Jose L. Muñoz, Juanjo Alins, Oscar Esparza, Jorge Mata

Introduction to Networks: Practices

Transport Control i Gestió a Internet (TCGI)
Universitat Politècnica de Catalunya (UPC)
Dp. Enginyeria Telemàtica (ENTEL)

Contents

Chapter 1

SWITCHING

1.1 Tools for the Practice

1.1.1 Send LLC1 Frames

`send-frame-LLC1.py`

The `send-frame-LLC1.py` application serves to send an LLC1 frame.

The application has to be run as root. This is because only the super user can launch applications that use LLC. To do so, type `sudo` and then the name of the application that to be run as root:

```
telematic@phyhost:~$ sudo send-frame-LLC1.py
Help of send-frame-LLC1.py v0.3 by JLM
     [-h, --help]
     -d, --dst_mac <destination mac>
     [-o, --outif] <ouput interface>  by default eth0
     [--ssap  <ssap>]                 by default 0x88 (use only even SAPs)
     [--dsap  <dsap>]                 by default 0x88 (use only even SAPs)
```

Example:

```
telematic@phyhost:~$ sudo send-frame-LLC1.py -d aa:bb:cc:11:22:33
```

Type a playload (text) and type enter to send the frame.

1.1.2 LLC1 Chat

For the following practices, we will use a simple chat client and chat server that uses LLC1 and that we have developed in the Python language. To properly execute "our chat service" you have to start first the server and after that you can start the client.

`server-chat-LLC1.py`

Our server application is called `server-chat-LLC1.py` and has to be run as root. This is because only the super user can launch applications that use LLC. To do so, type `sudo` and then the name of the application that to be run as root.

```
telematic@phyhost:~$ sudo server-chat-LLC1.py -h
Help of server-chat-LLC1.py v0.3 by JLM
  [-h, --help]
  [-i, --iface]  <listening interface> by defalut eth0
  [--sap  <sap>] <listening SAP>       by default 0x88 (use only even SAPs)
```

As you can observe, the server has no mandatory parameters. If you execute it with the default parameters, the server will receive and process Ethernet frames arriving to `eth0` that contain the LLC1 protocol with a destination SAP equal to 0x88:

```
root@phyhost:~# server-chat-LLC1.py
Listening on interface:  eth0
Listening on sap:  0x88
Waiting for the client...
```

The server process then, waits until a client sends an LLC1 frame. You can kill the server while it is running typing CRL+c. We must remark that you can use other parameters to start the server but the SAP must be always even. For example:

```
root@phyhost:~# server-chat-LLC1.py -i eth5 --sap 0x64
Listening on interface:  eth5
Listening on sap:  0x64
Waiting for the client...
```

View Interfaces

You can view the network interfaces of a Linux system with the `ifconfig` command.

```
root@phyhost:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:23:ae:1c:51:29
          inet addr:192.168.10.142  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::223:aeff:fe1c:5129/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:228688 errors:0 dropped:0 overruns:0 frame:0
          TX packets:121869 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:307726060 (307.7 MB)  TX bytes:8943824 (8.9 MB)
          Interrupt:22 Memory:f6ae0000-f6b00000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:841 errors:0 dropped:0 overruns:0 frame:0
          TX packets:841 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:73265 (73.2 KB)  TX bytes:73265 (73.2 KB)

wlan0     Link encap:Ethernet  HWaddr 00:21:6a:35:33:ac
          inet addr:192.168.10.131  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::221:6aff:fe35:33ac/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1462 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:169790 (169.7 KB)  TX bytes:11789 (11.7 KB)
```

client-chat-LLC1.py

After the server has been started, you can start the client. The client application is called `client-chat-LLC1.py` and it has to be run as root too. The client uses as default output interface `eth0` and 0x88 as source and destination SAP. However, it is mandatory that you specify the MAC address of the server. You can view the help menu of the client typing:

```
root@phyhost:~# client-chat-LLC1.py -h
Help of client-chat-LLC1.py v0.3 by JLM
     [-h, --help]
     -d, --dst_mac <destination mac>
     [-o, --outif] <ouput interface>  by default eth0
     [--ssap  <ssap>]                 by default LLC1 0x88 (use only even SAPs)
     [--dsap  <dsap>]                 by default LLC1 0x88 (use only even SAPs)
```

For example, to start the client with default parameters that connects to a server with MAC 00:23:ae:1c:51:29 type:

```
root@phyhost:~# client-chat-LLC1.py -d 00:23:ae:1c:51:29
Type Text: hello server!
.Waiting for the server...
```

Another example using other parameters:

```
root@phyhost:~# client-chat-LLC1.py -d 00:23:ae:1c:51:29 --ssap 0x54 --dsap 0x64
Type Text: hello server!
.Waiting for the server...
```

