AFDX Network Simulation in PtolemyII

1 Summary

The following document is a short documentation to explain how you can use PtolemyII's actors to simulate a (full-duplex) AFDX network in a distributed application. The important concepts in an AFDX network are: End-System (ES) or "daughter board" (connected to the CPIOM board of a computer), AFDX switches (SW) and virtual links (VL). The transmission of Ethernet frames in a VL queue is scheduled by the ES's Virtual Link Scheduler, as depicted in figure 1. The AFDX switch has a static routing table which specifies for each VL the output(s) port(s). In practice, there is an implicit routing based on the topology described by the nodes and switches. More details about AFDX network can be found in [AFDX12, GEIP12, TECH08]. Demos for AFDX network simulation can be found in ptolemy/actor/lib/qm/demo/AFDX. The actors implemented can be found in the Vergil's library under `MoreLibrairies->QuantityManager`: AFDX_ESs, AFDX_SW and VL.

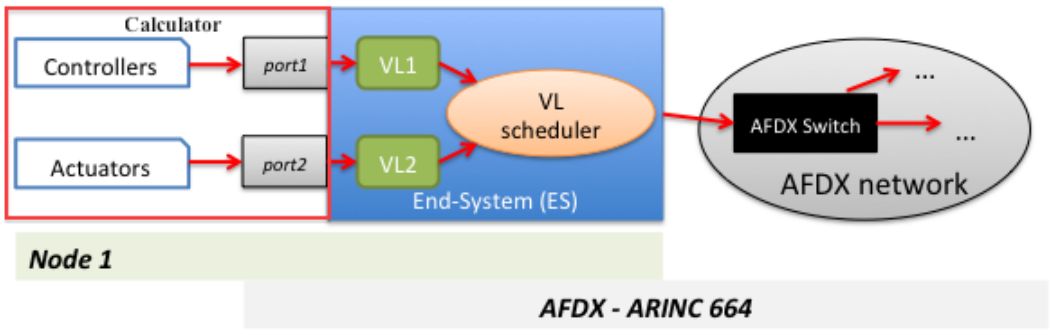


Figure 1: AFDX concept, devices and network

2 The AFDX simulator

2.1 Hypothesis based on the AFDX specification

Let us consider N a node (or computer), Nt an transmitting node and Nr a receiving node. We consider the followings hypothesis:

1. There is only one AFDX network, even if AFDX specifications considers (theoretically) that there are two redundant networks,
2. Each node (e.g. a Ptides platform or a DE composite actor) is connected to (only) one End-System (ES), and each ES is connected to only one switch (SW); but a same SW can be connected to several ESs.
3. Some SW(s) are directly connected to a ES, other SW(s) are in the "network heart" (only connected to other SWs).
4. A VL can link one Nt to one Nr (unicast) or to several Nr(s) (multicast). In the multicast case, each Nr node can be connected to distinct SW, and the path is given by a tree.
5. A VL has a source SW (connected to its ES), zero or more intermediate SW(s), one or more final SW(s) (one if it is an unicast, more if it is a multicast).
6. There can be several paths between two nodes (2 ESs), but a VL is a static path chosen by the designer.
7. The delay introduced by the traffic shaper of the end-system is considered as negligible.
8. The latency of the network physical link is negligible compared with the latency introduced by the AFDX devices (switches and end-systems).

2.2 Components of the AFDX network

There are three components, all found in `MoreLibrairies -> QuantityManager`:

- VL actor (see Figure 2.c), with the following parameters: `vlink` (name), `bag` and `trameSize` (size of frame); the parameter `schedulerMux` is the name of the ES (see Figure 2.d).
- AFDX_ESs, with a parameter `bitRate`, the bit rate of the traffic shaper of the end system. It simulates all the end systems of the AFDX network.
- AFDX_SW, with the following parameters: `inputBufferDelay`, `outputBufferDelay`, `technologicalDelay` and the `number` of ports. The default values are, respectively, 0.0, 0.0, 140us and 2 (see Figure 2.f). Add as many parameters i, $i = 0..(k-1)$ as the number of ports k: double-click on AFDX_SW, click on Add bouton. For each parameter, put the node name or the AFDX switch name the corresponding port is connected to, according with the network topology.

