Cupid: Fast and Reliable Convergecast-over-UWB Protocol for Dense Internet of Things

Jimin Park*, Geonhee Lee*, Jeongyeup Paek‡, and Saewoong Bahk*

*Department of Electrical and Computer Engineering and INMC, Seoul National University, Seoul, Republic of Korea
‡Department of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea
{jmpark, ghlee}@netlab.snu.ac.kr, jpaek@cau.ac.kr, sbahk@snu.ac.kr

***Abstract*—Concurrent transmission (CTX) has enabled low-power and low-latency data collection over multihop, demonstrating high efficiency across various wireless technologies such as Zigbee, BLE, and ultra-wideband (UWB) radio. However, most existing CTX-based protocols use logical hop-distance-based scheduling without considering the physical distances or the unique phenomenon of concurrent transmission, which may lead to frequent retransmissions. To further enhance the speed and reliability of data collection, we propose *Cupid*, an ultra-fast and reliable convergecast for multihop UWB wireless network that leverages in-band ranging, intra-hop grouping, and role differentiation within its collection topology to achieve its goals. Extensive evaluation via real experiments on a 36-node UWB testbed demonstrate that *Cupid* outperforms *Weaver*, a state-of-the-art UWB-based many-to-one data collection protocol, reducing latency by up to ∼29% while providing higher reliability and throughput across diverse application scenarios.**

***Index Terms*—Concurrent transmission, ultra-wideband, multihop, convergecast, wireless sensor network, Internet of Things**

I. Introduction

Wireless sensor network (WSN) is a critical component in many Internet of Things (IoT) systems, and has enabled the growth of a variety of surveillance and monitoring applications [1]–[3]. These systems typically feature a convergecast data flow where multiple low-power sensor nodes detect and transmit event or sensing information to one or more master nodes [4]. For this reason, routing and scheduling protocols for multihop data collection have long been major research topics in WSN [5]–[9]. However, there has recently been growing interest in *concurrent transmission* (CTX) technology due to its impressive latency and efficiency performance compared to traditional multihop routing [10]–[12].

Concurrent transmission is a technique that exploits generated interference by allowing nodes to transmit in a synchronized manner. Glossy [10], a pioneering CTX protocol, introduced network-wide flooding for time synchronization and data dissemination. Since then, it has become a core building block in many data collection protocols [11]–[13] and applications [14]–[18]. Initially explored over IEEE 802.15.4 narrowband radios, these protocols have demonstrated remarkably low latency and energy consumption while maintaining high reliability compared to traditional multihop routing [5], [19]. For these reasons, recent research has extended their applicability to other wireless technologies, such as Bluetooth Low Energy (BLE) [20] and Ultra-wideband (UWB) [12], [21]–[23].

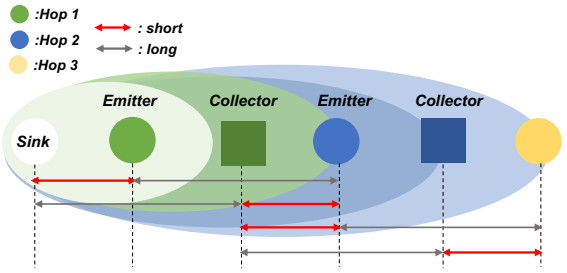


Fig. 1: Intra-hop grouping with relative distance.

Several CTX-based data collection protocols have been proposed in the literature. Crystal [11] enables data collection through repetitive one-to-all Glossy flooding. However, this approach is inefficient as it requires multiple flooding iterations to transmit data from multiple nodes. Weaver [12], a protocol introduced to overcome this challenge, is now considered the state-of-the-art in UWB-based data collection. By introducing a simple modification to the slot structure, it maintains high reliability while overcoming the inefficiencies of Glossy-based convergecast. Later, Speedcollect [13], a protocol leveraging network coding, was also proposed. However, it remains constrained to IEEE 802.15.4 radios.

Most existing protocols rely on *logical hop distance* to schedule CTX transmissions. Thus, multiple nodes within a hop transmit simultaneously, and differences in *physical distances and node positions* lead to unfair contention. As a result, packets from closer nodes are more likely to be successfully received while packets from more distant nodes require more retransmissions. This increases unnecessary collisions and energy consumption, causing distant nodes to experience delays and miss opportunities to participate in

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program(IITP-2025-2021-0-02048), Open RAN Education and Training Program (IITP-2024-RS-2024-00429088), and IITP-2025-RS-2024-00398157, supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). S. Bahk is the corresponding author.

relaying. Ultimately, these problems significantly reduce the overall efficiency of the network.

To address the problem, we propose *Cupid*, a fast and reliable CTX-based **C**onvergecast-over-**U**WB **P**rotocol for multihop wireless **I**oT with **D**ense deployment. The key idea is *intra-hop grouping* that leverages the physical distances between nodes obtained through UWB's precise *in-band ranging* capability. Specifically, *Cupid* divides nodes within each logical hop into two groups, called 'Emitter' and 'Collector,' based on the distances between nodes in the upper and lower parts of the hop (Fig. 1). Then, it assigns separate transmission schedules to each group to allow intra-hop overhearing. This approach improves the fairness in packet delivery for near and far nodes, reduces unnecessary retransmissions, and enables more active relaying. *Cupid* demonstrates high reliability and low latency, even in heavy traffic or long packet scenarios often ignored in prior CTX-based systems.

The contributions of *Cupid* can be summarized as follows:

- We propose *Cupid*, a convergecast protocol for multihop UWB wireless network that uses in-band ranging, intra-hop grouping, and role differentiation to enhance the speed and reliability of data collection.
- *Cupid* uses concurrent ranging to effectively obtain the relative distances between nodes, and divides nodes into two groups to isolate transmission schedules to avoid collisions.
- We implement *Cupid* on real embedded UWB devices and conduct extensive 36-node testbed experiments to demonstrate that *Cupid* outperforms CTX-based state-of-the-art protocols: *Crystal* and *Weaver*.