Operation

As you can observe, the client process expects that you type text. This text is used as payload of an LLC1 frame that is sent to the server using the specified SAPs and the MAC destination address provided in the command line. Then, the

server process receives the frame and displays the content. Meanwhile, the client process is waiting for data to come from the server. Next, the user on the server process is prompted to introduce some text that will be sent to the client process and so on.

```
root@phyhost:~# server-chat-LLC1.py
Listening on interface:  eth0
Listening on sap:  0x88
Waiting for the client...
Remote Text: hello server!
Type Text: hello client!
.Waiting for the client...
```

Some remarkable aspects of this simplified chat are that:

- The client is who always starts sending data and then, the communication is always alternate between the server and the client. In other words, type one sentence in each host each time.
- Another particularity of the client/server communication of our chat is that it requires user interaction in the server. We must stress that this is not a typical behavior of servers.
- Finally, you will need to start another server (with another LLC1 SAP) if you want to create a chat with another client because our server is not multi-client.

1.1.3 `simctl`

On the other hand, for the latest exercises we use an emulated scenario with virtual equipment (that works exactly the same as physical equipment). The simulation scenario that we will use is in Figure 1.1. To start this scenario, execute on your **phyhost** the following command:

```
telematic@phyhost:~$ simctl switching-vlan start
```

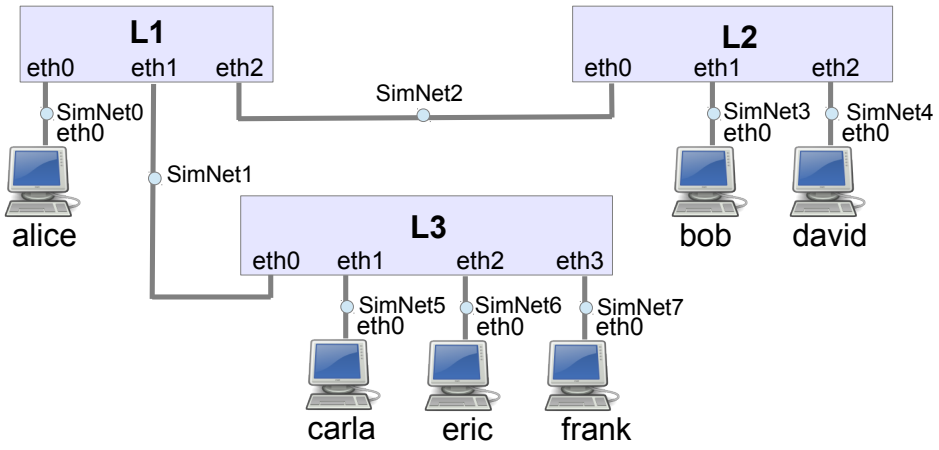


Figure 1.1: Scenario for testing Switching

It is very important that you run the simulation with your unprivileged user (**never with root!**). Starting the scenario might be slow so be patient. A machine might spent some time while booting. Then, it might appear a message telling us to retry, continue or abort. **Type always retry (r)**.

The simulation is started when a message telling us the total time elapsed appears:

```
telem@phyhost:~$ simctl switching-vlan start
......
Total time elapsed: 294 seconds
```

Then, you can get a console in any of the machines of the scenario with the get option. Example:

```
telem@phyhost:~$ simctl switching-vlan get alice
```

In the console that appears after typing the previous command type enter and then, login in the virtual machine with user root and password xxxx.

When you finish the exercise or if you need to shutdown your physical host **do not forget to stop the scenario**:

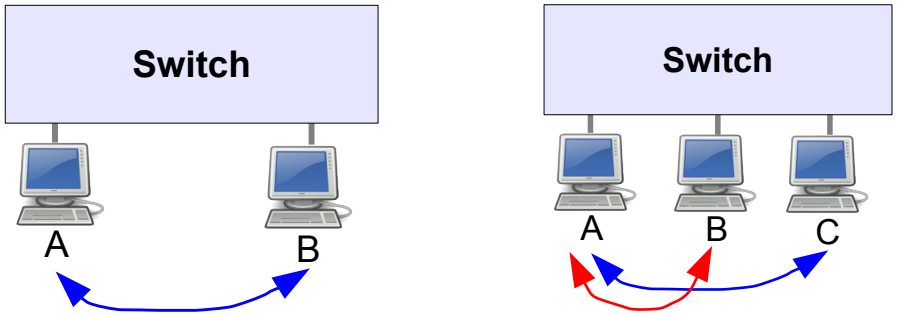
```
telem@phyhost:~$ simctl switching-vlan stop
```

If you have trouble starting or stopping the scenario, type CRL+c, then:

```
telem@phyhost:~$ simctl forcestop
```

And reboot the physical host.

