# Wishful IBFD Tutorial

# Initialization

The following example shows how the experimenter can create IBFD UPI and initiate it. The parameters below must be set prior to the experiment and before **start** command: Frequency, Transmitter Shift Frequency, Tx Gain, Rx Gain, Tx Sample Rate and Device Address. The UPI instructor initiates this parameters with the default values.

from IBFD\_agent import \*

Agent = IBFDAgent(ip\_address='10.33.136.138', ip\_port = 5022)

Agent.start()

# Write the experiment code here

Agent.stop()

# Start/Stop USRP

After initialization, **start** command runs the USPR and the parameters blow are still adjustable: Destination Address, Payload, collision detection average length, collision detection threshold and MAC settings. To terminate the experiment, the USRP should be stopped by **stop** command.

# Retuning duplexer

Prior to the experiment, the analog self-interference canceller should be tuned. This is needed only once after **start** command and can be done as following:

Agent.start()

Agent.si\_enable()

time.sleep(1)

Agent.retune\_duplexer(True)

time.sleep(2)

Agent.retune\_duplexer(False)

# Train the collision detector

The transmitting signal can be provided form either the host or the Microblaze. Due to the backoff time in CSMA/CD deployed on the Microblaze, for training the collision detector, the host should provide the transmitting signal. The following commands shows how one can train the collision detector and read the percent of the collision. The collision detection makes use of an average block. The averaging length and the detection threshold are adjustable through **set\_cd\_avg\_length** and **set\_cd\_threshold** commands.

Agent.start()

Agent.set\_cd\_avg\_length(8)

Agent.set\_cd\_threshold(6)

Agent.set\_mac(1) # set host as the source of transmitting data

Agent.si\_enable() # start transition

time.sleep(1)

Agent.set\_collision\_detection('reset') # reset the collision detector

time.sleep(2)

Agent.set\_collision\_detection('train') # train the detector

time.sleep(3)

Agent.set\_collision\_detection('fix\_alpha') # fix the detector alpha parameter

time.sleep(1)

Agent.set\_collision\_detection('start') # start detection

Agent.set\_mac(0) # set Microblaze as the source of transmitting data

time.sleep(3)

CD\_res = Agent.req\_cd\_result() # reading the detected value in percent

print(CD\_res)

# MAC setting

The MAC works in two modes; CSMA and CSMA/CD. To enable collision detection in the MAC protocol, the CD training phase should be down primitively as it is explained above. The CSMA/CD protocol is applicable only when the Microblaze is the source of the data, i.e. **Agent.set\_mac(2).**

# Setting host payload

The host payload is adjustable through the following commands. The experimenter can set an array of bytes (at most 118 elements). The following sample code shows how one can set the host payload.

Note: The Microblaze utilizes a ramp of size 118 as its payload.

Data = bytearray([0,1,2,3,4])

Agent.send\_payload(Data)

# Data request

After initialization and calling the start command, the experimenter can send request for IQ data or the received packet via **req\_iq\_data()** and **req\_recv\_pkt()** commands.

# Example

The Python code below, establishes a CSMA/CD protocol experiment with 2 terminals.

import time

from IBFD\_agent import \*

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Initialization

Agent1 = IBFDAgent(ip\_address='10.33.136.138', ip\_port = 5022)

Agent2 = IBFDAgent(ip\_address='10.33.136.139', ip\_port = 5022)

Agent1.device\_address(0)

Agent2.device\_address(1)

Agent1.dest\_address(1)

Agent2.dest\_address(0)

Agent1.set\_tx\_power(5)

Agent2.set\_tx\_power(5)

Agent1.set\_rx\_gain(0)

Agent2.set\_rx\_gain(0)

Agent1.set\_cd\_avg\_length(8)

Agent1.set\_cd\_threshold(4)

Agent2.set\_cd\_avg\_length(8)

Agent2.set\_cd\_threshold(4)

Agent1.set\_collision\_detection('reset')

Agent2.set\_collision\_detection('reset')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start USRPs

Agent1.start()

Agent2.start()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Retuning Duplexer

Agent1.si\_enable()

Agent1.retune\_duplexer(True)

time.sleep(3)

Agent1.retune\_duplexer(False)

Agent1.si\_disable()

Agent2.si\_enable()

Agent2.retune\_duplexer(True)

time.sleep(3)

Agent2.retune\_duplexer(False)

Agent2.si\_disable()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Train the collision detectors

Agent1.si\_enable()

Agent1.set\_mac(1)

time.sleep(1)

Agent1.set\_collision\_detection('reset')

time.sleep(2)

print('... Training')

Agent1.set\_collision\_detection('train')

time.sleep(3)

print('... Fix Alpha')

Agent1.set\_collision\_detection('fix\_alpha')

time.sleep(2)

Agent1.set\_collision\_detection('start')

Agent1.set\_mac(0)

Agent1.si\_disable()

Agent2.si\_enable()

Agent2.set\_mac(1)

time.sleep(1)

Agent2.set\_collision\_detection('reset')

time.sleep(2)

print('... Training')

Agent2.set\_collision\_detection('train')

time.sleep(3)

print('... Fix Alpha')

Agent2.set\_collision\_detection('fix\_alpha')

time.sleep(2)

Agent2.set\_collision\_detection('start')

Agent2.set\_mac(0)

Agent2.si\_disable()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Experiment: data exchange with CSMA/CD

Agent1.set\_mac(2) #enable CSMA/CD

Agent2.set\_mac(2) #enable CSMA/CD

Agent1.si\_enable()

Agent2.si\_enable()

Data1 = Agent1.req\_iq\_data()

print(Data1[‘IQ-COMPLEX’]) #print captured data

Data2 = Agent2.req\_recv\_pkt()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Terminate the experiment

Agent1.si\_disable()

Agent2.si\_disable()

Agent1.stop ()

Agent2.stop ()