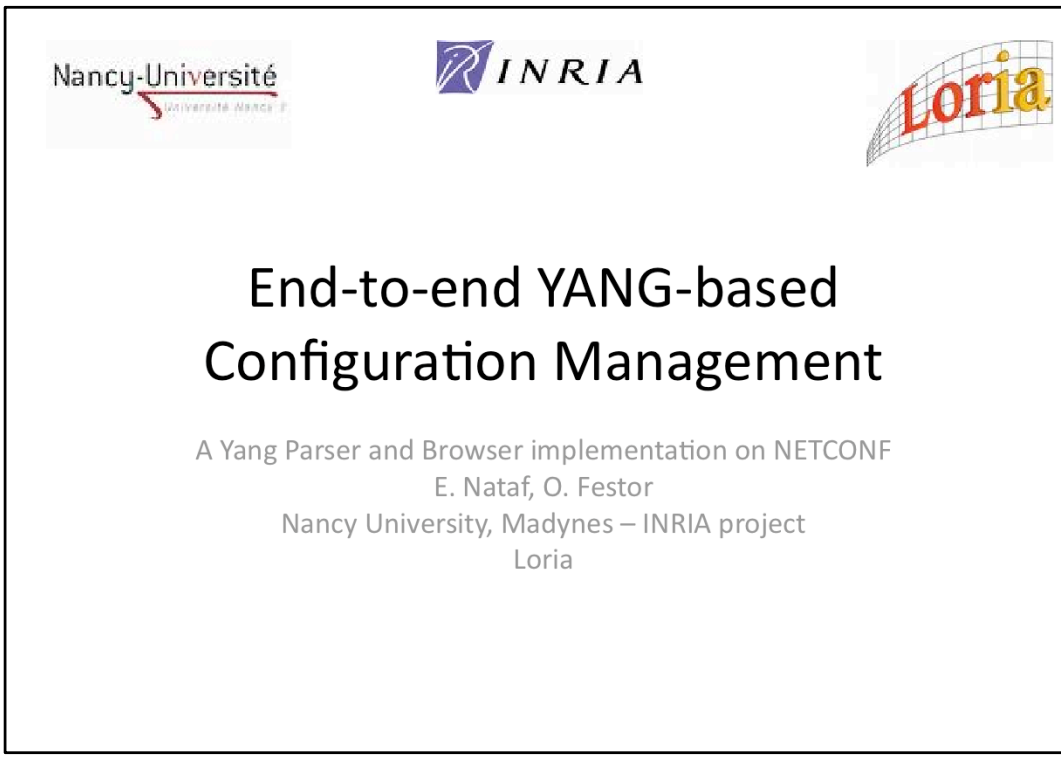
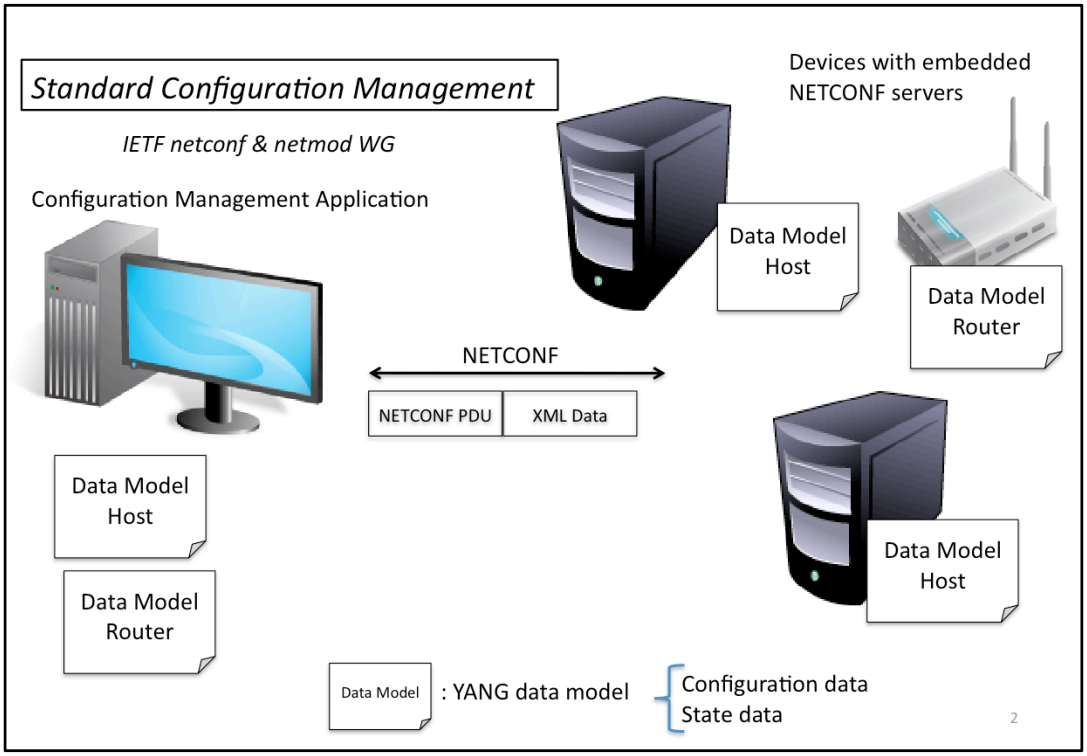


Networked resources are of increasing complexity and have to be configured properly to guarantee their operation. Within the IETF, current efforts are focused on both a protocol and a data model definition language for configuration management. The NETCONF protocol describes the communication between devices to be configured and configuration applications. NETCONF does not describe how configuration data is represented. This is addressed by the YANG data modeling language, the emerging proposal of the netmod standard working group.
We present in this paper the result of the integration of YANG and NETCONF in the ENSUITE open source framework. We illustrate this integration through a YANG-based navigation and edition application that works with YANG-enabled devices and interacts through NETCONF.

