HowTo for the ns-3 FTM interface

Christos Laskos

May 19, 2021

1 HowTo

This document focuses on how to use the interface and its components. It is divided in two sections. First the usage of the FTM interface in ns-3 and second the usage of the script used for the map generation for the wireless error model.

1.1 NS-3 Interface

In this section I go over the usage of the developed ns-3 FTM interface. This includes the basic usage of the interface, to using custom FTM parameters as well as using the implemented error models. Most of the code examples used in this section can be found in the 'ftm-example.cc' script.

1.1.1 Basic Use

The basic usage of the FTM interface consists of the following steps:

- Setting the correct time resolution
- Enabling FTM support on the MAC layer
- Creating and starting a bare-bones session

Setting the time resolution is very crucial when using the interface. The default time resolution in ns-3 is set to nano seconds and may lead to inaccuracies. The time resolution should be changed to pico seconds to ensure the best possible accuracy. How the time resolution can be changed is shown in listing 1.

```
Time::SetResolution (Time::PS);
```

Listing 1: Setting time resolution to pico seconds

The next step is to enable FTM support on the MAC layer. The functions for enabling and disabling FTM support reside in the RegularWifiMac class. With access to the RegularWifiMac instace the function 'EnableFtm' can be called on each node that should support FTM. FTM support can also be disabled by calling 'DisableFtm'. In listing 2 the enabling process through the RegularWifiMac is shown.

```
//access to the NetDevice for which to enable FTM
Ptr<NetDevice> net_device;
//get the WifiNetDevice from the NetDevice
Ptr<WifiNetDevice> wifi_net_device = net_device->GetObject<WifiNetDevice> ();
//get the WifiMac
Ptr<WifiMac> wifi_mac = wifi_net_device->GetMac ();
//get the RegularWifiMac
Ptr<RegularWifiMac> reg_wifi_mac = wifi_mac->GetObject<RegularWifiMac> ();
//enable FTM
reg_wifi_mac->EnableFtm ();
```

Listing 2: Enabling FTM through RegularWifiMac

If FTM support should be enabled on all nodes, the ns-3 attribute system should be used instead. The command for enabling and disabling FTM using the attribute system is shown in listing 3.

```
Config::SetDefault ("ns3::RegularWifiMac::FTM_Enabled", BooleanValue (true));
```

Listing 3: Enabling FTM using the attribute system

By default FTM support is always disabled and the user has to explicitly enable it.

After the correct time resolution is set and FTM is enabled a new FTM session can be created. To create a new FTM session, access to the RegularWifiMac instance of the initiating node/station is required. We then can call the 'NewFtmSession' method with the MAC address of the responding node/station as parameter. A pointer to a new FtmSession should be returned, if the session was created successfully. If the returned pointer has a value of 0, the session could not be created. This may be caused by the following:

- Not enabling FTM
- An already existing session with the specified responder
- A previous session has been denied and a new session with the specified responder is blocked for some time
- The specified responder address equals the initiator address

After successful creation of the FTM session, the 'SessionOverCallback' should be set. This way the user is able to access the sessions data once it has finished. At last the 'SessionBegin' method should be called to start the session. FTM parameters do not need to be set in this case. When no parameters are specified, the default ones will be used. An example of starting an FTM session is shown in listing 4.

```
//function structure to use for session over callback
void
SessionFinished (FtmSession session)
{
    session.GetMeanRTT ();
}

//access to RegularWifiMac of initiator
Ptr<RegularWifiMac> reg_wifi_mac;
Mac48Address resp_addr; //responder address
Ptr<FtmSession> session = reg_wifi_mac->NewFtmSession (resp_addr);
session->SetSessionOverCallback (MakeCallback (&SessionFinished));
session->SessionBegin ();
```

Listing 4: Creating and starting an FTM session

1.1.2 Setting FTM Parameters

In this section I explain how to set custom FTM parameters for each session individually and how to set default parameters for all created sessions. Setting parameters for each session individually can be done after a session has been created. Using the 'SetFtmParams' method a user can specify an 'FtmParams' object, which will be used for the session. It is import to note that the session setup has to be done, before the 'SessionBegin' method is called. An example for setting FTM parameters in an individual session can be found in listing 5.

```
//the FtmSession as previously created
Ptr<FtmSession> session;
//create an FtmParams header
FtmParams parameters;
parameters.SetNumberOfBurstsExponent (2); //4 bursts
parameters.SetBurstDuration (8); //4 ms burst duration
parameters.SetMinDeltaFtm (4); //400 us between frames
parameters.SetAsap (true); //ASAP
parameters.SetFtmsPerBurst (5); //5 FTMs per burst
parameters.SetBurstPeriod (4); //400 ms between bursts
session->SetFtmParams (parameters);
//possible further setup and SessionBegin ()
```

Listing 5: Adding FTM parameters to specific session

It is also possible to use the ns-3 attribute system to change the default FTM parameters for all created FTM sessions. In this case an 'FtmParamsHolder' object and an 'FtmParams' header need to be created. All the parameters are set in the 'FtmParams' header and afterwards it is passed to the 'FtmParamsHolder' object. Now the 'FtmParamsHolder' object can be set as the default FTM parameters in all sessions, using the attribute system as shown in listing 6. If this method is used, it is not necessary to set parameters individually as in listing 5.

```
//assume FTM parameters are already created and initialized
FtmParams parameters;
//create FtmParamsHolder object
Ptr<FtmParamsHolder> params_holder = CreateObject<FtmParamsHolder> ();
params_holder->SetFtmParams (parameters);
Config::SetDefault ("ns3::FtmSession::DefaultFtmParams", PointerValue (params_holder));
```

Listing 6: Setting default FTM parameters for all sessions

1.1.3 Usage of the Wired Error Model

The wired error model can be set in two ways. Either in the setup stage of the FTM session as shown in listing 5 or through the attribute system for all FTM sessions.

