Distributed TDMA for Mobile UWB Network Localization

Yanjun Cao, *Student Member, IEEE*, Chao Chen, David St-Onge, *Member, IEEE*, and Giovanni Beltrame, *Senior Member, IEEE*

***Abstract*—Many applications related to the Internet-of-Things, such as tracking people or objects, robotics, and monitoring require localization of large networks of devices in dynamic, GPS-denied environments. Ultra-WideBand (UWB) technology is a common choice because of its precise ranging capability. However, allowing access and effective use of the shared UWB medium with a constantly changing set of devices faces some particular challenges: high frequency of ranging measurements by the devices to improve system accuracy; network topology changes requiring rapid adaptation; and decentralized operation to avoid single points of failure.**

In this paper, we propose a novel Time Division Multiple Access (TDMA) algorithm that can quickly schedule the use of the UWB medium by a large network of devices without collisions in local network neighborhoods and avoiding conflicts with hidden terminals, all the while maximizing network usage. Using exclusively the UWB radio network, we realize a decentralized system for synchronization, dynamic TDMA scheduling, and precise relative positioning on a multi-hop network. Our system does not have special nodes (all nodes are equal) and it is sufficiently scalable for real-world applications. Our method can be applied to implement device localization services in large spaces without GPS and complex topologies, like malls, museums, mines, etc. We demonstrate our method in simulation and on real hardware in an underground parking lot, showing the effectiveness of its TDMA schedule for relative localization.

***Index Terms*—TDMA, multi-hop network, distributed localization, Ultra-WideBand, mobile ad-hoc network (MANET)**

I. Introduction

INDOOR localization and tracking have the potential to unlock a plethora of new concepts and applications for public space enhancement [1].

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant (No. 2019-05165), FRQNT "Stages internationaux - Énergie-Numérique-Aérospatiale" funding (No. 263380), SSHRC Insight grant "Observatoires de l'Inaccessible (pendant ce temps)" (No. 435-2018-1163), and Canada Council for the Arts grant "Point d'Origine, Étude No.2 Chambord (No. 1003-19-1138).

M. Cao was with the Department of Computer and Software Engineering, Polytechnique Montreal, Canada. He is now with Huzhou Institute of Zhejiang University, Huzhou, China.
E-mail: yanjun.cao@polymtl.ca, yanjundream@outlook.com

Dr. Beltrame is with the Department of Computer and Software Engineering, Polytechnique Montreal, Canada
E-mail: giovanni.beltrame@polymtl.ca.

M. Chen is with the Institute of Network Engineering, National Chiao Tung University, Taiwan
E-mail: cchen.cs07g@nctu.edu.tw.

Dr. St-Onge is with the Department of Mechanical Engineering, Ecole de technologie superieure, Canada
E-mail: david.st-onge@etsmtl.ca.

The rapid development of hardware and software technology needs to address a higher demand for accurate localization services, such as person tracking, crowdsensing, and device localization [2]. In particular, a scalable system with few or no dependencies on fixed infrastructure is highly desirable, allowing for faster deployment and reducing effort for users.

While many paths have been explored, it is still very challenging to deploy a scalable system with mobile and dynamic nodes [3]. Accurate tracking can be acquired with expensive sensors (e.g. multiple cameras, LiDARs, radars, etc.), but their cost limits their applicability for the Internet-of-Things (IoT). Building infrastructure support to provide localization-based services is the most common approach: satellite-based GPS or cellular systems (5G, LoRa) are mostly for outdoor use and can have low localization accuracy depending on the environment. Camera arrays and radio beacon setups can provide indoor localization, but can be expensive and labor-intensive to set up, limiting their scalability.

EM-based localization is usually lower cost and more suitable for IoT devices. Wifi, Bluetooth, and RFID have respective techniques for indoor positioning, however with limited accuracy [2]. Ultra-Wide Band (UWB) is also used for localization, using ultrashort pulses to achieve very accurate and high-frequency ranging as well as transmitting data, which makes it an ideal choice for indoor localization.

Anchor-based UWB localization has been widely commercialized [4], [5]. However, commercial systems mainly use Time Difference of Arrival (TDOA), a technique which requires nanosecond clock synchronization between anchors [6]. This limits the scalability of the system as a high level of synchronization accuracy is hard to maintain in a distributed system. In addition, the positions of agents are usually calculated by a server in a centralized fashion, requiring an available low-latency communication infrastructure. To maximize scalability, we target a distributed system with minimal infrastructure support.

Two way ranging (TWR) [7] is a more flexible ranging solution, enabling arbitrary pair of nodes to perform distance measurements at any time. The key challenge with TWR is the access control to a shared medium, i.e. the UWB communication channel. Ranging using TWR takes significantly longer than simply broadcasting a message, requiring a medium access control (MAC) mechanism to coordinate the measurements across all devices at a given location. Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA) are general strategies for MAC in many networks [8].

TDMA allows several devices to share the UWB channel by dividing the medium access into different time slots. For accurate localization, we want to maximize the number of ranging measurements, therefore requiring high utilization of the available time slots.

Our requirement of high channel usage for the system prevents the use of a contention-based TDMA [9] as the potential of collisions is very high. Another major category of distributed TDMA algorithms is based on request and response negotiations [10]. However, these techniques usually require several iterations to converge, which makes them difficult to apply in dynamic networks where the topology changes frequently. In this paper, we propose a novel distributed TDMA algorithm that allows conflict-free scheduling with almost full channel usage. In addition, our method supports multi-hop networks, where slots can be multiplexed and used by nodes that are two hops apart. Another major advantage of our method is that scheduling can be obtained and updated iteratively, making it suitable for dynamic networks of mobile devices. Without loss of generality, we apply our TDMA algorithm to a UWB network.

The mobility of devices poses challenges not only for TDMA scheduling, but also for the localization of nodes in the network. Cooperative localization can help improve the localization performance of a large scale network [11]. In our system, each node cooperates with its neighbors to construct a local topological map, which is broadcast to recover the global map of the network, allowing for the localization of a multi-hop device network. Our localization system has potential applications in crowdsensing, where privacy-preserving methods [12], [13] could be applied to avoid sharing localization data, and in the adaptive control of vehicular networks, where current software-defined networks rely on the centralized knowledge of topology and traffic conditions [14], [15].

The constraints of our application scenario (1. absence of fixed calibrated anchors, 2. large distributed network management, and 3. dynamic network topology) call for a new approach to the MAC control of UWB channels: one that optimizes UWB access to grant high measurement rates that are necessary for high localization accuracy. Our contributions can be summarized as:

- a scalable decentralized network management to be used for relative localization in mobile ad-hoc networks;
- a novel TDMA algorithm that allows fast convergence and full channel usage;
- a neighborhood topological map construction algorithm to be used for global cooperative map recovery.

II. Related Work

UWB technology has attracted substantial attention due to its high ranging accuracy. Beside the centralized usage of anchor-based systems [16], [17], point-to-point ranging started receiving more attention [3], [18]–[20]. These results show the advantage of UWB for multi-agent systems, but due to the small number of UWB nodes in the experiments, network usage has not yet been explored. Ridolfi et al. [21] analyzed the scalability of UWB-based localization and showed the huge impact of the coordination protocol on scalability. Qin et al. [6] designed a BLAS system that uses UWB for the localization of a multi-agent system. They divided the agents into parent and child groups: the parent agents act as moving anchors and create a coordinate frame for the system. They proposed a distributed clock synchronization in parent agents to maintain a high clock precision. This enables the child group to get the position by time-of-arrival (TOA) measurements. The biggest advantage of this system is the number of child agents is theoretically unlimited because they only passively receive the pings from the parents. Although the child nodes have no conflicts, the communication between parent agents still needs to be coordinated. Qin et al. use round-robin as distributed TDMA to achieve collision-free broadcasting from parent agents. However, this solution assumes all the parent agents are fully connected, which limits the scalability of the system. Macoir et al. [22] also uses an anchor-based TDOA strategy, but it is designed for relatively large scale networks. The authors divided the network into multiple small cells to cover large areas. Unlike the passive child nodes from [6], Macoir et al [22] schedule active slots for mobile tags to broadcast messages and use a server connected with the anchors to calculate the positions. The server is also responsible for slot assignment and thereby forming a centralized system.

Zhu and Kia [23] proposed a negotiation-based dynamic TDMA algorithm across the UWB network, G.M. ter Horst [3] developed an anarchic TDMA algorithm based on the DESYNC algorithm [24]. Both methods only consider one-hop collisions, so that collision can occur for hidden nodes at neighborhood boundaries.

The most widely used MAC control on UWB is the IEEE802.15.4-2011 protocol [25], integrated with the Decawave 1000 [7] chip, which is the most popular commercial UWB chip on the market. The protocol uses Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) or slotted ALOHA [9], [26] to avoid collisions. However, these two strategies are only applicable to lightly loaded networks that have a small probability of collisions. In our case, we want to maximize channel usage for accurate localization. To the best of our knowledge, this work is the first to apply dynamic TDMA for UWB localization in multi-hop ad-hoc networks with mobile devices.

Compared to some TDMA techniques available for communication networks, our application has several additional requirements: 1) the maximization of channel usage to increase the localization frequency; 2) rapid time slot scheduling to account for dynamic topologies; 3) decentralization to avoid the need for fixed infrastructure.

The TDMA slot assignment in a wireless network can be seen as an extension of the vertex colouring problem in graph theory, with the additional constraint of needing to avoid collisions in 2-hop neighborhoods [27]. The problem was proven to be NP-complete [28], and several heuristic solutions were proposed [27]–[29] to get the near-optimal results with full knowledge of the network topology.

When considering a distributed system where nodes only receive messages from neighbors, some works (FPRP [10], DRAND [30], DICSA [31], PCP-TDMA [32]) propose

negotiation-based algorithms to find the smallest frame with conflict-free scheduling. After scheduling, all nodes in a network need to agree on the same frame size to execute the slot schedule with the same frame reference. We believe that this strategy is not the best strategy for a highly dynamic network, as it would require fast global consensus of the frame size across the network. Furthermore, when a node makes a proposal during negotiation, it has to wait for the feedback of all neighbors, potentially leading to long convergence times if the neighborhood size is large or the network topology is complex. For similar reasons, practical mobile networks use fixed-length frames such as USAP [33] for military telephone networks and VeMAC [34] for vehicular ad-hoc networks (VANETs).

Since the size of the frame should be larger than the total number of nodes in the network, these protocols may have several unused time slots. Researchers found a balance between frame size stability and channel usage by doubling or halving the frame size, such as in USAP-MA [35] and Dynamic-TDMA [36]. Cao and Lee proposed VAT-MAC [37], a VANET with a changing frame size, relying on a roadside unit (RSU) infrastructure. In our method, we use a fixed frame size that equals the total number of nodes allowed in the system. Despite this static allocation, we have high channel usage since we always allocate all the available time slots to the neighborhood's devices.

