Towards the Simulation of WiFi Fine Time Measurements in NS3 Network Simulator

Anatolij Zubow, Christos Laskos, Falko Dressler

School of Electrical Engineering and Computer Science, TU Berlin, Germany

Abstract

WiFi has become the most widely used indoor positioning technology. The Fine Time Measurement (FTM) protocol introduced in the IEEE 802.11-2016 standard uses radio frequency based two-way time-of-flight (ToF) estimation, which promises precise indoor ranging and positioning. However, even with an ToF approach an exact indoor positioning is challenging due to the peculiarities of the propagation of the wireless signal such as signal attenuation, multipath propagation and signal fading. Moreover, the used WiFi hardware and its configuration like channel bandwidth also plays a major role. We present `FTM-ns3`, a software module which implements the 802.11 FTM protocol so that it can be used within the widely used ns3 network simulator. Moreover, we conducted experiments using commodity WiFi-FTM hardware, Intel 8260 and ESP32, and derived empirical error models which can be used in simulations to study the performance of novel FTM-based localization schemes under real channel propagation conditions while taking into account the specifics of the used WiFi hardware and configuration of FTM. Finally, we present results from simulations of a simple localization scheme based on FTM and multilateration which show the great influence of ranging inaccuracy introduced due to multipath propagation in typical indoor environments with line-of-sight (LoS) but strong multipath. Our module is provided to the community as open source and can be easily customized and extended.

Key words: WiFi Localization, IEEE 802.11, Fine Time Measurements, FTM, Network Simulation, NS3

1. Introduction

The knowledge about the location of a mobile device like a smartphone represents very valuable context information which is used by a variety of applications ranging from location-based services like user tracking [1] to location-aware communications [2]. An indoor localization system (ILS) based on existing and already deployed IEEE 802.11 (WiFi) infrastructure would be very promising as no special hardware would be required making the ILS very cost-efficient. Moreover, indoor localization might become ubiquitous to any device equipped with a WiFi chip (e.g., smartphone, tablet) like the global positioning system (GPS), which is used outdoors. However, such an ILS needs to be accurate, deployable, and universal [3]; often also making use of machine learning to overcome technical weaknesses [4].

Localization using WiFi is either based on measuring of the received signal strength (RSS) of surrounding WiFi APs or on the measurement of the two-way time-of-flight (ToF) between the mobile station and several co-located APs [5]. The latter is based on the Fine Time Measurement (FTM) protocol which was standardized with IEEE 802.11-2016 [6]. Some major WiFi chipset vendors like Intel, Qualcomm, or Espressif systems have already released devices that support the FTM protocol. Moreover, Android 9 and later support FTM.

Although FTM promises higher precision compared to RSS-based approaches, first experimental results show disappointing performance especially in environments with strong multipath and obstructed line-of-sight (LOS) [7, 8, 9, 10]. This was confirmed by our own experiments in the lab using the Intel 8260 and ESP32 chips. Studying the performance of ILS solutions based on the FTM protocol still depends on labor-intensive field experiments. To make the FTM protocol more accessible and to foster innovation, it is necessary to have a standard-compliant implementation of FTM in a widely used simulation toolkit. Moreover, error models to simulate inaccuracies are needed to study the performance of FTM-based ILS under real channel propagation conditions while taking into account the specific configuration of FTM.

In this paper, we present a standard compliant implementation of the WiFi-FTM protocol in the ns-3 network simulator [11] a widely used simulator in industry and academia for the research of networking protocols and communications technology. It extends our previous work [12] by having a more detailed modeling of ranging errors taking into account the channel bandwidth, wireless channel conditions like multipath propagation and the signal strength of the radio frames used by FTM. The empirical error models are based on actual experiments in the lab and field tests with commodity FTM hardware. This enables the rapid de-

Email addresses: `zubow@tkn.tu-berlin.de` (Anatolij Zubow), `laskos@tkn.tu-berlin.de` (Christos Laskos), `dressler@ccs-labs.org` (Falko Dressler)

Article published in Elsevier Computer Communications 210 (2023) 35-44. (10.1016/j.comcom.2023.07.028)

velopment of novel WiFi/FTM-based localization schemes in a well-controlled simulation environment. The `FTM-ns3` software package together with examples is provided to the community as open source.[1]

Our main contributions can be summarized as follows:

- We present `FTM-ns3`, a software module enabling support of 802.11 FTM protocol in the ns3 network simulator;
- we study the most relevant factors influencing the precision of FTM-based ranging by means of experiments using commodity FTM hardware;
- we introduce empirical error models for FTM-based ranging derived from results of extensive lab experiments and study of related work; and
- we evaluate our `FTM-ns3` module in a proof-of-concept using the example of a simple localization scheme based on FTM and multilateration.

2. Related Work

Our work is inspired by the experimental studies showing the performance of WiFi FTM in real-world environments. Bullmann et al. [9] evaluated FTM in realistic indoor scenarios using Intel 802.11ac WiFi hardware as well as Android smartphones in the 2.4 GHz ISM band together with 20 MHz channels. They discovered poor FTM performance in non-line-of-sight (NLoS) scenarios where they claim that environmental factors of the building like Shadowing from heavy metal fire doors affect the distance estimation process. At some measurement points they observed that the distance obtained from FTM ranging is bimodally distributed. With similar hardware Guo et al. [13] performed FTM ranging measurements indoors and in an outdoor open area with LOS. The authors conclude that the distribution of the RTT ranging error can be modeled as a Gaussian random process with zero mean and some variance.

Hashem et al. [14] performed measurements in two typical indoor environments: a college campus building floor and a work office floor. As hardware they used Google WiFi APs together with Google Pixel XL Android smartphones. Finally, Jathe et al. [8] performed FTM ranging experiments indoors in a long hallway with clear LOS but a strong multipath.

Barral Vales et al. [15] analyzed the performance of the FTM implementation provided by the low-cost IoT devices based on ESP32-S2 chipset. Results from several measurement campaigns in different indoor and outdoor scenarios reveal a large ranging error especially indoors, i.e., up to 5 m for 75% of the measurements when using the 20 MHz channel. Outdoors the error was lower, i.e., up to 1.5 m and 2.5 m for 75% of the measurements for 40 MHz and 20 MHz channels respectively.

Study	Mean error
Guo et al. [13, Figure 5]	1.27 m
Hashem et al. [14, Figure 5]	1.15 m
Bullmann et al. [9, Table 1]	1.76 m
Retscher [19, Figure 18]	1.41 m
Barral Vales et al. [15, Figure 5]	2.3 m
Xu et al. [16, Figure 8c/Eq. 32]	1.78 m
Aggarwal et al. [18, Figure 5] NLoS	6 m

Table 1: Reported FTM ranging accuracy.

Xu et al. [16] performed indoor FTM measurements using Intel 8260 chips for the APs and Google Pixel 3 smartphones for the client stations. They observed that the ranging error not only increases with the distance but that there is tendency to underestimate the distance. This is confirmed by our own observation where we analyzed the impact of the signal strength on the accuracy of FTM.

The latest FTM measurements campaign was performed by Dong et al. [17]. The results confirm the FTM RTT measurements are sensitive to environmental changes and analyzed the impact of different hardware, motion status, and signal path loss conditions. Another recent study by Aggarwal et al. [18] analyzes the effect of different hardware on ranging accuracy. The results show that depending on the AP and STA device used, ranging results are close to ground truth using the ASUS AP. The Compulab Fitlet AP is also close to ground truth, except when using the Xiaomi phone as STA. In contrast the Linksys AP overestimates the real distance. Table 1 summarizes the key results on the reported FTM ranging accuracy from different experimental studies.

Schepers et al. [20] performed the first security analysis of FTM using commodity WiFi hardware. They identified several weaknesses of FTM on both the logical and physical layer, and showed several attacks which allow an adversary to cause distance reductions or enlargements without changing the physical location of the stations. Some of the mentioned physical layer attacks exploit the sensitivity of FTM towards multipath which confirms our own observations.

