

Distributed Programming II

A.Y. 2016/17

Assignment n. 1 – part a)

The material for this assignment is in the *.zip* archive where you have found this file. Extract the archive to an empty directory that you will use as your working area and that we will call `[root]`.

DP2-NFFG is a (simplified) distributed system that manages *Network Function Forwarding Graphs (NF-FGs)*. The NF-FG concept and terminology used in the DP2 assignments is explained in the document file `[root]/intro.pdf`.

The Java interfaces defined in the package `it.polito.dp2.NFFG` (available under `[root]/doc` and at <https://pad.polito.it:8080/enginframe/dp2/assignments/lab1/doc/index.html>) give read-only access to some information about a set of NF-FGs and their verification process. The javadoc of the interfaces documents the kind of information that can be retrieved. The main interface, from which all the information can be accessed, is `NffgVerifier`. The methods in this interface can be used to get the information available about known NF-FGs (`getNffg` and `getNffgs`), and policies to be verified on them (`getPolicies`). These methods return sets of interfaces, by which all the available information about the returned NF-FGs and policies can be obtained.

The solution to part a) will be submitted along with the solution to part b).

Assignment description

1. Design an *XML* application that can be used to store all the information that can be retrieved by using the *Java* interfaces defined in package `it.polito.dp2.NFFG`, starting from interface `NffgVerifier`. The *XML* format must be such that all the data that can be retrieved using such interfaces can also be obtained from the *XML* document, and redundancies should be avoided. The *XML* application must be specified by means of a *DTD*, which must be saved in file `[root]/dtd/nffgInfo.dtd`, and by means of an *XML Schema*, which must be saved in file `[root]/xsd/nffgInfo.xsd`. The potential of *XML* schemas must be exploited, in order to accurately represent constraints on data types, and keep checks as much as possible within the *XML* processor (rather than in the application).
2. Write a short documentation of your design choices about the *DTD* and the schema (max 1 page) and save it as an ASCII file in `[root]/doc.txt`.
3. Write a valid *XML* file that references the designed *DTD* locally. The file must be saved as `[root]/dtd/nffgInfo.xml`.
4. Write a valid *XML* file that references the designed *schema* locally. The file must be saved as `[root]/xsd/nffgInfo.xml`.

Correctness verification

Before submitting your files, please verify their correctness. The submitted solution must at least satisfy the following requirements, in order to be considered acceptable:

- the file `nffgInfo.dtd` must be syntactically correct and the file `nffgInfo.xsd` must

be a valid XML Schema;

- the file `[root]/dtd/nffgInfo.xml` must be valid and must reference the DTD stored in file `nffgInfo.dtd`, in the same directory as the xml file.
- the file `[root]/xsd/nffgInfo.xml` must be valid with respect to the schema `nffgInfo.xsd` and must reference it, assuming it is stored in the same directory;
- the file `doc.txt` must exist.

The validity of the XML file can be checked by any XML validation program. For example, it can be checked by the Eclipse validate command or by running the `DomParseV` program (distributed in the *XML* examples bundle on the course site). Similarly, it is possible to check the validity of the XML file with respect to the schema and the validity of the schema.