Configuration manual

This manual explains how to manually configure OpenSAND network and platform.

OpenSAND Network Configuration

OpenSAND networking configuration is detailed in this page. It provides the necessary packages and commands in order to create the networking topology used by OpenSAND. It also includes necessary networking commands for setting up external networks & routing for.

This section provides step by step instructions to properly configure each component (SAT, GW, GW-PHY, GW-NET-ACC and ST) of your OpenSAND platform. You need to install this Linux package iproute2 on each component. You can install it from your command line terminal using built-in APT package manager by typing:

```
$ sudo apt-get install iproute2
```

The variables described hereafter will be used in the rest of this section to define Linux commands to run:

- `emu_iface`: emulation network interface
- `emu_iface_ip/mask-digits`: IP address of the emulation network interface and mask
- `emu_ip/mask-digits`: IP address of the the emulation network and mask
- `lan_iface`: terrestrial network interface
- `lan_iface_ip/mask-digits`: IP address of the terrestrial network interface and mask
- `lan_ip/mask-digits`: IP address of the terrestrial network and mask
- `tap_iface`: tap interface to create to connect an IP terrestrial network
- `br_iface`: bridge interface to create to connect an Ethernet terrestrial network
- `br_iface_ip/mask-digits`: IP address of the bridge interface and mask
- `br_ip/mask-digits`: IP address of the bridge network (aka internal network) and mask
- `interco_iface`: interconnect network interface
- `interco_iface_ip/mask-digits`: IP address of the interconnect interface
- `interco_ip/mask-digits`: IP address of the interconnect network

Before running each command, you need to replace [variable] by the value (without square brackets) specific to your platform.

Note that, these variables are related to the component on which you are running the command. For example, on a Gateway, `br_iface_ip` refers to the IP address of the bridge interface of the gateway side. When necessary, the component's name will be added as prefix to the variable to avoid confusion. So, in the previous example, we will used `gw_br_iface_ip` instead of `br_iface_ip`.

Configure interfaces
<table>
<thead>
<tr>
<th>Step</th>
<th>Components</th>
<th>Mode</th>
<th>Description</th>
<th>Shell commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SAT, GW, GW-PHY, ST</td>
<td>IP or Ethernet</td>
<td>Assign an IP address to the emulation interface, then bring it up and enable forwarding.</td>
<td><code>$ ip address add [emu_iface_ip]/[mask-digits] dev [emu_iface]</code>&lt;br&gt;<code>$ ip link set [emu_iface] up</code></td>
</tr>
<tr>
<td>2</td>
<td>GW, GW-NET-ACC, ST</td>
<td>IP</td>
<td>Configure tap, bridge and lan interfaces, then bring them up and enable forwarding.</td>
<td><code>$ ip tuntap add mode tap [tap_iface]</code>&lt;br&gt;<code>$ ip link add name [br_iface] type bridge</code>&lt;br&gt;<code>$ ip address add [br_iface_ip]/[mask-digits] dev [br_iface]</code>&lt;br&gt;<code>$ ip link set dev [tap_iface] master [br_iface]</code>&lt;br&gt;<code>$ ip address add [lan_iface_ip]/[mask-digits] dev [lan_iface]</code>&lt;br&gt;<code>$ ip link set [br_iface] up</code>&lt;br&gt;<code>$ ip link set [tap_iface] up</code>&lt;br&gt;<code>$ ip link set [lan_iface] up</code>&lt;br&gt;<code>$ sysctl -w net.ipv4.conf.[br_iface].forwarding=1</code>&lt;br&gt;<code>$ sysctl -w net.ipv6.conf.[br_iface].forwarding=1</code>&lt;br&gt;<code>$ sysctl -w net.ipv4.conf.[lan_iface].forwarding=1</code>&lt;br&gt;<code>$ sysctl -w net.ipv6.conf.[lan_iface].forwarding=1</code></td>
</tr>
<tr>
<td></td>
<td>GW, GW-NET-ACC, ST</td>
<td>Ethernet</td>
<td>Configure tap, bridge and lan interfaces then bring them up and enable forwarding.</td>
<td><code>$ ip tuntap add mode tap [tap_iface]</code>&lt;br&gt;<code>$ ip link add name [br_iface] type bridge</code>&lt;br&gt;<code>$ ip link set dev [tap_iface] master [br_iface]</code>&lt;br&gt;<code>$ ip link set dev [lan_iface] master [br_iface]</code>&lt;br&gt;<code>$ ip link set [br_iface] up</code>&lt;br&gt;<code>$ ip link set [tap_iface] up</code>&lt;br&gt;<code>$ ip link set [lan_iface] up</code></td>
</tr>
<tr>
<td>3</td>
<td>GW-NET-ACC, GW-PHY</td>
<td>IP or Ethernet</td>
<td>Conigue interconnect interface then turn it up and enable forwarding.</td>
<td><code>$ ip addr add [interco_iface_ip]/[mask-digits] dev [interco_iface]</code>&lt;br&gt;<code>$ ip link set [interco_iface] up</code></td>
</tr>
</tbody>
</table>

**Configure routing**

<table>
<thead>
<tr>
<th>Step</th>
<th>Components</th>
<th>Mode</th>
<th>Description</th>
<th>Shell Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GW, GW-NET-ACC, ST</td>
<td>IP</td>
<td>Enable IP forwarding</td>
<td><code>$ sysctl -w net.ipv4.ip_forward=1</code>&lt;br&gt;<code>$ sysctl -w net.ipv6.conf.all.forwarding=1</code></td>
</tr>
</tbody>
</table>
This manual explains how to manually configure the OpenSAND testbed by means of configuration files in the `/etc/opensand/` folder.