Sun et al. [21] propose an ILS based on FTM and RSS data, where they suggest a particle filter based fingerprinting approach to counter the NLoS and multipath problems faced by radio signal-based ToF. With that they are able to achieve a mean accuracy of 2 m, both in LoS and NLoS conditions. Similarly, Eberechukwu et al. [22] propose a deep neural network based fingerprinting system to counter the radio signal-based ToF problems. As feature input they use the mean FTM RTT and variance at each fingerprinted reference point for every AP. Using this approach, they achieve submeter localization accuracy indoors. Examining the performance of such system in `FTM-ns3` may be interesting for first system evaluation as well as comparison between simulated and real environment.

[1] `https://github.com/tkn-tub/wifi-ftm-ns3`

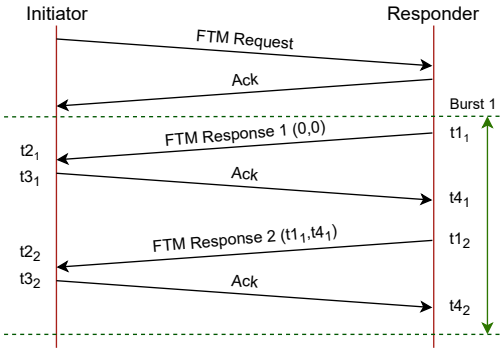


Figure 1: An FTM exchange with one burst.

A different approach to tackle multipath and weak LoS indoors was developed by Jiokeng et al. [23]. The authors combine FTM measurements together with the MUSIC algorithm and CSI measurements to correct the distance when the LoS is obstructed. In their indoor localization experiment they use least squares optimizer for determining the target location. Their FTM and MUSIC system shows a significant improvement in localization accuracy, achieving a median and 90-percentile localization error of 1.94 m and 3.77 m, while FTM only achieves a median and 90-percentile of 3.64 m and 5.79 m.

3. 802.11 FTM in a Nutshell

This section gives a short overview of the Fine Time Measurement (FTM) protocol as defined in the 802.11 standard [6]. The FTM protocol is used for performing high accuracy ranging between two stations (cf. Fig. 1). In order to mitigate the effect of clock synchronization error it uses a two-way time transfer (TWTT) protocol as two-way ranging (TWR) method. In an FTM exchange, one station is the initiator while the other is the responder. The FTM exchange starts with having the initiator sending an FTM request to the responder. In this request, the initiator transmits the parameters of the FTM session to be created. When the responder receives the request, it can either accept the request, change the parameters or deny the session. If the responder has accepted the initiator's request, an FTM session will be created between the two stations and ranging measurements can start. After the measurements have been performed, the session is closed. It is important to note that only the initiator of a session can determine the round-trip time (RTT).

High accuracy ranging in FTM is achieved by taking precise timestamps at the physical layer in picoseconds (ps) resolution, which gives them an accuracy of 0.03 cm. According to the standard [6], the timestamps should be taken as soon as the start of the preamble has been detected to make these timestamps as accurate as possible.

As shown in Fig. 1, the time measurement starts with the first frame the responder sends. As the session has just begun, the responder has no previous timestamps to transmit in the first FTM. Thus, both timestamps in the dialog number 1 are set to 0. A dialog consists of the FTM response sent by the responder and the ACK sent by the initiator. The timestamps taken by the responder are $t1$ and $t4$. The first is taken immediately before the responder starts transmitting its FTM response while the second represents the point in time when it receives the corresponding ACK frame. The initiator determines the timestamps $t2$ and $t3$, which represent the point in time the FTM response was received and the initiator started transmitting the corresponding ACK frame respectively. These four timestamps form an FTM dialog for which the RTT can be calculated. It is important to note that only $n-1$ RTT of n FTM exchanges can be calculated. The RTT is calculated as:

$$\mathrm{RTT} = (t4 - t1) - (t3 - t2) \tag{1}$$

The distance in cm between two stations is obtained as (note, that the RTT is given in ps):

$$d = \frac{\mathrm{RTT}}{2} \times 0.03 \frac{\mathrm{cm}}{\mathrm{ps}} \tag{2}$$

There will always be some fluctuation in the estimated RTT due to the limited bandwidth and environmental influences (cf. Section 4). Hence, in order to get higher accuracy, averaging is performed over the RTT values obtained within a session.

A WiFi station can have multiple active sessions with different stations. The stations do not need to be associated with each other in order to perform ranging. Instead, it is possible to perform an FTM exchange with a station in an unassociated state.

4. Factors Influencing the Precision of Ranging

There are many factors that have an impact on the accuracy of ToF-based ranging used in FTM, which are described in the following.

4.1. Channel Bandwidth

The detection of packet arrivals is a challenging task as a difference of only 1 ns could result in an error of 30 cm for the RF ranging system. Therefore, a fine resolution clock with 1 ns or higher is needed which is the case with FTM as it uses ps clock resolution. Another factor that limits the resolution of a ToF measurement is the sampling rate [24] and the channel bandwidth. This is known as range binning, which occurs when a matched filter is used to estimate the time of packet arrival with a sampling rate of up to $f_s = 2B$ where B is the channel bandwidth [25]. Sampling adds error to the estimate because the estimate space is divided up into range bins that are c/f_s wide where c is the speed of light. Sampling adds uniform range uncertainty in each bin of σ_s^2 [25]:

$$\sigma_s^2 = \frac{c^2}{12 f_s^2} \tag{3}$$

In the case of WiFi with a channel bandwidth of $B = 20\,\text{MHz}$ the variance due to sampling can be calculated to be $(4.68\,m)^2$. Continuous tracking, filtering, or averaging can be used to improve the resolution, but this is not bandwidth or power efficient. To reduce this error, the signal can be oversampled. To further improve the raw resolution super-resolution spectral signal processing techniques are being used today, e.g., [26]. Finally, the channel bandwidth itself can be increased as newer WiFi standards like 802.11n/ac/ax support 40, 80, or even 160 MHz.

In order to understand the ToF range estimation accuracy of commodity 802.11 hardware we performed our own experiments in the lab. The following hardware was used:

- Intel 8260 Network Interface Card (NIC)
- Adafruit ESP32-S2 System-on-Chip (SoC)

Both the Intel NIC and the ESP32 SoC are compliant to 802.11b/a/g/n/ac and support the FTM protocol. Note, that the ESP32 can be operated in standalone, whereby the Intel NIC requires a host computer. The WiFi devices were configured to operate in 2.4 GHz ISM band and use the 802.11n configuration. The setup consisted of a pair of WiFi nodes of the same type, Intel NIC or ESP32 SoC, where we replaced the antennas with coax cable with 50 dB attenuator. Such a configuration represents the operation at high signal strength, i.e., RSSI at around -40 dBm, mimicking perfect channel conditions without any distortions like multipath propagation. For the FTM experiment using the Intel NIC, we used the software provided by Gruteser.[2] Moreover, for the Intel NIC we were able to test two different bandwidth configurations, i.e., 20 (HT20) and 40 MHz (HT40). For each configuration, 2000 single-burst FTM ranging tests were performed. Hence, the reported RTT values are raw values, i.e., no averaging of multiple measurements was made.

The results for the Intel 8260 NIC are shown in Fig. 2. Here we present ΔRTT which is computed from of each RTT sample by subtracting the mean value measured over all RTT values, i.e., $\Delta\text{RTT} = \text{RTT} - \overline{\text{RTT}}$. This is because we are not interested in the absolute values which dependent on the length of the coax cable. Instead, we want to analyze the variations in RTT due to different channel bandwidth. From the results we see that the distribution of the ΔRTT values follows a normal distribution. Its standard deviation σ is around two times larger for 20 MHz channel as compared to a 40 MHz channel, i.e., 2563 ps and 1075 ps, respectively. Using Eq. (2), this translates into a distance of 38 cm and 16 cm, respectively.

Fig. 3 shows the results when using a pair of ESP32 SoCs. Here we were only able to take measurements on a 20 MHz channel as FTM ranging was not possible using 802.11n HT40. Interestingly, the standard deviation was only 1053 ps and hence comparable when using Intel on a

[2] `http://www.winlab.rutgers.edu/~gruteser/projects/ftm/Setups.htm`

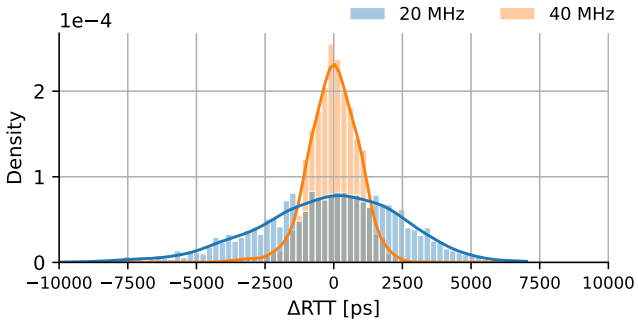


Figure 2: Ranging error in over-the-cable experiment (Intel 8260).

