SLIM Waveform Manual
Document Version 1.2

The Samraksh Company

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Contents

Introduction

"SLIM"(Synchronous Lightweight Interference Modeling) waveform is a waveform model that simulate a distributed TDMA (time division multiple access) MAC layer with a very flexible physical layer. The PHY layer consists of a lightweight model that can be configured for a large range of real life physical layers.

SLIM Overview

SLIM waveform provides a flexible and customizable abstract waveform that mimics real life waveforms to estimate their performance in simulation. Rather than implementing the details of a particular PHY layer to measure the performance metrics on a per-device basis, SLIM waveform takes a generic approach and uses apriori knowledge of simple performance metrics of a PHY layer to simulate system level performance.

We envision four types of use cases for SLIM:

- **Study emergent behavior:** A user can quickly implement a Waveform/PHY layer with a well known behavior using SLIM and simulate it in ns3 in vairous scenarios and configurations and study the network wide performance of the waveform along with other higher layer protocols.
- **Modify or adapt waveforms:** A user can implement a existing Wavform/PHY but then modify certains parameters like FEC ratios or datarate/bandwidth to understand tradeoffs;
- **Isolate higher layer protocols effects:** Many a times it becomes tough to isolate the performance/cost of a higher level protocol such as routing or transport from the effects (such as cost/stabilty) of lower level MAC/PHY protocols. This is especially try for complex lower layer protocols. SLIM provides a simple AND a wireless channel/phy model with well known characteristics that the user herself specifies.
- **Study custom or proprietary waveforms:** Many of the commercial radios have proprietary code or non standard implementations that a user might not want to reveal outside of the organization. SLIM provides for a way of studying those Waveforms under simulation, since only performance characteristics are needed rather than implementation details.

The key characteristics of the SLIM waveform are as follows:

- Models interference in the system including far-field interference effects;
- Simulates packet errors under highly varying SINR;

- Supports a wide range of modulation and FEC models;
- Supports packets structures consisting of different transmission modes;
- Supports customizable propagation models suitable for larege airborne networks including the curvature of the earth models (This feature is not available in the current release);
- Incorporates customizable antenna radiation pattern. (Although requires non trivial development of new radiation patterns)

Samraksh will provide configurations for SLIM Waveforms that corresponds to some well-known and commercially available PHY models. These PHY models can be readily used by users. In addition to those, SLIM waveform can easily be configured to model a custom PHY layer of a known waveform. Packet transmissions in SLIM are governed by a Transmission Vector (TxVector) configuration. A transmission consists of a one or more packet sections each with transmission characteristics that are specified by a SLIMMode. A SLIMMode is defined by the specifying physical layer characteristics such as bandwidth, data rates, BER, and FEC functions. Each SlimMode determines receivers decoding capabilities of that packet section. Users can create custom waveforms by creating one or more new SLIMModes, by providing SLIMMode parameters needed as abstract statistical models. These new modes can then be chained to form new TxVectors to be used by the custom waveform. Samraksh will create some examples of how the SLIM waveform can be used to simulate well know waveforms. The task is expected to be complete in next reporting period and will be released on the repository.

The rest of the document explains how to install and use the "SLIM" (i) with standalone ns-3 simulations, and (ii) with the Samraksh's Tuscarora framework.

Installation

Standalone installation for ns-3

SLIM waveform source files reside inside the Tuscarora package in "`/TuscaroraFW/ns-3.x/src/slim/`". To integrate these source files with the ns-3 simulator, we recommend creating a symlink of this directory under "`<ns3path>/src/`" using the following commands.

$ cd $< ns3path >$/src/
$ ln -s ${TUS}/ns-3.x/src/slim ${NS3_DIR}/src/

Additionally install SLIM tests using
$ ln -s ${TUS}/ns-3.x/scratch/slim*.cc ${NS3_DIR}/scratch/

Next, ns-3 configuration is updated and the ns-3 is rebuild via the following commands.