The overall goal of our system is to localize nodes within the network. We use least square optimization to find the coordinate of the nodes by carefully selecting reference nodes to do trilateration [38] and iterative multilateration [39], [40] with located nodes. However, considering the distributed and dynamic characteristics of our network, applying TWR between all nodes in the network in real-time can be challenging [41], [42]. Therefore, we propose a two-stage strategy: we create a local relative localization map and then merge the local maps to get the whole network spatial information.

III. System Overview

With the goal of accurate localization in a dynamic UWB ad-hoc network, we developed a fully decentralized system that does not require any fixed infrastructure. By designing a novel MAC protocol, nodes can make full use of the channel to get ranging measurements across the network. Our system includes four main modules: synchronization (to align the frame boundaries), distributed TDMA (to get collision-free slot assignments), relative localization, and global map merging.

Synchronization and distributed TDMA are related to the MAC protocol of the UWB network, while relative localization and global map merging combine the devices' ranging measurements and are executed in the time slots assigned by the MAC protocol.

Fig. 1 shows the system architecture from the perspective of a node with ID i. All nodes are identified by a unique ID, are considered equal in the network, and run the same software. The overall localization runs in a cycle with n time slots, where n is the total number of nodes in the network. This choice is without loss of generality: this number is usually known for a given application, and not all devices need to be active or present.

Each node is automatically assigned the slot with the same ID as the node. At the beginning of each frame, each node checks its synchronization with the node with the lowest ID in range, ensuring that all frames start at the same time.

All the time slots in a frame can be divided into: own time slot, extra time slots, and passive time slots. The purpose of distributed TDMA is to find conflict-free extra slots to improve node's localization capability. Every time a device is in its own slot (predefined as the ID of the device), the node broadcasts TDMA scheduling packets. Nodes listen to the network during passive slots.

Finally, the devices use relative localization to build a topological map of their neighborhood. Distributed TDMA helps nodes to gain extra slots, which enables them to do ranging and broadcast these measurements. Each node constructs the local map from its own ranging measurements as well as the measurements received from its neighborhood. Each node broadcasts its local map and merges the other nodes' maps when received, progressively converging to the global map.

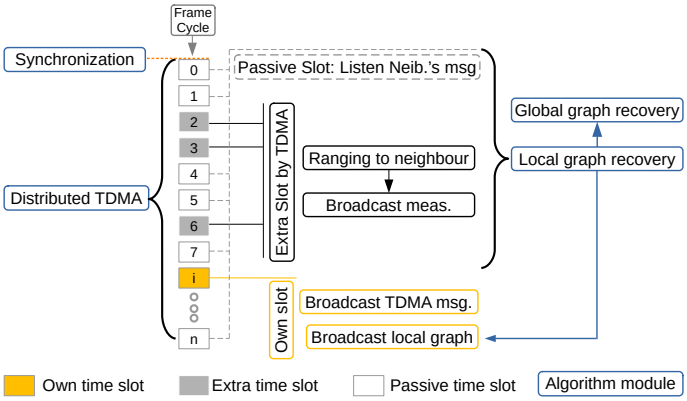

Fig. 1. System architecture of a node. Synchronization is checked at the beginning of each frame period. Distributed TDMA allows each node to be assigned with slots for ranging and broadcasting range measurements. Each node constructs a local node map from its own ranging measurements and the measurements received from its neighborhood.

IV. Frame-Based Synchronization

Synchronization is usually needed for systems with pure TDMA scheduling [43]: nodes need to have the same clock to wait for the time of their own slots. Considering that the network might be large and highly dynamic, it is not easy to effectively synchronize the entire system. In our case, we let the nodes synchronize the beginning of each frame instead of the true clock. We use two techniques to achieve distributed synchronization:

- Each node broadcasts the time offset to the starting of its frame;
- Each node listens to the offset of its neighbors and adjusts its offset using the neighbor with the smallest ID as reference.

The data usage used by transmitting an offset is less or at least not greater than for transmitting the clock. Using a reference (the lowest ID node in the neighborhood in our case) can also improve the synchronization speed with respect to a peer-to-peer gradient time synchronization setup [44].

V. Distributed TDMA

As long as all nodes in a neighborhood have the same frame start reference, the slot ID can be used as an index of the unique time period for controlling the access to the shared medium. Nodes also collect information from 2-hop neighbors to avoid conflicts at the interface of the neighborhood. Many TDMA algorithms [45], [46] use 2-hop neighborhood information to assign unallocated slots: our main contribution is a strategy to quickly allocate all these free slots with minimal conflicts in a distributed manner. To describe our algorithm, we define variables as shown in Table I.

TABLE I
NOTATIONS THROUGH THE PAPER

n	the largest ID in the system.
U	the set of node IDs. $U = \{u \mid u \in \mathbb{N}, 1 \leq u \leq n\}$
N_i^k	the *k-hop neighbor set*, consisting of the IDs of nodes exactly at k hops from node i.
aN_i^k	the *all neighbor set*, consisting of the IDs of nodes at most at k hops from node i. $aN_i^k = \bigcup_{j=1}^{k} N_i^j$
sS_i	the *send slots set*, consisting of the IDs of slots assigned to node i.
fS_i	the *free slots set*, consisting of the IDs of slots that are not assigned to node i. $fS_i = U \setminus sS_i$
cS_i	the *candidate slots set*, consisting of the IDs of slots that are not assigned to any node in aN_i^2. $cS_i = \bigcap_{j \in aN_i^2} fS_j$
sN_i	the *sibling neighbor set*, consisting of the IDs of nodes sharing the same candidate slot sets in 2-hop range of node i. $sN_i : \{j \mid cS_j = cS_i, \forall j \in aN_i^2, i \neq j\}$
$shdS_i$	the *shared slots set*, consisting of the IDs of shared candidate slots of neighbors of node i that are within 2 hops but 1) are not in sN_i 2) have cS_j that is not a superset of cS_i. $shdS_i = \bigcap_{j \in \Omega} cS_j$ $\Omega : \{k \mid k \in (aN_i^2 \setminus sN_i), cS_k \not\supset cS_i\}$

A. Frame structure

Each frame contains two cycles of communication slots, one for immediate neighbors, and one for 2-hop neighbors. The number of slots in each cycle is equal to the maximum number of nodes allowed in the system. This represents the lower bound of the frame size if all nodes are to communicate in a fully connected network [28]. We believe that this constraint is not critical as our system can assign arbitrarily large numbers of slots to each device and fully use the available bandwidth, as well as guaranteeing that all devices get at least one communication slot.

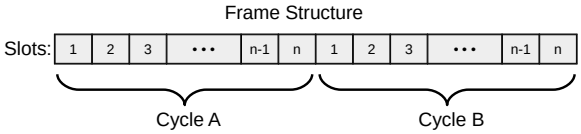

Fig. 2. The frame structure of the system. Each frame includes two cycles of slots from 1 to n. n is the maximum number of nodes allowed in the system.

As shown in Fig. 2, each frame has a two-cycle structure. Each cycle has n slots, with n is the maximum number of nodes that the system can support.

By default in each cycle, node i takes slot i, i.e. the slot with the same ID (called own slot). This ensures a lower bound of two slots per frame when all nodes share the same neighborhood, meaning we have a fully connected network. When a node moves and leaves some other nodes' communication range, certain time slots become idle. When idle slots are detected, the nodes dynamically assign these slots to improve their update frequency, as described in Section V-C.

The reason why we use two cycles (A and B) in each frame is to have a stable propagation of information of 2-hop nodes. Each node broadcasts its own ID, used and potential slots in its slot of cycle A, and the information of its neighbors in its slot of cycle B (see Fig. 3). Using this two-cycle broadcasting, each node acquires the latest 2-hop neighborhood information at each frame. This information is fed into the distributed TDMA scheduler for conflict-free scheduling. As the slots schedule for two cycles are the same, we explain one frame as n slot for simplicity.

B. Data packet format

We use different packet formats for the own slots in cycle A and B, as well as for extra slots. For own slots in cycle A the broadcast packet contains node ID, candidate slots (cS), and send slots (sS). Every node reached by the broadcast collects the information in its local memory. For own slots in cycle B, the packets contain every neighbor's cS and sS, effectively propagating 2-hop information across the network.

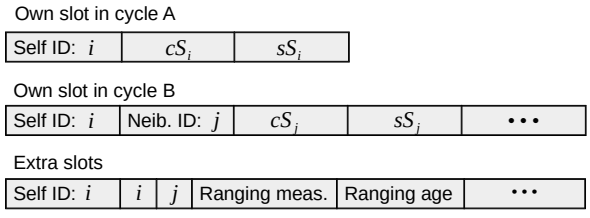

Fig. 3. Communication data packet format used in the three types of time slots in the system.

The extra slots are used for ranging with neighbors. After the ranging task is completed, each node broadcasts a timestamped ranging result. This transmission avoids repeating the same measurement from two directions and provides global knowledge of inter-node distance across a neighborhood.

C. Scheduling

With the goal to improve channel usage through rapid scheduling, we propose an algorithm that solves the scheduling problem in small number of iterations. By listening to the data broadcast by neighbors in their own time slots, each node knows the free slots it can take over. The key idea behind how we solve the assignment is to first find the unique free slots set for the node itself, and then evenly distribute the free slots with neighbors. As shown in Fig. 4 where two nodes i and j have candidate slots cS_i and cS_j respectively. The idea is that i takes $\{e|e \in cS_i, e \notin cS_j\}$; j takes $\{e|e \in cS_j, e \notin cS_i\}$, and then i and j evenly share the remainder $\{e|e \in cS_i, e \in cS_j\}$.

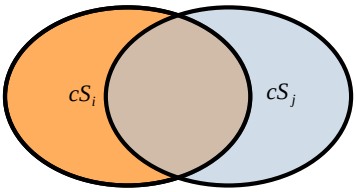

Fig. 4. Our approach to the assignment problem: the two ellipses stand for the free candidate time slots cS_i and cS_j for nodes i and j respectively. i takes $\{e|e \in cS_i, e \notin cS_j\}$; j takes $\{e|e \in cS_j, e \notin cS_i\}$, then i and j evenly share $\{e|e \in cS_i, e \in cS_j\}$.

However, when considering a real deployment with many nodes in a complex network topology, this is difficult to achieve. We propose Algorithm 1, that can safely and quickly complete the allocation of all free slots. We show a complex but representative example of four nodes with four different candidate time slots in Fig. 6.

The input of the algorithm is the received packets from all neighbors. The packets include cS and sS for all neighbors in 2-hop range. As described in Section V-A, this information is guaranteed to be received in one frame. This scheduling algorithm is executed once at the start of every frame based on the information received in last frame.

The algorithm has two key concepts among all definitions above: the *sibling neighbor set* (sN) and *shared free slots* ($shdS$). sN_i of a node i contains the neighbors within 2 hops which have the same candidate slots as i (as previously defined). One example of sibling neighbors is that nodes placed in close proximity share the same neighbors and 2-hop neighbors. $shdS_i$ are slots to be assigned, either shared between sibling neighbors or to i.