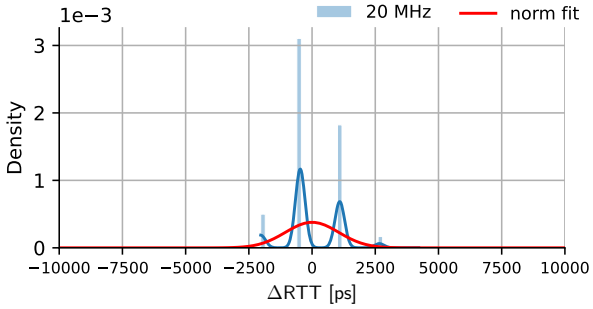


Figure 3: Ranging error in over-the-cable experiment (ESP32).

40 MHz channel. Contrary to the results for the Intel NIC we see a quantization of the ΔRTT at around 1.5 ns.

4.2. Signal Strength

Because of its discontinuous channel access due to usage of listen-before-talk a wireless technology like IEEE 802.11 uses self-contained frames, i.e., a preamble that precede the actual data allows the receiver to acquire the initial signal detection and synchronization in both time and frequency. The packet detection step is the first digital processing step at the receiver after the analog-to-digital conversion where the incoming discrete-time complex baseband samples are processed in order to detect the known preamble within the incoming stream. Since preambles usually contain repetitions of training symbols with good autocorrelation properties, conventional digital receivers apply correlation-based methods for packet detection [27]. Usually, the packet detection algorithm is implemented as a double sliding window as described in OFDM Wireless LANs [28]. The autocorrelation of L-STF short training symbols is used to return an estimated packet-start offset. There is often a second stage where this estimate is refined with symbol timing detection using the L-LTF.

However, the packet detection accuracy is impacted by the signal quality of the received signal which might result in too early or too late packet detection which is another factor that limits the resolution of a ToF measurement like FTM. Fig. 4 shows results from our link-level

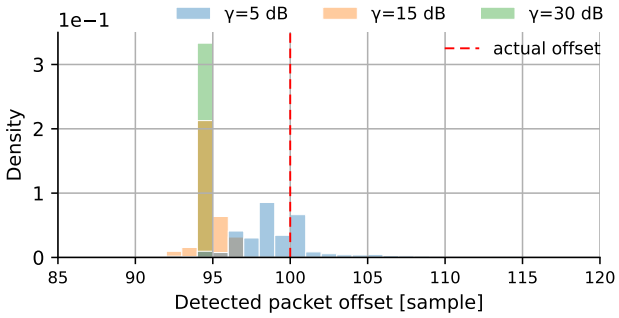


Figure 4: Detected packet offset in AWGN channel for different SNR.

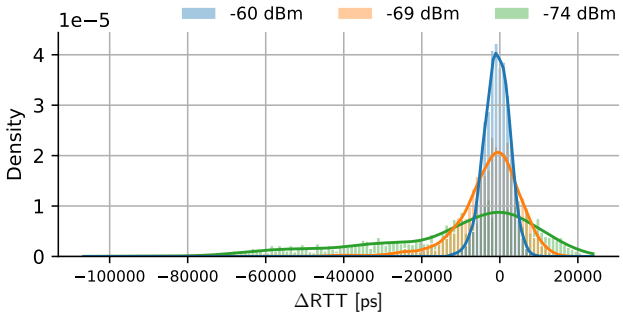


Figure 5: Impact of receive power P_{rx} on ranging error (Intel 8260).

simulations performed in Matlab using the WLAN toolbox. Here, we transmitted 802.11n packets delayed by 100 samples over an AWGN channel with different Signal-to-noise ratio (SNR) and analyzed the detected packet offset using Matlab's `wlanPacketDetect()` function. We used the recommended default value of 0.5 for threshold which favors false detections over missed detections considering a range of SNR values. From the results we see that for high SNR the packet detection algorithm reliably estimates for all packets the same packet offset of $\mu = 94$ ($\sigma = 0.5$). However, it is below the true offset of 100 but could be corrected with help of some calibration step. In contrast the situation is different for a signal with low SNR, i.e., $\gamma = 5\,\text{dB}$. Here the detected packet offset is no longer deterministic but random with $\mu = 100.7$ which is close to the true offset but some visible variation ($\sigma = 18.9$).

Based on the results from the simulation, we performed additional experiments with our WiFi hardware based on Intel 8260 NIC and ESP32 SoC. The setup is similar to the previous experiment except that we replaced the fixed attenuator by a variable one. For different attenuator configurations we performed FTM ranging and recorded the estimated RTT as well as the signal strength P_{rx} reported by the WiFi chip.

When using Intel 8260 NIC the distributions of ΔRTT for three different receive power levels P_{rx} are shown in Fig. 5. As expected, the ranging error, i.e., the standard deviation, is smaller for higher P_{rx}. Moreover, for high P_{rx} the ΔRTT values follow a normal distribution. This is not the case for lower P_{rx} values, e.g., -74 dBm, which show asymmetric distribution (skewness of -1.05). Moreover, in contrast to our simulation results there is a bias towards too low RTT value, i.e., the true distance is underestimated. Possible explanations might be the closed proprietary packet detection algorithm used by the Intel WiFi chipset. Based on the collected results for different P_{rx} values ranging from -34 to -82 dBm we performed curve fitting and tested different distributions and found out that the Johnson SU distribution [29] showed the best fit for ΔRTT. It has four-parameters $(\gamma, \delta, \xi, \lambda)$ which are specific for different P_{rx} values and are shown in Table 2.

We repeated the experiment using the ESP32 hardware.

P_{rx} [dBm]	γ	δ	ξ	λ
-34	3.185	5.478	6607.307	10570.049
-54	5.245	6.785	11337.622	13422.492
-60	2.956	5.965	7871.393	16622.557
-66	1.568	2.651	6467.335	10780.686
-72	1.684	1.538	9816.639	9491.488
-74	4.690	1.959	28439.427	6570.589
-76	7.140	2.171	38988.747	3882.820
-78	8.522	2.687	64794.774	7235.396
-80	10.561	3.346	99724.650	11258.496
-82	21.623	7.213	282943.146	33155.660

Table 2: Johnson SU distribution with parameter fit for different receive signal strength values P_{rx} (Intel 8260 NIC).

As can be seen from Fig. 6 the ΔRTT values of the FTM measurements having the same P_{rx} follow a normal distribution. However, the mean μ and standard deviation σ depend on P_{rx} (Table 3). With lower P_{rx} the distribution becomes wider, i.e., larger σ, and the mean value μ is shifted to the right resulting in an overestimation of the true RTT. An underestimation of the RTT occurs only rarely which is different to the results obtained with Intel 8260 NIC.

$P_{\mathbf{rx}}$ [dBm]	μ	σ
-42	0.0	1053.43
-69	4733.01	1634.68
-78	9809.9	3836.75

Table 3: Normal distribution parameters for different signal strength values P_{rx} (ESP32).

4.3. Multipath Propagation

In a typical scenario, indoor or outdoors, the wireless signal emitted by the transmitter is exposed to multipath propagation. Here the RF signals bounce off objects in the environment, causing the signal to arrive at the receiver through many paths. The consequence is that the received signal is the sum of the signals arriving along different paths. Except for the direct (Line of Sight, LOS) path all paths are the result of reflection and diffraction. Compared to LOS signal, the non-LOS signals are delayed, and the phase and amplitude of the signal is different. The result

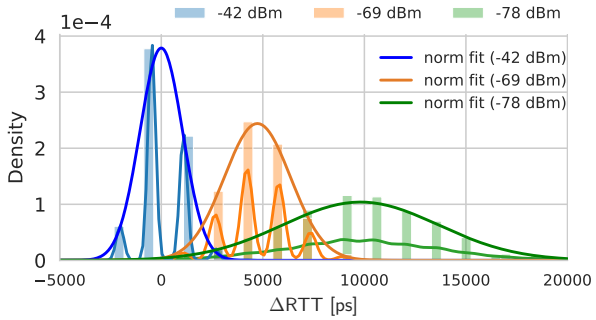


Figure 6: Impact of receive power P_{rx} on ranging error (ESP32).

