Indice

Neo4j Manager Report

1 – General overview

Once the service is started, it's possible to use either XML or JSON file to create a new graph, and after it's possible to choose even the other format to retrieve that. JAX-RS, starting from the annotated classes, is able to generate the response in both formats. Anyway, the annotated classes are generated by XJC starting from nffg.xsd schema.

2 – Neo4j database

The graph is saved into Neo4j database (it's used 2.3.1 version) by mapping XML/JSON entities to some kind of nodes (Referenceable, Pathable, Endpoint, NetworkFunction, NetworkElement, ConnectionPoint, MonitoringParameter, Flowspace, Specification, CtrlInterface, Flowrules, Action, Nffg) and relationships (PathRelationship, InfoRelationship). The InfoRelationship starts from a node and points to another one that contains info not useful in the calculation of paths. Instead the PathRelationship connects two pathable nodes.

3 – An example

Starting from these equivalents XML and JSON files:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<nffg xmlns="http://www.example.org/nffg/" id="nffg_0" version="0.1 D3.1">
        <endpoints>
                <endpoint id="ep_1">
                        <flowspace ingPhysPort="10">
                                <tcp src="80"/>
                        </flowspace>
                </endpoint>
                <endpoint id="ep_2">
                        <flowspace ingPhysPort="13">
                                <tcp src="80"/>
                        </flowspace>
                </endpoint>
                <endpoint id="ep_3">
                        <flowspace/>
                </endpoint>
        </endpoints>
        <network_functions>
                <network_function id="nf_1" functionalType="firewall">
                        <specification>
                                <deployment type="KVM"/>
                                <image uri="http://www.a.org/data.img"/>
                                <cpu numCores="7"/>
                                <memory size="10MiB"/>
                                <storage size="100MiB"/>
                        </specification>
```

```
                <connection_points>
                        <connection_point id="cp_2">
                                <port id="77" direction="out" type="10GbE"/>
                        </connection_point>
                        <connection_point id="cp_1">
                                <port id="79" direction="in" type="GbE"/>
                        </connection_point>
                </connection_points>
                <control_interfaces>
                        <control_interface id="ci_1">
                                <attributes>
                                        <attribute value="tcp://127.0.0.1:5555"/>
                                        <attribute value="Netconf"/>
                                </attributes>
                        </control_interface>
                </control_interfaces>
                <monitoring_parameters>
                        <parameter value="Measure_Script"/>
                </monitoring_parameters>
        </network_function>
</network_functions>
<network_elements>
        <network_element id="ne_1" type="BiSBiS">
                <eps-cps>
                        <ep-cp id_ref="cp_2">
                                <flowrules>
                                        <flowspace/>
                                        <actions>
                                                <action type="output" port="ep_3"/>
                                        </actions>
                                </flowrules>
                        </ep-cp>
                        <ep-cp id_ref="ep_1">
                                <flowrules>
                                        <flowspace/>
                                        <actions>
                                                <action type="output" port="cp_1"/>
                                        </actions>
                                </flowrules>
                        </ep-cp>
                        <ep-cp id_ref="ep_2">
                                <flowrules>
                                        <flowspace/>
                                        <actions>
                                                <action type="output" port="cp_1"/>
                                        </actions>
                                </flowrules>
                        </ep-cp>
                </eps-cps>
```

```
                        <monitoring_parameters>
                                <parameter value="Bandwith ep_1 cp_1 100mbit"/>
                                <parameter value="Delay ep_1 cp_1 50ms"/>
                        </monitoring_parameters>
                </network_element>
        </network_elements>
        <monitoring_parameters/>
</nffg>
```

```
{
        "id": "nffg_0",
        "version": "0.1 D3.1",
        "endpoints":
        {
                "endpoint": [
                {
                        "id": "ep_1",
                        "flowspace":
                        {
                                "ingPhysPort": "10",
                                "tcp":
                                {
                                        "src": 80
                                }
                        }
                },
                {
                        "id": "ep_2",
                        "flowspace":
                        {
                                "ingPhysPort": "13",
                                "tcp":
                                {
                                        "src": 80
                                }
                        }
                },
                {
                        "id": "ep_3",
                        "flowspace": {}
                }]
        },
        "network_functions":
        {
                "network_function": [
                {
                        "id": "nf_1",
                        "functionalType": "firewall",
                        "specification":
                        {
                                "deployment":
                                {
                                        "type": "KVM"
                                },
                                "image":
                                {
                                        "uri": "http://www.a.org/data.img"
                                },
                                "cpu":
                                {
                                        "numCores": 7
                                },
                                "memory":
                                {
                                        "size": "10MiB"
                                },
```