The process is formalized by Algorithm 1, which is also represented graphically in Fig. 5. This algorithm is fully decentralized, and it is executed by every node independently (for this reason, Algorithm 1 omits the subscript notation for the sets). After listening for neighbors (line 1) for one frame, each node stores the received messages, including candidate slots, send slots of its 1-hop and 2-hop neighbors.

Fig. 5 shows the cS sets of four nodes (n_i, n_j, n_k, n_p) as ellipses, marking the possible intersections between the ellipses with letters from A to G. Each intersection set represents a set of time slots.

We call this initial state of the system *step 0*. Alg. 1 proceeds with the initialization of some variables, including the creation

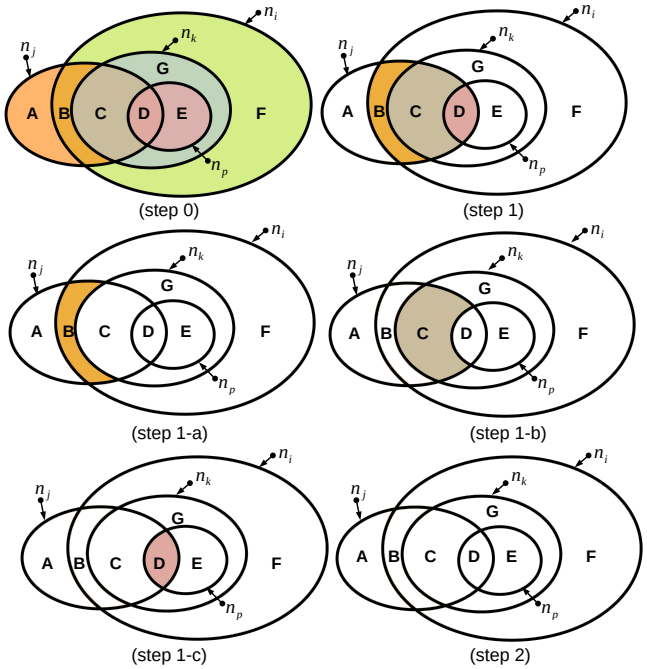

Fig. 5. Slots scheduling process corresponding to Alg. 1. The candidate slots set of four nodes (n_i, n_j, n_k, n_p) are shown as ellipses with color, marking the possible intersections between the ellipses with letters from A to G. The disappeared color means the slots in the set are assigned after a step (step 0 $\rightarrow$ step1 $\rightarrow$ step2).

Algorithm 1: TDMA schedule for slots distribution.

```
   input : rcvPackets
   output: cS, sS
1  listenNeighb(rcvPackets);
2  sS ⇐ ∅;
3  cS ⇐ {1, 2, 3, ..., n};
4  aN² ⇐ N¹ ∪ N²;
5  forall ele ∈ aN² do
6  |  cS ⇐ cS \ ele.sS
7  end
8  shdS ⇐ self.cS;
9  forall ele ∈ aN² do
10 |  if self.cS == ele.cS then
11 |  |  sN.add(ele);
12 |  end
13 |  else if self.cS ⊄ ele.cS then
14 |  |  shdS ⇐ shdS \ (self.cS ∩ ele.cS);
15 |  end
16 end
17 if size(sN) > 0 then
18 |  sS ⇐ distribute(shdS)
19 else
20 |  sS ⇐ sS ∪ shdS
21 end
22 cS ⇐ cS \ shdS
23 broadcast(cS, sS)
```

of an empty set of sending slots (sS, line 2), a candidate slots set (cS, line 3) initialized with all slots from 1 to n (with n the total number of nodes in the network), and a set that includes all 1-hop and 2-hop neighbors (aN^2, line 4). Then the cS is updated by removing all the send slots (sS) of its neighbors and 2-hop neighbors (lines 5-7). The remaining elements in

cS are the *free slots*, which initialize shared slots ($shdS$, line 8).

The next loop identifies sibling neighbors and shared slots. Going through all neighbors, if a node finds a neighbor ele with the same cS, it adds ele to the sN (lines 10-11). If the neighbor is *not* a sibling and its cS is not a superset of the node's cS, the node removes the common candidate slots elements (i.e. the intersection) from $shdS$ (lines 13-15). This rule makes sure the nodes can find unique shared slots to allocate. Take node k in Fig. 5 as an example: as no other node has the same cS, n_k does not have sibling neighbors. The cS_k of n_k are {C, D, E, G}, which are used to initialize $shdS_k$. When n_k compares its cS_k with those of n_j and n_p, the intersection {C, D, E} is removed from $shdS_k$. The comparison with n_i is skipped since the cS_i of n_i ({B, C, D, E, F, G}) is a superset of n_k's ({C, D, E, G}).

Similarly for n_i, its $shdS_i$ is {F} after removing {B, C, D, E, G} as intersections. Doing so for each node leads to unique $shdS$: {A} for n_j, {F} for n_i, {G} for n_k, and {E} for n_p.

After performing the above operations, the elements in $shdS$ are safe to use for each node. Each node shares its $shdS$ with its sibling neighbors if it has any (lines 18-19), or else adds $shdS$ to its own sS (lines 20-21). Then each node updates cS by removing the remaining $shdS$. The new sS and cS are broadcast in the next frame. At this point, some slots are still not assigned, like slots in {B, C, D}. These slots are assigned in the next frame.

After each node broadcasts its information in its own time slot, all nodes receive the results from step 0 and reach the state as shown in step 1 in Fig. 5. In step 1, following the same procedures for step 0, nodes n_i and n_j become sibling neighbors as they all have the same cS {B, C, D}. After checking for intersections n_k and n_p (that are not siblings), have $shdS$={B}, as step 1-a of Fig. 5 shows in orange. The sibling neighbors distribute their shared slots evenly (line 19), which means both n_i and n_j get part of {B} without collisions. Shared slots can be distributed by fairness, ID, or other rules, as long as the rule allows for unique assignments from local decisions. Nodes n_k and n_p still do not have sibling neighbors and therefore they just do the intersection check: n_k ends with $shdS$={C} at step 1-b and n_p with $shdS$={D} at step 1-c. All free slots are assigned and no candidate slots are left in step 2.

A major advantage of our algorithm is that we can quickly and safely assign all free slots, no matter how many slots in each set from A to G. This allows our system to have a fast response to dynamic topology changes.

Let us consider a specific example of a multi-hop network in Fig. 6: the four nodes (n_i, n_j, n_k, n_p) are within each other's communication range and form a cluster connected by solid black lines. Other surrounding nodes with ID 1 to 6 are 2-hop neighbors for the cluster. The dotted line between two nodes means they are in 2-hop range or that they are a hidden node [8] for each other. Node 1 is connected with n_i, and broadcasts in time slot 1 (its own slot). Therefore, n_i must be silenced in time slot 1, otherwise the node between n_i and 1 would detect a collision. As the Fig. 6 illustrates, there exist candidate slots {2, 3, 4, 5, 6} for n_i, {1, 2, 3, 4} for n_j, {3, 4, 5, 6} for n_k, and {4, 5} for n_p. It takes 3 iterations to allocate all candidate slots as listed in Table II.

TABLE II
SLOTS DISTRIBUTION PROCESS FOR EXAMPLE IN FIG. 6

		Step0	Step1	Step2	Step3
n_i	cS_i	{2,3,4,5,6}	{2,3,4,5,6}	{2,3,4}	$\emptyset$
	sS_i	$\{s_i\}$	$\{s_i\}$	$\{s_i,2\}$	$\{s_i,2\}$
n_j	cS_j	{1,2,3,4}	{1,2,3,4}	{2,3,4}	$\emptyset$
	sS_j	$\{s_j\}$	$\{s_j,1\}$	$\{s_j,1\}$	$\{s_j,1\}$
n_k	cS_k	{3,4,5,6}	{3,4,5,6}	{3,4}	$\emptyset$
	sS_k	$\{s_k\}$	$\{s_k,6\}$	$\{s_k,6,3\}$	$\{s_k,6,3\}$
n_p	cS_p	{4,5}	{4,5}	{4}	$\emptyset$
	sS_p	$\{s_p\}$	$\{s_p,5\}$	$\{s_p,5,4\}$	$\{s_p,5,4\}$

It can happen that the $shdS$ becomes empty after removing intersections with the cS of neighbors. This situation can cause a deadlock: we let the node aggressively takes all the cS when detecting no changes in cS for more than 3 frames to break the deadlock.

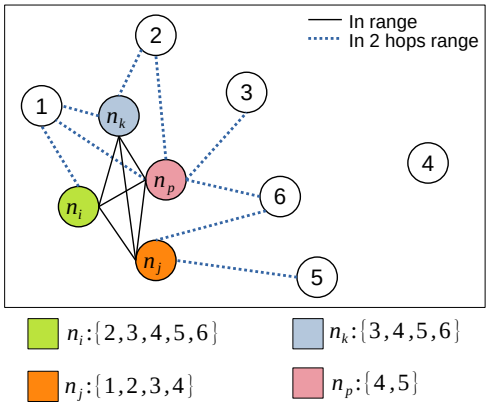

Fig. 6. Example of the candidate slots of four nodes (n_i, n_j, n_k, n_p) in a multi-hop network. The number sets at the bottom are the candidate slots. A black solid line indicates that nodes are in direct communication range, and the blue dotted line indicates they are at 2-hop distance.

D. Node arrival and departure

Algorithm 1 allocates all free slots, which means that in any 2-hop sub-graph, the shared channel medium is fully used. Therefore, when a new node enters the 2-hop neighborhood of a cluster of nodes, it generates unavoidable conflicts: at a minimum, its sending slot collides with the schedule of the cluster. We provide a two-step solution to quickly resolve the collision. Suppose node i enters the range of a cluster B, the first step is to have B release i's own slot. As we do not have precise clock synchronization, it is likely for nodes in B to receive a message from i. When a node in B receives a message from i, it adds i to its neighbor list and propagates its presence to the rest of B, freeing i's own slot.

A second step is to avoid collisions between the extra slots of i and B. We set that the node with the larger number of send slots must release the slots causing collisions. After i is allocated its own slot, i broadcasts its cS_i and sS_i and receives the same information from neighbors. When i finds it shares a slot s from sS_j with a 1- or 2-hop neighbor j (assume node $j \in B$, and $i \neq j$), namely $s \in (sS_i \cap sS_j)$, i checks

the number of its sS_i: if it has more slots in sS_i than the conflicting neighbor, it removes s from sS_i. Conversely, if j has more sS_j than i, j releases s. If both nodes have the same number of send slots, the node with the lowest ID releases s:

$$sS_i = \begin{cases} sS_i & \text{if } card(sS_i) < card(sS_j) \\ sS_i & \text{if } card(sS_i) = card(sS_j) \text{ and } i > j \\ sS_i \setminus s & \text{if } card(sS_i) = card(sS_j) \text{ and } i < j \\ sS_i \setminus s & \text{if } card(sS_i) > card(sS_j) \end{cases} \quad (1)$$

With this solution, collisions can be solved in one frame. Note that the collision-free part of the previous scheduling result is maintained, which allows the system to adapt to a new schedule. This behavior is replicated if multiple nodes join the neighborhood at the same time.

