FTM-Broadcast: Efficient Network-wide Ranging

Yann Busnel, Hervé Rivano

FTM-Broadcast: Efficient Network-wide Ranging

Yann Busnel
IMT Atlantique
SOTERN / IRISA
Rennes, France
ORCID: 0000-0001-6908-719X

Hervé Rivano
Univ Lyon, INSA Lyon, Inria
EA3720 CITI
69621 Villeurbanne, France
ORCID: 0000-0001-6112-7468

***Abstract*—Indoor geolocation has witnessed a significant advancement through the refinement of the 802.11 FTM (Fine Timing Measurement) protocol. Accurate indoor geolocation has numerous applications in areas such as asset tracking, indoor navigation, and location-based services. The standard 802.11 FTM protocol enables accurate indoor positioning by measuring the time-of-flight between a mobile device and multiple access points (APs). It can be generalized to device-to-device ranging. However, the conventional implementation of FTM suffers from increased complexity as the number of devices grows, limiting its scalability. FTM indeed involves a point-to-point exchange of messages between each pair of devices, leading to a quadratic increase in the number of messages as the number of neighboring devices increases. In this article, a breakthrough method is proposed to enhance the FTM protocol by leveraging broadcast communication, resulting in a substantial reduction in message complexity from quadratic to linear. By taking into account broadcast in the protocol, our approach eliminates the need for multiple individual exchanges and devises a mechanism where a single message from the mobile device is broadcasted to all neighbors simultaneously. Each message exchanged will then be useful for computing every pairwise time-of-flight, by piggybacking all timestamps, making the protocol more efficient and scalable. We conducted extensive simulated experiments to evaluate the performance of the enhanced FTM protocol. The results demonstrated the effectiveness of the proposed method, showcasing a substantial reduction in computational overhead compared to the conventional FTM implementation.**

***Index Terms*—Geolocation, FTM, Wi-Fi, Broadcast, Piggybacking**

I. Introduction

Obtaining accurate indoor location information presents a significant challenge, as GPS signals are often inaccessible indoors. For many years, alternative techniques have been actively pursued to address the challenge of indoor location tracking [1], in particular using wireless communication for ranging [2], [3]. One such technique is Fine Timing Measurement (FTM) specified in the 802.11-2016 standard [4], which has been further enhanced through the 802.11az amendment. FTM leverages Wi-Fi ranging to enable precise indoor location tracking. FTM relies on a Time of Flight (ToF)-based procedure. FTM enables ranging exchanges between an initiating station (ISTA) and a responding station (RSTA), facilitating indoor localization. However, FTM relies on unicast communication to compute the ranging distance.

FTM assumes that RSTAs, placed indoors, can be accurately configured to indicate their geographical position. Typically, these RSTAs are considered to be Wi-Fi Access Points (APs). However, it is important to note that other types of 802.11 devices can also act as RSTAs, such as digital signage, Wi-Fi cameras or printers. In most cases, these devices are located in fixed or rarely changing locations. However, configuring their geographical position accurately is a challenge. These objects usually do not have GPS sensors and, even if they did, GPS is not reliable indoors. Many users use manual and time-consuming techniques to establish the geolocation of reference points on the perimeter of the building using the GPS of their mobile devices or high resolution aerial images. Then, high-precision telemetry techniques such as laser or ultra-wideband are used to estimate the position and location of each sensor. This process has to be repeated each time an RSTA is moved or added. However, once the geolocation of one or more devices is known, the other sensors can use the MTF to self-locate [5], which greatly simplifies the deployment of an FTM-ready infrastructure.

However, the current version of FTM assumes that ISTAs perform an unicast exchange with each RSTA (often between 3 and 6) to estimate its position. If we consider use-cases with numerous mobile ISTAs, such as smartphones in large shopping malls or autonomous robots in mega warehouses, it seems unattractive to multiply unicast communications. In this paper, we propose to take advantage of the use of broadcast communication in wireless networks such as Wi-Fi, in order to save a significant amount of message exchanges, by reducing the complexity of the self-location protocol while guaranteeing a reduction in latency in most cases.

The remainder of this paper is organized as follows: Section II exposes the system model and the FTM background. Section III introduces then our network-scale 2-way ranging protocol. Section IV provides theoretical and experimental validation of the proposed method in various scenarios. Finally, Section V concludes the paper.

