```
network SERVICE
|
|--POLICY
|   |--name attribute
|   |--NF-FGname attribute
|   |--PolicyKind (REACHABILITY|TRAVERSAL)
|   |--ReachabilityPolicy
|   |   |--SRCnode
|   |   |--DSTnode
|   |--TraversalPolicy
|   |   |--SRCnode
|   |   |--DSTnode
|   |   |--TraversedFunctionalType
|   |
|   |- policyLogic (Positive|Negative)
|   |--VerificationResult
|       |-resultMsg
|       |-verificationTime
|       |-result (SATISFIED | NOTSATISFIED | UNVERIFIED)
|
|
|---NF-FG * element
    |-name attribute (ID)
    |-updateTime attribute
    |-Node * element
        |-name attribute (ID)
        |-FunctionalType(FW,DPI,NAT,SPAM,CACHE,VPN,WEB_SERVER,WEB_CLIENT,MAIL_SERVER,MAIL_CLIENT) attr.
        |-Link  element
            |-name  attribute  (ID)
            |-source  attribute (IDREF)
            |-destination attribute (IDREF)
```

Line of code (nffgInfo.xsd) | Description

5 | The SERVICE PROVIDER provide services (from 0 to infinite) described by NF-FGs and verify the correctness of the service through policies (from 0 to infinite) so in the schema has been chosen to have a root element called network_services wich implementation contains the sequence of possible nffgs and policies about these nffgs.Nffgs and policies model descriptions are icluded respectively in nf_fgType and Policy complexTypes

53 | Each Nffg (graph) is made of at least 1 node element (the molteplicity is indicated in the schema by a sequence of node elements
       while the declaration of node is done in nodeType element of the schema)
     and is characterized by a mandatory name (use=required) and the last updateTime (or creationTime)
     of the graph itself implemented by the simple built-in dateTime type (use=required too [update or creation]).
     A node name must to be unique inside the nffg-> this specification is respected by introducing
     the key=id_node right inside the scope of the nf_fg element(row 10).

67 | Each node has a mandatory name(use=required see above) and has a function
    inside the network which is represented by one of the strings listed in the enumeration defined
    in the functionType declaration at row 77 (restriction of a simple type).
    I chose to use enumeration attribute because it is the type that is better adapted to the specifics;
    furthermore each node is connected via links to 0 or more nodes, this is the reason why the node element
    contains a sequence of link elements(defined at row 94 in linkTypeRef).

94 | each link is an empty model complexType (this is the reason why it figure an empty sequence in the definition);
     it contains only attributes that characterize the link itself infact a link is characterized by
      a unique name that is mandatory to identify the node(use=required) and mandatorily
      by a source and a destination Node without which it would not exist.
      To avoid redundancy of data they are only referencies to real node elements; this is done
      by usign the keyref schema constructs at rows 41 and 46 of the xsd file.

    The name of the link has to be unique inside the whole nffg; this specification is respected by introducing
    the unique=id_node construct right inside the scope of the

nf_fg element(row 14) just like the id_node above.

104 | A policy could be a reachability policy or a traversal policy and its result is stored inside a verificationResult element. Just because both reachability that traversal share the same attributes, these are defined inside the generic policy element and only if is a traversal policy it will contain the sequence of functionType (belonging to the nodes) crossed by the source to the destination.

113 | A policy is characterized by a mandatory unique name like the other entities in the service provider system and it refers mandatorily to an exsisting graph (the string nffg_ref is defined as a keyref to a id_nffg in the schema). The specification about the uniqueness of the policy's name is respected introducing a key in the scope of the root element(each policy in the network_services element must have a unique name - row 36) The kind of logic about the policy is described by an enumeration type(policyLogicType) that is mandatory because cannot exists a policy without this description(use=required).

Policies need a source and a destination Node. Just like for the links, to avoid redundancy of data they are defined in the complexType as strings but they are just referencies to real node elements; this is done by usign the keyref schema constructs at rows 41 and 46 of the xsd file. Furthermore each policy has to describe the result of its verification and this is done by a verificationResult element defined in verificationresultType at row 127 as a sequence of 2 elements(resultMessage and result) and the attribute verificationTime(optional in the case the policy has not been verified ->verificationResult is present anyway but with some field absent or in some particular state)

127 | verificationResult is defined by the verificationTime attribute that is a simple dataTime schema type and indicate the time of the last verification of the policy who belongs to. The content model is completed

-verResult instead has to be mandatorily one of these strings: SATISFIED or VIOLATED accordingly the rules if the policy has been verified, or NOT-YET-VERIFIED if not.