If a node leaves the neighborhood, its slots should be reassigned. When a node i stops receiving messages from a neighbor j, i removes j from its neighbor list, as well as releasing all slots assigned to j. These time slots become candidate slots and participate in TDMA scheduling as described in Section V-C.

E. Fairness and extra slots

Considering n number of slots in one frame, to guarantee a fair slot allocation, if a node i has a number of send slots $card(sS_i) > 2n/card(aN_i^2)$, meaning more than twice the average allotment of its neighborhood, n_i releases as many slots as needed to reach $card(sS_i) = n/card(aN_i^2)$. The released slots become candidate slots for its 2-hop neighborhood and follow the scheduling algorithm in Section V-C.

The role of the extra slots depends on the application. For our localization purposes, each node performs two tasks in the extra slots:

1) ranging with one of its neighbors;
2) broadcast the ranging measurement.

Every node maintains an age list of its neighbors and selects the ranging target which has the oldest ranging measurement. It is worth noting that the measurements are also updated from the neighbors' measurement broadcast, avoiding repeated mutual ranging in short intervals.

VI. Relative Localization

Some wireless sensor network localization systems use anchors or landmarks with known positions, and use centralized control to propagate the position results [47]. We propose a collaborative localization system based on ranging using UWB sensors [40].

The proposed TDMA algorithm maximizes the use of the UWB channel, allowing ranging measurements between all nodes at high frequency. Using the ranging information, each node can construct a local topological map of its neighbors' positions. By broadcasting, receiving, and fusing the local maps, the nodes can converge towards a global map.

In this work, we consider the 2D localization for the nodes in the system. Node i has a neighbor set N_i^1, and it creates a graph of its surroundings as $G_i(V_i, E_i)$ where $V_i = \{v_t | t = i \text{ or } t \in N_i^1\}$ and $E_i = \{e_{kj} | k, j \in V_i, i \neq j\}$. For construction of a local map M_i, the goal is to find:

$$M_i = \{\mathbf{X}_j | \mathbf{X}_j = (x_j, y_j) \in \mathbb{R}^2,\ j \in V_i\}$$

Similarly, for the recovery of the global map, the goal is to find:

$$M_i^G = \{\mathbf{X}_j | \mathbf{X}_j = (x_j, y_j) \in \mathbb{R}^2,\ j \in W\}$$

where $W = \bigcup_{j \in N_i} V_j$.

A. Local map construction

To build the nodes' local maps, we use a similar method as the initialization stage of our previous work [40]. The difference in this work is that each node estimates the map locally. Given our hypothesis of a highly dynamic system, waiting for all ranging measurements to be propagated to build a global map as in [40] can introduce significant errors due to the use of outdated measurements. Therefore, each node creates a local map using the latest ranging measurements (which is therefore always up-to-date), and then merge the local maps at a later stage.

To build a local map, each node selects two neighbors as *reference nodes* to initialize the coordinate frame. The choice of reference nodes can greatly influence the localization accuracy. Yang [38] proposed the idea of the quality of trilateration to find appropriate reference nodes from nodes with known positions. Priyantha [11] proposed a relative localization algorithm by first selecting five reference nodes, and then applying an optimization process. In our case, the nodes do not have any neighbor with known positions. We propose a reference node selection that considering the expected trilateration performance, the number of neighbors, and the timestamps of the ranging measurements.

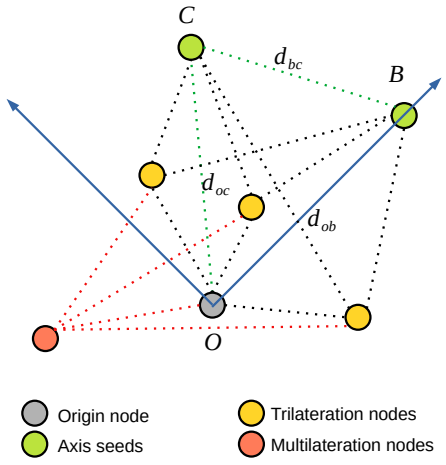

Fig. 7. Local map construction with nodes of different roles. The origin node and axis seeds (reference nodes) are used to create the coordinate frame. Nodes that have measurements to the reference nodes use trilateration to get their position. The rest estimate their position by multilateration to the nodes that already localized.

As showed in Algorithm 2, the system includes two parts: local map construction (lines 1-19) and global map recovery (lines 20-24).

Each node creates a map of its neighbors based on its own ranging measurements (named *rangeMeas*) as well as received measurements (named *receivedMeas*).

In Alg. 2, rangeMeas[tID, range, age] indicates an ego ranging measurement to the target neighbor tID, and receivedRange[sID, tID, range, age] indicates the received range measurement from node sID to node tID. All these range measurements are timestamped to allow the node to use the latest measurements.

Algorithm 2: Map construction for relative localization

input : neighbList, {rangeMeas[tID, range, age]}, {receivedMeas[sID, tID, range, age]}, receivedMaps
output: localMap, globalMap

1 $Origin \Leftarrow (0,0)$;
2 $X\_Seed \Leftarrow Top(sortedCommonNeighb(Origin) \cap sortedRange(Origin) \cap sortedRangeAge())$;
3 $Y\_Seed \Leftarrow Top(sortedCommonNeighb(Origin) \cap sortedRange(Origin, X\_Seed) \cap sortedRangeAge())$;
4 $CS \Leftarrow coordinateFrame(Origin, X\_Seed, Y\_Seed)$;
5 $locatedNodes.add(Origin, X\_Seed, Y\_Seed)$;
6 $suspendSet \Leftarrow \emptyset$;
7 **for** $ele \in neighbList \setminus locatedNodes$ **do**
8 **if** $rangeTo(Origin, X\_Seed, Y\_Seed)exist$ **then**
9 $elePos \Leftarrow trilateration(Origin, X\_Seed, Y\_Seed)$;
10 $locatdNodes.add(elePos)$;
11 **else**
12 $suspendSet.add(ele)$;
13 **end**
14 **end**
15 **for** $ele \in suspendSet$ **do**
16 $elePos \Leftarrow multilateration(ele, locatedNodes)$;
17 $locatdNodes.add(elePos)$;
18 **end**
19 $globalOptimization(locatedNodes, rangePairs)$;
20 **for** $map \in receivedMaps$ **do**
21 **if** $overlayVertexNum(globalMap, map) > 3$ **then**
22 $globalMap \Leftarrow Merge(globalMap, map)$;
23 **end**
24 **end**

The construction of the local map is divided into three stages. Consider a node i: the first stage is to find reference nodes to create the coordinate system (lines 1-5). Node i uses its position as the origin of the coordinate frame (line 1). i then identifies another node *X_Seed* to define its x axis. The selection of X_Seed is based on three criteria (line 2):

1) the number of common neighbors with node i should be large, which is found by sorting $size(N_i^1 \cap N_j^1)$, where $j \in N_i^1$;
2) the node should be as far as possible from i, which is selected by sorting e_{ij}, where $j \in N_i$, $e_{ij} \in$ rangeMeas;
3) the measurement should as recent as possible, which is selected by sorting the ages of e_{ij}, where $j \in N_i$, $e_{ij} \in$ rangeMeas.

Then node i uses similar criteria to select another node *Y_Seed* to define the y axis. Suppose node i select node j as X_Seed and k as Y_Seed. The mutual distances between i, j, and k are $[d_{ij}, d_{jk}, d_{ik}]$. i calculates the coordinates of j and k as:

$$\begin{aligned}\mathbf{X}_i &= (0,0),\\ \mathbf{X}_j &= (d_{ij},0),\\ \mathbf{X}_k &= \left(\frac{d_{ik}^2+d_{ij}^2-d_{jk}^2}{2d_{ij}}, \pm\sqrt{d_{ik}^2-\left(\frac{d_{ik}^2+d_{ij}^2-d_{jk}^2}{2d_{ij}}\right)^2}\right)\end{aligned} \tag{2}$$

which corresponds to line 4 in Alg. 2. Note that we use the positive value for the Y coordinate of $\mathbf{X}_k$. Then X_Seed and Y_Seed are added to *locatedNodes* with their coordinates.

In a second stage, each node localizes the remaining neighbors with respect to the itself and the two reference nodes by trilateration (lines 6-14). Sometimes neighbors do not have ranging measurements to all reference nodes, or the ranging measurements are outdated. In this case, the node cannot perform trilateration is added to *suspendSet* (line 12) to be reexamined at a later stage. The nodes with successful trilateration are put into locatedNodes (line 10).

In a third stage, each node attempts to localize the nodes j with $j \in$ suspendSet using the existing ranging measurements with least squares multilateration (lines 15-17) [40]:

$$\mathcal{X}_\mathbf{j}^* = \arg\min_{\mathcal{X}_j} \sum_{k \in locatedNodes} (d_{jk} - \|\mathcal{X}_j - \mathbf{X}_k\|)^2, \tag{3}$$

with $\mathcal{X}_j^*$ the computed coordinates of node j. Following this procedure, a node can acquire the coordinates of all neighbors that can be located in the local map. These nodes are added to locatedNodes. However, these coordinates do not consider the error model of the ranging sensor.

UWB sensors have TWR error that depends on many factors [48]. Lederberger et al. [49] introduce a Gaussian process error model for UWB ranging. However, this model requires knowledge of the relative angle between UWB antennas, making it impractical for our system. We apply a least square optimization to the coordinates of all nodes except for the origin and the y coordinate of X_Seed (that are all zero). Equ. 4 shows the $globalOptimization$ step of line 19 in Algorithm 2: all mutual distance measurements for nodes in locatedNodes are used to find the optimal coordinate estimations. The coordinates from the previous stages are used as the initial value for the least square optimization.

$$\begin{aligned}&\mathcal{X}^* = \min_{\mathcal{X}} \sum_{i,j \in locatedNodes} (d_{ij} - \|\mathcal{X}_i - \mathcal{X}_j\|)^2\\ &\mathcal{X}^* : [x^*_{\text{X\_Seed}}, \mathcal{X}_t^*, ..], t \in locatedNodes \setminus \text{X\_Seed}\end{aligned} \tag{4}$$

B. Global map recovery

Once a node i has built its local map M_i, it broadcasts it to its neighbors. When a node i receives a map from a neighbor j, it merges j's map M_j to its local map M_i if the maps share common nodes.

The key to merge different maps is to estimate translation, rotation and reflection with respect to a given axis [50], [51]. In our case, reflection on the x axis might be induced since nodes always select the positive value for the Y coordinate

of the Y_Seed node. Therefore, compared to a classical 2D transformation, we add an extra parameter related to reflection to the transformation matrix. Suppose node i has its own map M_i and receives map M_j from node j. The goal is to find the transformation $\mathcal{T}_j^i = [\mathbf{R}, \mathbf{T}, \mathbf{F}]$ from j to i to place the nodes that only exist in M_j in i's coordinate frame:

$$\begin{aligned} \mathcal{T}_j^i =& [\mathbf{R}, \mathbf{T}, \mathbf{F}] \\ \mathbf{R} =& \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}, \theta \in (-pi, pi] \\ \mathbf{T} =& \begin{bmatrix} t_x \\ t_y \end{bmatrix}, t_x, t_y \in \mathbb{R} \\ \mathbf{F} =& \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}, \gamma = 1 \; or \; -1 \end{aligned} \tag{5}$$

where $[\mathbf{R}, \mathbf{T}, \mathbf{F}]$ correspond to the rotation, translation, and reflection matrices, respectively.