II. FTM Principle and Related Works

A. System Model

We consider a spatial area containing a set of n devices (called nodes in the following) that can communicate through a wireless communication channel, using the IEEE 802.11-2016 [4]. Each node is identified by their id d_i, $i \in [\![n]\!]$, and located in the three-dimensional space. Let $p_i = (x_i, y_i, z_i)$ be respectively the position (x_i, y_i) of node d_i in the 2D

plane, and z_i its altitude. Each node d_i is equipped with an omnidirectional antenna. It is able to communicate with other nodes located within a ball of radius ρ_{p_i}. Let $\mathcal{N}$ denotes the set of all reachable nodes in the ball centered on d_i.

To take a step forward in the direction of a realistic network model, a MAC layer is modeled, namely a classical CSMA/CA with RTS/CTS-like mechanism [6] and clear channel assessment (CCA). Before any transmission, a node senses the channel. If the channel is occupied by the transmission to or from a neighboring node, its transmission is delayed by a backoff, which is randomly picked in range $[0; \beta]$, with β as a user parameter. The MAC layer is ideal, meaning it is lossless, and no simultaneous transmission happens in the transmission range of a node.

B. Fine Timing Measurement

FTM introduces a novel exchange of unassociated user-targeted content between an initiating station (ISTA) and a responding station (RSTA). Unlike previous provisions in the 802.11 standard, which mainly served general information or radio parameter exchanges, FTM leverages unassociated exchanges for user-centric purposes, such as indoor localization.

ISTA negotiates ranging session parameters with RSTA, followed by a series of bursts. In each burst, RSTA sends a frame at time t^1, received by ISTA at time t^2. ISTA responds with an acknowledgement frame at time t^3, received by RSTA at time t^4. RSTA then communicates the values (t^1, t^4) to ISTA, enabling ISTA to compute the Time of Flight (ToF): $(t^4 - t^1) - (t^3 - t^2)$ and determine its distance d from RSTA using:

$$d = \frac{(t^4 - t^1) - (t^3 - t^2)}{2} \times c \tag{1}$$

where c refers to the speed of light.

Additionally, ISTA can request the Location Configuration Information (LCI) from RSTA, which consists of geographical coordinates. The exchange is repeated multiple times, and ISTA typically retains the smallest ToF as it represents the direct line of sight (LoS) path with the shortest time.

Figure 1 presents the chronogram of the protocol as published in the standard. In order to symmetrize the knowledge, so that each participant in the exchange is able to estimate the distance between them, it is necessary to piggyback t^2 and t^3 in the last acknowledgement message from ISTA to RSTA.

In the following, we will refer to the original version of FTM above as FTM-UC.

Although the 802.11 standards do not define a specific method for location computation, it is evident that the position is calculated solely based on the ToF and LCI values obtained from the frame exchanges with local RSTAs. Various techniques exist for location computation, including geometric methods, matrix-based approaches, and Kalman filters. In this paper, we therefore exclude the exchanges of the pairwise distance calculation rounds.

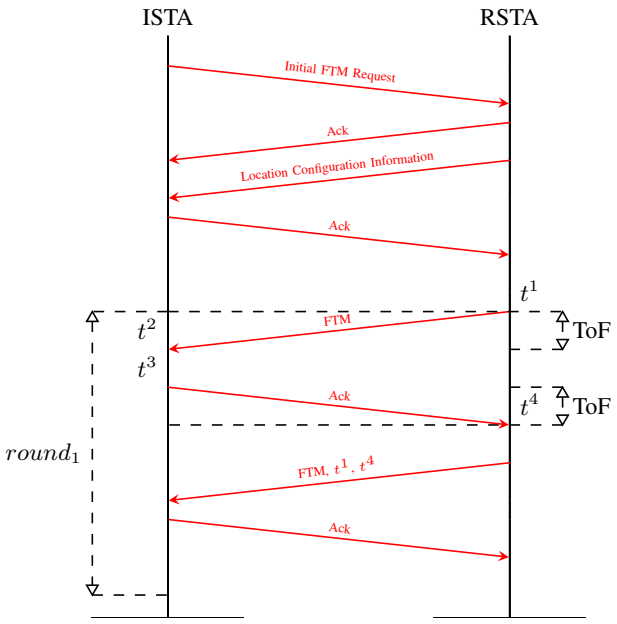


Fig. 1: FTM protocol

III. A NETWORK-SCALE 2-WAY RANGING PROTOCOL

A. Intuition of the overall design

The main idea of our proposal is to use broadcasting to reduce the number of messages exchanged during pairwise exchanges. Thus, instead of initiating a 2-way ranging with all its neighbors within communication range, each node of the network will broadcast its message without targeting any destination a priori. Thus, any node in the network that receives a previously broadcast message can consider it as the initial message of its own FTM with the concerned emitter.