$ ./waf configure
$ ./waf
$ ./waf build

Along with Tuscarora

Tuscarora installer also installs SLIM waveform automatically. Using a configuration file passed as a parameter to Tuscarora binarys using –WFConfig parameter the SLIM can be created on all or a selected subset of the nodes. For details on creating and using configuration files please refer to the Tuscarora User Manual.

Components of the SLIM waveform

"SLIM" employs a simple distributed and randomized TDMA scheme. Using "SLIM", each node continuously listen to the channel for any transmission that it can decode unless it transmits a message. Each node individually selects random slots to transmit its messages. In the case of two or more nodes selecting the same slot, unless one of the transitions capture receivers with a high SINR, the transmissions results in a collision preventing successful reception of all colliding transmissions.

As depicted in Fig. 2.1, SLIM waveform consists of:

- a channel that consists of a propagation loss model and and propagation delay model,
- a phy layer that consists of an interference helper,
- a net device that implement MAC layer and an interface for the interaction with upper layers.

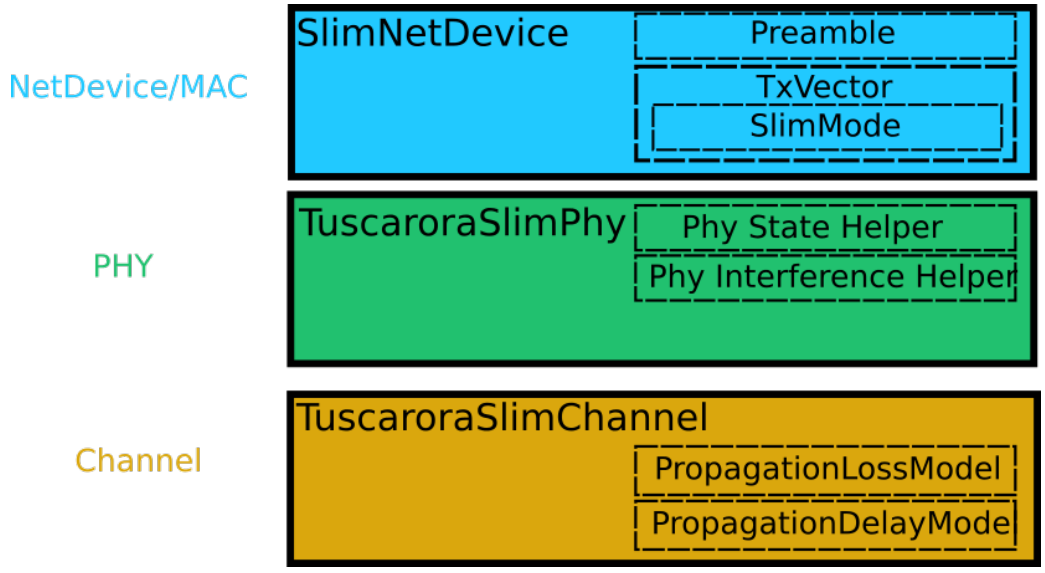

Figure 2.1: Waveform modules

Adapting SLIM Waveform

SLIM waveform can be adapted for a large class of waveforms whose characteristics under constant SINR are known apriori.

First an error function is derived from SlimModeError which simply implements a *GetBER(double sinr)* function. An example of this is given in *TuscaroraDsssDbpskModeError*.

In the next stage, the user creates the *SlimMode* objects with unique names and setting their parameters. The input variables that should be specified for each mode is listed in table 2.1 along with their defaults.

