REDES E SISTEMAS AUTÓNOMOS

AUTONOMOUS NETWORKS AND SYSTEMS

PRACTICAL GUIDE 1 – IPFS

Objectives

- Set up IPFS emulator
- Emulate the content distribution with different file sizes.
- Observe the content distribution with static and mobile datasets
- Analyse metrics and log to understand the different events
- Compare the performance of the distribution in a mobility environment with different protocols (IPFS and Bittorrent)

Duration

2 weeks

1. Download and install VirtualBox

1.1. Download and install VirtualBox:
https://www.virtualbox.org/wiki/Downloads

1.2. Download the virtual machine (VM) to be used in this practical guide from:
https://filesender.fccn.pt/?s=download&token=74a6c706-6ac2-4704-801b-e92d9116eadf

1.3. Import the VM **Labcom.ova** in VirtualBox.

1.4. Launch the VM and test it. User is "labcom" and password is "labcom".

2. EmuCD Introduction

2.1. EmuCD is a distributed emulator where each network node is built in a container that consumes a datatrace (real or from simulation) that defines the node's mobility and connectivity. The nodes interact directly with the container's operating system, updating the network conditions at each step of the emulation. In this way, the emulator allows the development and testing of protocols like IPFS. More info in
https://www.mdpi.com/1999-5903/12/12/234

2.2. When running the emulation, all the containers are running in the same bridge called "simipfs". Run the following to observe it:
`docker network ls`
Run "ip a" to verify that the bridge name in the OS level is
`br-04894679da29`
Annotate what is the network range used by the bridge. The containers will obtain IPs from that LAN.

2.3. The emulator working directory is
`~/emucd_ipfs/`

2.4. The datasets to be used by the emulation are stored in:
`~/emucd_ipfs/dataset/`

2.5. The docker containers are built in the folder:
`~/emucd_ipfs/docker/`

2.6. And the directory to launch the emulator and plot output metrics is:
`~/emucd_ipfs/sim_manager/`

NOTE

Download from **e-learning** the following support files and copy to the VM to the folder
`~/emucd_ipfs/sim_manager/`:
- app2.py
- plotMetrics2.py

3. EmuCD first test

3.1. In order to run each test, you need <u>always</u> to follow this procedure:
- Go to the datasets directory and run the following to create the input for the emulator (example using "1node.csv" dataset):
NOTE: We need to force remove all the files every time, so there are no conflicts with the previous run.

```
cd ~/emucd_ipfs/dataset/
rm -rf out/
python3 invertData.py 1node.csv
```

- Then navigate to the docker folder and build the containers:

```
cd ~/emucd_ipfs/docker/
rm -rf src/
./build.sh
```

- Go to the manager folder and launch the emulator

```
cd ~/emucd_ipfs/sim_manager/
python3 app2.py
```

Finally, open Firefox and navigate to **http://localhost:5050/**

3.2. Start the simulation pressing "Start". The simulation takes 30sec to start.

3.3. Follow the progress both in the browser and in the terminal
- The simulation with this dataset runs for 10min;
- In the first minute (60sec) you notice that the onboard unit - OBU (node 102 blue) is far from the road site unit - RSU (node 101 yellow). Open the 1node.csv file and check that in the first 2 lines both 101 and 102 nodes don't have neighbours;
- After 60sec, the OBU moves closer to the RSU (in the csv they are neighbours of each other now) and the content will start distributing between them.
kjn

3.4. You can stop the simulation anytime, doing CTRL+C on the terminal running **app2.py** (press just once and wait – the script will try to stop and remove the containers).

4. Plot the metrics and analyse the logs

4.1. In the "sim_manager" folder, run the following:

```
python3 plotMetrics2.py obuMetrics_2022-03-18_##:##:##
```

4.2. Observe the progress of the file transfer for the several OBUs.

4.3. How long did it take to transfer the file?

4.4. Observe the logs of both RSU (node_101.log) and OBU (node_102.log) in

```
~/emucd_ipfs/logs/
```

Find in the log when the peers are detected and when the "want" messages occur

5. Change the content file

5.1. By default, the emulator is transferring the file "file.pdf" located in the home directory to the container nodes. This file has a size of 50MB. The emulator is expecting this specific file, and so, its IPFS CID is hardcoded into the code.