In addition, each message sent can be used to carry all the information known locally up to that time. Any message will be piggybacked with the set of timestamps stored within the current round.

Figure 2 depicts the timeline of our protocol for a clique of 5 nodes. In the first phase of the round, each node responds to the initial broadcast of Node 0 by transmitting the timestamps of all broadcast messages received so far (t^2_*). Each timestamp t^3_* will be considered as the timestamp t^1 for those pairs that had not yet communicated during phase 1. Thus, at the end of phase 1, any pair of nodes will have made a round trip communication. As with the classic FTM protocol, phase 2 of the algorithm is used to transmit to all the other nodes in range the send and receive timestamps not yet broadcast in phase 1.

At the end of phase 2, all the nodes have the 4 timestamps, enabling them to calculate their distance to any other node in range using Equation 1.

It is important to note that the order in which messages are sent between phases 1 and 2 does not matter. The transmitted

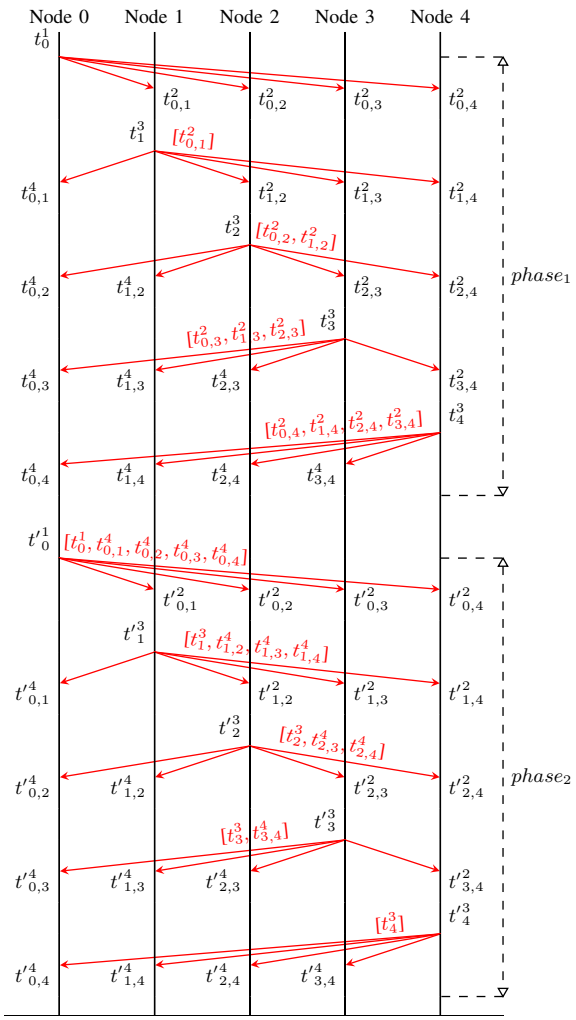


Fig. 2: FTM-BC protocol

data can be done in any order, with no consistency between the two phases. The only constraint is that all messages in phase 1 must be completed before phase 2 is initialized.

B. Precise design of the protocol

The algorithm of the protocol FTM-BC consists of two parts: the packet processing (Algorithm 1: packet processing), where received and emitted packet information is recorded and analyzed, and the sending of packets (Algorithm 2: Sending packet) based on specific conditions. Let us describe each part in detail.

1) Packet Processing: The algorithm takes as input three data structures: $reception$, $emission$, and $received\_at$. These dictionaries store information about received packets, emitted packets, and reception times, respectively.

The algorithm receives a packet p from source s at time t and extracts relevant information such as the packet ID (p_{id}), current phase, and lists of sent and received packets timestamps ($sent_s$ and $received_s$).