The remainder of this paper is organized as follows: We provide background of this work in §II, and discuss related prior work in the literature in §III. In §IV, we propose the design of *Cupid*, and in §V we present evaluation results. Finally, we summarize the paper in §VI.

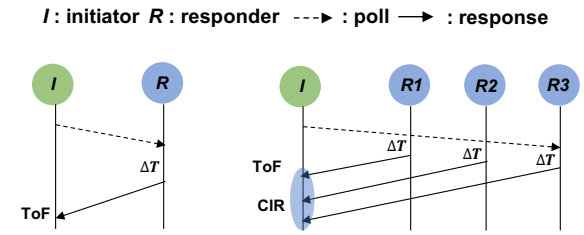
II. Background

We first introduce the *concurrent transmission* technique, and provide some background on ultra-wideband (UWB) wireless technology that motivates our work.

A. Concurrent Transmission (CTX)

CTX leverages *constructive interference* and *capture effect* to ensure that, despite multiple nodes transmit simultaneously, at least one packet is received with high probability.

Constructive interference [24] occurs when identical packets transmitted by different nodes arrive within a precise time window (e.g., 0.5μs for IEEE 802.15.4 2.4 GHz [10]), enabling successful reception. Glossy [10] employed this phenomenon to establish a fast one-to-all flooding where a single sink node initiates the process. Nodes relay packets from the sink to downstream hops and determine their logical hop distance based on the number of times the received packet has been relayed. Subsequently, each node performs flooding by repeatedly following the transmission(T)–reception(R) slot structure defined by its logical hop. Many subsequent CTX-based protocols [11], [12], [18] have utilized Glossy's idea to develop protocols that support diverse traffic patterns.



(a) Two-way ranging **(b)** Concurrent ranging

Fig. 2: UWB ranging methods.

Capture Effect [25], [26] means, even if distinct signals collide, the stronger signal prevails and can be received. By leveraging this effect, it is possible to design protocols for many-to-one [11]–[13] and many-to-many [14]–[18] traffic structures where multiple transmitters are simultaneously sending different packets.

B. IEEE 802.15.4 UWB

UWB offers high precision distance measurements with its ultra-short pulse communication ($\leq 2\,ns$) and bandwidth of over 500 MHz. Although interest in UWB research declined after an initial surge, the past decade has seen a resurgence due to Qorvo's DW1000 chipset [27] commercialization. UWB provides a much higher data rate of 6.8 Mbps compared to the traditional IEEE 802.15.4 narrowband, which is another significant advantage. Despite these advantages, however, research on communication protocols for UWB has been relatively limited, and a fully standardized network stack has yet to be established.

Moreover, due to UWB's use of wide bandwidth, its communication range is inherently short, necessitating the development of a robust multi-hop communication structure. Consequently, there is a pressing need for the design of an effective multi-hop MAC protocol tailored for UWB. Rather than focusing solely on conventional MAC protocol designs, recent research [11], [12], [21], [22] has shifted towards proposing simple yet highly effective CTX-based protocols leveraging UWB's unique properties.

C. UWB Ranging

The most common method for UWB distance measurement is *two-way ranging* (TWR) as illustrated in Fig. 2a. First, the initiator intending to measure the distance sends a poll message. Upon receiving this poll message, the responder sends a response message after a predetermined fixed delay ΔT. The initiator then uses the transmission timestamp of the poll message TX_{poll}, the reception time of the response message $RX_{response}$, and the ΔT to calculate the time of flight (ToF) as follows:

$$ToF = \frac{(TX_{poll} - RX_{response}) + \Delta T}{2}$$

This method provides centimeter-level ranging accuracy, making it a popular choice for many systems utilizing UWB for positioning and localization. However, since TWR is designed for one-to-one ranging, it faces scalability issues in dense systems with large number of nodes. For example, to measure the distances between N nodes, at least $2 \times \binom{N}{2}$ message exchanges are required.

To address the issue of one-to-many distance measurements, *concurrent ranging* has been proposed [28], [29]. As shown in Fig. 2b, the initiator sends a single poll message, and each responder replies after a fixed delay ΔT. The initiator then analyzes the channel impulse response (CIR) to match peaks with nodes and calculate their distances. However, this method requires significant prior information to match CIR peaks to nodes, limiting scalability.

In our study, we simplify concurrent ranging for relative hop-based ranging, enabling efficient distance measurements with minimal overhead.

III. Related Work

State-of-the-art CTX-based data collection protocols are represented by Crystal [11] and Weaver [12], both of which have proven their performance over UWB.

• **Crystal** aims for energy-efficient and reliable data collection from multiple nodes in IoT applications with sporadic and sparse traffic. It constructs a many-to-one convergecast system by repeatedly executing Glossy flooding in three distinct phases: After the (1) initial S-phase for network synchronization, Crystal alternates between (2) T-phases for data transmission and (3) A-phases for acknowledgments. During the S- and A-phases, the sink broadcasts synchronization and ACK packets, which are rebroadcast by receiving nodes to flood the network. In the T-phase, nodes wishing to transmit data initiate flooding, but only one data packet can be successfully delivered to the sink per phase. Nodes that fail to transmit successfully will attempt retransmission in the next T-phase. If a predetermined number of consecutive empty T–A phases occur, Crystal terminates the data collection. However, the protocol's reliance on repeated one-to-many flooding for many-to-one convergecast causes excessive transmissions and contention, leading to energy inefficiency and collisions. As a result, Crystal is not well-suited for IoT systems requiring high throughput or low latency.

• **Weaver** addresses the inefficiency of repetitive flooding by introducing a simple yet effective approach. In most one-to-all flooding, nodes rely on a repetitive T–R structure to enable data dissemination. Weaver replaces this with a T–R–R three-slot structure, allowing efficient many-to-one data collection within a single *epoch*. The first RX slot is used for receiving upward packets to the sink, while the second RX slot handles downward messages from the sink. By separating upward and downward transmissions, Weaver reduces collisions. Additionally, Weaver utilizes two types of ACKs to reduce unnecessary retransmissions. A bitmap-based *global ACK* is transmitted by the sink to explicitly indicate whether packets have been successfully received at the sink. A *local ACK* is piggybacked within all data transmissions, which contains the originator ID of the most recently received packet. After receiving a local ACK containing its ID, a node calculates the timout for the global ACK and temporarily suppresses retransmissions during this period, thereby minimizing collisions. This modification to the slot structure enables Weaver to be faster and more energy-efficient than Crystal while maintaining high reliability in convergecast scenarios.