**Configuration of OpenSAND Platform**

The `topology.conf` file is an XML file containing information used to create sockets to emulate satellite carriers as well as some basic routing. The root component should be a `<configuration component="topology">` and the relevant children are detailed below:

### sarp

The `<sarp>` component is used for basic routing and maps a destination MAC address to an OpenSAND entity ID. Two children are of relevance:

- `<default>` holds an entity ID that will receive the packets whose destination is not found in the table;
- `<ethernet>` holds, as children, the mapping between a MAC address and an entity ID. The provided MAC address must be the address of the bridge/TAP interface created using the "opensand-network" tool.

**Example of section:**

```xml
<sarp>
  <!-- The default destination terminal if no one is found, -1 for none -->
  <default>-1</default>
  <ethernet>
    <!-- The broadcast MAC address -->
    <terminal_eth mac="ff:ff:ff:ff:ff:ff" tal_id="31"/>
    <!-- IPv6 multicast (** for any Byte) for Neighbour Discovery -->
    <terminal_eth mac="33:33:33:33:33:33" tal_id="31"/>
    <!-- MAC for IPv4 multicast -->
    <terminal_eth mac="01:00:5E:33:33:33" tal_id="31"/>
    <terminal_eth mac="f2:3b:0f:96:4e:68" tal_id="1"/>
    <terminal_eth mac="06:12:c6:13:c7:cc" tal_id="0"/>
  </ethernet>
</sarp>
```

The terminal ID 31 is reserved for broadcast/multicast addresses.

### sat_carrier

The `<sat_carrier>` component holds the information to emulate the physical carriers and group
them into spots. All children must be <spot> components that specify, as arguments, the spot ID and the associated gateway ID. These <spot> components contain a <carriers> child that lists eight <carrier> children defining the following attributes:

- **id**: a numerical value, each <carrier> should have a different id than any other <carrier>;
- **type**: either ctrl_out, ctrl_in, logon_out, logon_in, data_out_st, data_in_st, data_out_gw, or data_in_gw. All eight have to be defined in a <carriers> component. The nomenclature in/out is thought relative to the satellite (*_in is data coming from entities to the satellite, *_out is data going from the satellite to other entities);
- **ip_address**: the IP address (not network) on the emulation network where data of the given type should be sent. This address must be consistent with addresses specified when launching entities ("-a" option). Note that all type *_in carrier must specify the IP address of the satellite, type ctrl_out and data_out_st must use multicast addresses and logon_out and data_out_gw must specify the address of the gateway whose ID has been passed to the <spot> component;
- **is_multicast**: true for ctrl_out and data_out_st, false for the rest.

Example of section:

```xml
<sat_carrier>
  <spot id="1" gw="0">
    <carriers>
      <!-- SAT to GW/ST control (CR, TBTP, SoF) -->
      <carrier id="0" type="ctrl_out" ip_address="239.137.194.221" port="55000"
        ip_multicast="true"/>
      <!-- GW/ST to SAT control (CR, TBTP, SoF) -->
      <carrier id="1" type="ctrl_in" ip_address="10.42.0.15" port="55001"
        ip_multicast="false"/>
      <!-- SAT to GW Logon -->
      <carrier id="2" type="logon_out" ip_address="10.42.0.13" port="55002"
        ip_multicast="false"/>
      <!-- ST to SAT Logon -->
      <carrier id="3" type="logon_in" ip_address="10.42.0.15" port="55003"
        ip_multicast="false"/>
      <!-- SAT to ST Data -->
      <carrier id="4" type="data_out_st" ip_address="239.137.194.222" port="55004"
        ip_multicast="true"/>
      <!-- ST to SAT Data -->
      <carrier id="5" type="data_in_st" ip_address="10.42.0.15" port="55005"
        ip_multicast="false"/>
      <!-- SAT to GW Data -->
      <carrier id="6" type="data_out_gw" ip_address="10.42.0.13" port="55006"
        ip_multicast="false"/>
      <!-- GW to SAT Data -->
      <carrier id="7" type="data_in_gw" ip_address="10.42.0.15" port="55007"
        ip_multicast="false"/>
    </carriers>
  </spot>
</sat_carrier>
```

You can have the same spot served by several gateways, but you must specify several <spot> components using the same spot ID but changing the gateway ID.

**spot_table**
The `<spot_table>` component is a map of which terminal belongs to which spot. The children of relevance are:

- `<default_spot>` holds the spot ID that will be used for terminals that are not specified otherwise;
- `<spot>` maps a spot ID, provided as parameter, to a list of terminals, provided in the `<terminals>` child.

Example of section:

```
<spot_table>
  <spot id="1">
    <terminals>
      <tal id="1"/>
      <tal id="3"/>
    </terminals>
  </spot>
  <default_spot>1</default_spot>
</spot_table>
```

**gw_table**

The `<gw_table>` component is a map of which terminal belongs to which gateway. The children of relevance are:

- `<default_gw>` holds the gateway ID that will be used for terminals that are not specified otherwise;
- `<gw>` maps a gateway ID, provided as parameter, to a list of terminals, provided in the `<terminals>` child.

Example of section:

```
<gw_table>
  <gw id="0">
    <terminals>
      <tal id="1"/>
      <tal id="3"/>
    </terminals>
  </gw>
  <default_gw>0</default_gw>
</gw_table>
```