First, the algorithm updates all the dictionaries:

Algorithm 1: Protocol FTM-BC: packet processing

Data: $reception$ dict
$node$:$packet\ id$:$(sending\ time, reception\ time)$
Data: $emission$ dict $packet\ id$:$emission\ time$
Data: $received\_at$ dict $packet\ id$:$node$:$reception\ time$
$self \leftarrow$ packet p from s at time t
$p\_id, phase, sent_s, received_s \leftarrow p$
/* record reception of p */
$reception[s][p\_id] \leftarrow (\emptyset, t)$
/* update emission times from s */
for p_i, t_i *in* $sent_s$ **do**
  if $p_i \in reception[s]$ **then**
    $(\emptyset, t^r) \leftarrow reception[s][p_i]$
    $reception[s][p_i] \leftarrow (t_i, t^r)$
/* update reception times by s */
for p_i, t_i *in* $received_s[self]$ **do**
  $received\_at[p_i][s] \leftarrow t_i$
/* check if ToF from s is known */
if $\exists p_1 s.t.\ s \in received\_at[p_1]$ *and* $p_2 \in reception[s]\ s.t.\ reception[s][p_2] \neq (\emptyset, \_)$ **then**
  $t_1 \leftarrow emission[p_1]$
  $t_2 \leftarrow received\_at[p_1][s]$
  $t_3, t_4 \leftarrow reception[s][p_2]$
  $\rightarrow ToF(s, self) = \frac{(t^4 - t^1) - (t^3 - t^2)}{2} c$
/* check if reaction is needed */
if $self$ = *0* **then**
  if *phase+1 not prepared* **then**
    prepare packet($phase$+1)
    mark next phase as prepared
else
  if *never reacted to phase* **then**
    prepare answer packet($phase$)
    mark $phase$ is answered
delay next packet to $now + phase\_delay$

Algorithm 2: Protocol FTM-BC: Sending packet

Data: $buffer$ = list of packets to send
Data: run when sending is possible
for *each packet* p *in buffer* **do**
  if $p.time\_to\_send = now$ **then**
    load unsend data from $emission$ and $received\_at$ dictionaries into p
    send p

- The reception of the packet p is recorded in the $reception$ dictionary, with the sending time set to an empty value ($\emptyset$).
- The emission times of previous packets from s are retrieved in the $sent_s$ list and the $reception$ dictionary for source s is updated.
- The reception times by source s are updated similarly.

When all needed timestamps from s have been received, the Time of Flight (ToF) can be computed.

The last part of the algorithm checks whether the node has already managed the necessary reaction to the current phase. On the one hand, if we are on the $root$ node initiating the

FTM-BC (here, set to 0 without losing generality), it prepares the next packet to be sent at the start of the next phase. On the other hand, if we are on another node, it checks whether a response message has already been sent. If not, it prepares the answer packet for the current phase and marks the phase as answered.

Finally, the algorithm delays the sending of the next packet by the current time plus the $phase\_delay$.

2) Sending Packet: The algorithm takes as input a buffer, which is a list of packets to send. It is executed only when the underlying MAC layer signals that an emission is possible (*e.g.* when the backoff counter of the packet reaches 0 in a CSMA/CA mechanism).

For each packet p in the buffer, the algorithm checks if it is time to send the packet ($p.time\_to\_send = now$). If so, the algorithm loads the unsent data from the $emission$ and $received\_at$ dictionaries into the packet p and sends it.

C. Practical implementation issues

The latter sending mechanism yields an issue for an implementation in a real networking stack. It is usually not possible to modify the payload of a packet once it is in the lower layers buffers. It may need that the protocol is implemented in the driver of the network interface, similarly to actual FTM implementations [7]. Some Ultra-Wide Band chipset (*e.g.* Decawave DW1000 chipset [8]) performs such last moment modifications for precise timestamping. Nevertheless, we conjecture that it is also possible to circumvent most of these issues by adapting techniques that have been used in different settings facing similar challenges, such as network coding experiments [9]. An optimized and a practical implementation will be investigated in future works.

IV. Validation and evaluation

In this section, we will analyze the asymptotic behavior of our protocol, before presenting intensive simulation results.

A. Theoretical analysis

In this section, we will formally analyze the performance of the algorithm in terms of communication complexity, number of messages and time.