5.2. To check the IPFS CID of the file, in the home directory run the following:
```
./go-ipfs/cmd/ipfs/ipfs --offline add --hash=blake2b-256 file.pdf
```

5.3. Browse to `~/emucd_ipfs/sim_daemon/` and open **app.py.** Find the CID in the code.

5.4. Rename the current "file.pdf" to "50MB.pdf" (to maintain if needed later) and create a new file with a size of 10MB:
```
head -c 10M /dev/urandom > file.pdf
```

5.5. Now, the CID needs to be changed according to the new "file.pdf" we just created, to use it in the emulation. Check the CID of the new file and change "app.py" accordingly.

5.6. Rerun the simulation with the new file (follow the exact procedure described in **3.1**) and plot the metrics.

6. Change the dataset

6.1. Rerun the simulation following the procedure described in **3.1** but now with dataset "5nodes.csv"

6.2. Plot the metrics and analyse the CSV of the dataset. Why did all the nodes connect instantly at the start?

6.3. Change the dataset so that the number of neighbours on each node is more limited (e.g. not all the nodes are in coverage of the RSU, forcing some of them to get the content from other vehicles).

6.4. Rerun the simulation with your custom dataset and observe the metrics. When did each node get the file? Analyse the log files of all the nodes and crosscheck with the "want" messages the timing each node is requesting the content.

2nd week

7. EmuCD Bittorrent version

7.1. In this step, we will explore the EmuCD Bittorrent version. In this version, each container (node) is running a Bittorrent client, instead of IPFS, for sharing the content. The "file.pdf" used for the simulation must be placed in the new "torrents" folder located in environment of this version:
```
~/emucd_bittorrent/torrents/
```

7.2. For the first test, use the file with 50MB size. In the previous folder copy "50MB.pdf" to the required name "file.pdf":
```
cp 50MB.pdf file.pdf
```

7.3. Contrary to the IPFS protocol, Bittorrent torrent is dependent on the filename. Thus, every time you rename a file, even if the content is the same, you must create the ".torrent" file:
```
ctorrent -t -u http://172.20.0.1:6969/announce -s file.torrent \
file.pdf
```

7.4. Now, repeat the procedure in **3.1** to start the simulation, but inside `~/emucd_bittorrent/` and with `app3.py`. Use the dataset "1node.csv" for this first test.

7.5. **Important note: In the Bittorrent version of the emulator, the content is pre-loaded in the RSUs, so there is no time elapsed to distribute the file into the RSUs at the beginning of the simulation. Thus, the OBUs can receive blocks from the very beginning of the simulation (contrary to the IPFS).**

8. IPFS vs Bittorrent

Download plots.py from **e-learning** and add it to the VM home directory. You can use the script to generate plots from two separate simulations to facilitate the comparison between the performance of each protocol.

Usage: python3 plots.py <path_to_file1> <path_to_file2>
Example: python3 plots.py
emucd_bittorrent/sim_manager/obuMetrics_2025-02-25_15:00:50
emucd_ipfs/sim_manager/obuMetrics_2025-02-25_14:50:25.csv

8.1. We will now compare the performance of IPFS and Bittorrent. Rerun both versions with the 50MB file and "5nodes.csv" dataset (remember the procedure in **5.** to set IPFS with the 50MB file). Leave the simulation running for at least 3 minutes so the output metrics file has enough points to be plotted.

8.2. From the "sim_manager" folder in both environments, plot the metrics. Compare the average time in both cases for the content to be distributed to all the nodes. Which one seems to be faster? Remember to consider the note described in **7.5.**

8.3. Run now the following combinations for both simulations (always for at least 3 minutes) and plot the metrics in the end:
- 10MB file, "aveiro_mobility.csv" dataset
- 50MB file, "aveiro_mobility.csv" dataset
Take note on the average time to complete the distribution of the nodes, and if there are cases that the content distribution is not completed.

8.4. Take conclusions about the performance of each protocol, especially in the mobility environment.

9. More about IPFS

9.1. Analyse in more detail the evolution of the last simulation of IPFS (50MB file, "aveiro_mobility.csv" dataset) with the help of the logs stored in
`~/emucd_ipfs/logs/`

9.2. There is a log file for each node. For example, open an RSU "node_106", search for a want-have message, and with the "local" and "from" fields identify which node is requiring a block (through the hash of the peer). Open the logs of the OBUs to identify which one has the hash you are looking for (local).

9.3. Extend the analysis from the previous point to discover if the nodes are getting blocks from different RSUs.

9.4. Bonus: can you find if an OBU requested the same block more than once?

Hint about the node IDs in the "aveiro_mobility.csv" dataset:
RSUs: 105, 106, 126
OBUs: 152, 160, 186, 187

9.5. To enhance your analysis, you can use the following Python script to parse the log files:
https://gist.github.com/pedrorito/898f2cc4d4e525a9a5bf2f5a7bab18b3