A node can start merging two maps if the maps have more than three common vertices (lines 20-24 in Alg. 2). We use $V(M_i \cap M_j)$ to indicate the common vertices between M_i and M_j. We use a least squares optimization to find the optimal transformation $\mathcal{T}^*$:

$$\begin{aligned} \mathcal{T}^* = \min_{[\mathbf{R},\mathbf{T},\mathbf{F}]} \sum_{t \in V_o} (\mathbf{X}_t^i - (\mathbf{R} \cdot \mathbf{F} \cdot \mathbf{X}_t^j - \mathbf{T}))^2 \\ \text{where} \;\; V_o = V(M_j \cap M_i). \end{aligned} \tag{6}$$

$\mathbf{X}_t^i$ stands for the coordinates of node t in node i's coordinate frame. With $\mathcal{T}^*$, the node can transform the vertices in $V(M_j \setminus M_i)$ to its coordinate frame to get a merged map M_i^G:

$$\begin{aligned} m_{j \setminus i} &= \{\mathbf{X}_t^i | t \in V(M_j \setminus M_i), \mathbf{X}_t^i = \mathbf{R} \cdot \mathbf{F} \cdot \mathbf{X}_t^j - \mathbf{T}\} \\ M_i^G &= M_i \cup m_{j \setminus i}, \end{aligned} \tag{7}$$

where $m_{j \setminus i}$ indicates the transformed map in n_i's coordinate frame consisting of the vertices that are unique to M_j.

VII. Experiments

A. Simulations

We perform a set of simulations to assess the scalability of the proposed algorithm. The algorithm is run in a custom simulator written in Python and run on a laptop with 16GB of memory and an Intel i7-6700HQ processor. We simulate different numbers of nodes in an arena with a size of $50m \times 50m$. The communication range between nodes is $5m$. The nodes are randomly distributed in the arena, as shown in Fig. 8. We use three different orders of magnitude (10, 100, 1000) for the number of nodes in the system to simulate different deployment densities. We allow a maximum of 50 frames for the TDMA scheduling. The length of each time slot (as t_{slot}) is defined as 3 ms (the time for an UWB ranging operation [3]).

Fig. 9 shows the scheduling process for a system with a distribution of 100 nodes in Fig. 8, meaning there are 100 slots in a frame. Each node starts with a send slots set with only its own slot, and they have an average of 79 candidate slots available to assign. The number of candidate slots decreases quickly over time, which means the slots are assigned. All candidate slots are assigned around the 6th frame and each node is given 27 slots on average. The initial peak in the send slots suggests some form of collisions. Collisions are resolved in one frame (by releasing slots) and therefore the number of send slots quickly decreases.

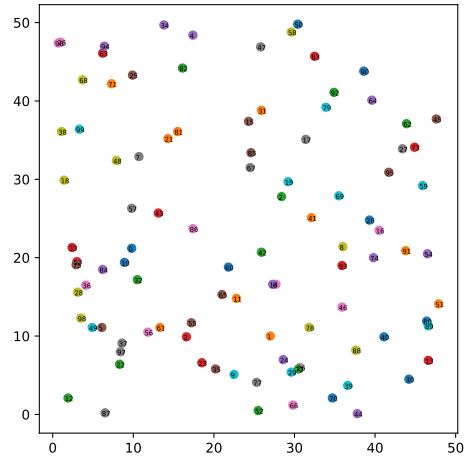

Fig. 8. Random distribution of 100 nodes in a area of 50m*50m arena.

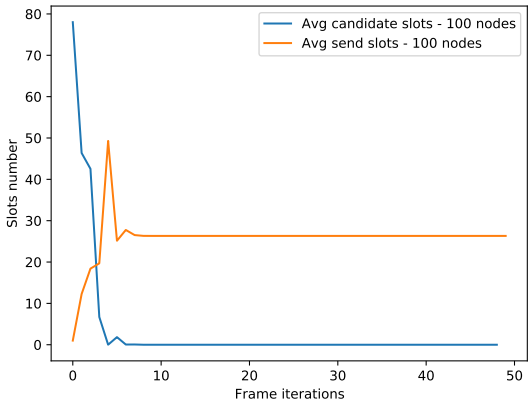

Fig. 9. Scheduling process of a system with 100 nodes, distributed in an area of 50m*50m. The average number of send slots and candidate slots are plotted. At the start, each node has 1 send slot and an average around 79 candidate slots. Around frame 6, all the candidate slots are assigned and each node got around 27 slots on average.

The distribution with 1000 nodes is quite dense as shown in Fig. 11. Using the same setup for the arena and slot size, we run each experiment 30 times and show the aggregate results in Fig. 10 and Table III.

Fig. 10 also plots the average number of frames needed to assign all the candidate slots ("frames" in the plot). We can see the increase in the number of frame iterations is fundamentally linear even for an exponential increase in the number of nodes. The average number of send slots increases significantly from 10 to 100 nodes, but due to the increase in density, it is basically unchanged from 100 to 1000 nodes.

In Table III, N_Frame_cy is the number of slots in a cycle of one frame, which has the same value of the number of nodes (N_Nodes) in the network, and N_Neighbors is the average number of neighbors of each node. $Density$ is the ratio between the "communication area" occupied by all nodes

and the total area of the arena [52], as shown in Equ. 8. As the size of the arena is fixed, it is linearly related to the number of nodes.

$$Density = \frac{N * \pi R^2}{L^2} \tag{8}$$

where $R = 5$ and $L = 50$ in our experiment.

The average (Avg) and standard deviation (Std) for frame iterations (Frames) and resulting send slots (sS) listed in Table III correspond to Fig. 10.

Finally, we report some additional measurements:

$$\begin{aligned} \text{T\_Frame\_cy} &= t_{slot} \cdot \text{N\_Frame\_cy} \\ \text{Avg\_T\_Frame\_cy} &= 2 \cdot \text{Avg\_Frame} \cdot \text{T\_Frame\_cy} \\ \text{N\_}sS &= \frac{\text{Avg\_}sS}{\text{T\_Frame\_cy}} \\ \text{LN\_}sS &= (1 + \text{N\_Neighbors}) \cdot \text{N\_}sS \\ \text{Total\_N\_}sS &= \text{N\_}sS \cdot \text{N\_Nodes} \end{aligned} \tag{9}$$

T_Frame_cy is the time taken for each frame cycle; Avg_T_Frame is the average time needed for scheduling considering two cycles; N_sS is the average number of slots per node per second, which indicates the number of range each node can measure per second; LN_sS is the total number of slots used by a local neighborhood per second, including the node and its neighbors; Total_N_sS is the total number of slots per second across the whole network, indicating the total number of ranging measurements can be made in the system.

From Table III, we can see the number of frames used in scheduling (Avg_Frames) does not increase significantly between 100 and 1000 nodes. However, the time duration (Avg_T_Frame) is still large compared to a 100 nodes configuration: this result is acceptable considering the extremely high density of devices (see Fig. 11 for an intuition).

In terms of network usage, N_sS is equal to $1/t_{slot}$ (=333.3) when there are no neighbors. For the 10 nodes configuration, which is quite sparse, the average number of slots (N_sS) for each node is 263. The slots used by local neighborhoods on average (LN_sS) is 336.4. This is because the average number of neighbors (N_Neighbors) is only 0.28.

For the other configurations, the LN_sS have a similar value. This is due to the fact that some of the slots are used by 2-hop neighbors. To find the maximum of Total_N_sS, we consider the maximum number of nodes N_{max} allowed in the arena without any intersections of their communication ranges, i.e. when each node can make full use of the channel:

$$N_{max} < \frac{L^2}{\pi R^2} = \frac{50 \cdot 50}{\pi \cdot 5 \cdot 5} = 31.8 \tag{10}$$

N_{max} is less than the ratio between the area of the arena and the communication area. Therefore, the maximum of Total_N_sS can not be larger than $N_{max} \cdot 1/t_{slot}$, which is 10600. We can see in the configurations of 100 and 1000 nodes, the results are close to full channel usage.

B. Hardware setup

We test the system with a physical implementation with 12 nodes, as shown in Fig. 12-b. Except for the unique ID of each module, all modules are identical. Each module consists of a Raspberry Pi 3 A+ with a Decawave 1000 [7] UWB module from Pozyx [4] as its ranging and communication sensor. All processing is performed on-board, with the exception of the global map construction, which is done on a separate computer used by the system operator. All communication between nodes is implemented through the UWB channel, while a 802.11 network is used for control and debugging. Please note that since the ceramic antenna of the Pozyx modules is

TABLE III
SCALABILITY STUDY

	10 nodes	100 nodes	1000 nodes
N_Frame_cy , N_Nodes	10	100	1000
N_Neighbors	0.28	2.29	28.73
Density	0.31	3.14	31.41
Avg_Frames	1.00	6.41	20.76
Std_Frames	0.0	1.25	9.46
Avg_sS	7.89	27.01	28.28
Std_sS	0.89	1.71	8.66
T_Frame_cy	0.03 s	0.30 s	3.0 s
Avg_T_Frame	0.06 s	3.84 s	124.8 s
N_sS	263	90.03	9.43
LN_sS	336.4	296.2	280.3
Total_N_sS	2630.0	9003.33	9433.33

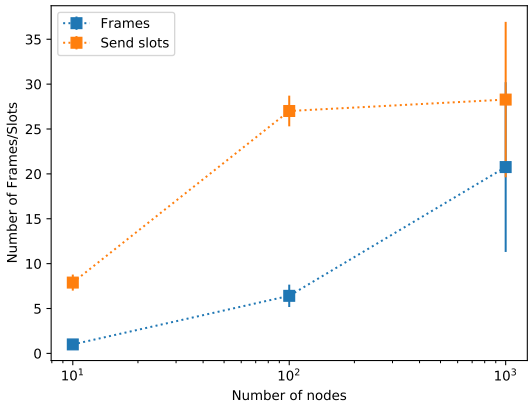

Fig. 10. Scalability study with 10, 100, and 1000 nodes. Frames stands for the average number of frames used before all the candidate slots are distributed. Send slots shows the average number of result slots each node gets.

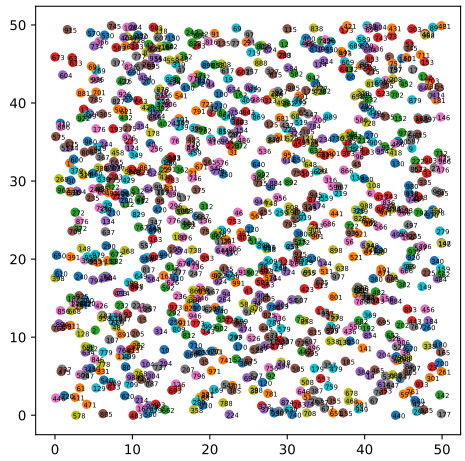

Fig. 11. Random distribution of 1000 nodes in a area of 50m*50m arena.

directional, we place the modules upright as shown in Fig. 12-a to minimize orientation-related issues.