1) Communication complexity in terms of messages:

Theorem 1 (Communication complexity of FTM-UC– Messages): Let $\overline{\mathcal{N}}$ be the average degree of the nodes in the underlying communication graph. Each round of the FTM-UC protocol requires $2n\overline{\mathcal{N}}$ messages to estimate the complete set of pairwise distances of the participating nodes. The worst case corresponds to a complexity of $2n(n-1)$ messages, while the best case has a complexity of $4(n-1)$ messages.

The proof of Theorem 1 is basic graph theory arguments and is omitted here.

Theorem 2 (Communication complexity of FTM-BC– Messages): Each round of the FTM-BC protocol requires exactly $2n$ messages to estimate all the pairwise distances of the participating nodes.

Proof. As introduced in Section III, by design, each node sends exactly 2 messages per round (1 in each phase of FTM-BC). Thus, the number of messages exchanged during one round is exactly $2n$, that concludes the proof. □

Thus, whatever the topology of the graph concerned, with $n \geq 2$, the complexity of FTM-UC is greater than FTM-BC, and significantly higher most of the time. In the case of very high density graphs, it even goes from quadratic complexity to linear complexity.

2) Communication complexity in terms of information exchanged:

Theorem 3 (Communication complexity of FTM-UC– Information): Let $\overline{\mathcal{N}}$ be the average degree of the nodes in the underlying communication graph. For the protocol FTM-UC, the complexity of communication, in terms of the amount of information transmitted, is $2n\overline{\mathcal{N}}$ timestamps per round.

Proof. To determine the distance between any two participating nodes, 4 timestamps are required. As introduced in Section II-B, each pair of nodes exchanges 4 messages over each edges in the graph. Then, as the number of messages to compute all the pairwise ranging is $2n\overline{\mathcal{N}}$ (*cf.*, Theorem 1), the overall number of timestamps sent is the same. □

Theorem 4 (Communication complexity of FTM-BC– Information): Let $\overline{\mathcal{N}}$ be the average degree of the nodes in the underlying communication graph. For the protocol FTM-BC, the complexity of communication, in terms of the amount of information transmitted, is $n(\overline{\mathcal{N}}+1)$ timestamps per round.

Proof. In this proof, we separate the processing of timestamps for sent and received messages.

In the case of FTM-BC, some timestamps are identical for several neighbors (*cf.*, Section III). In fact, as the FTM-BC protocol is based on broadcast communication, only one emission will happen, and the timestamps t^1 and t^3 are the same for all the neighbors of a given node i. Therefore, these timestamps only need to be transmitted once by node (during phase 2) to reach the entire $\mathcal{N}_i$.

Furthermore, as explained in Section III-A, each timestamp t^3_* will be considered as the timestamp t^1_* for those pairs that had not yet communicated during phase 1. Thus, for each given node i, only one timestamp will be transmitted (t^1_i for the initiator and t^3_i for the other participating nodes). This optimization by design reduces by a factor of 2 the number of all the timestamps t^1 and t^3 for sent messages. This represents n timestamps to be sent for the overall round.

On the other hand, the reception time of the same broadcast message depends on the receiving node. Thus, all the reception timestamps (t^2_* and t^4_*) will have to be transmitted during phases 1 and 2 of the protocol. This represents exactly $n\overline{\mathcal{N}}$ timestamps (*cf.*, Proof of Theorem 3).

Finally, the overall amount of timestamps exchanged is:

$$n + n\overline{\mathcal{N}} = n(\overline{\mathcal{N}}+1),$$

that concludes the proof. □

B. Performance evaluation

1) Simulator software: Both FTM-BC and FTM-UC have been implemented in a dedicated simulator build on the SimPy3 discrete event simulation framework[1]. For the sake of reproducibility, the code of the simulator is available on gitlab[2]. All nodes are autonomous agents that can communicate only through an ideal communication layer, also implemented as an autonomous agent.

While the implementation of the nodes is close to what a real implementation would look like, the communication layer only mimics an ideal CSMA/CA-based MAC layer and emulates transmission and propagation times. It is however possible to gradually increase the realism of the network model up to a certain point. To simulate a very precise Wi-Fi stack, *e.g.,* to take account of interference, it will nevertheless be necessary to switch to a real network simulator such as NS3[3] for which a recent implementation of FTM has been proposed [10].

2) Simulation settings: For each simulation, the network is represented as a graph of the radio connectivity. In this paper, all nodes have the same communication range. The graph is therefore a symmetric unit disk graph. The node with the identifier 0 has the special role of initiating the FTM-BC protocol or a BFS protocol that allows each node to discover its neighborhood before running FTM-UC.

