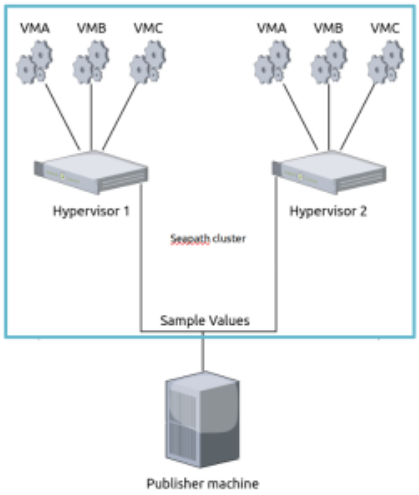
Simulating IEC61850 traffic for Seapath

Setting the context

The following document describes the integration of a IEC61850 traffic simulation with a Seapath cluster. The main idea is to generate IEC61850 Samples Value (SV) from a standalone machine (aka “publisher”), outside the Seapath cluster. The SV will be transit to the Seapath cluster, and be proceed inside one or many virtual machines by a receiver (aka “subscriber”).



The IEC61850 norm defined, in a 50Hz electric network, that each SV should be paced at a rate of 4KHz, and thus, to wait 250µs between each sent of SV (4.8KHz and 208µs in a 60Hz network). In this document, we will use the 50Hz frequency as reference.

So, to be accurate, the simulation IEC61850 traffic should be paced the most close to 4KHz between the emission of the SV on the publisher to it reception on the subscriber. Note that:

- If your pacing is too quick (<< 250µs), the subscriber will not have enough time to proceed the SV, and will result of SV dropping.
- If your pacing is too low (>> 250µs), the subscriber will be waiting between each SV sent, introducing latency in the simulation.

Requirements

Hardware

Any machines that could a run a recent Linux RT distribution can be used as a publisher machine. But for better performance, we recommend at least a 4 cores CPU, in order to have enough cores to isolate RT process. X86 or ARM CPUs can be used, but we recommend at least Intel i5 or ARM Cortex-A53 based CPUs for better performance. Note that on recent platforms, CPUs comes with implementation of “performance” and “efficient” cores, running at different speed, and so could reduce system determinism.

A 1 Gigabit network interface should be use for the SV sent, isolated from any network to avoid interference.

Software

A Real Time kernel is highly recommended to increase the publisher determinism. Tests with a Debian 12 distribution with an RT kernel showed good performance.

For traffic generation, we recommend to use the package **Bittwist**: https://github.com/ayeowch/bittwist

You can also use the package **Tcpreplay** (https://github.com/appneta/tcpreplay), but note that tests showed that Bittwist gives more performance over Tcpreplay in a RT context.

For increasing determinism, it is highly recommended to isolate on a specific core:

- The IRQ interface process used to send SV. If your network interface provides many TX queues, we recommend to pin, if possible, one core per IRQ process.
- The bittwist process, with the lowest process priority.

For example, to start traffic generation, on a specified core, with lowest priority, you can use following command:

```
taskset -c <CORE> chrt --fifo 1 bittwist -i <INTERFACE> <PCAP>
```

TODO: Add a part on PCAP generation