SeaLinx Guide

Table of Contents

SeaLinx Overview

SeaLinx adopts a layer-based design with the following components

- Application layer
- Transport layer
- Network layer
- MAC layer
- Physical layer
- Core component

An upper layer program must be linked with one program in the direct lower layer, so that a packet can be routed correctly through the stack.

Core Component

The core component is responsible for dispatching data among layers. The component waits for connections from protocol stack layers. It can be started by running

sealinx-core -p <core port>

where <core port> is the port that the core is listening on.

Settings for the core can be found in the file settings.ini in the same folder as where sealinx-core is running. A sample content of the file follows

[General Settings]
MacAddress = 1
NetworkAddress = 1
PacketSize = 1024

where

- MacAddress is the MAC address of the node
- NetworkAddress is the Network address of the node
- PacketSize is the maximum packet size that is transmitted between layers.

Physical Layer

The physical layer interprets packets from acoustic modems and sends data to them. OFDM modems are handled by sealinx-ofdm-cmd. The syntax of the command is

./sealinx-ofdm-cmd -p <core port>

where <core port> is the port that the core is listening on.

There are two configuration needed for the module: config_ofdm.cfg and config_ser.cfg.

config_ser.cfg specifies the port and baud rate used to connect to a modem. The first line is the port, and the second line is the baud rate. Most of the time, failure to connect to a modem happens because of incorrect serial settings.

config_ofdm.cfg specifies the setting of the modems. Each line consists of a parameter name and its value, which are separated by a colon as follows

<variable name> : < variable value>

There are three parameters currently used:

- OFDM_ACOUSTIC_RATE: OFDM Modem Acoustic Sending Rate (bps)
  OFDM_ACOUSTIC_RATE : 2500
- OFDM_MTU: OFDM Modem Maximum Transmission Unit (bytes)
  OFDM_MTU : 1280
- OFDM_IFGT: OFDM Modem Inter Frame Guard Time (ms)
  OFDM_IFGT: 110

Mac Layer

In the given package, there are two MAC programs: sealinx-mac and sealinx-uwaloha. Sealinx-mac is a dummy MAC layer and sealinx-aloha implements the ALOHA protocol.

The syntax of sealinx-mac is

sealinx-mac -i <protocol ID> -p <core port>

where

- <protocol ID> is the identifier of the protocol, which will be filled in the type field in the MAC header.
- <core port> is the port that the core is listening on.

With this syntax, multiple MAC layers can be connected to one, given that their identifiers are different.

The syntax of sealinx-uwaloha is

sealinx-uwaloha -i <protocol ID> -p <core port> -c <aloha config file> -t <arp table file>

where

- <protocol ID> and <core port> are the same as with sealinx-mac
- <aloha config file> is the configuration file of aloha.
- <arp table file> is the file specifying the ARP table.

A sample aloha config file follows

UWALOHA_SOCK_TIMEOUT : 10
UWALOHA_ACK_TIMEOUT : 25
UWALOHA_RETX_MAX : 3
UWALOHA_BROADCAST_ADD : 99

Where

- UWALOHA_SOCK_TIMEOUT is the timeout length on reading the core socket.
- UWALOHA_ACK_TIMEOUT is the ACK time out.
- UWALOHA_RETX_MAX is the maximum number of retransmissions.
- UWALOHA_BROADCAST_ADD is the broadcast address. This is deprecated and will be removed in future releases.

An ARP table file consists of lines, each specifies a MAC address and the corresponding network address as follows

<mac address> : <net address>

Network Layer

The packet provides three network layer programs: sealinx-net, sealinx-sroute and sealinx-droute.

- sealinx-net is a dummy network layer.
- sealinx-sroute implements static routing.
- sealinx-droute implement dynamic routing.

The syntax for sealinx-net is

sealinx-net -i <protocol ID> -p <core port> -m <MAC protocol ID>

where

- <protocol ID> is the identifier of the protocol, which will be filled in the type field in the network header.
- <core port> is the port that the core is listening on.
- <mac protocol ID> is the identifier of the underlying MAC protocol.

The syntax of sealinx-sroute is

sealinx-sroute -i <protocol ID> -p <core port> -m <MAC protocol ID> -c <routing table file>

where

- The first three parameters are the same as sealinx-net
- <routing table file> is the file listing the routes in the network.

A routing table file consists of lines, each include three numbers: the source node, the relay node and the destination node in the following format:

<source node> : < relay node> : <dest node>

The syntax of sealinx-droute is

sealinx-droute-i <protocol ID> -p <core port> -m <MAC protocol ID> -c <dynamic routing config>

The first three parameters are the same as sealinx-net. The last parameter is the configuration file of dynamic routing. A sample file follows

[Protocol Parameters]
StableHelloPeriod = 180
UnstableHelloPeriod = 20
NumRoutingEntries = 10
EntryTimeout = 190
MinimumUpdateWait = 10

MaximumUpdateWait = 30

where

- StableHelloPeriod: The frequency of sending HELLO messages when the routing table is stable. A routing table is determined to be stable if at least two HELLO message from its neighbors which do not incur any update to the local routing table.
- UnstableHelloPeriod: The frequency of sending HELLO message when the routing table is unstable. A routing table is unstable after initialization or after the node receives a HELLO message from one of its neighbors, which changes the table.
- NumRoutingEntries: the maximal number of routes, which should be at least the number of nodes.
- EntryTimeOut: The validity interval of a routing entry. After a routing entry is updated, the countdown timer associated with it is reset to EntryTimeOut. If this timer times out, the entry is deleted.
- MinimumUpdateWait, MaximumUpdateWait: the minimum/maximum time that a node needs to wait before it broadcast changes to its routing table. After a change to the routing table is detected, a node set up a timer, whose interval is from MinimumUpdateWait to MaximumUpdateWait, to broadcast the changes.