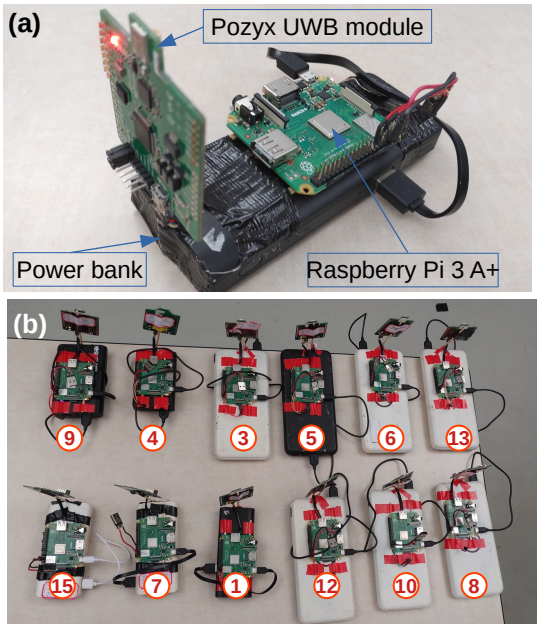

Fig. 12. The hardware modules used as nodes in our experiments. Each module consists of a battery, a Raspberry Pi 3 A+, and a Pozyx module based on the Decawave 1000 UWB chip. The numbers in (b) are the node IDs. All the processing except for the global map recovery is done onboard and all the communication between nodes passes through the UWB channel.

We conduct 3 sets of experiments to validate the system. To evaluate the performance of TDMA scheduling and rescheduling, we consider three situations: new nodes joining, nodes leaving, and multi-hop scenarios.

To show the fast response of our system to changes in high-traffic conditions, we test the effect of joining and leaving nodes in a static, fully connected scenario, by placing the nodes in proximity and turning on/off the nodes that are joining/leaving.

We also test our algorithm in a dynamic environment by moving the nodes from a fully connected network to a multi-hop network. This result shows the conflict-free scheduling and slot multiplexing usage in a complex network topology.

Finally, we conduct a comprehensive experiment to test the spatial map construction for a dynamic network with mobile devices. This last experiments shows that the system can localize the devices using ranging in the assigned time slots. Each node can construct its own local spatial map and recover the global map using local communication.

C. New nodes joining

Without TDMA (or with Carrier Sense [25]), the system may encounters a large number of collisions since all nodes attempt to achieve high channel usage.

We place twelve modules together on a desk as shown in Fig. 12-b. Starting the nodes in sequence, we can emulate new nodes joining the network. We start nodes 1, 3, 4, and 5 at first. After two minutes, we start nodes 6, 8, 12, and 13. Finally, we start the remaining nodes: 7, 9, 10, and 15.

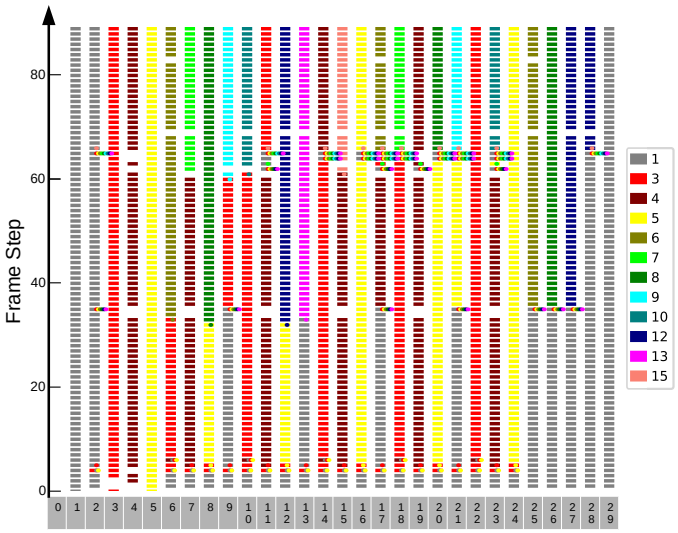

Fig. 13. TDMA scheduling results for a system with new nodes joining. Nodes group {1, 3, 4, 5}, {6, 8, 12, 13} and {7, 9, 10, 15} are started gradually. This result is from the perspective of node 4, but it is the same for everyone in the network as they are fully connected. The X-axis is the slot ID and the Y-axis is the frame iteration, representing the time. A total of 30 slots are configured. We can see that when a new group of nodes join the system near frame 35 and 53, the system can quickly adapt to the new schedule. All slots are occupied all the time to have full channel usage.

The scheduling process results are shown in Fig. 13. The numbers from 0 to 29 on the x axis represent the slot IDs. We use 30 slots in our system (with slot 0 is reserved for the processing of TDMA algorithm). We select the 30 to have a more visible scheduling result to show. The system can be viewed as a system supporting a maximum of 30 nodes, but only 12 nodes are studied. Each slot has a duration of 50 ms due to firmware limitations of the Pozyx module. Ter Horst et al. [3] show that a 3 ms slot is enough to perform ranging measurements using the same Decawave 1000 UWB chip, but for a different type of module. Our system could be set to similar values on different hardware.

Having 30 slots of 50 ms means each frame is 3 seconds due to our organization of two cycles per frame (see Fig. 2). On the x axis, we show the 30 slots of the first cycle (cycle A) of the frame as the scheduling is identical for both cycles. Considering 12 nodes and 50 ms slot size, each node can be assigned at least 2 slots per frame (one own slot and one extra slot). Therefore, the lower bound slot use for each node is 4 slots every 3 seconds. The y axis is the frame count, increasing in time, meaning that the scheduling forms a grid showing the assignment of slots in time with colours assigned according to the assigned node identity. This reporting is a general way to show the scheduling speed, and it is independent from the slot size, which can differ based on the system configuration.

From Fig. 13, we can see that slot scheduling can quickly adapt to network changes and obtain a conflict-free schedules with full slot use in two to three frames. During the first 3 frames, almost all slots except slots 3, 4, and 5 are assigned to the node with ID 1 (gray color). This is because when the first 4 nodes are started, they first synchronize. However, since the ID of node 1 is the smallest in the neighborhood, node 1

TABLE IV
SLOT DISTRIBUTION FOR NEW NODES MERGING SCENARIO

ID	Stage 1	Stage 2	Stage 3
1	1,2,9,13,17,21,25,26,27,28,29	1,2,28,19	1,2,29
3	3, 6, 10, 14, 18, 22	3,9,10,14,18,22	3,11,22
4	4, 7, 11, 15, 19, 23	4,7,11,15,17,19,23	4,14,19
5	5, 8, 12, 16, 20, 24	5,16,20,21,24	5,16,24
6		6,25	6,1,25
8		8,26	8,20,26
12		12,27	12,27,28
13		13	13
7			7,18
9			9,21
10			10,23
15			15

TABLE V
SLOT SCHEDULE FOR NODES LEAVING SCENARIO

ID	Stage 1	Stage 2	Stage 3
6	6,17,28	6,17,28	1,5,6,14,17,24,28,29
8	8,19	8,19	2,7,8,16,19,25
12	12,22	12,22	3,10,12,20,22,26
13	13,23	9,13,15,18,23	4,9,11,13,15,18,21,23,27
1	1,2,24,29	1,2,7,10,20,21,24,29	
3	3,11,25	3,11,25	
4	4,14,26	4,14,26	
5	5,16,27	5,16,27	
7	7,18		
9	9		
10	10,20,21		
15	15		

enters the TDMA scheduling phase directly, while the other three nodes synchronize to node 1 instead.

Therefore, node 1 does not receive any candidate slots or sending slots information from the other nodes and takes all idle slots. Note that node 1 does not occupy slots 3, 4, 5 since its neighbors still broadcast heartbeat messages (i.e. they notify node 1 of their presence) during synchronization. After the slot usage fairness check described in Section V-E, node 1 releases some slots for its neighbors: nodes 3, 4, and 5 can get these extra slots after they are synchronized. One can see that node 3 gets slot {6, 10, 14, 18, 22} as extra slots starting from frame 3. Node 4 and 5 get {7, 11, 15, 19, 23 } and {8, 12, 16, 20, 24 }, respectively. All the remaining slots are occupied by node 1. The four nodes share slots 6 to 25 periodically: this is because they are sibling nodes, and get a shared slot set of {6-25} during scheduling.

Around frame 35, we turn on four additional modules with ID 6, 8, 12, and 13. We can see that once the new nodes are detected, their own slots {6, 8, 12, 13} are released immediately. As the number of neighbors increases, the older nodes notice they are using too many slots during their fairness check and begin to release slots. The change of slot owner is reflected with color changes for columns {9, 17, 21, 25, 26, 27}. Some collisions do occur during scheduling on these slots, which are indicated by the colored dots next to the columns. We can see that the collisions are resolved within five frames (mostly in two frames), leading again to full slot usage. Following a similar process, we turn on the remaining four nodes, which reach a new schedule after 3 frames. The resulting schedule for each node during the three stages of the experiment is shown in Table IV. Please note some slots have a short white gap: this is due to the initial synchronization of the newly added nodes.

D. Nodes leaving

We tested the system with the same setting as the nodes joining experiment, but starting with all modules turned on, after which we turn off two groups of 4 nodes in sequence. First, nodes with ID 7, 9, 10, 15, followed by nodes 1, 3, 4, 5. The scheduling results are shown in Fig. 14, with the same axes and content format as Fig. 13.

We can see that the slots are always fully used for the three stages: the system adapts to the new schedule as soon as the nodes detect the departure of a set of nodes. The free slots are allocated by the remaining nodes and reach a conflict-free schedule. The scheduling result of each node in every stage is listed in Table V.

Please note that the white gaps are mainly caused by the nodes leaving the system: we have set a timeout of 3 frames without messages from a node before considering that it has left the neighborhood to avoid rescheduling due to network noise. A short gap can also be caused by synchronization induced by the departure of the node with the smallest ID (which is used as a reference). This happens during the transition from stage 2 to stage 3, when the node 1 is turned off. In this scenario, node 13 took three frames to synchronize with the new reference node 6, leading to the gap for slot 13.