3 How to deploy the AFDX network

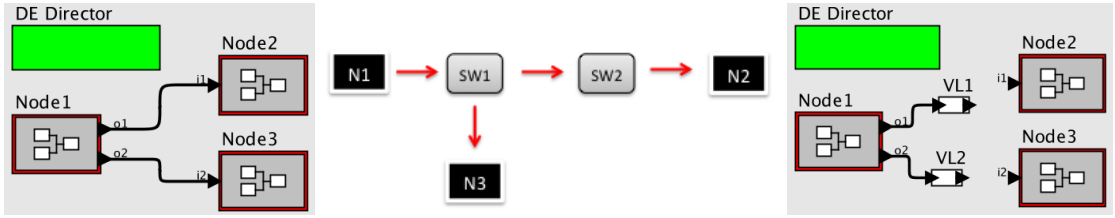
Let us consider the (functional) model of Figure 2.a), already validated as a centralized system (no latency in the transmission data). The goal is now consider that the three nodes are distributed and the data are transmitted through an AFDX network whose topology is represented in Figure 2.b. The bit rate of the traffic shaper of the end system ES1 (connected to Node1) is 100 Mbit/s. The input and output buffer delay of the switches is 0.0. Two virtual links will be used:

- VL1 (from o1 to i1, going through AFDX switches SW1 and SW2), with a bag of 32ms and size of frame of 600,
- VL2 (from o2 to i2, going through AFDX switch SW1), with a bag of 16ms and size of frame of 300.

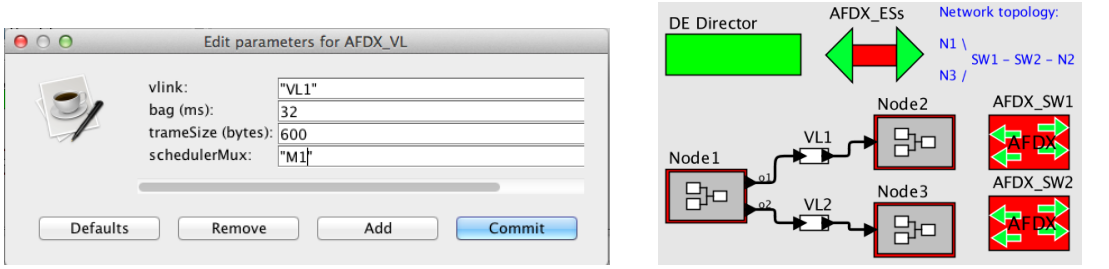
We describe here the steps required to build a model adding an AFDX network to the original model of Figure 2.a:

1. be sure there is a DE director inside each Node connected to the network in the original model. Otherwise, add a DE director;
2. for each data sent through the network (functional link), add a VL actor, see Figure 2.c;
3. put the information concerning the VLs in the VL actors (see Figure 2.d for VL1). Make the logical connections between the output ports of VL1 and VL2 and the respective inputs ports of Node2 and Node3.
4. put an AFDX_ESs quantity manager and two AFDX_SW quantity manager in the top level model. Rename the switches as AFDX_SW1 and AFDX_SW2. You can also add an annotation showing the network topology (see Figure 2.e). Choose the bit rate for AFDX_ESs. For each AFDX switch, put the right number of ports, and for each port, put the name of the node or other switch according with the topology as described in section 2.2. Figure 2.f depicts the default values for AFDX_SW1 and Figure 2.g after have changed the number of ports and have added parameters `0`, `1` and `2` with, respectively, values AFDX_SW2, AFDX_ESs and Node3. Figure 2.h shows the final values for AFDX_SW2.
5. finally, indicate at each receiving port all the quantity managers representing the virtual link, i.e. the network communication (from the sender port to the received port) by right clicking on the receiving port, `Customize->Configure/ add a parameter` and putting the name of the quantity manager actor. The parameters must be added in this order: first, the AFDX_ESs, then each switch crossed by the corresponding virtual link. In fact, the order in which quantity managers are specified is also the order in which they process the tokens from the sender (see information on receiving port of Node2 and Node3 in Figure 2.i). Notice that the color of the port is the color of the last quantity manager crossed (you can change the color of a quantity manager).

