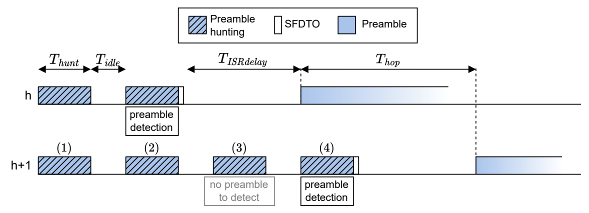
The initial delay T_{idle} has also repercussions on preamble TX, causing a misalignment between the time it begins and the time when receivers first attempt detection, hampering latency and reliability. We prevent this by inserting a T_{idle} delay for initiators, effectively realigning preamble transmissions and detection attempts.

Does energy efficiency actually improve? Based on T'_F, we can now directly compute the energy consumption in an OFF round. Figure 10b clearly shows that Sniff can cut consumption by more than half in OFF rounds, which is expected to be the most frequent operation in the classes of systems we target (§2). The value $T_{idle} = 26\,\mu$s, yielding the lowest latency and hence T'_F, consequently achieves the best performance in terms of energy, reduced to 104 μJ from the 217 μJ without Sniff (§5.5). Glossy configured with $N = 1$ would consume 4× more, and 4.4× with $N = 2$. STATETIME reveals that the Sniff variant meets our expectations also when FLICK is ON, reducing the per-hop energy consumption from 17.4 μJ to 7.9 μJ.

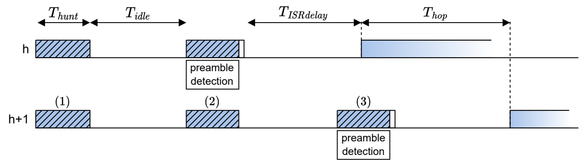
5.7 Multiple Initiators

FLICK has demonstrated unprecedented performance across the three dimensions of reliability, latency, and energy efficiency when a *single* node initiates a flood. The question now becomes whether the same holds when multiple FLICK ON conditions are disseminated concurrently, a key notion to determine how systems can exploit our primitive. Contention is indeed a well-known hampering factor for the reliability of systems based on conventional, packet-based CTX [10, 24].

Methodology. We analyze this aspect by carrying out dedicated experiments in which a predefined number U of nodes act as a FLICK initiator. We explore $U \in \{1, 5, 10, 30, 60\}$, i.e., up to 3/4 of the network, chosen at random at the beginning of each epoch. To determine whether exploiting the Sniff mode is still an option at the



(a) Further increasing T_{idle} only increases latency, until detection attempt (3) replaces (4).



(b) T_{idle} matches $T_{ISRdelay}$ more closely, yielding low per-hop latency T_{hop} despite sparser detection attempts.

Figure 11: Understanding per-hop latency with Sniff: alignment of preamble TX from hop h and hunting at $h + 1$.

increase of contention, we consider both continuous preamble hunting and $T_{idle} = 26\,\mu$s, the best Sniff configuration identified in §5.6. Performance are evaluated in terms of *i)* on condition detection rate (*CDR*) at non-initiators, *ii)* maximum latency, computed as the time elapsed from the beginning of a Flick on round to the last detection, and *iii)* network-wide average energy consumption.

Results. Table 1 reports the average of these metrics across 325200 Flick rounds, providing clear indications. First and foremost, reliability is virtually unaffected by the level of contention. Even in the rather extreme condition with $U = 60$ nodes concurrently originating a flood, Flick ensures *CDR* > 99.999%, regardless of the Sniff configuration adopted. This result is in contrast to mainstream packet-based CTX, and largely determined by the fact that reliability depends only on the correct detection of a preamble, rather than the successful decoding of a full packet. Second, both latency and energy consumption improve significantly with U. This trend, surprising at first sight, is actually expected; at the increase of the number of initiators the maximum node-initiator distance unavoidably reduces, with clear benefits in terms of flood latency and energy efficiency. Finally, while $T_{idle} = 26\,\mu$s confirms to be the most efficient solution, the impact of Sniff decreases at the increase of U, another side effect of the reduced node-initiator distance. As shown in Figure 9, the closer a node is to an initiator, the faster it can detect a preamble, minimizing the time the radio remains in preamble hunting (hence consumption) independently from the Sniff configuration adopted.