Table 2.1: Attributes of SlimMode

Attribute	Short Description	Default Value
UniqueName	A unique name for the mode	”UNINITIALIZEDMODE”
Bandwidth	The bandwidth(Hz) of the signal being modeled.	20000
Tpb	Time it takes to transmit one bit of information. Only needed for modes specifying payload.	1000ns
Duration	Total time it takes to transmit the section. Only needed for fixed length sections of a packet. Calculated internally for payload.	1000ns
FecF	The fraction of bits that the FEC can correct. A double value.	0.0
ErrorModPtr	Pointer to the SlimModeError object implementing SINR vs. BER characteristics. The default is of type TuscaroraDsssDbpskModeError.	NULL

Next, *SlimTxVector* objects that represent the transmission characteristics of packets are created. This object holds a vector of previously created *SlimMode* objects each representing a part of the packet such as preamble or a specific level header. All of the modes have fixed sizes except for the payload whose size is the determined at the time of the transmission. The position of the payload relative to other parts of the packet in the vector is explicitly specified using *SetPayloadModePosition(uint8_t k)*.

Table 2.2: Attributes of SlimTxVector

Attribute	Short Description	Default Value
UniqueName	A string containing a uniqueName of the TxVector. Used for comparison purposes.	”UNINITIALIZEDTXVECTOR”
PayloadModePosition	The index of the section containing the payload.	0

Each *TuscaroraSlimPhy* object holds a list of *SlimTxVector* objects. This lists determines the capa-

bilities of the PHY to transmit/receive packets encoded with the underlying transmission characteristics.

In the final stage, *SlimTxVector* objects created in the previous stage is added to this list. Along with these characteristics a channel number is defined for the PHY. This can be changed at any instant and only phy object operating on the same channel are able to communicate with each other.

SlimNetDevice

The *SlimNetDevice* is the top layer providing an interface to the upper layers using the waveform as a communication device. *SlimNetDevice* has various attributes that can be customized on the fly. These are summarized in table 2.3.

Table 2.3: Attributes of TuscaroraSlimNetDevice

Attribute	Short Description	Default Value
TxVectorIndex	The index of the SlimTxVector to be used for transmitting packets in the list of supported SlimTxVectors of the attached phy layer.	0
ContentionWindowSize	Average number of slots in between two selected transmission slots.	10
SlotDuration	The size of each slot(in ns3::Time type).	1ms
TxQueue	The pointer to a queue object used as the transmit queue in the device.	DropTailQueue

Upper layers pass the packets using *SlimNetDevice*'s *Send* routine. These packet gets queued up and are transferred to the *TuscaroraSlimPhy* one by one during the chosen transmission slots. Outgoing packages are stored in a customizable queue. The default queue is of "DropTailQueue" type and can be customized using any standard queue object through the attribute "TxQueue" of the "slim-net-device".

Multiple packets can be transmitted in a given transmission slot depending on the size of the packets. At the end of each transmission, if there are remaining packets in the *TxQueue* and the time it takes to transmit the next packet is less than the remaining time of the slot, *SlimNetDevice* starts the transmission of the next packet. Otherwise, *SlimNetDevice* waits for the next transmission slot.

The slot timing and boundaries are synchronous but the slot selection algorithm is distributed and independent. The next transmission slot is selected randomly from a range of [1-*ContentionWindowSize*] slots. The max limit of this range determines the average frequency of slot selection and can be adjusted on the fly using the attribute *ContentionWindowSize*. The default value of *ContentionWindowSize* is ten slots.

On the receiving side, *SlimNetDevice* checks the destination field of the packets and drops the packets that are not destined for the node. In addition to this, *SlimNetDevice* offers a promiscuous packet sniffer and callback. If this mode is used, all packets are passed to the chosen callback

regardless of the destination.

TuscaroraSlimPhy

The *TuscaroraSlimPhy* implements the PHY layer functions for the "SLIM" waveform. *TuscaroraSlimPhy*'s main duties are: (i) keeping the state of the physical layer, (ii) setting the modulation type for the packets and calculating their transmission duration, (iii) keeping track of the interference level using the *SlimInterferenceHelper*, and (iv) deciding on whether a packet is successfully received or had unrecoverable errors using *SlimInterferenceHelper* and the error characteristics of the *SlimMode*s making up the packet. *TuscaroraSlimPhy* has various attributes that can be customized on the fly. These are summarized in table 2.4.