```
                        "storage":
                        {
                                "size": "100MiB"
                        }
                },
                "connection_points":
                {
                        "connection_point": [
                        {
                                "id": "cp_2",
                                "port":
                                {
                                        "id": 77,
                                        "direction": "out",
                                        "type": "10GbE"
                                }
                        },
                        {
                                "id": "cp_1",
                                "port":
                                {
                                        "id": 79,
                                        "direction": "in",
                                        "type": "GbE"
                                }
                        }]
                },
                "control_interfaces":
                {
                        "control_interface": [
                        {
                                "id": "ci_1",
                                "attributes":
                                {
                                        "attribute": [
                                        {
                                                "value": "tcp://127.0.0.1:5555"
                                        },
                                        {
                                                "value": "Netconf"
                                        }]
                                }
                        }]
                },
                "monitoring_parameters":
                {
                        "parameter": [
                        {
                                "value": "Measure_Script"
                        }]
                }
        }]
},
"network_elements":
{
        "network_element": [
        {
                "id": "ne_1",
                "type": "BiSBiS",
                "eps-cps":
                {
                         "ep-cp": [
                        {
                                "id_ref": "cp_2",
                                "flowrules": [
                                {
                                        "flowspace": {},
                                        "actions":
                                        {
```

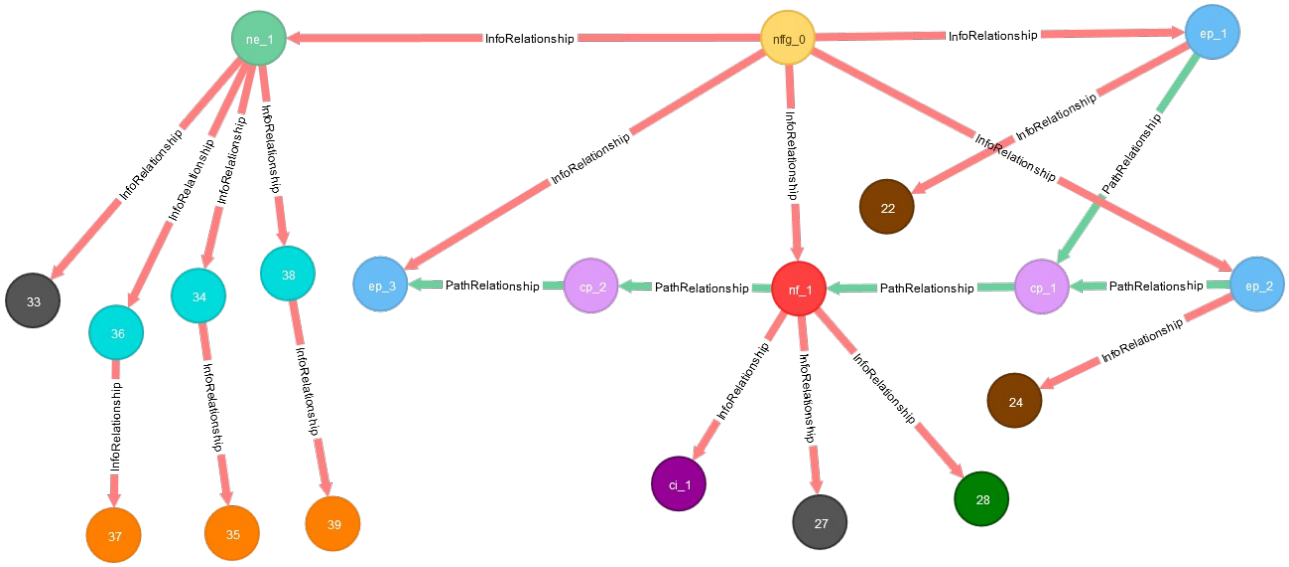
```
                                                "action": [
                                                {
                                                        "type": "output", "port": "ep_3"
                                                }]
                                        }
                                }]
                        },
                        {
                                "id_ref": "ep_1",
                                "flowrules": [
                                {
                                        "flowspace": {},
                                        "actions":
                                        {
                                                "action": [
                                                {
                                                        "type": "output", "port": "cp_1"
                                                }]
                                        }
                                }]
                        },
                        {
                                "id_ref": "ep_2",
                                "flowrules": [
                                {
                                        "flowspace": {},
                                        "actions":
                                        {
                                                "action": [
                                                {
                                                        "type": "output", "port": "cp_1"
                                                }]
                                        }
                                }]
                        }]
                },
                "monitoring_parameters":
                {
                        "parameter": [
                        {
                                "value": "Bandwith ep_1 cp_1 100mbit"
                        },
                        {
                                "value": "Delay ep_1 cp_1 50ms"
                        }]
                }
        }]
    },
    "monitoring_parameters": {}
}
```


we obtain the following representation into the database. The query to get it is:

match (n {nffgId: 'nffg_0'})-[]-(r) return n, r

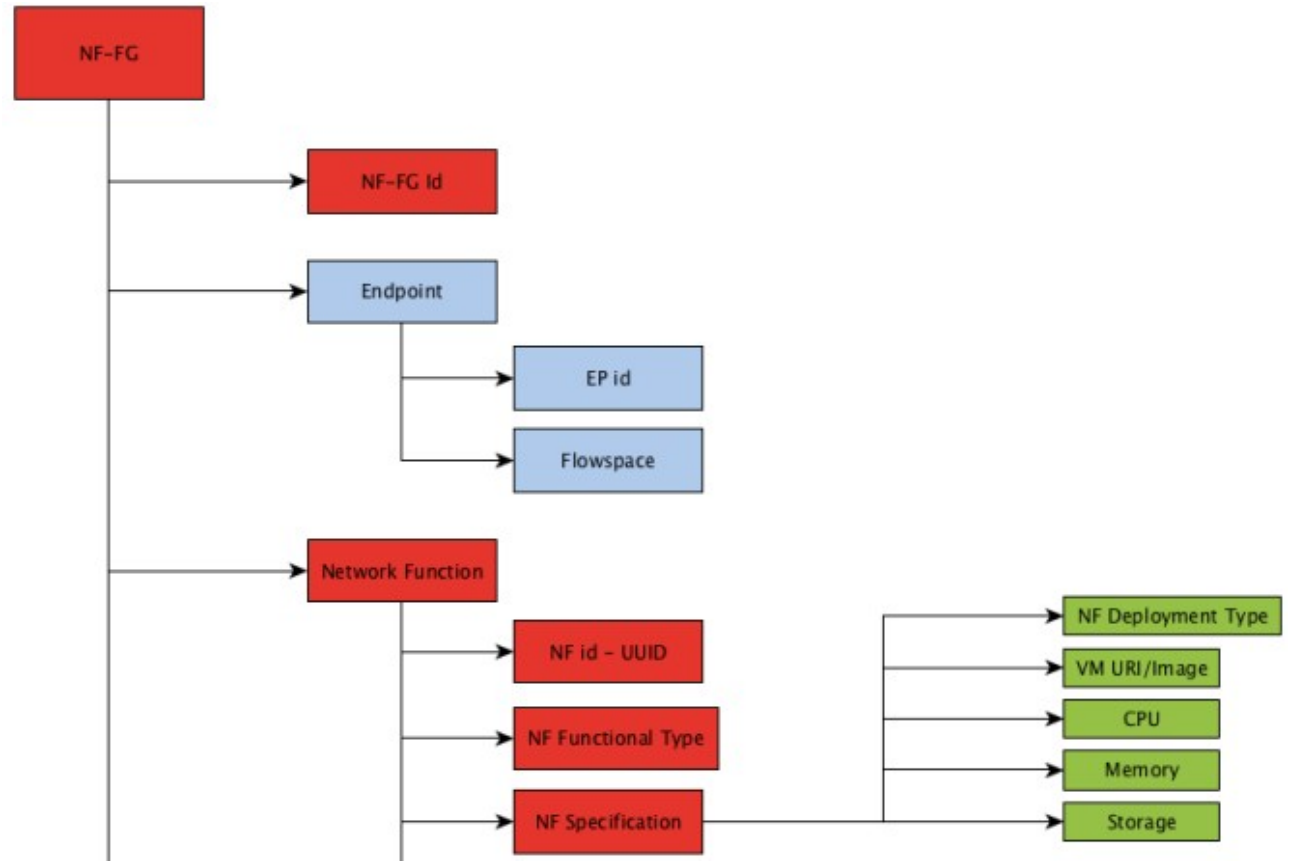
while the one to get the complete content of the database is:

match (n) return n



Yellow node is a nffg element that is connected to all network_function (red) nodes, to all endpoint (blue), to all network_element (light green) nodes and eventually to its monitoring_parameters (black) nodes. The cyan nodes are flowrules elements, the orange ones are action elements, the light purple ones are connection_points elements, the purple ones are control_interface elements, the green ones are specification elements and finally the brown ones are flowspace elements.

4 – Schema structure



NF-FG Id, EP Id, NF Id, CP Id, port Id, CI Id and NE have to be unique within a certain graph. The endpoint is an external reference point (or interface) of the service (or the NF-FG) to the existing network. The NF represents the compute element that performs the Network Functionality. The "connection point" (CP) is the external reference point of each NF element, which allows describing how the NF element is connected to other elements in the NF-FG (i.e. other NFs or Endpoints). Finally the Network Element (NE) is an abstract representation used to describe the interconnection between the different elements in the NF-FG (e.g. endpoints and/or connection points) as a virtualized SDN Forwarding Element. Therefore, it describes the interconnection between the NFs and represents the networking resources of the NF-FG. The endpoints are also connected to them, being the incoming and outgoing reference points of the NF-FG. The NE is used to abstract the whole interconnection as a Big Switch and uses the flow-rules to describe the

connections between the Nfs. The PathRelationship is obtained from the element 'action' contained in element 'flowrules', that has to be read in this way:

```
<ep-cp id_ref="cp_2">
        <flowrules>
                <flowspace/>
                <actions>
                        <action type="output" port="ep_3"/>
                </actions>
        </flowrules>
</ep-cp>
```

All the incoming packets to cp_2 that match with the eventually rules described in the flowspace element, are directed toward ep_3. Obviously the id_ref and port attributes, have to refer to existing endpoints/connection points.

5 – Concurrency

The program is concurrent. In Neo4j all operations have to be performed in a transaction, and transactions use a read-committed isolation level, which means they will see data as soon as it has been committed and will not see data in other transactions that have not yet been committed. Therefore it's possible to handle correctly GET requests, POST requests and DELETE requests at the same time. Instead the possible conflicts between two DELETE requests or between two POST requests are handled in a different way. In the first case with an explicit write lock at database level, whilst in the second one adding the keyword synchronized to the CreateNffg method, because the creation of a graph is more critical.

6 – Bean validation

Once the XML/JSON file is received, JAX-RS, is able to create an instance of annotated class Nffg from its content. Then, thanks to the custom @Graph annotation, the bean is validated, and if it doesn't pass this phase, a message which describes the validation error is sent to the client.

7 – HttpMessage

The annotated class HttpMessage is used to send to the client a structured message when the server, for some reason, isn't able to send the requested resource. Exceptions thrown by the program are first intercept by classes that implement the interface ExceptionMapper<>, and then by a filter, which acts only on those generated automatically by JAX-RS and would arrive at the client in the form of html files, violating the expected format. These intercepted exceptions, are then mapped into an HttpMessage and a response containing it is sent.

8 – Packages

it.polito.nffg.neo4j.jaxb contains the annotated classes generated by XJC starting from nffg.xsd.
it.polito.nffg.neo4j.config contains a configuration class and a filter one.
it.polito.nffg.neo4.constraints contains the annotation Graph used during the validation phase.
it.polito.nffg.neo4.exceptions contains custom exceptions and exception mappers.
it.polito.nffg.neo4.manager contains neo4j library and a java client.
it.polito.nffg.neo4.resources contains the resources classes in which there are the mapped methods to the HTTP GET, POST and DELETE.

9 – Java and Swagger client

It's possibile to use the java client through some ANT tasks that relies on client.properties file.