Figure 7: Emulated 2-tap multipath channel over cable.

of multipath is a randomly changing received signal power which is termed as fading.

Multipath propagation distorts the ToF-based ranging used by FTM. First, in absence of a LOS component, i.e., in a pure non-LOS environment, the measured distance typically overestimates the actual distance as the length of the reflections is being measured. Second, it is not uncommon that the indirect paths have higher power than the direct path [30]. Third, the multipath propagation complicates the proper detection of packet arrival time resulting in too late or too early detection.

Results from experimental studies analyzing the ToF range estimation accuracy using commodity 802.11 hardware (Intel 8260), e.g., [9, 10, 13, 14, 19], show that under real indoor conditions the FTM ranging error is around 1-2 m (RTT of 6.6-13.3 ns) which is 2.6-5.2× larger than the error we obtained with our wired setup (cf. Section 4.1) and, thus, cannot be explained with the limitations due to the channel bandwidth or weak signal. One potential reason for this could be the distortions to the wireless signal caused by transmitting it over a wireless channel that experiences multipath propagation.

To explore this further, we conducted our own experiments. First, we conducted measurements over cable with an emulated 2-tap multipath channel (Fig. 7). Here we analyzed the impact of the cable lengths, d_1 and d_2, as well as the impact from attenuation on the second path (reflection). Fig. 8 shows the results when using ESP32. We can clearly see that even in such a simplistic multipath channel with just a single reflection the ranging error becomes very high. This is especially the case when the reflection is strong, i.e., has small attenuation. Moreover, in most of the cases the true distance and hence the RTT is overestimated, however, for one configuration we saw even an underestimation, i.e., $\Delta\text{RTT} < 0$.

In addition, we took a closer look at the results obtained by Jathe et al. [8], who performed over-the-air ranging ex-

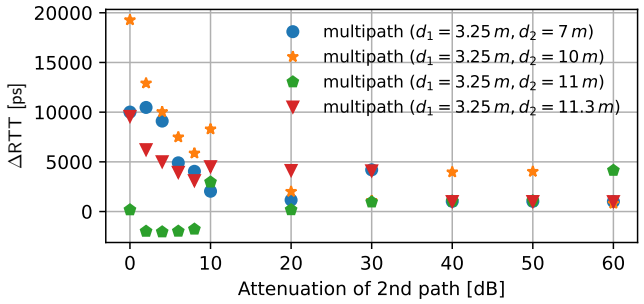


Figure 8: Results from emulated multipath channel over cable (ESP32).

periments indoors in a long hallway. The propagation was characterized by having a clear LOS with additional multipath components. As WiFi hardware they used the Intel 8260. We used their provided dataset to compute the ranging error of each data point. As the reported distance values are mean values calculated over 20 FTM rangings, the impact of channel bandwidth was already averaged out. We converted both the ground truth distance and the measured distance into RTT (Eq. (1)) from which the RTT error, ΔRTT, was computed. The results are shown in Fig. 9. As can be seen, the distribution of ΔRTT is very wide. Moreover, the values are not normally distributed. Instead, it is an exponentially modified normal distribution,[3] i.e., sum of independent normal and exponential random variables, with parameters $\mu = -5478$, $\sigma = 2821$, and $\lambda = 0.000183$. Looking at the resulting error model, it is clear that its peak is negative, at about -2000 ps. This means that there is a tendency to underestimate the true distance by 30 cm. Also, it is possible that sometimes the distance is overestimated significantly. This is likely caused by the multipath propagation in the indoor environment, resulting in detecting the signal too late or in the former case, too early.

To confirm this and to investigate if there are differences between WiFi chips, we performed our own indoor measurements in a small hall ($9\,m \times 18\,m$) with LOS propagation and additional reflections from stone floor and glass facade. Fig. 10 shows the results for both the Intel and the ESP32 hardware. We can clearly see the impact from the different hardware. While the results for Intel are similar to the ones obtained in the long hallway (Fig. 9), the results for ESP32 are different. Here the multipath channel leads to significant overestimation of the true distance in most of the cases. Moreover, the distribution of ΔRTT values is wider and is a mixture of two normal distributions ($\mu = \{6030, 16492\}$ and $\sigma = \{2942, 8794\}$) with weights $w = \{0.572, 0.427\}$.

[3]The Kolmogorov-Smirnov test for goodness of fit gives a p-value of 0.86.

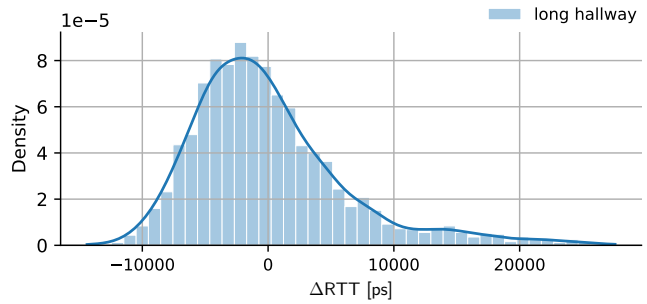


Figure 9: Ranging error in long hallway (Intel 8260, data from [8]).

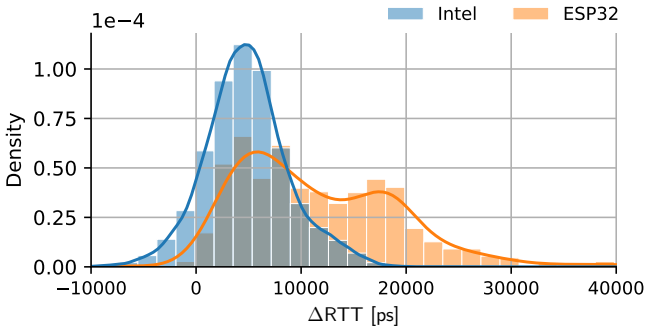


Figure 10: Ranging error in small hallway (Intel 8260 and ESP32).

5. FTM Extension for NS3

In the following, we present our `FTM-ns3` system. We implemented it in form of an ns-3 extension to support WiFi-FTM. We discuss the design, the supported error models to account for channel bandwidth, multipath propagation, and signal quality, and the actual implementation.

5.1. Design

The goal of our work is to extend the WiFi module of the ns-3 network simulator to support the FTM protocol in a standard compliant way. Therefore, we added the support for FTM to the `RegularWifiMac` class. The actual logic for handling FTM requests and responses and the corresponding FTM sessions is provided by the `FtmManager` class. Each Wifi node has exactly one `FtmManager` instance if support for FTM has been enabled. Every `FtmSession` has its own parameters and can have an error model defined in `FtmErrorModel` if specified. Different FTM sessions can use different error models. The provided error models and their limitations will be discussed in next section.

Fig. 11 shows the class diagram of the main FTM components. The extension of the `RegularWifiMac` class is kept as minimalistic as possible. The `RegularWifiMac` class receives all action frames and processes them accordingly, including FTM frames. If an action frame with category value public action is received, it is further processed to determine if it is an FTM request or response frame. In such a case it is handed over to the `FtmManager`

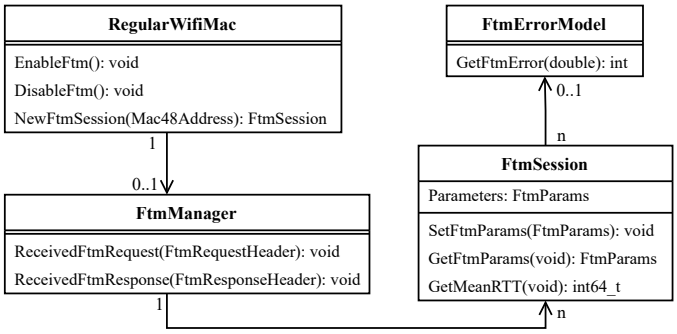


Figure 11: Class diagram showing the components of `FTM-ns3`.

to determine to which session it belongs. If the frame does not belong to an existing session and it is an FTM request, a new session is created. Finally, the received FTM frame is forwarded to the correct `FtmSession` by the manager.