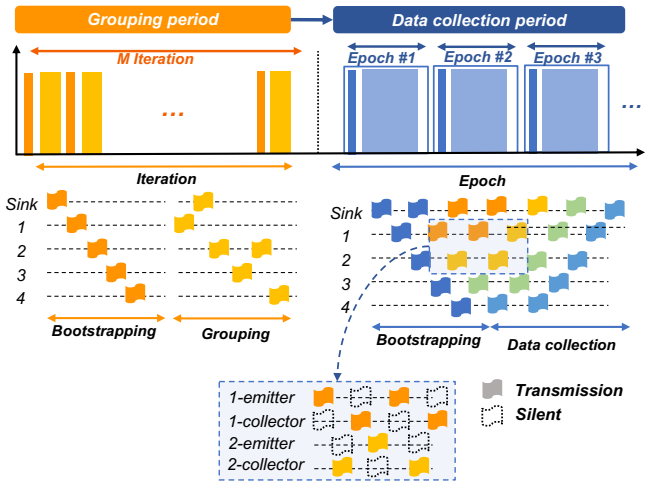


Fig. 3: Overview of *Cupid*'s transmission schedule.

However, like other CTX-based protocols, *Weaver* also has the limitation of assigning the same TX schedule to nodes within the same hop without considering the physical location of nodes. Nodes closer to the upper hop have a higher chance of successful packet delivery to the upper hop compared to those farther away. Conversely, nodes closer to the lower hop are better positioned to receive packets from the lower hop while having a disadvantage when transmitting simultaneously with nodes closer to the upper hop. However, if all nodes within the same hop have same TX schedule, no packet delivery nor overhearing is possible among them, which hinders efficient relaying and leads to unnecessary retransmissions. We propose *Cupid* to solve this problem.

IV. *Cupid* Design

In this section, we present the design details of *Cupid* including a high-level overview, ranging and grouping mechanisms, and the data collection protocol.

A. Overview

Cupid utilizes the physical distances between nodes to address the challenges associated with logical hop-based transmission scheduling. Specifically, *Cupid* incorporates *intra-hop grouping* which classifies nodes within a single hop into two groups, 'emitter' and 'collector', each operating with distinct transmission and reception schedules. Nodes closer to the

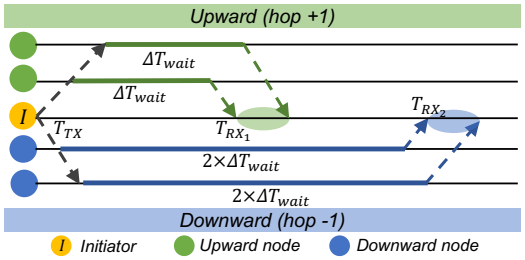


Fig. 4: One to multi-node ranging for shortest distance measurement.

upper hop are assigned as *emitter*, while those closer to the lower hop are assigned as *collector* (Fig. 1).

Cupid operates in two phases (Fig. 3): the *grouping period (§IV-B)* and the *data collection period (§IV-C)*. Each period is organized into epochs, beginning with a *bootstrapping* process where the sink floods the network with synchronization packets. Nodes determine their logical hop from the relay count of received synchronization packets, a critical step in most CTX-based protocols like Crystal and Weaver. Since logical hop count can vary between epochs, each node calculates its *virtual hop distance* during the *grouping period* based on the relative distances to upper and lower hops. This *virtual hop distance* is then used as a threshold in the *data collection period* for each node to assign itself to a group, ensuring efficient scheduling for data transmission and reception.

B. Grouping Period

Emitters and collectors are grouped based on their relative distances to the *nearest node* in either the upward or downward hop. Grouping requires measuring distances between multiple nodes. While UWB allows precise ranging, doing so for all $O(n^2)$ pairs of nodes undermines the simplicity of CTX systems and introduces significant overhead. To address this, we employ a simplified one-to-many *concurrent ranging* for more efficient grouping process.

Concurrent ranging typically requires receiving multiple responses and conducting an analysis of the channel impulse response (CIR). However, since the information required for grouping is not the distances to all nodes but only the distance to the closest node within the target hop, we early-terminate this process. When multiple response messages are received by the initiator of ranging, we assume that the packet from the closest node is received first, and only one distance is calculated for the corresponding node. At this time, the nodes respond with a smallest packet (header-only, with payload length of 0) to minimize the probability of collisions.

The ranging process is illustrated in Fig. 4. It begins with the initiator transmitting a poll message containing its node ID and logical hop information. Among the neighboring nodes that receive the poll message, only those located within the ± 1 hop upward or downward relative to the initiator participate in concurrent ranging. Nodes in the upper hop respond after a small delay ΔT_{wait}, while nodes in the lower hop respond after a delay twice as long. The Time of Flight (ToF) for the first received response from upward hop is calculated as:

$$ToF_{up} = \frac{\mathrm{T}_{RX_1} - \mathrm{T}_{TX} - \Delta \mathrm{T}_{wait}}{2}$$

where, T_{TX} is the transmission timestamp of the poll packet and T_{RX_1} represents the timestamp of the first received response among the multiple concurrently transmitted responses. Similarly, the ToF for the downward hop is calculated using the second response as:

$$ToF_{down} = \frac{\mathrm{T}_{RX_2} - \mathrm{T}_{TX} - 2 \times \Delta \mathrm{T}_{wait}}{2}$$

Using this approach, the initiator determines the distance to the closest node within each hop and obtains relative proximity information for both the upward and downward hops.