Overall, these results not only demonstrate that Flick can support systems where multiple on floods are initiated concurrently, but provide interesting insights on how to effectively exploit our primitive. Indeed, effort should *not* be made to reduce the amount of contention; on the contrary, system designers should explore whenever possible ways to increase the initiators of an on flood.

6 EXPLOITING FLICK

Now that we have evaluated the characteristics of our global network switch primitive, we are in the position to assess its impact on the representative classes of systems we considered, for which we have already outlined how Flick could be exploited (§2).

6.1 Replacing Conventional Packet Floods

In event-triggered systems, the event packet-based flood can be *directly* replaced by a Flick round (Figure 4, top). Based on our experimental results (§5), in off rounds this reduces energy consumption, of the flood alone, by a factor 4.4× w.r.t. the Glossy configuration ($N = 2$) matching the 5-nines reliability of Flick. In on rounds, the per-hop energy consumption is similarly reduced by 4.7× on average, while latency is abated by nearly a 10× factor. Of course, the *overall* gains depend on the specific protocol. Still, the improvement in energy consumption is particularly significant given that the case when no event is actually detected (Figure 2b) is the common one. For instance, in the ETC application used in the evaluation of WCB [21], an event is generated in only 13% of the epochs when these have a duration of 60 s, and a mere 0.3% when using a shorter epoch duration of 1 s. Therefore, the use of Flick could amplify the energy gains unlocked by ETC and enable new tradeoffs between reactivity of control vs. energy consumption.

6.2 Preceding Conventional Packet Floods

An alternative way to exploit Flick is for it to *precede* a conventional CTX packet flood (Figure 4, bottom). In this scheme, nodes flicking the on switch with a Flick round effectively announce their intent to transmit a packet in the subsequent flood, forcing the other nodes to remain awake to participate in it; vice versa, an off round enables the entire network to safely return to sleep. In the latter case, the energy gains are exactly the same we discussed above, i.e., a 4.4× reduction. In the on case, energy consumption is actually higher, as the cost of Flick adds to the one of the packet flood.

In the case of distributed schedule updates, the off case is likely far more frequent than the on case. In LWB, nodes are unlikely to continuously request transmission in the contention slot, except during the initial bootstrap; using Flick clearly reduces the energy cost of this functionality. However, the relative impact w.r.t. the overall protocol depends on the application traffic pattern and is inversely proportional to the number of data floods scheduled in each LWB epoch. A similar argument holds for SociTrack, where the relative energy impact of the contention flood used to announce a new group member decreases as the number of group members increases. In this case, however, Flick enables again new tradeoffs between reactivity vs. energy. For instance, given the 4.4× reduction in energy consumption, the contention-based flood could be scheduled at the same energy cost but with a corresponding 4.4× increase in frequency, increasing the speed at which joining nodes are included in the ranging schedule and improving the overall accuracy of SociTrack as a tool to monitor human interactions.

In the case of aperiodic data collection, the most complex among our representative examples, benefits are harder to assess because

Table 1: Performance vs. number of initiators U.