5.2. Modeling Ranging Errors

Our `FTM-ns3` extension supports three error models which can be parameterized.

The first one is the wired error model which models the impacts of the WiFi hardware, e.g., ESP32 or Intel 8260, and its configuration, i.e., channel bandwidth, on the ranging accuracy. Currently for the Intel 8260 NIC two channel bandwidths are available which can be selected for simulations: 20 and 40 MHz. The model is based on our observations from experiments over coax cable (cf. Section 4.1). In case of the Intel 8260 NIC the RTT error is drawn according to normal distribution with $\mu = 0$, i.e., zero mean, and standard deviation of $\sigma = 2563$ ps and $\sigma = 1075$ ps for 20 and 40 MHz, respectively. For the ESP32 operating on a 20 MHz channel the RTT error is also normally distributed with $\mu = 0$ and $\sigma = 1053$ ps.

The second model is a wireless error model which extends the wired error model to additionally model the impact of multipath signal propagation which is of great importance especially when performing simulations of indoor scenarios. This model is based on the observations made in Section 4.3. First, the impact from multipath on the FTM accuracy introduces a bias (RTT offset) which depends on the locations of transmitter and receiver. That means that as long as the nodes' locations are fixed the introduced RTT offset stays the same and does not change over time. Therefore, the wireless error model needs to know the position of the WiFi nodes to which it is attached (cf. Fig. 12). This position is given to the `GetBias` function of the `FtmMap`, which returns the bias for a given position.

The `FtmMap` stores the pre-computed RTT bias for each possible node location. The precomputed RTT bias is generated as follows. The RTT bias of closely positioned WiFi nodes is typically observed similar or correlated. Therefore, the bias can be obtained via interpolation in the following way. A uniformly spaced grid is generated using a pre-defined de-correlation distance d, with e.g., $d = 25\,cm$

when using 2.4 GHz carrier frequency. For each grid point a random value is generated according to the distribution of the selected environment (e.g., long hallway, small hall) and WiFi hardware being used. In case of Intel 8260 and the long hallway it is an exponentially modified normal distribution with $\mu = -5478$, standard deviation of $\sigma = 2821$ and $\lambda = 0.000183$ is generated. When using ESP32 in the small hall wireless environment the RTT error is drawn from the mixture of two normal distributions (cf. Section 4.3). The values between the grid points are generated by interpolation using a cubic spline with a resolution of 1 cm which is sufficient from practical point of view. After adding to the RTT the bias obtained from the `FtmMap`, the error calculated by the wired error model is added as well.

The third model is a wireless signal strength error model which extends the wireless error model to additionally account for the impact of the signal receive power which has a significant impact on the FTM ranging accuracy. Moreover, it is dependent on the WiFi hardware being used. Our model is based on the results we obtained from experiments over cable with different receive power levels (Section 4.2). In case of Intel 8260 the RTT error is drawn according to Johnson SU distribution with parameters shown in Table 2. Note, the parameters $(\gamma, \delta, \xi, \lambda)$ are different for different receive power values. For the ESP32 the RTT error is normally distributed with (μ, σ) parameters selected from Table 3.

In summary, when using the most realistic error model, i.e., the wireless signal strength error model, the RTT in the simulation, $\widetilde{\text{RTT}}$, is computed as follows:

$$\widetilde{\text{RTT}} = \text{RTT} + h + w + p, \tag{4}$$

where RTT is the ground truth RTT as computed by Eq. (1), h, w and p are random variables representing the RTT errors due to multipath propagation, channel bandwidth and receive signal power, respectively. Note, while w and p change over time even in stationary setup it is not the case with h which changes over space. All three errors are additive.

Fig. 12 shows the class diagram. While `WiredFtmErrorModel` class directly inherits from the `FtmErrorModel` base class, the `WirelessFtmErrorModel` is a subclass of `WiredFtmErrorModel`, and the `WirelessSigStrFtmErrorModel` is a subclass of `WirelessFtmErrorModel`. All error models implement the `GetFtmError` method, which is used by the `FtmSession` to obtain the error for each RTT measurement. First, the RTT is calculated according to Eq. (1). Thereafter the RTT error returned from the `GetFtmError` method is added to it to account for ranging inaccuracy. It is important to note, that the `FtmSession` is unaware of the error model currently being used. As shown in Fig. 11, the `FtmSession` holds only a reference to the base class `FtmErrorModel`. By default, the base class is used by every session, which always returns zero RTT error.

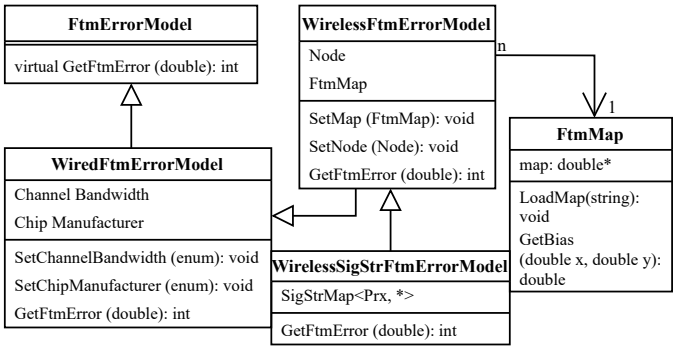


Figure 12: Class diagram of the ranging error models.

5.3. Implementation

In this part, we will give some implementation details of `FTM-ns3` like FTM framing, usage of time stamps, integration with 802.11 PHY/MAC layers. For the FTM implementation, the support of action frames in the 802.11 module of ns3 is required. The module supports action frames but is limited to block ack, mesh, multihop, self-protected, and vendor-specific action. Hence, we extended it to also support public action frames, in which only the FTM request and FTM response are supported. For this support, the `WifiActionHeader` class has been extended. All of the required FTM specific frames, like request, response, parameters and TSF sync, have been implemented and can be found in the 'ftm-header' files. The FTM request and response headers are implemented as defined by the 802.11 standard. The TSF sync info header also complies with the standard and is being transmitted in the first frame of every burst instance. It is not used actively and always has a value of 0, because time is always synchronized in the simulator. It is added to comply to standard and to have an accurate overhead representation of the FTM protocol.

The FTM parameters header is also implemented as defined in the standard but two of its fields are used differently. The partial TSF timer is a time value specified in ms and the initiator indicates to the responder when the FTM measurements should begin. For example, when the TSF timer value is set to 10, it means that the measurements should begin in 10 ms. This was done out of simplicity and because initiator and responder always have synchronized time. The other is the format and bandwidth field, which is not used and is always set to 0. All of the other fields are used as they are defined in the standard.

Next, we discuss the integration of FTM to the 802.11 PHY layer. It is needed in order to retrieve the time stamps of incoming and outgoing packets related to an FTM measurement. The way this is done, is by connecting the `FtmManager` of each WiFi node to the `PhyTxBegin` and `PhyRxBegin` callbacks of the `WifiPhy`. These callbacks are fired when the preamble of a packet has either been successfully transmitted or received. The main difference to the definition in the standard is, that the time stamps are set after the preamble has been received. This leads

```
Fixed parameters
    Category code: Public Action (4)
    Public Action: FTM Request (0x20)
    Trigger: 0x01
Tagged parameters (11 bytes)
    Tag: Fine Time Measurement Params
        Tag Number: Fine Time Measurement Params (206)
        Tag length: 9
        FTM Params (Subset 1 of 3): 0x7100
            .... .... .... ..00 = Status Indication: 0x0
            .... .... .000 00.. = Value: 0x00
            .... .... 0... .... = Reserved1: 0x0
            .... 0001 .... .... = Number of Burst Exponent: 0x1
            0111 .... .... .... = Burst Duration: 0x7
        FTM Params (Subset 2 of 3): 0x1500000a
            .... .... .... .... .... .... 0000 1010 = Min Delta FTM: 0x0a
            .... .... 0000 0000 0000 0000 .... .... = Partial TSF timer: 0x0000
            .... ...1 .... .... .... .... .... .... = Partial TSF no pref: 0x1
            .... ..0. .... .... .... .... .... .... = ASAP Capable: 0x0
            .... .1.. .... .... .... .... .... .... = ASAP: 0x1
            0001 0... .... .... .... .... .... .... = FTM per burst: 0x02
        FTM Params (Subset 3 of 3): 0x0a0000
            .... .... .... .... .... ..00 = Reserved2: 0x0
            .... .... .... .... 0000 00.. = Format and Bandwidth: 0x00
            0000 1010 0000 0000 .... .... = Burst Period: 0x0a00
```

Figure 13: FTM request packet displayed in Wireshark.

to some inaccuracies by having the preamble detection period in the calculated RTT. To remove this delay, the preamble detection duration is subtracted twice during RTT calculation. It is removed twice, because two frames are transmitted for each measurement, the FTM response and its Ack.