Indeed, our implementation of FTM-UC runs in two phases.

1) Node 0 starts a BFS, each node discovers its neighbors.
2) Then each node initiates the FTM-UC exchange with all its neighbors that have a larger identifier.

After a 4 packets FTM-UC exchange, both the initiating and responding nodes knows the ranging between them.

Several metrics are presented.

- The number of packet transmissions are given to validate the theoretical analysis.
- The node completion time is the time elapsing between sending or receiving the first FTM message and receiving or sending the last packet, when the ranging to all its neighbors is known. If the nodes were mobile, this time would distort the ranging information obtained. The greater the time, the greater the errors introduced into a multilateration calculation.
- The overall completion time is the time needed for computation of all ranging information in the network. If the nodes were mobile, the ranging process would have to be repeated periodically to update the location information. The greater the overall completion time, the lower the update frequency. Even if each location calculation is perfect, its accuracy degrades over time as a function of the speed of the nodes.

3) Simulation results: In the following, we report simulation results on 2 to 200 nodes lines, 2 to 200 cliques and 50 nodes random graphs for which the communication range vary from 1 (sparse graph) to high values (when the graph becomes a clique)

Figure 3 reports the number of packets sent by each method. As expected, it validates the theoretical analysis. FTM-BC sends 2 packets per node, as FTM-UC 2 packets per edge.

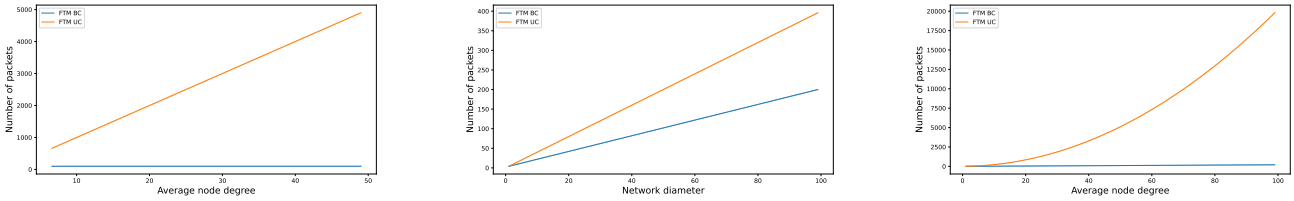
Figure 4 and Figure 5 reports the overall completion time. For both protocols, a high degree implies more contention for accessing the medium, hence larger delays due to the random backoff in the CSMA/CA layer. This slows down both FTM-BC and FTM-UC. For FTM-UC however, the degree has an additional and more significant impact since one transaction has to be performed with each neighbor. Sparse networks with large diameters and low degrees are the most efficient configurations for FTM-UC. It indeed is able to perform several ranging in parallel. Oppositely, the two waves produced by FTM-BC induces a delay that is proportional to the diameter time the $phase\_delay$ parameter. Overall, for very sparse networks, FTM-UC can complete the ranging faster than FTM-BC by taking advantage of spatial reuse in radio networks. Note that FTM-UC needs that a BFS is done beforehand so that each node knows its neighborhood, in particular their ID. But, since there are other mechanisms, potentially more effective, that can provide this information or that this information can be needed by other protocols, the time taken by BFS is not counted here for sake of fairness.

Figure 6 reports the average node completion time, only on the 50 nodes random graphs, since the results on lines and cliques are directly related to the overall completion time. More than 120 topologies have been simulated because the randomness of CSMA/CA have a significant impact on the result, in particular for FTM-UC, as one can see in the min-max discrepancy. FTM-UC is faster in most of the sparse network cases, which is expected since each FTM-UC transaction is run in a row and each node loops over its neighbors (with higher IDs) while each node waits that the first FTM-BC wave is completed in its neighborhood before answering to the second one. Besides, the waiting time implemented ($phase\_delay$ in Algorithm 1) might be over dimensioned and need to be carefully optimized. However, the difference between the mean node completion time with FTM-BC and FTM-UC is much lower in sparse networks than in dense ones.