	Sniff disabled ($T_{idle} = 0\,\mu s$)			*Sniff enabled* ($T_{idle} = 26\,\mu s$)		
U	*CDR* [%]	Latency [μs]	Energy [µJ]	*CDR* [%]	Latency [μs]	Energy [µJ]
1	99.9998	349.38	96.03	99.9998	368.81	63.34
5	99.9995	159.19	58.57	99.9997	184.64	45.90
10	99.9995	113.18	50.48	99.9998	135.79	41.81
30	99.9998	69.28	41.53	99.9998	94.99	36.23
60	99.9997	48.27	31.21	99.9999	76.41	29.27

Flick is used repeatedly within the same epoch, based on the unpredictable and unknown amount of data to collect, and the number of senders potentially differs in each epoch. For this reason, and to concretely demonstrate the feasibility of integrating our primitive in existing protocols, we next present a quantitative evaluation of Flick-enhanced variants of Crystal and Weaver against their original counterparts.

7 FLICK-ENHANCED APERIODIC DATA COLLECTION

We begin by concisely describing how we integrated Flick into Crystal and Weaver. For the latter, we use the publicly available original version in [20], based upon the TSM layer presented in the same paper. We already exploited TSM for the Glossy version used in our comparison against Flick (§5). Therefore, for uniformity, we reimplemented Crystal atop this TSM-based Glossy, a task simplified by the features of TSM and the public availability of a Crystal version for UWB [12]. We discuss the salient protocol and configuration details along with the rest of experimental setup, followed by a report of the quantitative results we obtain.

7.1 Experimental Setup

Protocols. We already provided a high-level description of Crystal [8, 9, 12] as part of the related work and motivation (§2). As for Weaver [20], it focuses on low latency by exploiting individual CTX as the fine-grained protocol building blocks, instead of the monolithic Glossy floods common in the literature and among the systems considered here [2, 7, 8, 17, 21]. Multiple floods initiated simultaneously at different senders are dynamically woven into a single one based on a repeating TX-RX-RX pattern that *i)* decouples upward data flows towards the sink from their downward acknowledgments, and *ii)* enables local ones (L-ACKs) between hops to suppress wasteful retransmissions as data moves closer to the sink. A global G-ACK from the sink, performed every Y rounds, provides network-wide confirmation of packet reception; the value of Y determines the tradeoff between confirmation timeliness and contention with data flows. As the number of senders is not known a priori, a termination procedure is executed at each node including the sink, based on local knowledge of the hop-count, acquired during a short bootstrap flood initiated by the sink at the beginning of each epoch, and of the number of slots elapsed from the last G-ACK. These determine the number of slots that must elapse without the appearance of a new packet to trigger termination. Upon deciding termination, the sink floods a special packet to put the network to sleep; however, when this is lost at some nodes, these can autonomously enter sleep based on the knowledge above.

Mechanics of Flick integration. In both Crystal and Weaver, termination is arguably the most expensive portion of the protocol, both in terms of latency and energy; yet, it is also crucial to reliability as it ensures that no packet is erroneously lost by the network prematurely entering sleep.

Flick improves matters by offering an *instant and reliable termination* mechanism. In Crystal, this is exploited (Figure 3b) to remove completely the original termination phase at the expense of an additional Flick round before every data transmission flood (T). In Weaver, Flick is scheduled at the end of the initial bootstrap and of each G-ACK, both initiated by the sink, again completely removing termination at the expense of the extra Flick to check if other senders are present. In both cases, Flick enables all nodes to instantly and reliably determine whether to stay awake or enter sleep.

Protocol configuration. The configuration of Crystal relies on two parameters; the number R of empty T and A floods during termination, and the number N of packet retransmissions in the underlying Glossy layer. For the latter we use $N = 2$ because, as mentioned in §5, this enables Glossy to match the 5-nines reliability achieved by Flick. For the same reason, we use $R = 2$ for Crystal (Figure 3a). In Weaver, we use $Y = 6$ as it strikes the best tradeoff over the large network diameter of our setup. All protocols run on the same radio configuration used in §5.