In addition to the timestamps required in each measurement, the received signal strength is also required for the wireless signal strength error model. To retrieve the signal strength of each measurement, the `FtmManager` is also connected to the `MonitorSnifferRx` callback of the `WifiPhy`. This callback is fired when a packet has been received successfully and includes the received signal strength in dBm. If the received packet is an FTM response frame, the signal strength is added to the current measurement using the dialog token of the frame. During RTT calculation the stored signal strength in each FTM measurement is used as an input to the `GetFtmError` method. This input is only used by the wireless signal strength error model to determine the parameters for the Johnson SU distribution by finding the closest match between the received signal strength of the measurement and the available P_{rx} values shown in Table 2. The FTM retransmissions have been handled in a way that an FTM frame can be transmitted as many times as needed. This is possible because the time stamps are renewed every time the frame is transmitted or received, even if the dialog token already exists and has time stamps. In this case the time stamps for that dialog will be overridden with the newest ones, but the ones transmitted in the frame using the followup dialog token are not set again if they have already been set at the initiator. This makes handling retransmissions simple.

In order to analyze the correctness of our FTM protocol implementation, we replicated the simple scenario shown in Fig. 1 with two WiFi nodes where one node was triggering FTM ranging. We enabled the tracing functionality of ns3 in order to capture all transmitted frames during the simulation. Those traces we later analyzed using the Wireshark tool. As can be seen from Fig. 13 Wireshark was able to correctly display the FTM request frame.

Our software implementation of FTM extension for ns3 together with examples is provided to the community as open source under GPL license: `https://github.com/tkn-tub/wifi-ftm-ns3`.

6. Performance Evaluation

In this section, we present results from simulations we performed in ns3 using our `FTM-ns3` module. We begin with performing ranging experiments between a pair of WiFi nodes illustrating the impact of the distance, the wireless channel propagation model as well as the FTM error model on the ranging accuracy. Therefore, we selected four scenarios with different degrees of realism ranging from a very simple to a realistic setup. We configured `FTM-ns3` to use the error models for the Intel 8260 NIC. Moreover, the long hallway was set as wireless environment. Afterwards, we analyze the performance of a simple localization scheme based on multilateration which uses the distances obtained from FTM ranging to estimate the 2D position of a mobile WiFi node.

6.1. Ranging Accuracy

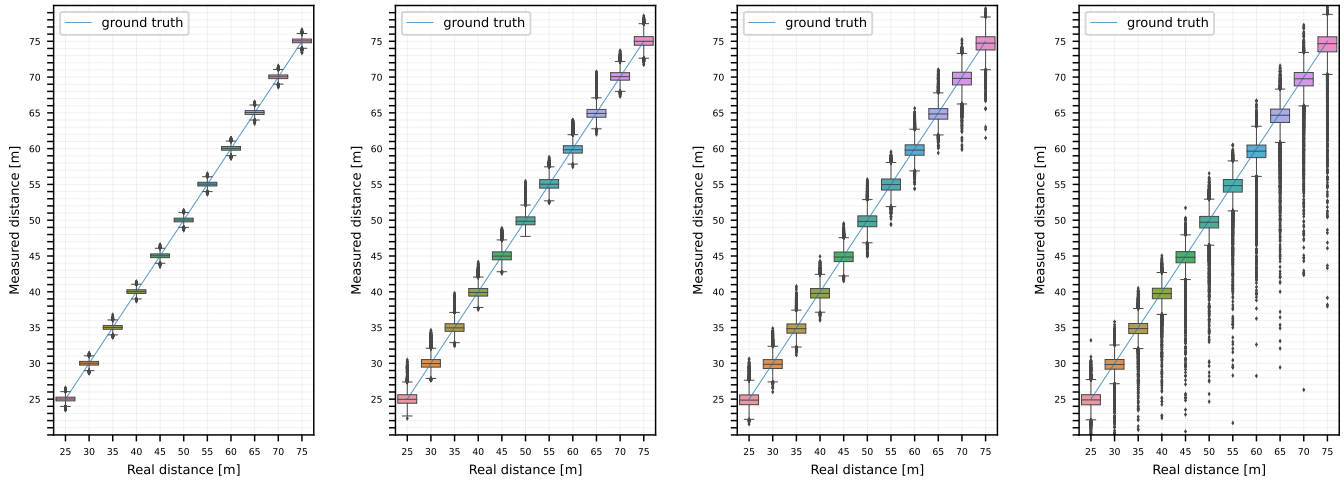
First, we analyze the FTM ranging accuracy for the four different scenarios depicted in Table 4. The objective is to understand the impact from the used wireless channel and FTM error models. The simplistic scenario $S1$ represents the best-case situation for FTM ranging. Here the receive signal power is constant at -40 dBm and therefore does not depend on the distance. Moreover, the wired FTM error model is being used here. A more realistic scenario is $S2$ where the wireless FTM error model is used. In $S3$ the radio propagation accounts for the attenuation due to distance resulting in long-distance links having low receive signal strength and SNR. In addition, the FTM error model also accounts for the impact of signal strength. Finally, $S4$ extends $S3$ by simulating small-scale fading.

In all experiments, the FTM responder was placed at a fixed location while the FTM initiator node was placed randomly on a circle with different radius r ranging from 25 to 75 m. For each r the initiator performed ranging operation at 180 random locations on the circle whereas the number of FTM measurements at each location was set to $F = 79$. Hence, in total 14220 ranging measurements were collected and analyzed for each r.

The results are shown in Fig. 14. We see that in the simplistic scenario $S1$ the measured distance via FTM ranging shows only a small variation around the real distance which also do not depend on the distance between the pair of nodes. This is because of the used wired FTM error model where the FTM ranging accuracy is only determined by the channel bandwidth. The situation is different in scenario S2 where the wireless FTM error model is being used. Here the ranging accuracy is distorted as the impact of the multipath channel propagation is modeled in the wireless FTM error model. In scenario S3 the ranging error further increases, especially for larger distances, which is

Scenario	Propagation model	Fading model	FTM error model	FTM map
1. constant power + wired	FixedRssLossModel (-40 dBm)	none	wired	no
2. constant power + wireless	FixedRssLossModel (-40 dBm)	none	wireless	yes
3. average power + wireless	ThreeLogDistancePropagationLoss-Model	none	wireless signal strength	yes
4. instantaneous power + wireless	ThreeLogDistancePropagationLoss-Model	NakagamiPropagationLoss-Model	wireless signal strength	yes

Table 4: The four evaluated scenarios (802.11n PHY/MAC, BW=20 MHz).



(a) S1: constant power + wired. (b) S2: constant power + wireless. (c) S3: average power + wireless. (d) S4: instant. power + wireless.

Figure 14: Ranging accuracy for all four scenarios.

caused by the used FTM error model which also takes the receive signal strength into account. Finally, the highest ranging error is observed in scenario S4 which is because of the additional simulation of fast-scale fading. Here we can clearly see the bias towards underestimating the true distance.