On the sender side, *SlimNetDevice* create a *SlimTxVector* setting packet properties such as the modulation class and FEC characteristics and set the transmit power level and the payload size. Then, the packet is passed down using *TuscaroraSlimPhy*'s *SendPacket* routine that further sends the packet down to the channel and schedules the end of the transmission for state changes. All devices operating on the same channel number as the transmitting device are invoke for packet reception. The channel also calculates the power level at each receiver using the chosen *PropagationLossModel*. The antenna gains relative to the isotropic antenna are calculated based on the *AntennaModel* and the relative angle between the devices and added to the received power.

Table 2.4: Attributes of TuscaroraSlimPhy

Attribute	Short Description	Default Value
EnergyDetectionThreshold	The energy of a received signal should be higher than this threshold (dbm) to allow the PHY layer to detect the signal.	-96.0
TxGain	Transmission gain (dB).	1.0
RxGain	Reception gain (dB).	1.0
TxPowerLevels	Number of transmission power levels available between TxPowerStart and TxPowerEnd included.	1
TxPowerEnd	Maximum available transmission level (dbm).	16.0206
TxPowerStart	Minimum available transmission level (dbm).	16.0206
RxNoiseFigure	Loss (dB) in the signal to noise ratio due to non-idealities in the receiver.	0
StateHelperPtr	The pointer to the helper class object that keeps the state of the phy layer	PointerValue ()
ChannelSwitchDelay	Time it takes to switch channels.	250000ns

ChannelNumber	Channel number among multiple orthogonal channels.	1
Frequency	The central operating frequency.	2407
AntennaPtr	The pointer pointing to the associated antenna object.	IsotropicAntennaModel

On the reception side, *TuscaroraSlimPhy* receives packets from the channel using *StartReceivePacket* function. The average reception power of all packets are recorded using the *SlimInterferenceHelper* in a linked list together with their start and ending time instances. Although all packets are considered as a possible interferer, only for packets that have a larger power than *EnergyDetectionThreshold* are considered for reception and an *EndReceive* event is scheduled. *EndReceive* function determines whether the reception was indeed successful by considering the modulation characteristics specified by the associated *SlimTxVector*.

In calculating the packet error rate, *SlimInterferenceHelper* takes into account the cumulative list of interference events throughout the packet duration as kept by the and the particular modulation and FEC scheme employed.

Each packet consists of sections with possibly different error characteristics due to the underlying modulation and FEC characteristics. For instance, for many waveforms the preamble section is more vulnerable to interference compared to the rest of the packet. Similarly, due to compatibility issues, certain headers might have different characteristics than the payload. These characteristics are abstracted in a *SlimMode*.

The *SlimModeError* for each *SlimMode* implement an error function describing the relationship between the SINR and the BER characteristics excluding the FEC. Total number of errors in the section is calculated based on the *SlimModeError* under varying SINR. This number is compared with the total number of errors that can be corrected by the FEC to determine whether packet can be successfully received or not.

The defaults in the *SlimNetDevice* uses a single modulation and FEC rate for the entire packet. Since the packet consists of a single section, the packet success rate entirely depends on the random number of bit errors in the packet and error correction capabilities of the FEC algorithm.

TuscaroraSlimChannel

The *TuscaroraSlimChannel* object connects the *TuscaroraSlimPhy* objects of various nodes to each other. Its main duties are: (i) calculating the received power levels on each node using the propagation loss model, (ii) calculating the time instance with which the packet reaches to the nodes using the propagation delay model, (iii) initiating reception on each node's phy layer. *TuscaroraSlimChannel* has various attributes that can be customized on the fly. These are summarized in table 2.5.