V. Conclusion and Future works

In this article, we present an improvement to the geolocation protocol included in the 802.11.2016 standard, namely Fine Timing Measurement (FTM). The main idea is to optimize the two-way handshake of the classic protocol by taking into account the omnidirectional communication of the devices involved in the use of this wireless communication standard. We leveraged the protocol with broadcast communication, enabling the number of messages exchanged to be drastically reduced, and dividing the amount of information required to be transmitted by 2.

[1]SimPy 3: https://simpy.readthedocs.io
[2]Gitlab of our code: https://gitlab.inria.fr/hrivano/ftm_broadcast
[3]NS3: https://www.nsnam.org/



(a) 50 nodes random graphs (b) 2-100 nodes lines (c) 2-100 nodes cliques

Fig. 3: Number of packets

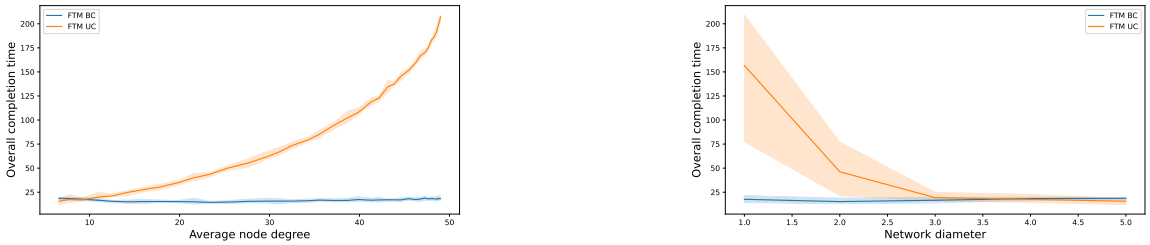


Fig. 4: Overall completion time - 50 nodes random graphs, min-max values

Fig. 5: Overall completion time - 2-100 nodes cliques (left) and lines (right), min-max values

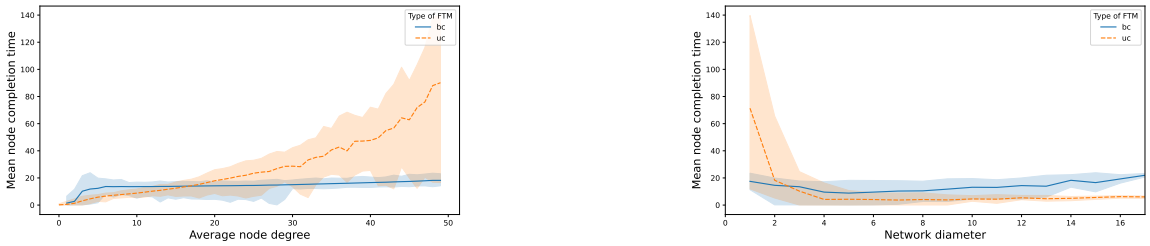


Fig. 6: Mean node completion time - 50 nodes random graphs, min-max values

REFERENCES

[1] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Comm. Surveys and Tutorials*, 2019.

[2] M. Von Tschirschnitz and M. Wagner, "Synchronization-free and low power tdoa for radio based indoor positioning," in *IPIN*, 2018.

[3] P. Corbalán and G. P. Picco, "Ultra-wideband concurrent ranging," *ACM Transactions on Sensor Networks (TOSN)*, 2020.

[4] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 2016.

[5] J. Henry, N. Montavont, Y. Busnel, R. Ludinard, and I. Hrasko, "A Geometric Approach to Noisy EDM Resolution in FTM Measurements," *MDPI Computers*, 2021.

[6] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "Macaw: A media access protocol for wireless lan's," in *ACM SIGCOMM*, 1994.

[7] V. Barral Vales, O. C. Fernández, T. Domínguez-Bolaño, C. J. Escudero, and J. A. García-Naya, "Fine Time Measurement for the Internet of Things: A Practical Approach Using ESP32," *IEEE Internet of Things Journal*, 2022.

[8] J. Cano, G. Pagès, E. Chaumette, and J. LeNy, "Clock and Power-Induced Bias Correction for UWB Time-of-Flight Measurements," *IEEE Robotics and Automation Letters*, 2022.

[9] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "Xors in the air: Practical wireless network coding," *IEEE/ACM Transactions on Networking*, 2008.

[10] A. Zubow, C. Laskos, and F. Dressler, "FTM-ns3: WiFi Fine Time Measurements for NS3," in *WONS*, 2022.