Performance metrics and aperiodic traffic profile. We compare Crystal and Weaver against their Flick-enhanced variants in terms of reliability, latency, and energy consumption. For each protocol under consideration, we analyze these metrics for a different number U of initiators. Indeed, this is a crucial parameter for this class of systems, which must minimize energy consumption when no packet is to be transmitted ($U = 0$) while enabling quick and reliable delivery to the sink when possibly a packet is sent by many nodes at the same time ($U > 0$). We analyze this aspect in two ways. First, we evaluate separately the performance for a number $U \in \{0, 1, 2, 5, 10, 20, 30, 60\}$ of initiators, chosen at random. These values include those we used to evaluate Flick in §5.7, along with the values $\{2, 20\}$ present in the real-world aperiodic traffic profile from [9]. We use this profile (Table 2) in the final part of our evaluation, to assess the overall energy consumption when a mix of different values of U occur over time during data collection, providing a concrete example of the magnitude of improvements that could be attained in practice.

Table 2: Aperiodic traffic profile used for comparing Crystal and Weaver against their Flick-enhanced variants.

		U					
		0	1	2	5	10	20
epochs	#	84.3K	15.5K	2.2K	606	46	1
	%	82.1	15.1	2.2	0.14	0.038	0.005

Table 3: ***PDR*** **[%] vs. number of initiators** ***U*****.**

	Crystal		Weaver	
U	w/o Flick	w/ Flick	w/o Flick	w/ Flick
1	100	100	100	100
2	100	100	100	100
5	100	100	100	100
10	99.995	100	100	100
20	100	100	100	100
30	99.995	100	100	100
60	99.998	100	100	100

We run ≈2000 repetitions for each combination of protocol variant and U. In all experiments, the sink is node 119, yielding the maximum 10-hop network diameter (Figure 7).

7.2 Reliability

A first key question our analysis aims to address is whether integrating Flick in existing protocol stacks and, most importantly, relying on a single Flick off to terminate collection, impairs reliability. We study this aspect by analyzing the packet delivery rate (*PDR*) at the sink. Table 3 demonstrates that Flick does not degrade performance in practice. On the contrary, Flick-enhanced stacks ensure *perfect PDR* regardless of the level of contention, notably outperforming the original Crystal variant where rare losses occur at the increase of U. As a matter of fact, the 5-nines reliability provided by Flick (§5.2) is unmatched by packet-based CTX floods when different packets are transmitted concurrently, as in Crystal T phases and in the collection flow of Weaver. Therefore, leveraging this primitive instead of concluding a collection round after not receiving new data at the sink for a predefined time limits the risk of early termination, alongside reducing its energy cost and duration, as discussed next.

7.3 Latency

To exploit Flick, we periodically insert dedicated slots during collection, as in Figure 3b. At first glance, this approach may appear detrimental in terms of latency. This is certainly the case when considering *collection latency*, i.e., the time from the epoch start until the sink has received the last packet. Another question is the impact on *termination latency*, i.e., the time elapsed from the epoch start until a node enters sleep mode. If collection latency tells when the application can start processing the received data packets, with a direct impact on its performance, termination latency indicates how long the active portion of an epoch is expected to last, fundamental to schedule radio operations and to energy consumption. To ascertain whether Flick poses a threat to either type of latency, we study both in relation to the number of initiators per epoch, and therefore packets to collect, U. The average values aggregated over all epochs, as well as over all nodes for termination latency, are reported in Table 4.

Results show that the collection latency in Crystal is only marginally affected by the presence of Flick, which contributes a constant increase for each packet whose magnitude is determined by the small, sub-ms maximum duration T_F. This additional time is even lower for Weaver because, unlike Crystal, multiple packets can be received by the sink between Flick rounds.

As far as termination latency is concerned, Flick improvements are significant for low values of U, achieving our main objective. Nonetheless, as a stress test, we experiment also with very high values of U. The two variants of Crystal match at $U = 30$, and even at $U = 60$ the one with Flick achieves a latency that is only 27 ms higher than the original, less than a 3% increase. The termination latency of Weaver with Flick is likewise smaller than the original until $U = 60$. Finally, we analyze how much termination latency differs across nodes, as the maximum value of the active portion of an epoch is ultimately what concerns programmers when scheduling operations. As expected, while in the original protocols the termination latency varies by few ms depending on the node hop distance from the sink, in the Flick-enhanced variants all nodes terminate simultaneously with the off round.