Table 2.5: Attributes of TuscaroraSlimChannel

Attribute	**Short Description**	**Default Value**

PropagationLossModel	A pointer to the propagation loss model attached to this channel.	PointerValue ()
PropagationDelayModel	A pointer to the propagation delay model attached to this channel.	PointerValue ()

Configuration

"SLIM" waveform can be used both within Tuscarora framework simulations and with stand-alone ns-3 simulations.

Configuration with Tuscarora Framework

Standard "`runOrDebug.sh`" script provides automatic configuration for the SLIM waveform using the standard waveform configuration file of the "Tuscarora" framework. The waveform name to be used in the configuration file for the "SLIM" waveform is "SlimNetDevice".

An example configuration file for "SLIM" is available in "`TuscaroraFW/Config/slim-test.cnf`". This file can be used as:

```
$ ./runOrDebug.sh Gossip – WFConfig  ${TUS}/Config/slim-test.cnf
```

For further details on the usage of "`runOrDebug.sh`" please refer to the TuscaroraUserManual.

Configuration with ns-3 Simulations

In this section, a configuration example for the "SLIM" waveform to be used in a stand-alone ns3 simulation is presented. The source code for the example is available in "`TuscaroraFW/ns-3.x/scratch/slim-example.cc`"

Creation of the Nodes and the Waveform

The main function inside "`slim-example.cc`" begins by creating the nodes and a node container that holds these objects. Next it creates a mobility model and attaches to the nodes. In this example, a scenario with stationary nodes are simulated and hence a *ns3::ConstantPositionMobilityModel* is getting created.

Next, a common "TuscaroraSlimChannel" object that is shared among the nodes is created and is configured with a propagation loss model and a propagation delay model of "FriisPropagationLossModel" and "ConstantSpeedPropagationDelayModel" respectively. Any standard ns3 propagation loss and propagation delay model can be used with the "SLIM" waveform.

Next, inside a “for” loop, individual “SlimNetDevice” and “TuscaroraSlimPhy” objects are created and configured for each node. The loop starts with creation of the net device to be attached to the node. Next the PHY layer, i.e. *TuscaroraSlimPhy* is created and customized as follows:

- A pointer to the mobility model object is passed to the *TuscaroraSlimPhy* object to be stored in the PHY.
- A list of operation modulation and slim rates that is supported by the mode *TuscaroraSlimPhy* needs to be created and specified. In the example script a *txModeList* consisting of only 1 Mbps is created, and all nodes are configured using this list.
- The channel object that the *TuscaroraSlimPhy* operates on is registered.
- The channel object defines multiple bands and the corresponding channel number to be used by the *TuscaroraSlimPhy* is set.
- The *TuscaroraSlimPhy* and *SlimNetDevice* are registered with each other.

Next, a queue that stores the outgoing packets before they are transmitted needs to be registered with the *SlimNetDevice*. In this example, a drop tail queue is selected with a maximum queue size of 100 packets.

Finally, *SlimNetDevice* gets attached to the node before proceeding with the next node.

In addition to the “SLIM” configuration, the main function in the “`slim-example.cc`” also configures the nodes, physical layout and the mobility of these nodes, internet stacks, and echo server and client applications that will generate traffic to be carried with the *SlimNetDevice*.

Setting up applications

This scenerio simulates 20 nodes. An IPv4 stack is deployed on the nodes with IP addresses ”10.1.3.x”.

A single echo server application is deployed on node 0 and all nodes carry echo client applications pinging the echo server once every second. The server and the client apps are set-up to start working at interval t=[2-15]s.

Data Logging

The logging components can be enabled or disabled as desired. Only, the logging component for the *UdpEchoClientApplication*s is activated. This will create information about sent and received packets on each UdpClient.

Dynamic Changes to the Simulation

All nodes start with the default contention window size hence access to the channel randomly with the same frequency. In this simulation, at every second one of the nodes independently change their access to the channel.

Support

Support for Tuscarora is provide mainly (at this point) through email. Please email tuscarora-support@samraksh.com