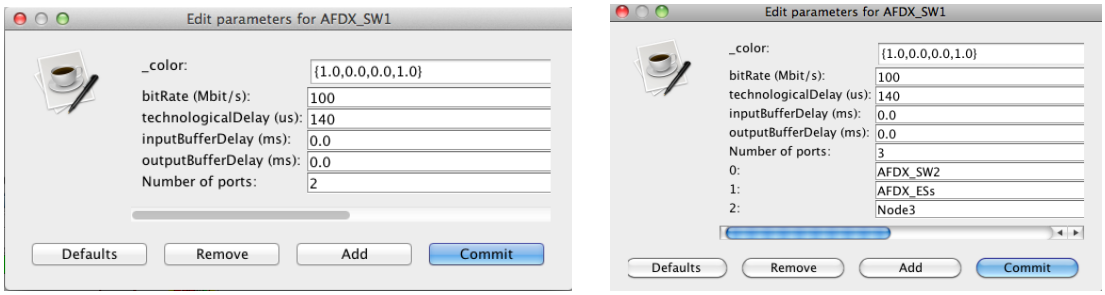
You can find more information about a quantity manager actor in [QM]. For a more complex topology, see the demo /ptII/ptolemy/actor/lib/qm/demo/AFDX/AFDX.xml.



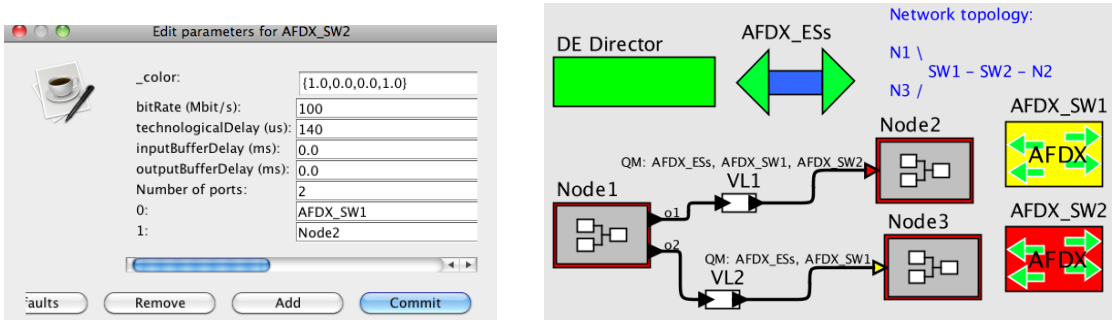
a) initial model b) network topology c) adding VL actors



d) VL configuration e) adding AFDX_ESs and AFDX_SW



f) AFDX_SW1 default values g) AFDX_SW1 final values



h) AFDX_SW2 final values i) quantity managers configuration

Figure 2: Adding an AFDX network to a functional model

4 Developper corner

The implementation of the AFDX network, based on quantity manager and parameter actors, has the following classes:

- `AFDXESs.java`: manages all end-systems required in an AFDX network.
- `AFDXSwitch.java`: simulates the behavior of an AFDX switch with respect to our hypothesis made on section 2.1.
- `AFDXVlCfg.java`: implements the virtual-link configurator used by the user to configure the different virtual-links for each node of the application.

- `AFDXVlink.java`: implements a virtual-link for an AFDX Network

For the integration and the deployment of actors in PtolemyII and Vergil you should also read the `qm.xml`, `AFDX*Icon.xml` and `makefile` files in ptolemy/actor/lib/qm.

5 Authors

- Gilles Lasnier, ISAE/DMIA, post-doctoral researcher (gilles.lasnier@isae.fr)
- Janette Cardoso ISAE/DMIA, professor (janette.cardoso@isae.fr)

The authors are from the DMIA team of the *Institut Supérieur de l'Aéronautique et de l'Espace* (ISAE), 10, avenue Edouard Belin, 31055, Toulouse, France.

References

[AFDX12] ARINC Specification. 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network `https://www.arinc.com/cf/store/catalog_detail.cfm?item_id=1270`

[GEIP12] ARINC Protocol Tutorial (GFT-640A). GE Intelligent Platforms. `http://defense.ge-ip.com/library/detail/1955`

[TECH08] AFDXARINC 664 Tutorial. techSAT. 700008_TUT-AFDX-EN_1000. 2008. `www.techsat.com/fileadmin/.../TechSAT_TUT-AFDX-EN.pdf`

[QM] Using a Quantity Manager for modeling a Network. `ptolemy/actor/lib/qm/doc-files/QuantityManager.pdf`