The analysis confirms that Flick can be easily integrated in these collection protocols, yielding low overhead for collection latency, and nearly always improving termination latency.

7.4 Energy Consumption

Our evaluation thus far shows that Flick-enhanced protocols achieve remarkable performance in terms of reliability and latency, often performing better than the original. The third pillar of our investigation is energy consumption, where we expect Flick to make major improvements. We report the network average consumption over all collection epochs, but also the average minimum and maximum values across them, as they convey concisely the impact of hop distance from the sink.

Table 5 presents the results. As expected, leveraging Flick is particularly advantageous with $U = 0$, enabling Crystal to consume *one third of the original.* In Weaver, Flick reduces consumption by more than half, despite the fact that this prootocol is more optimized in this respect. The question is whether using Flick also pays off when there are packets to collect, and what is the break-even point w.r.t. U. In Crystal, Flick enables lower consumption than the original for $U \leq 20$; nonetheless, it only consumes 4.4% more even with $U = 60$. Similarly, the Flick-enhanced version of Weaver outperforms the original for $U \leq 30$, with only a 1.9% increase for $U = 60$. The energy improvement in terms of ratio of the average energy consumption of the original protocols over the Flick-enhanced versions is shown in Figure 12.

Even when looking at the maximum energy consumption per epoch, the one of Flick-enhanced protocols does not increase substantially w.r.t. the network average, despite the large network

Table 4: Latency vs. number of initiators U.

(a) Crystal.

	Collection [ms]		Termination [ms]		Termination max. [ms]	
U	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK
0	—	—	33	8	36	8
1	10	11	48	24	51	24
2	25	27	62	40	65	40
5	70	75	107	86	110	86
10	145	153	181	165	184	165
20	293	309	328	321	331	321
30	441	466	476	477	480	477
60	884	935	919	946	923	946

(b) Weaver.

	Collection [ms]		Termination [ms]		Termination max. [ms]	
U	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK
0	—	—	31	10	34	10
1	11	12	45	25	52	25
2	17	18	50	31	59	31
5	26	28	59	42	66	42
10	39	42	71	56	78	56
20	67	73	98	87	105	87
30	96	104	125	118	134	118
60	184	201	211	214	223	214

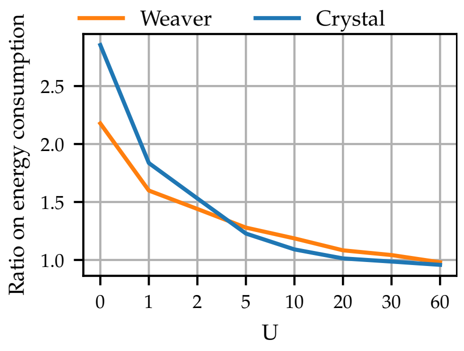


Figure 12: Energy consumption ratio of the original protocols vs. FLICK-enhanced versions with varying U.

we experiment on. Interestingly, using FLICK increases the minimum consumption of WEAVER nodes for $U \geq 10$ w.r.t. the original protocol. This is due to our strategy for integrating FLICK, as it replaced the original termination policy in its entirety. In WEAVER, autonomous termination includes a condition for nodes to enter sleep and save energy when they detect they already forwarded all packets (if any) coming from farther hops in the collection topology. This strategy could potentially be reinstated in conjunction with FLICK. Nonetheless, this seems relevant only for high values of U, while for low values the version with FLICK clearly outperforms the original. Given the consistent improvements brought by our proposed primitive, its use in the context of sparse traffic patterns is bound to yield advantages as well. We quantify them next.