In the setup stage a new WiredFtmErrorModel should be created. The default bandwidth used is 20 MHz. This can be changed to 40 MHz, to achieve more accurate results. It is also possible to set a custom mean and standard deviation for the gaussian distribution by using the 'SetMean' and 'SetStandardDeviation' functions. After the wired error model has been set up, it can be added to the session. This procedure is shown in listing 7.

```
Ptr<WiredFtmErrorModel> wired_error = CreateObject<WiredFtmErrorModel> ();
//using 40 MHz channel error
wired_error->SetChannelBandwidth (WiredFtmErrorModel::Channel_40_MHz);
session->SetFtmErrorModel (wired_error);
```

Listing 7: Using the WiredFtmErrorModel in session setup

Setting the wired error model for all sessions using the attribute system is done in two steps. First we create a new WiredFtmErrorModel to be used and change its parameters if needed. Afterwards it is being set in all sessions using the attribute system. This is shown in listing 8.

```
Ptr<WiredFtmErrorModel> wired_error = CreateObject<WiredFtmErrorModel> ();
//further setup of the error model if required
Config::SetDefault("ns3::FtmSession::FtmErrorModel", PointerValue (wired_error));
```

Listing 8: Setting the WiredFtmErrorModel through the attribute system

1.1.4 Usage of the Wireless Error Model

The usage of the wireless error model is only possible in the setup phase of the session. This is due to it depending on the position of the node. Setting up the wireless error model requires access to the node it is attached to and an FtmMap. The FtmMap needs to be loaded from a previously created '.map' file. The procedure to create '.map' files is explained in section 1.2. After the map is loaded the WirelessFtmErrorModel can be created and the Node and FtmMap passed to it. The Node object can be accessed from the NetDevice or WifiNetDevice, used in listing 2, with the GetNode method. The default bandwidth used is again 20 MHz. At last the wireless error model can be passed to the session it will be used in. This entire procedure is shown in listing 9.

```
Ptr<WirelessFtmErrorModel::FtmMap> ftm_map = CreateObject<WirelessFtmErrorModel::FtmMap> ();
//load FtmMap from previously created map file
ftm_map->LoadMap ("FTM_Wireless_Error.map");
//get the initiator node
Ptr<Node> node = net_device->GetNode();
//create WirelessFtmErrorModel
Ptr<WirelessFtmErrorModel> wireless_error = CreateObject<WirelessFtmErrorModel> ();
wireless_error->SetFtmMap(ftm_map);
wireless_error->SetNode(node);
//further setup of the error model if required
session->SetFtmErrorModel(wireless_error);
```

Listing 9: Using the WirelessFtmErrorModel

It is also possible to set the same FTM map for all created WirelessFtmErrorModels using the attribute system. First the map needs to be loaded as shown in the beginning of listing 9. We will assume the FtmMap object is already created and the map is loaded. Afterwards it can be set using the attribute system as shown in listing 10.

```
Config::SetDefault ("ns3::WirelessFtmErrorModel::FtmMap", PointerValue (ftm_map));
```

Listing 10: Setting the FtmMap for all Wireless error models

1.2 FTM Map Generator

The FTM maps used for the WirelessFtmErrorModel are created by a python script. This script can be found under 'src/wifi/ftm_map' and is called 'ftm_map_generator.py'. It was created and only used with python version 3.6, so it is recommended to use version 3.6 or later. The script can either be used to create new maps or to read existing maps and display them visually. This section will focus on the basic usage of the script. Further usage options, can be displayed using the '-h' argument.

1.2.1 Map Creation

In this section we are creating a square map with 10 m × 10 m size. For this operation we use the '--dim' argument to pass the dimension for both axes. With this argument a map with $[-\frac{x}{2}, +\frac{x}{2}]$ on both axes is created, where x is the specified size in meters. In this case we want to use 10 as argument. This creates a map with axes dimensions of [-5, 5]. The default bias value in the generator script is set to 10000 ps. If the exponentially modified normal error distribution should be used for the map generation, the '--heavy_multipath' option should be added. A custom file name can be specified with the '-o' argument. We will name this map '10x10'. The .map file ending is appended automatically, if it has not been specified. If no file name is specified the default file name is used. Beware that map files may get very large with increasing dimensions. The programm call for this example is shown in listing 11.

```
python ftm_map_generator.py --dim 10 -o "10x10"
```

Listing 11: FTM map creation

1.2.2 Map Visualization

Previously created maps can be visualized using the map generator script. This can either be done with the '--read' argument, which tries to read the map from the default file name. If a custom map name was chosen, the '--readfile' argument followed by the file name is to be used. The programm call to read our previously created 10x10 map is shown in listing 12.

```
python ftm_map_generator.py --readfile "10x10.map"
```

Listing 12: Visualizing an FTM map