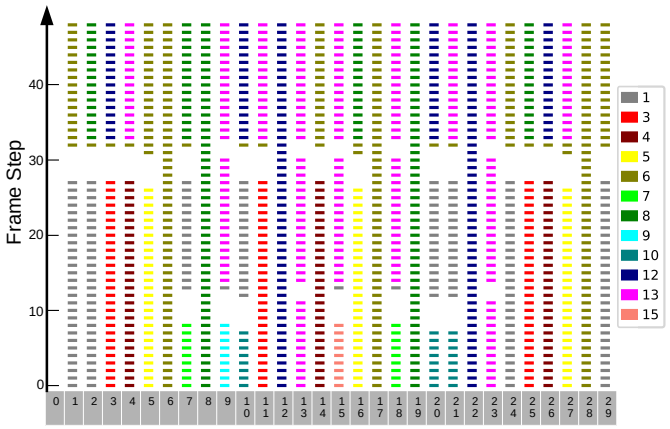

Fig. 14. Scheduling result of experiment that nodes leave the network. Starting from all modules powered on, we first turn off nodes with the ID of 7, 9, 10, 15 near the frame 10, and then turn off nodes 1, 3, 4, 5 near frame 30. This plot is based on the logs from node 13. We can see when nodes left near frame 10 and 30, the released slots are allocated by the rest nodes and reach a conflict-free schedule.

E. Multi-hop TDMA scheduling

We tested the system in the indoor parking lot at Polytechnique Montreal as shown in Fig. 15. The parking lot has very thick walls and makes it easy to set up a multi-hop network. We first put all modules together in one parking spot as depicted in Fig. 16-a. They form a fully connected network with a TDMA schedule shown at the bottom of Fig. 16-a and

in the column "Stage 1" of Table VI. We can see they reach a conflict-free schedule.

Fig. 15. Multi-hop experiment test field at parking place.

In a second stage of the experiment, we move the modules to other parking spots, forming a multi-hop network topology as shown in Fig. 15. The connectivity graph of the nodes is shown in Fig. 16-b with lines between nodes indicating a data and ranging connection.

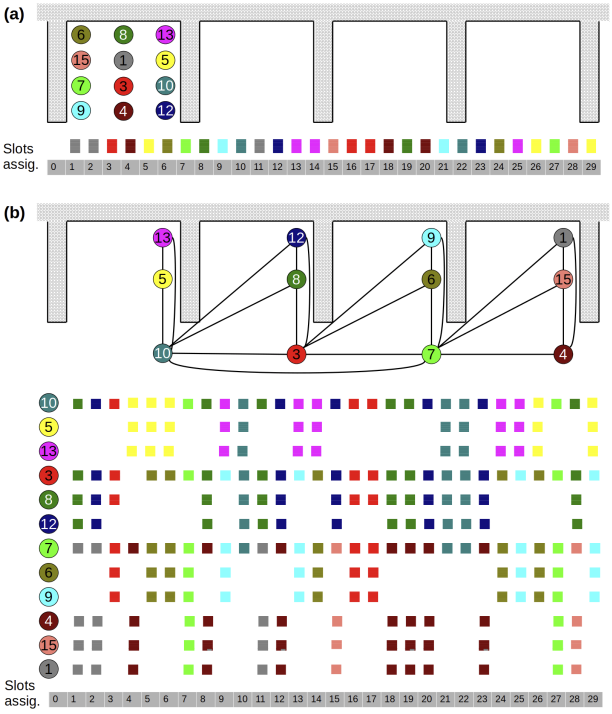

Fig. 16. Multi-hop network TDMA experiment. Twelve modules are moved from (a) to (b). The slots scheduling is shown with the color block under the graph. In sub-figure (a), all nodes have the same scheduling as they are fully connected. When moving nodes to form the topology in (b), they build a multi-hop network. The bottom part of the figures shows the slot assignment (1 to 29) of a cycle. The boxes colors follow the tags colors (circles). For instance, in slot 1 of (b) node 8 (green) and 1 (gray) use the channel simultaneously. This is possible as they are three hops apart in the topology map.

The scheduling result is displayed in the color block of Fig. 16-b, as well as listed in the column "Stage 2" in Table VI. The slot schedule shown in Fig. 16 is plotted using the log of each node, which includes its own time slot and the slot schedule received from its neighbors.

TABLE VI
SLOT DISTRIBUTION FOR MULTI HOP NETWORK

ID	Stage 1	Stage 2
10	10, 22	10,21,22
5	5,26,29	4,5,6,26,29
13	13,14,25	9,13,14,24,25
3	3,16,17	3,16,17
8	8,19	1,8,11,18,19,28
12	12,23	2,12,15,20,23
7	7,27	7,27
6	6,24	6,7,14,24,26
9	9,21	9,13,25,29
4	4,18,20	4,8,12,18,19,20,23
15	15,28	15,28
1	1,2,11	1,2,11

Taking the top line as an example, node 10 receives the slot assignment of all its neighbors, namely neighbors 5, 13, 3, 8, 12. Compared with the following row, node 5 only has neighbors 10 and 13. The advantage of our algorithm can be seen from nodes that are three hops away: as an example, nodes 5 and 6 share slot 26 (there are two colors on column 26 of Fig. 16-b) without collisions (the nodes are 3 hops apart, see the graph in Fig. 16-b).

For this schedule, 21 slots out of 29 are multiplexed (shared among multiple nodes), leading to full channel utilization.

F. Map construction

So far, we have proved that our system can effectively perform real-time TDMA scheduling with full channel usage. In this experiment, we explore the relative localization capability of the system.

In the first stage, we put all nodes in a fully-connected grid formation, as shown in Fig. 17 and in Fig. 18-a. Using all available ranging measurements, the nodes can easily recover their local map. We show the map constructed by 4 randomly selected nodes (all the remaining maps are similar) in Fig. 19. We can see that they all have the same relative structure, but with different coordinates as each node uses its own coordinate frame.

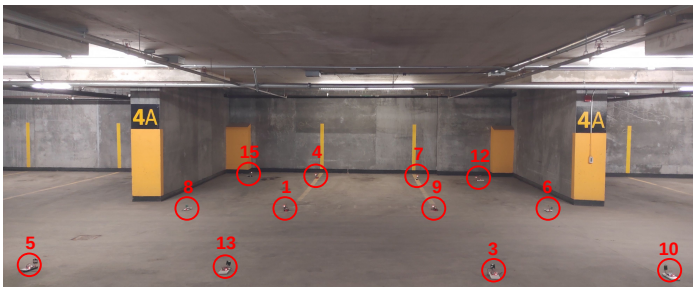

Fig. 17. Node distribution at the start of map construction experiments. Nodes are placed in a grid formation occupying an area around $8\,\mathrm{m} \times 12\,\mathrm{m}$.

When the nodes are moved to the configuration in Fig. 18-b, the network topology changes and the nodes reschedule the TDMA assignment with the same logic as in the previous experiments. For this new configuration, we randomly select 3 nodes on the left side and 3 nodes on the right side to show their local map in Fig. 20. We can see they are able to construct

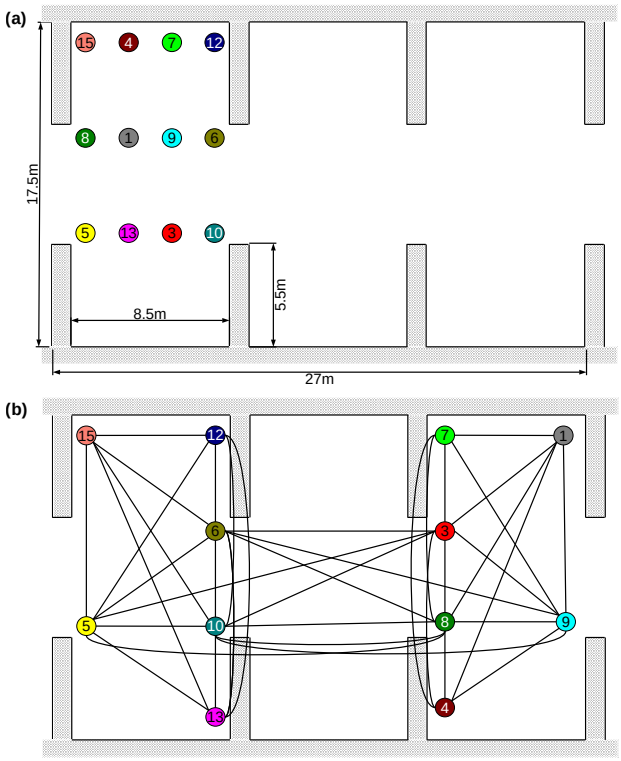

Fig. 18. The experiment involves two stages, corresponding to sub-figure (a) and (b). All nodes start in a grid formation (a), the network is then fully connected. They are then moved to configuration (b), showing a maximum of three hops.

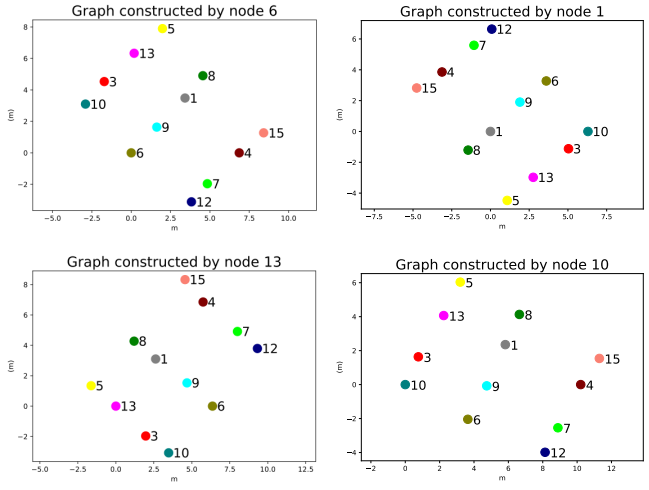

Fig. 19. Local map construction of nodes 6, 1, 13, and 10. They build their own coordinate system and get the relative localization for other nodes. As they are fully connected, the relative localization is the same, which is also the same for the rest nodes.

the correct relative map. Using the local map from node 5 and 1, node 8 can recover the global map as shown in Fig. 21.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel distributed TDMA algorithm that allocates all the idle time slots quickly to maximize the communication and ranging channel usage on

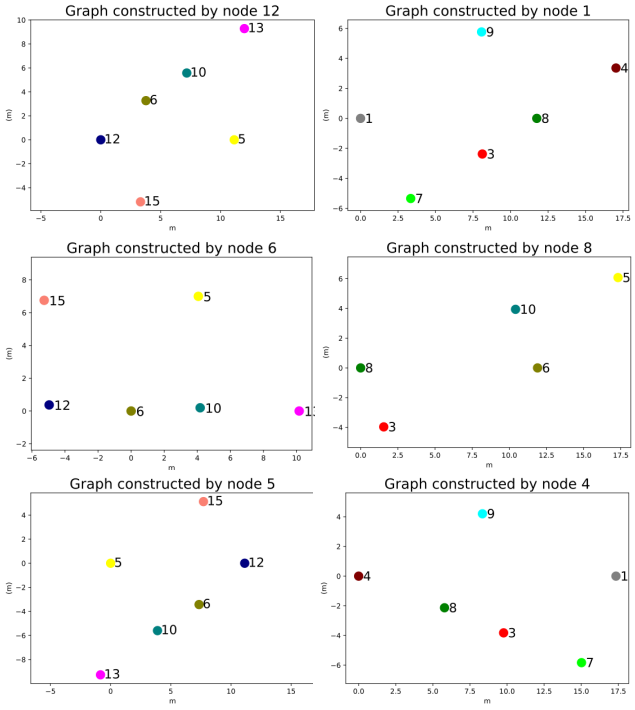

Fig. 20. Local map construction of nodes 12, 1, 6, 8, 5, and 4 at stage 2. Every node build a relative location for its neighbors. We can see the map shows part of the map in Fig. 18-b.