Grouping Algorithm. To address potential issues caused by packet collisions or timestamps acquired from non-nearest nodes during the ranging process, *Cupid* performs multiple iterations of ranging rather than relying on a single measurement. Additionally, since CTX-based systems dynamically (re-)assign the logical hop in each epoch based on how the synchronization packets are relayed, group assignment must account for the dynamics in channel conditions or logical topology. For instance, a node near the boundary between hops 2 and 3 may function as a collector when assigned to hop 2 but as an emitter when assigned to hop 3.

To resolve this challenge, *Cupid* employ an algorithm to compute a *virtual hop distance* which reflects a node's relative proximity to adjacent hops. During each grouping iteration, nodes use their logical hop value and distance measurements to calculate their *virtual hop distance*. For example, a node with a logical hop of 2 but closer to the lower hop will have a *virtual hop distance* closer to 3. This approach bridges the gap between logical and physical hop distance, enabling precise grouping even as logical hops dynamically shift across epochs.

Grouping iterations account for four scenarios during concurrent ranging: (1) successful measurements of both upper and lower node distances, (2) successful measurement of only the upper node's distance, (3) successful measurement of only the lower node's distance, and (4) failure to measure both distances. In the first case, distances are compared to determine group assignment. In the second and third cases, the node is assigned to the group corresponding to the successful measurement. If both measurements fail, the node is placed in a middle group.

Based on these results, nodes calculate their *virtual hop distance* using their logical hop count and by comparing the distances to the upward hop (d_{up}) and the downward hop (d_{down}): (a) If $d_{\text{up}} < d_{\text{down}}$ or only d_{up} is successfully measured, the node is assigned the *logical hop +0.25* value. (b) If $d_{\text{up}} > d_{\text{down}}$ or only d_{down} is successfully measured, the node is assigned the *logical hop + 0.75* value. (c) If $d_{\text{up}} = d_{\text{down}}$ or both measurements fail, the node is assigned

Algorithm 1 Dynamic Grouping for Virtual Hop

Input: N nodes, M iterations, d_{hop} (logical hop)
Output: Group assignments based on virtual hop distance
for $i = 1$ to M **do** ▷ Epoch iterations
  for $n = 1$ to N **do** ▷ Process each node
    Measure d_{up} and d_{down}
    Compute virtual hop distance:
    if d_{up} and d_{down} are both successfully measured **then**

$$\text{hop}_i^{\text{Virtual}} = \begin{cases} \text{hop} + 0.25 & \text{if } d_{\text{up}} < d_{\text{down}} \\ \text{hop} + 0.75 & \text{if } d_{\text{up}} > d_{\text{down}} \\ \text{hop} + 0.5 & \text{if } d_{\text{down}} = d_{\text{up}} \end{cases}$$

    else if d_{up} is successfully measured, but d_{down} fails **then**
      $\text{hop}_i^{\text{Virtual}} = \text{hop} + 0.25$
    else if d_{down} is successfully measured, but d_{up} fails **then**
      $\text{hop}_i^{\text{Virtual}} = \text{hop} + 0.75$
    else
      $\text{hop}_i^{\text{Virtual}} = \text{hop} + 0.5$
    end if
  end for
end for
Final Assignment:
$\text{hop}_{\text{avg}}^{\text{Virtual}} \leftarrow \frac{1}{M} \sum_{i=1}^{M} \text{hop}_i^{\text{Virtual}}$ ▷ Virtual hop distance

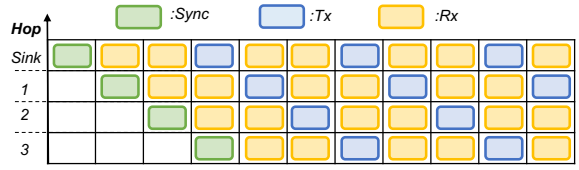
the *logical hop + 0.5* value. The final *virtual hop distance* is a real number computed as the average of these values over m iterations, which serves as the grouping threshold. For example, if the virtual hop distance is '2.8', it means that the node is in hop 2 but is closer to hop 3 than hop 1. This procedure is summarized in Alg. 1.

During the data collection period, after bootstrapping, each node compares its *virtual hop distance* to its current logical hop count to dynamically adjust its group. If the *virtual hop distance* is closer to its logical hop count than 'hop count + 1', the node is classified as an emitter; otherwise, if it is closer to 'hop count + 1', it is classified as a collector.

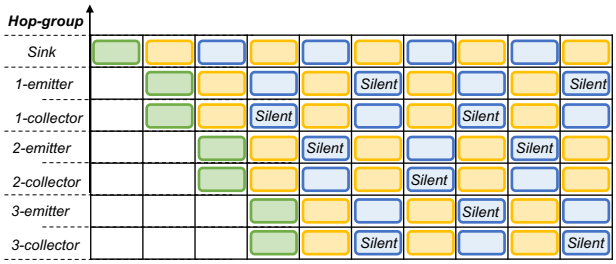
C. Data collection

Slot Schedule. Once the grouping process is completed, the network transitions into the data collection period which operates on an *epoch* basis. During this period, each node autonomously configures its transmission and reception schedule according to its logical hop and the virtual hop distance. Specifically, each *epoch* begins with a bootstrapping procedure, during which the sink floods the network with an initialization packet to synchronize all nodes and establish the logical hop. Once a node receives a bootstrapping packet, it synchronizes and operates in a repetitive two-slot structure consisting of transmission (TX) and reception (RX) based on its logical hop.