7.5 Aperiodic Data Collection

Both Crystal and WEAVER have been originally designed to efficiently support aperiodic traffic. To provide the reader with a concrete example of the performance improvements unlocked by FLICK-enhanced stacks under this traffic pattern, we consider the real-word traffic profile in Table 2. Specifically, for each communication protocol, we compute the average energy consumption aggregated over the entire dataset as $E = \frac{\sum_{u=0}^{N} e(u) \times z(u)}{\sum_{u=0}^{N} z(u)}$, where $e(u)$ is the network-wide average energy cost for a given number U of initiators, directly informed by our analysis in §7.4, and $z(u)$ is the number of epochs with U nodes concurrently transmitting, as reported in Table 2.

Table 6 summarizes the results and confirms our expectations; under aperiodic traffic, exploiting FLICK as a primary asset to quickly and reliably terminate data collection brings significant energy improvements. In WEAVER, it halves consumption w.r.t. to the original protocol. In Crystal, where especially in the absence of traffic the impact of termination is even higher (Figure 3a), exploiting FLICK abates energy costs by a 2.5× factor.

Table 5: Energy consumption vs. number of initiators U.

(a) Crystal.

	Minimum [mJ]		Network average [mJ]		Maximum [mJ]	
U	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK
0	2.82	1.00	3.12	1.09	3.63	1.18
1	3.90	2.16	4.23	2.30	4.76	2.49
2	4.97	3.33	5.32	3.48	5.81	3.71
5	8.17	6.76	8.60	7.01	9.34	7.35
10	13.50	12.50	14.04	12.88	14.96	13.37
20	24.25	23.87	24.85	24.52	26.00	25.34
30	34.55	35.19	35.57	36.07	37.12	37.20
60	66.00	68.66	67.13	70.08	69.22	72.29

(b) Weaver.

	Minimum [mJ]		Network average [mJ]		Maximum [mJ]	
U	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK	w/o FLICK	w/ FLICK
0	2.84	1.37	3.15	1.45	3.54	1.53
1	3.98	2.53	4.56	2.85	5.39	3.26
2	4.12	3.02	5.06	3.51	6.16	4.06
5	4.30	3.98	6.17	4.81	7.23	5.54
10	4.41	5.21	7.70	6.50	8.89	7.40
20	4.50	7.91	10.83	9.99	12.37	11.16
30	4.67	10.67	14.14	13.57	16.05	15.04
60	5.86	19.57	24.28	24.75	27.26	27.03

Table 6: Energy consumption with the aperiodic traffic profile in Table 2.

Protocols	Energy [mJ]
Crystal	3.37
Crystal w/FLICK	1.37
WEAVER	3.42
WEAVER w/FLICK	1.73

8 CONCLUSIONS AND FUTURE WORK

The binary decision of whether to be awake to disseminate a packet or instead enter sleep mode is a crucial one, directly impacting performance and efficiency in several classes of CTX-based systems. We presented FLICK, a novel primitive that, like the flick of a switch, can instantaneously and reliably establish an on or off binary condition across the entire network, without requiring additional hardware. We illustrated our design, targeting UWB radios, and evaluated its performance on a 78-node network in the Cloves [13] testbed, confirming that the binary condition can be established globally with 5-nines reliability in only a few tens of μs per hop and with very low energy consumption. We have shown how these unprecedented characteristics can be exploited in the aforementioned classes of systems and offered a detailed quantitative evaluation for one of them, demonstrating at once how to integrate Flick in existing systems and the improvements it unlocks.

As for opportunities of future work on this topic, we observe that although we focused on UWB radios, our preamble-based technique may be applicable to other radios with a different PHY layer, especially as low-power transceivers are becoming increasingly efficient and configurable. Moreover, the ability to take instant global binary decisions may find use in contexts other than CTX-based systems, e.g., in combination with conventional network stacks [23], or directly at the application layer as a network-wide OR or AND operator (§2), e.g., to globally set the values of state variables. We intend to facilitate this and other potential developments by publicly releasing our code as open source [1].