6.2. 2D Localization Accuracy

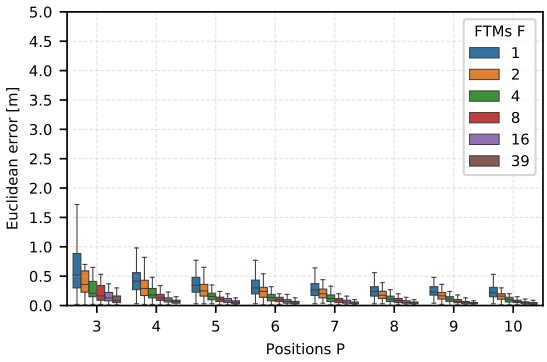
In a second experiment, we study the impact of the ranging error introduced by FTM on a more complex localization problem. We implemented a localization scheme based on multilateration where a mobile WiFi station knowing its 2D location performs FTM ranging while moving to a fixed anchor node with unknown position, e.g., some WiFi AP. The goal is to determine the 2D location of that fixed node. Therefore, the mobile station performs FTM ranging at randomly selected locations P (positions) with varying number of FTM measurements F. We again evaluated the four different scenarios from Table 4 to test the accuracy of the localization. We analyzed two different strategies. The first approach tries to average out the fluctuations in the ranging measurements by taking the mean of the F FTM rangings performed at each position to derive the distance to each AP. We refer to this approach as `baseline`. The second approach termed as `advanced` tries to improve the distance measurement accuracy by weighting the F FTM rangings according to their signal strength (RSSI): $\tilde{d} = \sum_{i=1}^{F} w_i \times d_i$, where d_i is the distance measured during the i-th measurement and the weight w_i is computed as $w_i = \frac{p_i}{\sum_{j=1}^{n} p_j}$ and p_i is the receive signal strength.

Finally, the distances obtained by both strategies are used to compute the 2D location of the WiFi node by means of multilateration which tries to minimize the error function. Each of the experiments was repeated 100 times.

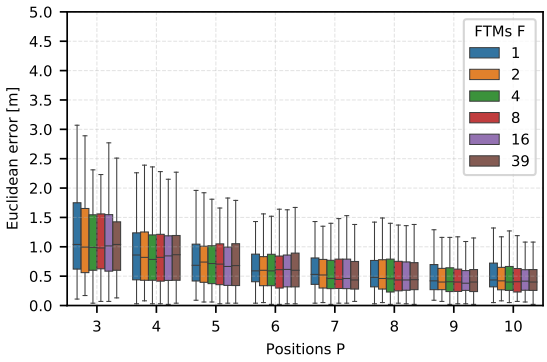
6.2.1. Baseline

We begin by analyzing the accuracy of localization in scenario $S1$ (Fig. 15a). We can observe that by increasing the number P of locations the accuracy of the localization can be increased significantly. A similar improvement can be achieved by increasing the number of rangings F at each location. Hence a trade-off can be made by decreasing P and increasing F to achieve similar accuracy. For the best possible accuracy in $S1$ P and F should be as high as possible.

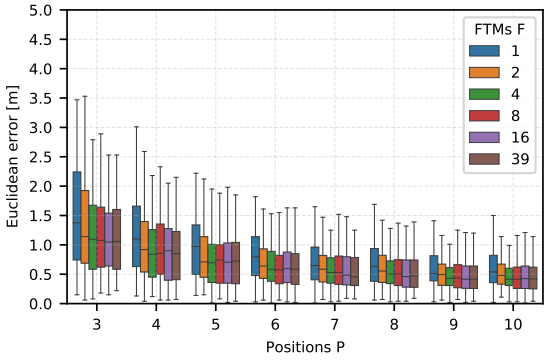
Next, the same experiment is repeated for scenario $S2$ where the wireless FTM error model is used (Fig. 15b). From the results we see that F has only a small impact on the localization accuracy. This is because the distortions due to multi-path effects cannot be averaged out by repeat-



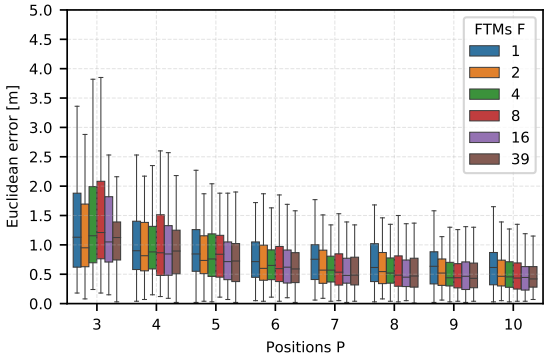
(a) S1: constant power + wired.



(b) S2: constant power + wireless.



(c) S3: average power + wireless.



(d) S4: instant. power + wireless.

Figure 15: Localization accuracy for different number of locations P, number of rangings F, and error models.

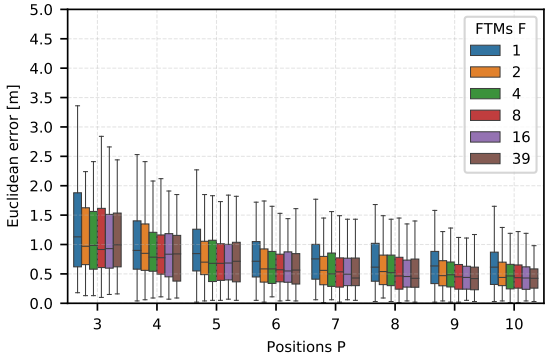


Figure 16: RSSI-aware localization with measurement weighting based on signal strength, using scenario S4.

ing the ranging multiple times for the same location. Only the increase in the number of locations P from which the ranging was performed improves the localization accuracy noticeably. However, we see a saturation with increasing P beyond 7 yielding only minor accuracy improvements.

In scenario $S3$ we see similar results but with a slightly higher localization error. This is because the signal strength of the WiFi signal is taken into account by the FTM error model. An increase in P results in an increase in accuracy up to a specific point. After that point the improvements are only minor. Moreover, an increase in F yields better accuracy for most of the values of P. Thus, we can attribute an accuracy increase with increasing F. In contrast to $S2$, in which F did not improve localization accuracy.

In the last scenario $S4$ the overall localization accuracy decreases the most of all scenarios. This is mainly due to the simulation of small-scale fading which creates additional distortions. When using a low value of P, it can be observed that an increasing F can result in both an accuracy increase and a decrease. This is due to the changing environment with fast-scale fading, in which a measurement may be influenced positively or negatively. A large enough F can correct this by being exposed to enough environment changes to average out the effect of fast-scale fading. For higher P an increase in localization accuracy is observed, as in the previous scenarios. Additionally, an increasing F also has accuracy improvements if P is large enough.

6.2.2. Advanced

Finally, we analyze the localization accuracy of the advanced strategy which takes the signal strength into account when estimating the distance. We present results for scenario $S4$ only as it represents the most realistic and challenging environment. The results are shown in Fig. 16. Comparing the results to the `baseline` strategy (cf. Fig. 15d), we see that for a low F and low P the localization is more accurate with a reduced variance. For $F = 1$ both strategies perform equally. Increasing F and P we observe that the `advanced` localization yields more stable results with less variance as F increases, compared

to `baseline`. This variance reduction is mostly observable up to $P = 7$ and afterwards both strategies perform mostly equal. These results show that our `advanced` scheme offers a noticeable improvement in localization accuracy for low P. This first improvement introduces the potential for future work to further improve localization accuracy using different algorithms and makes `FTM-ns3` a powerful tool for testing localization algorithms quickly and efficiently.

7. Conclusion

In this paper, we presented `FTM-ns3`, a software module for the ns3 network simulator to support the IEEE 802.11 FTM protocol. Empirical error models for FTM are provided which allow to study the performance of FTM-based localization schemes by taking into account ranging inaccuracies due to channel conditions, e.g., multipath propagation, and used FTM parameters, e.g., channel bandwidth. Moreover, we present results from simulations of a localization scheme based on multilateration which reveal its performance in an indoor environment with LOS and strong multipath. Our module can be used for benchmarking different FTM-based localization schemes.

As future work, we will focus on updating the wireless error model to support additional environments with either non-LOS or obstructed-LOS, e.g., shopping center or outdoors. We also plan to extend `FTM-ns3` to model the characteristics of newer WiFi chips supporting wider channel bandwidth, e.g., 80 or 160 MHz, as defined in IEEE 802.11ac/ax which is available when using the 5 GHz spectrum band. Finally, as newer generations of WiFi make heavy use of antenna beamforming we plan to investigate its impact on the FTM ranging accuracy.

Acknowledgements

We would like to thank S. Roesler and L. Pusch for their contributions in conducting the experiments. This work has been supported in part by the project AvaRange funded by the German Research Foundation (DFG) under grant DR 639/22-1.

References

[1] J. Xiao, Z. Zhou, Y. Yi, L. M. Ni, A Survey on Wireless Indoor Localization from the Device Perspective, ACM Computing Surveys (CSUR) 49 (2016) 1–31. doi:10.1145/2933232.