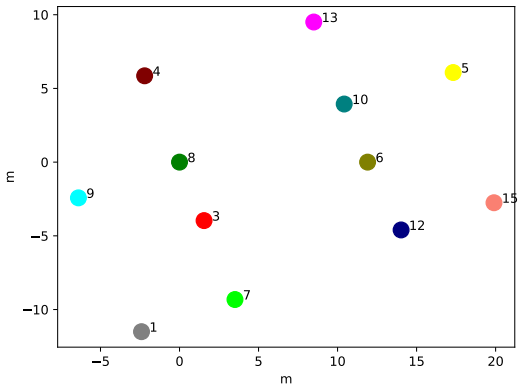

Fig. 21. Global map recovery for node 8 when receiving local maps from node 5 and 1.

mobile networks. For us, this high channel usage grants the required high rate of measurement from UWB devices to reach good localization accuracy. Our fast scheduling strategy also makes the system suitable for a network with changing topology. We presented a fully decentralized system that is able to build a topology map in a UWB-only network. We split the localization problem in the network into local relative localization and global map merging. Although our system is designed for UWB localization, our TDMA algorithm can also be applied to any networks requiring high channel usage, with static or mobile nodes. We acknowledge that our system is limited by the fixed size of the slots required in its frame structure initialized at the start. It limits the TDMA scheduling process to the update rate of a frame cycle. In future works, we will

implement an adaptive frame size to improve the scheduling update rate. Furthermore, our neighborhood topological map may fail if a node is moving fast with respect to our TDMA cycle. To address this limitation and improve the localization and tracking performance, we will fuse measurements from different sensors, such as Inertial Measurement Units (IMUs) or cameras.

REFERENCES

[1] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1327–1346, 2017.

[2] G. M. Mendoza-Silva, J. Torres-Sospedra, and J. Huerta, "A Meta-Review of Indoor Positioning Systems," *Sensors*, vol. 19, no. 20, p. 4507, Jan. 2019, number: 20 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/19/20/4507

[3] T. ter Horst, "Ultra-Wideband Communication and Relative Localisation for Swarming Robots: Ultra-Wideband Communication and Relative Localisation for Swarming Robots," 2019.

[4] "Pozyx - centimeter positioning for Arduino." [Online]. Available: https://www.pozyx.io

[5] D. Zito and D. Morche, "UWB Radios — The maturity age?" in *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, Jun. 2016, pp. 1–4, iSSN: null.

[6] Q. Shi, X. Cui, S. Zhao, S. Xu, and M. Lu, "Blas: Broadcast relative localization and clock synchronization for dynamic dense multi-agent systems," *IEEE Transactions on Aerospace and Electronic Systems*, 2020.

[7] V. DW1000 User Manual, "2.11.(decawave, 2017)," 2019.

[8] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley & Sons, 2010.

[9] N. Abramson, "THE ALOHA SYSTEM: another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*, ser. AFIPS '70 (Fall). Houston, Texas: Association for Computing Machinery, Nov. 1970, pp. 281–285.

[10] Chenxi Zhu and M. Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," in *Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No.98CH36169)*, vol. 1. San Francisco, CA, USA: IEEE, 1998, pp. 322–331.

[11] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 340–341.

[12] Y. Liu, H. Wang, M. Peng, J. Guan, and Y. Wang, "An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning," *IEEE Internet of Things Journal*, 2020.

[13] Y. Liu, T. Feng, M. Peng, J. Guan, and Y. Wang, "Dream: Online control mechanisms for data aggregation error minimization in privacy-preserving crowdsensing," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[14] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 80–86, 2017.

[15] W. Quan, N. Cheng, M. Qin, H. Zhang, H. A. Chan, and X. Shen, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 653–656, 2018.

[16] A. Prorok and A. Martinoli, "Accurate indoor localization with ultra-wideband using spatial models and collaboration," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 547–568, 2014, 05.

[17] J. Li, Y. Bi, K. Li, K. Wang, F. Lin, and B. M. Chen, "Accurate 3D Localization for MAV Swarms by UWB and IMU Fusion," *arXiv:1807.10913 [cs]*, Jul. 2018, arXiv: 1807.10913.

[18] K. Guo, X. Li, and L. Xie, "Ultra-wideband and Odometry-Based Cooperative Relative Localization With Application to Multi-UAV Formation Control," *IEEE Transactions on Cybernetics*, pp. 1–14, 2019.

[19] F. M. Martel, J. Sidorenko, C. Bodensteiner, M. Arens, and U. Hugentobler, "Unique 4-DOF Relative Pose Estimation with Six Distances for UWB/V-SLAM-Based Devices," *Sensors*, vol. 19, no. 20, p. 4366, Oct. 2019.

[20] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm," p. 7, 2020.

[21] M. Ridolfi, S. Van de Velde, H. Steendam, and E. De Poorter, "Analysis of the Scalability of UWB Indoor Localization Solutions for High User Densities," *Sensors (Basel, Switzerland)*, vol. 18, no. 6, Jun. 2018.

[22] N. Macoir, M. Ridolfi, J. Rossey, I. Moerman, and E. De Poorter, "Mac protocol for supporting multiple roaming users in mult-cell uwb localization networks," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 588–599.

[23] J. Zhu and S. Kia, "A SPIN-based dynamic TDMA communication for an UWB-based infrastructure-free cooperative navigation," *IEEE Sensors Letters*, pp. 1–1, 2020.

[24] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: self-organizing desynchronization and tdma on wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 11–20.

[25] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, Sep. 2011, conference Name: IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006).

[26] "ALOHA packet system with and without slots and capture | ACM SIGCOMM Computer Communication Review." [Online]. Available: https://dl.acm.org/doi/abs/10.1145/1024916.1024920

[27] J. Yeo, H. Lee, and S. Kim, "An efficient broadcast scheduling algorithm for tdma ad-hoc networks," *Computers & operations research*, vol. 29, no. 13, pp. 1793–1806, 2002.

[28] G. Wang and N. Ansari, "Optimal broadcast scheduling in packet radio networks using mean field annealing," *IEEE Journal on selected areas in Communications*, vol. 15, no. 2, pp. 250–260, 1997.

[29] S. Ramanathan, "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," in *Proceedings of INFOCOM '97*, vol. 2, Apr. 1997, pp. 900–907 vol.2, iSSN: 0743-166X.

[30] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, Oct. 2009, conference Name: IEEE Transactions on Mobile Computing.

[31] B. Dezfouli, "DICSA: Distributed and concurrent link scheduling algorithm for data gathering in wireless sensor networks," *Ad Hoc Networks*, p. 18, 2015.

[32] Y. Xu, K.-W. Chin, and S. Soh, "A Novel Distributed Pseudo TDMA Channel Access Protocol for Multi-Transmit-Receive Wireless Mesh Networks," *arXiv:1607.02045 [cs]*, Jul. 2016, arXiv: 1607.02045.

[33] C. D. Young, "USAP: a unifying dynamic distributed multichannel TDMA slot assignment protocol," in *MILCOM '96, Conference Proceedings, IEEE Military Communications Conference, 1996*, vol. 1, Oct. 1996, pp. 235–239 vol.1.

[34] H. A. Omar, W. Zhuang, and L. Li, "VeMAC: A TDMA-Based MAC Protocol for Reliable Broadcast in VANETs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1724–1736, Sep. 2013, conference Name: IEEE Transactions on Mobile Computing.

[35] C. Young, "USAP multiple access: dynamic resource allocation for mobile multihop multichannel wireless networking," in *MILCOM 1999. IEEE Military Communications. Conference Proceedings (Cat. No.99CH36341)*, vol. 1, Oct. 1999, pp. 271–275 vol.1.

[36] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio, "Dynamic TDMA slot assignment in ad hoc networks," in *17th International Conference on Advanced Information Networking and Applications, 2003. AINA 2003.*, Mar. 2003, pp. 330–335.

[37] S. Cao and V. C. S. Lee, "A Novel Adaptive TDMA-Based MAC Protocol for VANETs," *IEEE Communications Letters*, vol. 22, no. 3, pp. 614–617, Mar. 2018, conference Name: IEEE Communications Letters.

[38] Z. Yang and Y. Liu, "Quality of Trilateration: Confidence Based Iterative Localization," in *2008 The 28th International Conference on Distributed Computing Systems*, Jun. 2008, iSSN: 1063-6927.

[39] S. Hadzic and J. Rodriguez, "Utility based node selection scheme for cooperative localization," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, Sep. 2011, pp. 1–6.

[40] Y. Cao, M. Li, I. Švogor, S. Wei, and G. Beltrame, "Dynamic range-only localization for multi-robot systems," *IEEE access*, vol. 6, pp. 46 527–46 537, 2018.

[41] S. Han, S. Lee, S. Lee, J. Park, and S. Park, "Node distribution-based localization for large-scale wireless sensor networks," *Wireless Networks*, vol. 16, no. 5, pp. 1389–1406, Jul. 2010.

[42] J. Liu and Y. Zhang, "Error Control in Distributed Node Self-Localization," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, p. 162587, Dec. 2007.
[43] M. Hadded, P. Muhlethaler, A. Laouiti, R. Zagrouba, and L. A. Saidane, "Tdma-based mac protocols for vehicular ad hoc networks: a survey, qualitative analysis, and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2461–2492, 2015.
[44] L.-A. Phan, T. Kim, T. Kim, J. Lee, and J.-H. Ham, "Performance Analysis of Time Synchronization Protocols in Wireless Sensor Networks," *Sensors*, vol. 19, no. 13, p. 3020, Jul. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/13/3020
[45] L. Van Hoesel and P. Havinga, "A lightweight medium access protocol (lmac) for wireless sensor networks," in *1st Int. Workshop on Networked Sensing Systems (INSS 2004)*, 2004.
[46] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient collision-free medium access control for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. Los Angeles, California, USA: Association for Computing Machinery, Nov. 2003, pp. 181–192.
[47] A. K. Paul and T. Sato, "Localization in wireless sensor networks: A survey on algorithms, measurement techniques, applications and challenges," *Journal of Sensor and Actuator Networks*, vol. 6, no. 4, p. 24, 2017.
[48] D. Ltd, "Aps011 application note: Sources of error in dw1000 based two-way ranging (twr) schemes," 2014.
[49] A. Ledergerber and R. D'Andrea, "Ultra-wideband range measurement model with Gaussian processes," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. Mauna Lani Resort, HI, USA: IEEE, Aug. 2017, pp. 1929–1934.
[50] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review: Multiple-Robot Simultaneous Localization and Mapping," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, Jan. 2016. [Online]. Available: http://doi.wiley.com/10.1002/rob.21620
[51] A. Cornejo and R. Nagpal, "Distributed range-based relative localization of robot swarms," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 91–107.
[52] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 287–294.