Cupid builds upon the TX-RX two-slot structure of conventional Glossy-based protocols, while newly introducing the *silent mode* to TX slots. In *silent mode*, TX slots are replaced with RX slots, allowing nodes within the same hop to receive data. To isolate transmissions between different groups within each hop, *silent mode* alternates across groups within a hop. This process results in the formation of a four-slot structure

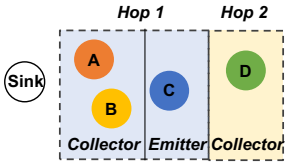


(a) Slot Structure of *Weaver*

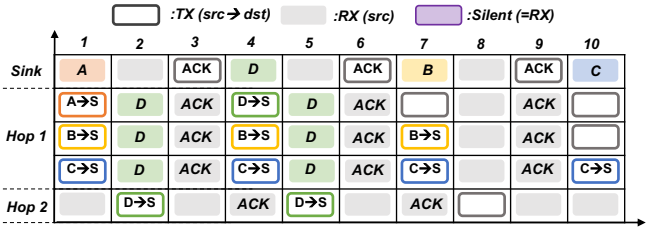


(b) Slot structure of *Cupid*

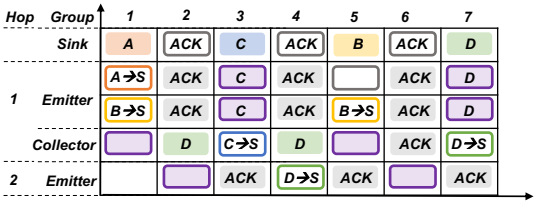
Fig. 5: *Weaver* vs. *Cupid*, comparison of TX schedule structure.



(a) Example of topology



(b) Slot Schedule of *Weaver*



(c) Slot Schedule of *Cupid*

Fig. 6: *Weaver* vs. *Cupid*, comparison of TX and RX slot schedules during data collection on an example topology.

for each group, as illustrated in Fig. 5b. Allocation of the *silent mode* is based on the logical hop and group, and the order differs between even-numbered and odd-numbered hops within the same group, ensuring collision-free transmission. This approach improves fairness in reception within the hop and effectively utilizes *overhearing*, ultimately enhancing the overall data delivery efficiency of the network.

Example. Fig. 6 illustrates the effectiveness of group-based

Fig. 7: UWB Transceiver based on Qorvo DW1000 radio.

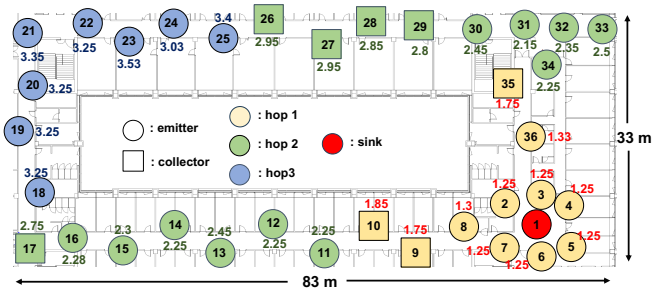


Fig. 8: CLOVES testbed with a snapshot of grouping result. Colors indicate the logical hop, and shapes indicate the group: circles for 'emitter' and squares for 'collector'. Real numbers are virtual hop distances.

transmission separation in a multi-hop topology. In this example, hop 1 consists of three nodes (A, B, C) while hop 2 contains only one node (D). Given this topology, in *Weaver*, packets from nodes closer to the sink, such as A and B, are more likely to be received first (Fig. 6b). This is because *Weaver* adopts a three-slot structure (TX-RX-RX) (Fig. 5a) that separates upward data packets from downward ACKs but nodes within the same hop still attempt to transmit simultaneously. This causes node C, which is part of the same hop 1 but farther from the sink, to repeatedly attempt retransmissions. Furthermore, as illustrated in slot index 4 of Fig. 6b, the relaying of packets from a lower-hop node D by node A can further delay the transmission of node C's packets.

In contrast, *Cupid* adopts a two-slot structure (TX-RX) and introduces group-based intra-hop transmission separation (Fig. 6c), resulting in a more efficient data collection structure. By alternately assigning *silent mode* to each group, even distant nodes like node C can transmit data more quickly. For instance, while *Weaver* requires four transmissions to deliver a packet from node C, *Cupid* successfully delivers the same packet in a single transmission. Furthermore, this approach enables faster relaying of data from lower-hop nodes like node D. In summary, *Cupid* leverages intra-hop grouping to reduce contention, enhance fairness, and achieve faster, more efficient data collection.

V. Evaluation

We compare *Cupid* against *Crystal* [11] and *Weaver* [12], the two state-of-the-art CTX-based convergecast protocols, through real experiments on a UWB testbed.

A. Experiment Setup

Experiments are performed on the CLOVES [30] testbed with EVB1000 boards (Fig. 7). These boards feature the DW1000 [27] UWB radio chipset and an STM32F105 ARM Cortex M3 MCU. We use UWB radio channel 4 with a center frequency of 3993.6 MHz and bandwidth of 1331.2 MHz, resulting in a pulse duration of approximately 1 ns. Preamble code 17 is used, along with the preamble repetition frequency (PRF) of 6.4 MHz and the preamble acquisition chunk (PAC) set to 8. Data rate is configured to the maximum possible 6.8 Mbps.

A total of 36 nodes are used for the experiments as shown in Fig. 8. Unless stated otherwise, Node 1 marked in red serves as the sink node, while the remaining 35 nodes participate in data collection, forming a multi-hop structure of up to 3 hops (we later change the sink node and vary the number of source nodes to diversify the experiments). We implement *Cupid* in Contiki-NG open-source operating system[1]. For slot scheduling across nodes, we utilize the Time Slot Manager (TSM) implemented in *Weaver* and apply it to *Cupid*. By default, we use the longest frame sizes (126 bytes, including the TSM header) supported by the system for the experiments.

For comparison schemes, we use Crystal and Weaver.

- **Crystal.** The duration (W) of each phase is determined based on the packet length and topology depth, referring to the configurations provided in [11].
- **Weaver.** *Cupid* and *Weaver* share common features, such as the bootstrapping process, two types of ACKs, and a termination period, which are configured identically for fair comparison. For instance, the global ACK batching interval is set to Y=4 as in the original *Weaver* paper [12], and a unified slot duration of 813μs is used based on the packet size of 126 bytes.

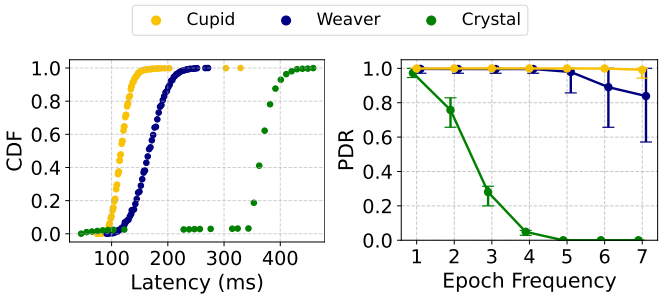
Finally, all results are statistics from running at least 2,000 epochs per single setup, case, and scheme.