[2] R. Di Taranto, S. Muppirisetty, R. Raulefs, D. Slock, T. Svensson, H. Wymeersch, Location-Aware Communications for 5G Networks: How location information can improve scalability, latency, and robustness of 5G, IEEE Signal Processing Magazine 31 (2014) 102–112. doi:10.1109/msp.2014.2332611.

[3] M. Kotaru, K. Joshi, D. Bharadia, S. Katti, SpotFi: Decimeter Level Localization Using WiFi, ACM SIGCOMM Computer Communication Review (CCR) 45 (2015) 269–282. doi:10.1145/2829988.2787487.

[4] A. Zubow, S. Bayhan, P. Gawłowicz, F. Dressler, DeepTxFinder: Multiple Transmitter Localization by Deep Learning in Crowdsourced Spectrum Sensing, in: IEEE International Conference on Computer Communication and Networks (ICCCN 2020), IEEE, Virtual Conference, 2020, pp. 1–8. doi:10.1109/ICCCN49398.2020.9209727.

[5] F. Zafari, A. Gkelias, K. K. Leung, A Survey of Indoor Localization Systems and Technologies, IEEE Communications Surveys & Tutorials 21 (2019) 2568–2599. doi:10.1109/comst.2019.2911558.

[6] IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Std 802.11-2016, IEEE, 2016. doi:10.1109/IEEESTD.2016.7786995.

[7] A. Awad, T. Frunzke, F. Dressler, Adaptive Distance Estimation and Localization in WSN using RSSI Measures, in: 10th EUROMICRO Conference on Digital System Design - Architectures, Methods and Tools (DSD 2007), IEEE, Lübeck, Germany, 2007, pp. 471–478. doi:10.1109/DSD.2007.20.

[8] N. Jathe, M. Lütjen, M. Freitag, Indoor Positioning in Car Parks by using Wi-Fi Round-Trip-Time to support Finished Vehicle Logistics on Port Terminals, IFAC-PapersOnLine 52 (2019) 857–862. doi:10.1016/j.ifacol.2019.11.237.

[9] M. Bullmann, T. Fetzer, F. Ebner, M. Ebner, F. Deinzer, M. Grzegorzek, Comparison of 2.4 GHz WiFi FTM- and RSSI-Based Indoor Positioning Methods in Realistic Scenarios, Sensors 20 (2020) 4515. doi:10.3390/s20164515.

[10] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, F. Bai, Verification: Accuracy Evaluation of WiFi Fine Time Measurements on an Open Platform, in: 24th ACM International Conference on Mobile Computing and Networking (MobiCom 2018), ACM, New Delhi, India, 2018, pp. 417–427. doi:10.1145/3241539.3241555.

[11] NS-3 Consortium, ns-3 documentation, https://www.nsnam.org/, Accessed: 2021-05-23.

[12] A. Zubow, C. Laskos, F. Dressler, FTM-ns3: WiFi Fine Time Measurements for NS3, in: 17th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2022), IEEE, Virtual Conference, 2022, pp. 1–7. doi:10.23919/WONS54113.2022.9764460.

[13] G. Guo, R. Chen, F. Ye, X. Peng, Z. Liu, Y. Pan, Indoor Smartphone Localization: A Hybrid WiFi RTT-RSS Ranging Approach, IEEE Access 7 (2019) 176767–176781. doi:10.1109/access.2019.2957753.

[14] O. Hashem, M. Youssef, K. A. Harras, WiNar: RTT-based Sub-meter Indoor Localization using Commercial Devices, in: 18th IEEE International Conference on Pervasive Computing and Communications (PerCom 2020), IEEE, Austin, TX, 2020, pp. 1–10. doi:10.1109/percom45495.2020.9127363.

[15] V. Barral Vales, O. C. Fernandez, T. Dominguez-Bolano, C. J. Escudero, J. A. Garcia-Naya, Fine Time Measurement for the Internet of Things: A Practical Approach Using ESP32, IEEE Internet of Things Journal 9 (2022) 18305–18318. doi:10.1109/jiot.2022.3158701.

[16] S. Xu, Y. Wang, M. Si, A Two-Step Fusion Method of Wi-Fi FTM for Indoor Positioning, Sensors 22 (2022) 3593. doi:10.3390/s22093593.

[17] Y. Dong, D. Shi, T. Arslan, Y. Yang, Error Investigation on Wi-Fi RTT in Commercial Consumer Devices, Algorithms 15 (2022) 464. doi:10.3390/a15120464.

[18] S. Aggarwal, R. K. Sheshadri, K. Sundaresan, D. Koutsonikolas, Is wifi 802.11mc fine time measurement ready for prime-time localization?, in: 28th ACM International Conference on Mobile Computing and Networking (MobiCom 2022), 16th ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WINTECH 2022), ACM, Sydney, Australia, 2022, pp. 1–8. doi:10.1145/3556564.3558234.

[19] G. Retscher, Fundamental Concepts and Evolution of Wi-Fi User Localization: An Overview Based on Different Case Studies, Sensors 20 (2020) 5121. doi:10.3390/s20185121.

[20] D. Schepers, M. Singh, A. Ranganathan, Here, there, and everywhere, in: 14th ACM Conference on Security and Privacy

in Wireless and Mobile Networks (WiSec 2021), ACM, Abu Dhabi, United Arab Emirates, 2021, pp. 78–89. doi:10.1145/3448300.3467828.

[21] M. Sun, Y. Wang, K. Liu, C. De Cock, W. Joseph, D. Plets, Smartphone-based WiFi FTM Fingerprinting Approach with Map-aided Particle Filter, in: IEEE International Conference on Indoor Positioning and Indoor Navigation (IPIN 2022), IEEE, Beijing, China, 2022, pp. 1–8. doi:10.1109/ipin54987.2022.9918110.

[22] P. Eberechukwu, H. Park, C. Laoudias, S. Horsmanheimo, S. Kim, DNN-based Indoor Fingerprinting Localization with WiFi FTM, in: 23rd IEEE International Conference on Mobile Data Management (MDM 2022), IEEE, Paphos, Cyprus, 2022, pp. 367–371. doi:10.1109/mdm55031.2022.00082.

[23] K. Jiokeng, G. Jakllari, A. Tchana, A.-L. Beylot, When FTM Discovered MUSIC: Accurate WiFi-based Ranging in the Presence of Multipath, in: 39th IEEE International Conference on Computer Communications (INFOCOM 2020), IEEE, Virtual Conference, 2020, pp. 1857–1866. doi:10.1109/infocom41043.2020.9155464.

[24] K. I. Ahmed, G. Heidari-Bateni, Improving Two-Way Ranging Precision with Phase-offset Measurements, in: IEEE Global Telecommunications Conference (GLOBECOM 2006), IEEE, San Francisco, CA, 2006, pp. 1–6. doi:10.1109/glocom.2006.496.

[25] S. Lanzisera, D. Zats, K. S. J. Pister, Radio Frequency Time-of-Flight Distance Measurement for Low-Cost Wireless Sensor Localization, IEEE Sensors Journal 11 (2011) 837–845. doi:10.1109/jsen.2010.2072496.

[26] X. Li, K. Pahlavan, Super-Resolution TOA Estimation With Diversity for Indoor Geolocation, IEEE Transactions on Wireless Communications (TWC) 3 (2004) 224–234. doi:10.1109/twc.2003.819035.

[27] V. Ninkovic, D. Vukobratovic, A. Valka, D. Dumic, Preamble-Based Packet Detection in Wi-Fi: A Deep Learning Approach, in: 92nd IEEE Vehicular Technology Conference (VTC 2020-Fall), IEEE, Virtual Conference, 2020, pp. 1–5. doi:10.1109/vtc2020-fall49728.2020.9348765.

[28] J. Heiskala, J. Terry, OFDM Wireless LANs: A Theoretical and Practical Guide, SAMS, 2001.

[29] N. L. Johnson, Systems of Frequency Curves Generated by Methods of Translation, MAA Biometrika 36 (1949) 149. doi:10.2307/2332539.

[30] Q. H. Spencer, B. D. Jeffs, M. A. Jensen, A. L. Swindlehurst, Modeling the statistical time and angle of arrival characteristics of an indoor multipath channel, IEEE Journal on Selected Areas in Communications (JSAC) 18 (2000) 347–360. doi:10.1109/49.840194.