```
XMLfileForPOST=nffg
XMLfileForGET=out
XMLfileForPaths=paths
XMLfileForProperty=property
parameterMediaType=application/xml
parameterNffg=667
parameterSrcNode=ep_1
parameterDstNode=ep_2
parameterDirection=incoming
```

XMLfileForPost: name of the file that contains the graph to send to the server.
XMLfileForGet: name of the file in which the retrieved graph will be saved.
XMLfileForPaths: name of the file in which the retrieved paths will be saved.
XMLfileProperty: name of the file in which the property response will be saved.
ParameterMediaType: media type used. Also from it, the extension for the above files is obtained.
ParameterNffg: Id of the graph to retrieve (used by paths/property requests also). 'all' for all graphs.
ParameterSrcNode: Id of the source node in paths/property requests.
ParameterDstNode: Id of the destination node in paths/property requests.
ParameterDirection: direction considered in the calculation of paths and in property requests.

The Swagger client is hosted on http://localhost:8047/Project-Neo4jManager/index.html and is self-sufficient and self explanatory.

10 – server.properties file

```
graphDBPath=neo4j/db/nffg.graphdb
neo4jJarsDirectory=neo4j/lib
otherJarsDirectory=WebContent/WEB-INF/lib
schemaForBinding=nffg.xsd
schemaForValidating=nffg.xsd
```

This file is loaded during startup into unique instance of PropCache enum (it's a singleton, just like the Neo4jLibrary, because it's an enumeration and the JVM guarantees this feature). Within this enum, it's needed to set manually the path of the project, because the system property 'user.dir' doesn't work well in all classes (maybe due to an Eclipse bug). From it it's possible to get the above relative paths of the first three properties.

```
private void loadFileProperties() throws IOException
{
	prop = new Properties();
	prop.setProperty("my.user.dir", "C:/Users/Vastu/workspace/ProjectNeo4jManager");
	br = new BufferedReader(new FileReader(prop.getProperty("my.user.dir") +
	     "/server.properties"));
	prop.load(br);
	br.close();
}
```

11 – Development environment

Windows 7 Ultimate 64 bit
Neo4j Community Edition 2.3.1 64 bit
Eclipse Mars.1 4.5.1 64 bit
Jdk 7u80 64 bit
Tomcat 8.0