B. Grouping Results

To demonstrate that the grouping algorithm works as intended, we present the accuracy of ranging to the nearest node along with a snapshot of the grouping results in Fig. 8. The accuracy of ranging is calculated on the basis of the actual distance to the nearest node in the upper or lower hop as the ground truth. After taking 100 distance measurements, the average error is found to be 1.31 m. Although this error exceeds the precision of state-of-the-art UWB ranging techniques [31], it aligns with the expected average error of approximately 1.2 m (4 ns) due to the 8 ns timestamp resolution of the simplified early-termination[2] used in *Cupid*.

[1]Contiki-based software package for DecaWave EVB1000/DWM1001 platforms, https://github.com/d3s-trento/contiki-uwb

[2]State-of-the-art UWB ranging employs double-sided two-way ranging [31] with multiple packet exchanges for more precise measurement, which we relax for lower overhead.



(a) CDF of latency (ms) **(b)** PDR vs. epoch frequency

Fig. 9: Latency and PDR results under maximum traffic (O=35). (Errorbars in (b) are for 95^{th} and 5^{th} percentiles.)

This demonstrates that the distance to the nearest node is successfully measured using simplified concurrent ranging.

Furthermore, the grouping snapshot (Fig. 8) and the performance results (§V-C) confirm that the level of error is sufficient for the grouping process and reliable data collection. In Fig. 8, the real numbers next to each node represent the *virtual hop distances* calculated after 10 iterations. For example, node 28 is primarily assigned to hop 2, but its *virtual hop distance* of 2.75 indicates it is physically closer to hop 3 than hop 1, and it has been correctly assigned to the 'collector' group (square shape). Additionally, distance-based grouping reflects non-line-of-sight (NLOS) conditions and signal strength [32]. For instance, in hop 2, nodes 17, 26, 27, and 28 belong to the collector group, while nodes 14, 15, and 16, despite being closer to hop 3, are assigned to the emitter group due to lack of communication with node 18. This confirms that *Cupid*'s distance measurement accurately captures the communication environment.

During the grouping period, only the shortest packets (payload length = 0) are transmitted to ensure accurate distance measurements, with a slot duration of 460 μs. After M iterations, N nodes are assigned their averages, requiring at least $(\text{Bootstrapping} + 3 \times N) \times \text{slot duration} \times M$. In our proof-of-concept implementation with N=35 and M=10, the bootstrapping iteration is set to 10, resulting in a total time of less than 0.5 s to collect all necessary grouping information.

C. Performance Comparison Results

We present the key performance results: (1) epoch latency, (2) packet delivery ratio (PDR), and (3) the impact of varying traffic load, for the three compared schemes.

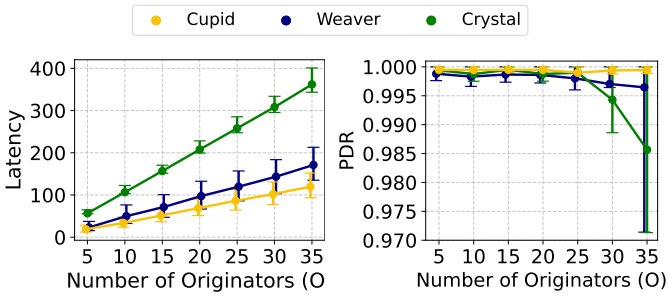
Latency Performance. First, we analyze the latency distribution of the three protocols under maximum traffic conditions; i.e. all 35 nodes generate/originate data and transmit to the sink (O = 35). Latency is defined as the time required for data collection from all 35 nodes in each epoch, measured from the start of the epoch to the moment the sink receives the last data packet of that epoch (one unique data packet per source node per epoch). Fig. 9a is the cumulative distribution function (CDF) of the latency for 2,000 epochs per scheme. It clearly shows that *Cupid* exhibits the lowest latency compared to the other two schemes, with median of 119.6 ms and most epochs completing within ~200 ms. In contrast, *Weaver* experiences median and approximate tail latency of 168.3 ms/250 ms, and *Crytal* being the worst with those at 360.3 ms/450 ms. On average, *Cupid* has 66.8% and 29.0% lower latency compared to Weaver and Crystal, respectively.

These latency results have an important implication for system throughput: How fast can the system collect data from every node in the network. For example, above results imply that *Cupid* can collect 5 packets per node per second while Crystal can only do 2 packets per node per second. Alternatively, the latency result determines the *stable minimum epoch period* that the system can use for data collection. *Epoch period* represents the maximum duration of a single epoch before which the sink stops waiting for incoming packets for that epoch. If set too short, the epoch may terminate prematurely before receiving all packets, including potential retransmissions, resulting in low PDR. If set too long, it will slow down the system resulting in low overall throughput. Thus, it must be set as low as possible for high throughput but high enough to ensure reliable data collection.

Epoch Period. To assess the impact of epoch period on reliability, we measure packet delivery ratio (PDR) with varying epoch frequency. PDR is the ratio of unique packets successfully received at the sink to the total number of those packets sent by all nodes during each epoch. *Epoch frequency* is simply an inverse of epoch period; e.g., an epoch period of 200 ms equals running 5 epochs in one second. Fig. 9b plots the result. It first shows that, as epoch frequency increases, PDR decreases for all protocols due to shorter epoch periods causing data collection to end prematurely. Notably, however, *Cupid* achieves significantly higher PDR compared to *Weaver* and *Crystal*, especially at high epoch frequency. While *Weaver*'s PDR drops to 89.1% at an epoch frequency of 6, *Cupid* maintains 99.9% under the same conditions. These results, consistent with Fig. 9a, indicate that longer data collection latencies increase sensitivity to changes in the epoch period, and the lower latency of *Cupid* allows it to maintain higher reliability and throughput than *Weaver* and *Crystal*.