Transport Layer

In the package, there is only a dummy transport layer: sealinx-tra, whose syntax is

sealinx-tra -i <protocol ID> -p <core port> -m <MAC protocol ID> -n <net protocol ID>

where

- <protocol ID> is the identifier of the protocol, which will be filled in the type field in the network header.
- <core port> is the port that the core is listening on.
- <mac protocol ID> is the identifier of the underlying MAC protocol.
- <net protocol ID> is the identifier of the underlying network protocol. Note that the network protocol must be registered with the MAC protocol.

Application Layer

In the application layer, we include a Poisson traffic generator: sealinx-tx-poi. In the sender, the syntax is

sealinx-tx-poi -i <app ID> -p <core port> -m <MAC protocol ID> -n <net protocol ID> - t <transport protocol id> -l <packet length> -r <traffic rate> -d <destination node>

where

- <app ID> is the identifier of the application, which will be filled in the service type field.
- <core port> is the port that the core is listening on.
- <mac protocol ID> is the identifier of the underlying MAC protocol.
- <net protocol ID> is the identifier of the underlying network protocol. Note that the network protocol must be registered with the MAC protocol.
- <transport protocol ID> is the identifier of the underlying network protocol. Note that the network protocol must be registered with the network protocol.
- <packet length> is the size of the packet that is sent.
- <traffic rate> is the rate of sending, in packets per second.
- <destination node> is the network address of the destination.

Note that multiple upper layer protocol can be registered with a lower layer one.

Quick Start Guide

The rest of the guide assumes that users have some experience with the Linux operating system.

Software Installation and Compilation

- Download the CRI project archive – sealinx-dev-1.0.0.0.tar.gz
- Extract the archive

```
tar xvzf sealinx- dev-1.0.0.0.tar.gz
```

- Compile the sample code

```
cd sealinx- dev-1.0.0.0.tar.gz
./configure
make install DESTDIR=`pwd`/tmp
```

- At this points, the sample programs can be found under the *tmp/usr/local/bin* folder in your working directory

Running the Protocol Stack

This section shows you how to run a sample combination of protocols: UW-Aloha, Static Routing, dummy transport layer and Poisson Traffic Generator on OFDM modems.

- Copy programs *sealinx-core, sealinx-ofdm-cmd, sealinx-uwaloha, sealinx-sroute* and *sealinx-tx-poi* from your *sealinx-2013-01-14/bin* to a directory, say *~/sealinx-bin*. Assuming you are in the project directory

```
cp sealinx- dev-1.0.0.0 /bin/sealinx-{core,ofdm-cmd,uwaloha,sroute, tx-poi} ~/sealinx-bin
```

- Copy configuration files *settings.ini, config_ser.cfg, config_net.cfg, config_uwaloha.cfg, config_arp.cfg, config _ofdm.cfg* to *~/sealinx-bin*

```
cp sealinx- dev-1.0.0.0/bin/config_{conn,ser,net,uwaloha,arp,ofdm}.cfg ~/sealinx-bin
cp sealinx- dev-1.0.0.0/bin/settings.ini ~/sealinx-bin
```

Please refer to previous pages for detailed information on how to change the settings.

- Copy the dummy transport layer that you have just compiled to *~/sealinx-bin*

```
cp sealinx-2013-01-14/sealinx-1.0.0.0/tmp/usr/local/bin/sealinx-tra ~/sealinx-bin
```

- Run the programs
  - Change your directory to *~/sealinx-bin*

```
cd ~/sealinx-bin
```

  - Run the core

```
./sealinx-core -p 12345 &
```

Make sure that port 12345 has not been used by another program. Otherwise, pick another port number.

  - Run the modem driver

```
./sealinx-ofdm-cmd -p 12345 &
```

Don’t forget the **dot** before the slash.

- Run UW Aloha

```
./sealinx-uwaloha -i 2 -p 12345 -c config_uwaloha.cfg -t config_arp.cfg &
```

- Run static routing

```
./sealinx-sroute -i 3 –p 12345 -m 2 -c config_net.cfg &
```

- Run the dummy transport layer

```
./sealinx-tra -i 4 -p 12345 -m 2 -n 3 &
```

- Run the Poisson traffic generator
  - On senders

```
./sealinx-tx-poi -i 5 –p 12345 -m 2 -n 3 -t 4 -l 200 -r 0.05 –d 1 &
```

  - On receivers or relays

```
./sealinx-tx-poi -i 5 –p 12345 -m 2 -n 3 -t 4 &
```

Note: on relays nodes, you do not need to run the application layer.

You can put all the above-mentioned commands to run SeaLinx in a script as follows:

```
#!/bin/sh
if [ $# -lt 2 ]; then
   echo "$0 <port number> <send|recv>"
   exit 1
fi
PORT=$1
TYPE=$2
./sealinx-core -p $PORT &
sleep 1
./sealinx-ofdm-cmd -p $PORT&
./sealinx-uwaloha -i 2 -p $PORT -c config_uwaloha.cfg -t config_arp.cfg &
./sealinx-sroute -i 3 -p $PORT -m 2 -c config_net.cfg&
./sealinx-tra -i 4 -p $PORT -m 2 -n 3 &
if [ "$TYPE" = "send" ]; then
   sleep 20
   ./sealinx-tx-poi -i 5 -p $PORT -m 2 -n 3 -t 4 -l 200 -r 0.05 -d 1 &
else
   ./sealinx-tx-poi -i 5 -p $PORT -m 2 -n 3 -t 4 &
fi
```

Note: The first sleep command in the above script is necessary to guarantee that the core is loaded before all other layers.