Appendix A: Input and Output files specifications

Your program needs to consider the following input and output files, similarly to the Open-Timaeus Net tool that will be used for the integration phase (as explained in Appendix B).

Input:

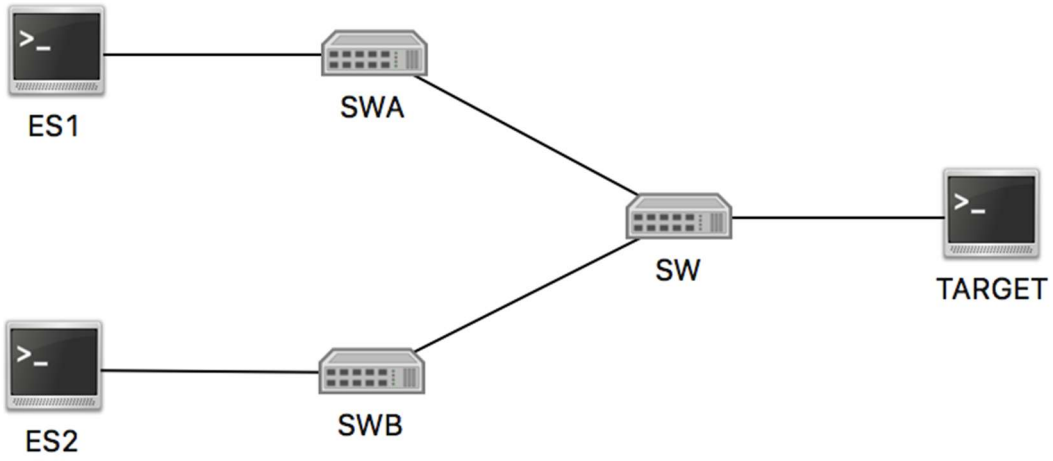
- The input is a single xml file (unified format)
- This single file is containing both the network architecture, and the traffic
- See samples at the end of this document.

Output:

- The output is an xml file, containing the main results:
  - End to End Delays
  - Jitters
  - Backlogs & Delays per port
  - Load

Input XML File

The unified input file is a single file gathering all information; architecture, traffic, constraints, etc. We illustrate this input on the following network:



The file contains a set of « xml nodes »

```
<?xml version="1.0" encoding="UTF-8"?>
<elements>

  // here some nodes !

</elements>
```

Network Node

One xml node is describing the main network properties :

- The name of the network : it's optionnal ; if no name is provided, then the name of the file is used as the network name
- The « global » overhead for each flow ; if no overhead is provided, 0 is used.
- The shortest path policy : DIJKSTRA is the default ; possible shortest Path are:
  - DIJKSTRA (Dijkstra A*)
  - X_Y (matrix rooting, first X then Y)
  - Y_X (matrix rooting, first Y then X)
  - NONE (no rooting is performed)
- The technology in the Open-Timaeus-Net sense : can be AFDX, Ethernet, SpaceWire, etc.
  - Each Open-Timaeus-Net packing supports a limited set of technologies
- The global transmission capacity
  - Default transmission capacity unit is the bit per second (bps)

Example :

```
<network name="Z_2ES_SE" overhead="67"
    shortest-path-policy="DIJKSTRA" technology="Ethernet"
    transmission-capacity="100Mbps" />
```

Station Node

One xml node per station

- Each station has a name
- Each station has a service policy ; possible service policy are :
  - FIRST_IN_FIRST_OUT
  - STATIC_PRIORITY
- Each station may have its own transmission capacity
  - Default transmission capacity unit is the bit per second (bps)

Example :

```
<station name="ES1" service-policy="FIRST_IN_FIRST_OUT"
    transmission-capacity="100Mbps" />
```

Switch Node

One xml node per switch

- Each switch has a name
- Each switch has a service policy
  - FIRST_IN_FIRST_OUT
  - STATIC_PRIORITY
- Each switch has a technological latency
  - Default latency unit is the micro second (µs)
- Each switch has a switching technique (switching-technique) ; available techniques are:
  - STORE_AND_FORWARD,
  - CUT_THROUGH,
  - WORM_HOLE
- Each switch may have its own transmission capacity
  - Default transmission capacity unit is the bit per second (bps)
- Each switch can have a redundancy mode ; this mode can be:
  - MAIN (the default)
  - BACKUP

Example :

```
<switch name="SW" service-policy="FIRST_IN_FIRST_OUT"
    tech-latency="60" transmission-capacity="100Mbps"/>
```

Link Node

One xml node per link

- Each link has a name, a source (from) and a destination (to)
- Ports are mandatory
- Each link may have its own transmission capacity
  - Default transmission capacity unit is the bit per second (bps)

Example :

```
<link name="AFDX Edge 1"
from="ES1" fromPort="0" to="SWA" toPort="0" transmission-capacity="100Mbps"/>
```

Flow Node

One xml node per flow

- Each flow has a name
- Each flow has a source
- Each flow has a pay-load : pay-load min is optionnal
  - Default payload unit is the byte
- Each flow has a period and a jitter
  - Default period and jitter unit is the milli second (ms)
- Each flow has a deadline (default is the period)
  - Default deadline unit is the milli second (ms)
- Each flow can have a priority (used for STATIC_PRIORITY policy)
  - Priority value depends on the technology priority model ; for instance, for AFDX, it can be “Low” or “High”
- Each flow has a set of targets : path can be explicited, as in the following example ; else paths will be computed by Open-Timaeus-Net based on a shortest path algorithm.
  - Each target can have a redundancy mode ; this mode can be:
    - MAIN (the default)
    - BACKUP

Example :

```
flow name="F1" source="ES1"
    max-payload="1000"
    min-payload="1000"
  period="1" deadline="1" jitter="0"
  priority="Low" >
     <target name="TARGET" mode="MAIN">
       <path node="SWA"/>
       <path node="SW"/>
       <path node="TARGET"/>
     </target>
</flow>
```

Available Units and conversions

Each attribute has a default unit. For instance, default transmission capacity unit is the bit per second (bps).
You might specifiy another unit, and then Open-Timaeus-Net will perform the conversion.
This is supported only for time and traffic units.
Available traffic units: (pay attention to the case!)

- bps
- Mbps
- Gbps

Available time units:

- s
- ms
- µs

The output XML File

The output file contains a set of « xml nodes »

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
    <delays>
        // delays nodes
    </delays>
    <jitters>
        // jitters nodes
     </jitters>
    <backlogs>
        // backlogs nodes
    </backlogs>
    <load>
        // load nodes
    </load>
</results>
```

Delay node: it contains the End-to-end delay for each flow, then for each target. Delay default unit is the µs. Example:

```
<flow name="Flow1">
        <target name="ES2" value="323"/>
</flow>
```

Jitter node: it contains the jitter value for each flow, then for each target. Jitter default unit is the µs. Example:

```
<flow name="Flow1">
        <target name="ES2" value="141"/>
</flow>
```

Backlog node: it contains the backlog value for each switch, then for each port. It contains also a switch consolidation (consolidation of all the ports). Additional information can also be set like the local delay on each port or the buffer size of the switch. Backlog default unit is the byte. Example:

```
<switch name="Switch 1">
  <port num="2" backlog="100" delay="146" />
  <total backlog="100" buffer="1000" percent="10%" />
</switch>
```

Load node: it contains for each link (edge), the traffic in bits and the ratio Vs the link capacity. In case of full-duplex network, two nodes are provided (one for ech sense). Example:

```
<edge name="Edge1">
  <usage type="direct" value="8536000" percent="8.5%" />
  <usage type="reverse" value="853600" percent="0.85%" />
</edge>
```