Impact of Traffic Load. To evaluate the impact of varying traffic load on each system, we adjust the number of data originators, O, among the total 35 nodes, to {5, 10, 15, 20, 25, 30, 35} nodes while all 35 nodes still participate in forwarding. The originators are randomly selected and rotated for each value of O to ensure equal transmission opportunities across nodes. The impact of traffic load is assessed when epoch period set to 250 ms for *Cupid* and *Weaver*, and 1000 ms for *Crystal* (to provide sufficient time for retransmissions).

Fig. 10a plots the data collection latency (with min/max error bars) with varying number of originators. Overall, latency increases linearly as the number of source nodes increases, and *Cupid* demonstrates consistently lower latency compared to *Crystal* and *Weaver* with the difference becoming more



(a) Average epoch latency (ms) **(b)** PDR (Packet Delivery Ratio)

Fig. 10: Comparison of latency and PDR under different traffic loads (number of originators, O) with fixed epoch period. (Errorbars are for 95^{th} and 5^{th} percentiles, and are moved slightly in x-axis to avoid overlap.)

TABLE I: Packet delivery ratio (PDR) and latency (ms) comparison for different topologies via different sink node configurations

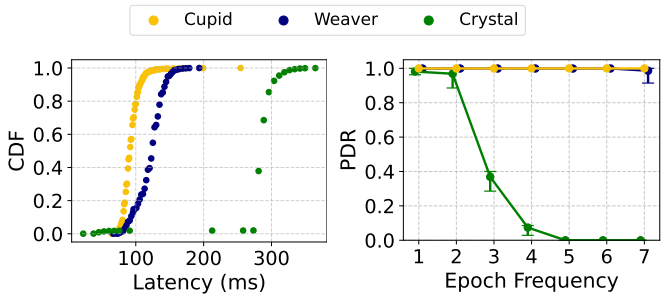
Scheme	Metric	Sink Node ID			
		1	5	15	25
Crystal	PDR	0.9809	0.9724	0.9901	0.9398
	Latency (ms)	360.280	359.801	369.832	353.154
Weaver	PDR	0.9976	0.9998	0.9996	0.9962
	Latency (ms)	168.299	119.770	114.588	108.876
Cupid	PDR	**0.9999**	**0.9999**	**0.9998**	**0.9998**
	Latency (ms)	**119.623**	**93.186**	**106.486**	**104.684**

significant as the number of origin nodes increases while the ratio being consistent with Fig. 9a. In addition, Fig. 10b plots the PDR. All schemes achieve reasonably good PDR when the number of originators are below 30. However, it does gradually decrease as the traffic load increases, and *Crystal* starts to fall when the number of originators go above 30. This result is consistent with that in Fig. 9b.

These superior performances can be attributed to *Cupid*'s intra-hop grouping mechanism, which prioritizes transmissions based on relative distances within the same hop. By effectively reducing contention and ensuring fairness in packet reception, *Cupid* demonstrates its ability to lower latency and maintain high reliability even under challenging network conditions.

D. Different Settings

Impact of Topology Variation. To investigate whether *Cupid*'s performance can generalized on different physical topologies, we alter the sink node. This effectively creates a drastically different logical hop and virtual hop distance convergecast structure. The experimental results under high-traffic conditions (O=35) are presented in Table I. *Crystal* exhibits significantly higher latencies exceeding 350 ms and lower PDRs compared to *Weaver* and *Cupid*. In contrast, both *Weaver* and *Cupid* maintained a PDR above 99%, with most cases achieving a near perfect 99.9%. Notably, *Cupid* achieves the lowest latency for all cases, outperforming *Weaver* and demonstrating its enhanced reliability and throughput in data collection.



(a) CDF of latency (ms) **(b)** PDR vs. epoch frequency

Fig. 11: Latency and PDR results for short packets (26 Bytes). (Errorbars in (b) are for 95^{th} and 5^{th} percentiles.)

Impact of Packet Length. Finally, results with shorter packets (23 B) are shown in Fig. 11. All three protocols exhibit a reduction in latency when using short packets. Specifically, *Cupid* completes data collection within an average of 92.2 ms, followed by *Weaver* at 120.0 ms while *Crystal* requires 278.7 ms[3]. This difference in latency impacts reliability at higher epoch frequencies. Both *Cupid* and *Weaver* maintain a high PDR of 99.9% up to an epoch frequency of 6, while *Crystal* exhibited a significant decline in PDR as the epoch frequency increases. Furthermore, at an epoch frequency of 7, *Weaver* begins to exhibit a drop in PDR, whereas *Cupid* continues to sustain a PDR above 99.9%. The results show that *Cupid* provide lower latency and higher PDR compared to *Weaver* and *Crystal* even for short packets.

VI. Conclusion

We proposed *Cupid*, a novel CTX-based multihop convergecast protocol over UWB radio. *Cupid* leverages the low-overhead concurrent ranging capability of UWB to redesign intra-hop scheduling via grouping, which improves fairness in the reception of simultaneously transmitted packets within the same hop. This approach contributes to improved network robustness and significant latency reduction. Evaluations on a real 36-node UWB testbed demonstrate that *Cupid* reduces latency by up to 66.8% over *Crystal* and up to 29.0% over the state-of-the-art prior work *Weaver* while providing similar or better reliability and increasing overall throughput. In future work, we plan to explore UWB MAC protocols for *Cupid* to optimize the slot size depending on packet size and further reduce latency. We also aim to experiment *Cupid* in more diverse and larger-scale environments and application scenarios to validate its scalability and robustness.

References

[1] R. Tan, G. Xing, J. Chen, W.-Z. Song, and R. Huang, "Quality-driven volcanic earthquake detection using wireless sensor networks," in *Proceedings of the 31st IEEE Real-Time Systems Symposium*, 2010, pp. 271–280.

[3]A few outlier data points below 100 ms for Crystal are due to a large number of packets being lost and never being received for latency calculation.