1.2 Practices



(a) One Chat. (b) Two Chats.

Figure 1.2: LLC1 Chats.

Exercise1– The goal of this practice is to test transmissions of LLC1 over Ethernet with physical equipment (your physical host and switches).

1. Create a chat between two hosts using the parameters that you consider necessary (as shown in Figure 1.2a).

   Capture traffic with `wireshark` on your physical interface, exchange 5 o 6 text messages in the chat and explain the protocol fields of the traffic captured (MAC addresses, LLC fields, SAPs, etc.). Explain also how the chat works.

2. Using three hosts, create two chats as shown in Figure 1.2b.

   Explain the parameters that you have used.

Exercise2– The goal of this practice is to test a LLC over Ethernet transmission with emulated equipment (hosts and switches). In particular, we will use the simple LLC1 client/server chat. Since you are root on the virtual machine you can simply execute them as:

```
root@alice:~# client-chat-LLC1.py -d fe:fd:00:00:02:00
root@bob:~# server-chat-LLC1.py
```

1. Using **alice** and **bob**, create a chat between these hosts as shown in Figure 1.2a using the parameters that you consider necessary.

   Capture traffic with `wireshark` in `SimNet2` in the **phyhost** and set the display filter: `!ipv6 and !igmp`

   Exchange 5 o 6 text messages in the chat and explain the protocol fields of the traffic captured (MAC addresses, etc.).

2. Using **alice**, **bob** and **carla**, create two chats as shown in Figure 1.2b.

   Explain the parameters that you have used.

Exercise3– The goal of this practice is to study switching and the MAC learning process of switches using the emulated scenario. To do so, we are going to use a simple application called `send-frame-LLC1.py` that you can find in the directory /tmp. This application can be used to send LLC1 frames (use the -h option to see how it works).

1. Get consoles in **L1**, **L2** and **L3**. These are Linux boxes used to build switches with the brctl utility. Using the brctl commands show and showmacs previously explained detail the current configuration and state of each switch.

2. In **L1** type the following command:

   ```
   root@L1:~# watch -n 5 brctl showmacs br1
   ```

   The previous command shows the MAC table of the switch every 5 seconds. Remember that the MAC table contains the MAC addresses learned by the switch. On the other hand, the ageing in the switches is set to 1 minute.

   Capture in `SimNet1` and `SimNet2` and send a frame from **alice** to **bob** using `send-frame-LLC1.py`.

   Explain the traffic captured in each Tap and the time evolution of the MAC addresses of **alice** and **bob** (only these) in the MAC table of **L1**.

3. Wait for a minute and repeat the experiment sending:
   - One frame from **alice** to **bob**.
   - Before 60 seconds of the previous transmission, send another frame from **alice** to **bob**.
   - Before 60 seconds of the previous transmission, send another frame from **bob** to **alice**.

   Explain the traffic captured in each Tap and the time evolution of the MAC addresses of **alice** and **bob** (only these) in the MAC table of **L1**.

4. Increase the ageing in all the switches to 10 minutes. Using **alice**, **bob** and **carla**, create two chats as shown in Figure 1.2b. Capture traffic in `SimNet2` and `SimNet1`, exchange 5 or 6 text messages in each chat and explain the traffic that you observe in each Tap.

Exercise4– The goal of this practice is to study and test VLANs. In the first exercise you have to configure a port level VLAN. In the second exercise, you have to configure switching between tagged and untagged frames. Finally, in the last exercise, you have to configure switching between trunks.

1. In **L3**, modify the bridge configuration (removing the previous one) to create one VLAN for the ports 1 and 3 and another VLAN for the ports 2 and 4.

   Explain your configuration. Do you need any VLAN id? Test your configuration with `send-frame-LLC1.py` and the L2 broadcast address.

2. Create a configuration in the switches in which all the hosts send untagged frames but:
   - **alice** and **carla** belong to one L2 network (use VLAN id 10).

- **bob**, **david**, **eric** and **frank** belong to another L2 network.

Explain your configuration.

Capture traffic in the interfaces `SimNet0` and `SimNet1`. Then, test your configuration appropriately using the L2 broadcast address and `send-frame-LLC1.py`. Discuss the results and the format of the VLAN frames.

3. Create a configuration in the switches in which all the hosts send untagged frames but they belong to the following VLANs:
   - VLAN 10: **alice**, **bob** and **carla**.
   - VLAN 20: **david**, **eric** and **frank**.

   Explain your configuration.

   Capture traffic in the trunk interfaces (`SimNet1` and `SimNet2`) and in any other `SimNet` interface that you consider useful. Then, test your configuration appropriately using `send-frame-LLC1.py` and the L2 broadcast address. Discuss the results and the format of the VLAN frames.