ACKNOWLEDGMENTS

This work is partially supported by the Italian government via the NG-UWB project (MIUR PRIN 2017).

REFERENCES

[1] 2023. https://github.com/d3s-trento/contiki-uwb.
[2] A. Biri et al. 2020. SociTrack: Infrastructure-Free Interaction Tracking through Mobile Sensor Networks. In *Proc. of MobiCom.*
[3] DecaWave Ltd. 2017. DW1000 Data Sheet, version 2.19.
[4] DecaWave Ltd. 2017. DW1000 User Manual, version 2.18.
[5] P. Dutta et al. 2010. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proc. of SenSys.*
[6] F. Ferrari et al. 2011. Efficient Network Flooding and Time Synchronization with Glossy. In *Proc. of IPSN.*
[7] F. Ferrari et al. 2012. Low-power Wireless Bus. In *Proc. of SenSys.*
[8] T. Istomin et al. 2016. Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks. In *Proc. of SenSys.*
[9] T. Istomin et al. 2018. Interference-Resilient Ultra-Low Power Aperiodic Data Collection. In *Proc. of IPSN.*
[10] O. Landsiedel, F. Ferrari, and M. Zimmerling. 2013. Chaos: Versatile and Efficient All-to-all Data Sharing and In-network Processing at Scale. In *Proc. of SenSys.*
[11] R. Lim et al. 2017. Competition: Robust Flooding Using Back-to-Back Synchronous Transmissions with Channel-Hopping. In *Proc. EWSN.*
[12] D. Lobba et al. 2020. Concurrent Transmissions for Multi-hop Communication on Ultra-wideband Radios. In *Proc. of EWSN.*
[13] D. Molteni et al. 2022. Poster abstract: Cloves: A Large-scale Ultra-wideband Testbed. In *Proc. of SenSys.*
[14] R. Piyare et al. 2017. Ultra Low Power Wake-Up Radios: A Hardware and Networking Survey. *IEEE Communications Surveys & Tutorials* 19, 4 (2017).
[15] J. Polastre, J. Hill, and D. Culler. 2004. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of SenSys.*
[16] F. Sutton et al. 2015. Zippy: On-Demand Network Flooding. In *Proc. of SenSys.*
[17] F. Sutton et al. 2017. The Design of a Responsive and Energy-Efficient Event-Triggered Wireless Sensing System. In *Proc. of EWSN.*
[18] F. Sutton et al. 2019. BLITZ: Low Latency and Energy-Efficient Communication for Event-Triggered Wireless Sensing Systems. *ACM Trans. on Sensor Networks* 15, 2 (2019).
[19] P. Tabuada. 2007. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Trans. on Automatic Control* 52, 9 (2007).
[20] M. Trobinger et al. 2020. One Flood to Route Them All: Ultra-fast Convergecast of Concurrent Flows over UWB. In *Proc. of SenSys.*
[21] M. Trobinger et al. 2021. The Wireless Control Bus: Enabling Efficient Multi-hop Event-Triggered Control with Concurrent Transmissions. *ACM Trans. on Cyber-physical Systems* 6, 1 (2021).
[22] D. Vecchia et al. 2019. Playing with Fire: Exploring Concurrent Transmissions in Ultra-wideband Radios. In *Proc. of SECON.*
[23] X. Vilajosana et al. 2020. IETF 6TiSCH: A Tutorial. *IEEE Communications Surveys & Tutorials* 22, 1 (2020).
[24] Yin W. et al. 2012. Exploiting constructive interference for scalable flooding in wireless networks. In *Proc. of INFOCOM.*
[25] M. Zimmerling, L. Mottola, and S. Santini. 2020. Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. *Comput. Surveys* 53, 6 (Dec. 2020).