[2] D. Tracey and C. Sreenan, "A Holistic Architecture for the Internet of Things, Sensing Services and Big Data," in *IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, 2013, pp. 546–553.

[3] E. Sisinni, A. Saifullah, and S. Han, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *Industrial Internet of Things: Challenges, Opportunities, and Directions*, vol. 14, no. 11, p. 4724–4734, July 2018.

[4] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The Tenet Architecture for Tiered Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 6, no. 4, pp. 34:1–34:44, Jul. 2010.

[5] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*, 2009, p. 1–14.

[6] H.-S. Kim, H. Kim, J. Paek, and S. Bahk, "Load Balancing under Heavy Traffic in RPL Routing Protocol for Low Power and Lossy Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 964–979, Apr. 2017.

[7] H.-S. Kim, J. Paek, D. E. Culler, and S. Bahk, "PC-RPL: Joint Control of Routing Topology and Transmission Power in Real Low-Power and Lossy Networks," *ACM Transactions on Sensor Networks*, vol. 16, no. 2, Mar. 2020.

[8] N. D. Tan and V.-H. Nguyen, "EE-TLT: Energy-efficient routing protocol using two-level tree-based clustering in wireless sensor network," *Journal of Communications and Networks*, vol. 25, no. 6, pp. 734–749, 2023.

[9] M. Park and J. Paek, "On-demand Scheduling of Command and Responses for Low-power Multihop Wireless Networks," *Sensors*, vol. 21, no. 3, Jan 2021.

[10] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'11)*, 2011, pp. 73–84.

[11] T. Istomin, A. L. Murphy, G. P. Picco, and U. Raza, "Data Prediction + Synchronous Transmissions = Ultra-low Power Wireless Sensor Networks," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems (SenSys'16)*, 2016, pp. 83–95.

[12] M. Trobinger, D. Vecchia, D. Lobba, T. Istomin, and G. P. Picco, "One flood to route them all: Ultra-fast convergecast of concurrent flows over UWB," in *Proceedings of the 18th ACM Conference on Embedded Networked Sensor Systems (SenSys'20)*, 2020, pp. 179–191.

[13] E. K. William and M. C. Chan, "SpeedCollect: Data Collection Using Synchronous Transmission for Low-Power Heterogeneous Wireless Sensor Network," in *In Proceedings of the 19th International Conference on Embedded Wireless Systems and Networks (EWSN'22)*, 2022.

[14] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power Wireless Bus," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys'12)*, 2012, pp. 1–14.

[15] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Chaos: Versatile and Efficient All-to-All Data Sharing and In-Network Processing at Scale," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys'13)*, 2013, pp. 1–14.

[16] M. Mohammad and M. C. Chan, "Codecast: Supporting Data Driven in-Network Processing for Low-Power Wireless Sensor Networks," in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'18)*, 2018.

[17] C. Herrmann, F. Mager, and M. Zimmerling, "Mixer: Efficient manyto-all broadcast in dynamic wireless mesh networks," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys'18)*, 2018, pp. 145–158.

[18] J. Debadarshini and S. Saha, "SyncCast: Real-Time Self-Adaptive and Fault-Tolerant All-to-All Data-Sharing in IoT," in *IEEE 31st International Conference on Network Protocols (ICNP'23)*, 2023, pp. 1–11.

[19] H.-S. Kim, J. Ko, D. E. Culler, and J. Paek, "Challenging the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL): A Survey," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2502–2525, Sep. 2017.

[20] B. A. Nahas, S. Duquennoy, and O. Landsiedel, "Concurrent transmissions for multi-hop Bluetooth 5," in *In Proceedings of the 16th International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2019, pp. 130–141.

[21] D. Lobba, M. Trobinger, D. Vecchia, T. Istomin, and G. P. Picco, "Concurrent transmissions for multi-hop communication on ultra-wideband radios," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2020, pp. 132—-143.

[22] D. Vecchia, P. Corbalan, T. Istomin, and G. P. Picco, "Playing with Fire: Exploring Concurrent Transmissions in Ultra-wideband Radios," in *IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2019.

[23] M. Trobinger and G. P. Picco, "0A Case for Ultrawideband Concurrent Transmissions in Wireless Control," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 39 651–39 664, 2024.

[24] C.-H. Liao, Y. Katsumata, M. Suzuki, and H. Morikawa, "Revisiting the So-called Constructive Interference in Concurrent Transmission," in *IEEE 41st Conference on Local Computer Networks (LCN)*, 2016.

[25] K. Leentvaar and F. Flint, "The Capture Effect in FM Receivers," *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 531–539, 1976.

[26] J. Lu and K. Whitehouse, "Flash Flooding: Exploiting the Capture Effect for Rapid Flooding in Wireless Sensor Networks." in *IEEE INFOCOM*, 2009.

[27] Decawave, "DW1000 UWB Device Datasheet," 2021, https://www.decawave.com/dw1000-datasheet.

[28] C. Pablo and G. P. Picco, "Concurrent Ranging in Ultra-wideband Radios: Experimental Evidence, Challenges, and Opportunities," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2018.

[29] P. Corbalán and G. P. Picco, "Ultra-wideband Concurrent Ranging," *ACM Transactions on Sensor Networks (TOSN)*, vol. 16, no. 4, pp. 1–41, 2020.

[30] M. David, P. Gianpietro, T. Matteo, and V. Davide, "Poster abstract: Cloves: A Large-scale Ultra-wideband Testbed." in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2023, pp. 808–809.

[31] D. Coppens, A. Shahid, S. Lemey, B. V. Herbruggen, C. Marshall, and E. D. Poorter, "An Overview of UWB Standards and Organizations (IEEE 802.15.4, FiRa, Apple): Interoperability Aspects and Future Research Directions," *IEEE ACCESS*, vol. 10, Jun 2022.

[32] Y. Chen, J. Wang, and J. Yang, "Exploiting Anchor Links for NLOS Combating in UWB Localization," *ACM Transactions on Sensor Networks*, vol. 20, no. 3, May 2024. [Online]. Available: https://doi.org/10.1145/3657639