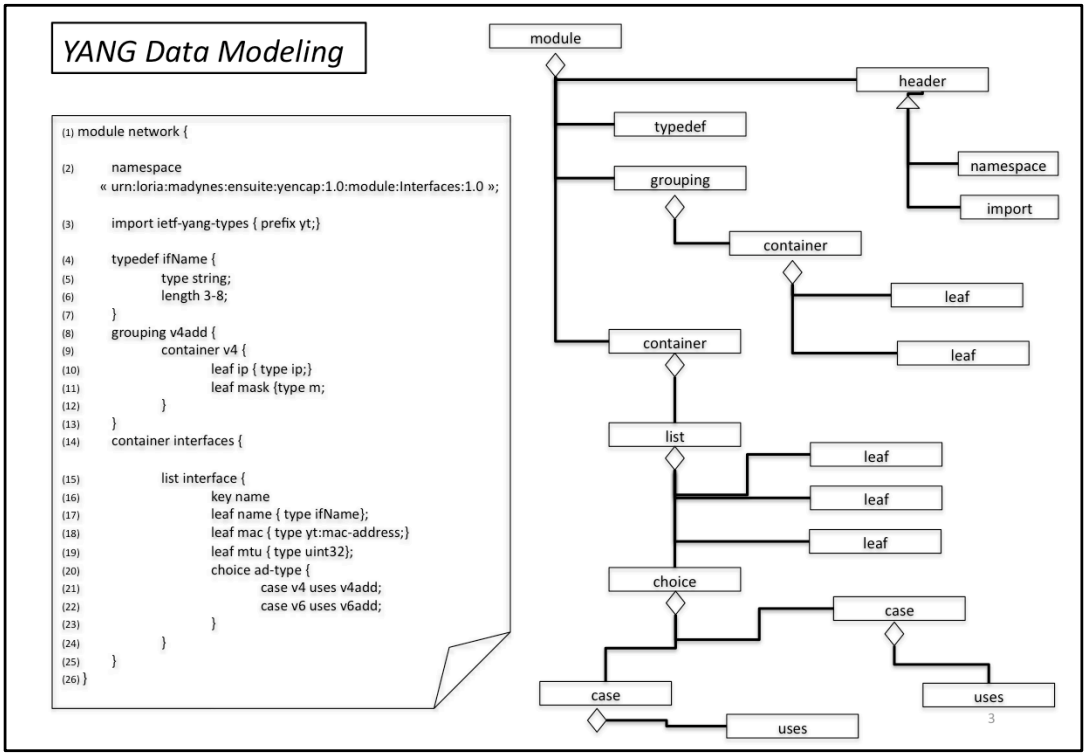


The importance of configuration management is increasing with the growing size and the complexity of network resources and applications. In the Internet context, the netconf working group has designed the NETCONF protocol [1] as a standard to manage configuration of network devices. This protocol is tailored to configuration operation i.e. setting and/or getting configuration data values to/from devices with an RPC mechanism. Data values are transmitted as XML documents. The standardization body acknowledges this should be improved by a data modeling language that will describes these data values.

Several specification languages exist to model XML document structure like XML Schema or RelaxNG [5,6]. Even if these schema languages are powerful, the netmod WG chose to define its own language that can control the evolutions and which is descriptive enough and more focused on configuration management.

YANG [2] is the data modeling language proposed by the netmod working group. YANG can be compared to SMI [3] in the SNMP [4] framework because it is a data modeling language where data values are distributed and accessible through a protocol. In the same way, a YANG specification is a reference document used by device vendors and application developers.

The objective of this paper is to demonstrate the feasibility of an End-to-End YANG-aware management framework and to describe how it can be implemented in an open source framework. First we describe the YANG language, focusing on its major concepts. Secondly we present a parser for YANG specifications : jYang, an open source implementation we provide to the community. The third part shows how we did integrate YANG into the used NETCONF server. Finally we show a YANG browsing application and its functionalities to get and edit configuration data.



Data models written with YANG describe hierarchical organization of configuration data. A YANG module is one data model related to a specific configuration purpose, as for example network configuration, or a generic, reusable, set of data models. The left part of the figure shows a YANG module called network. A module must first defines its name space (line 2) that must be unique among all YANG models. If needed, a module can import other YANG models (line 3, the "ietf-yang-types" reference is a YANG module [7] with useful types intended to be used by other modules).

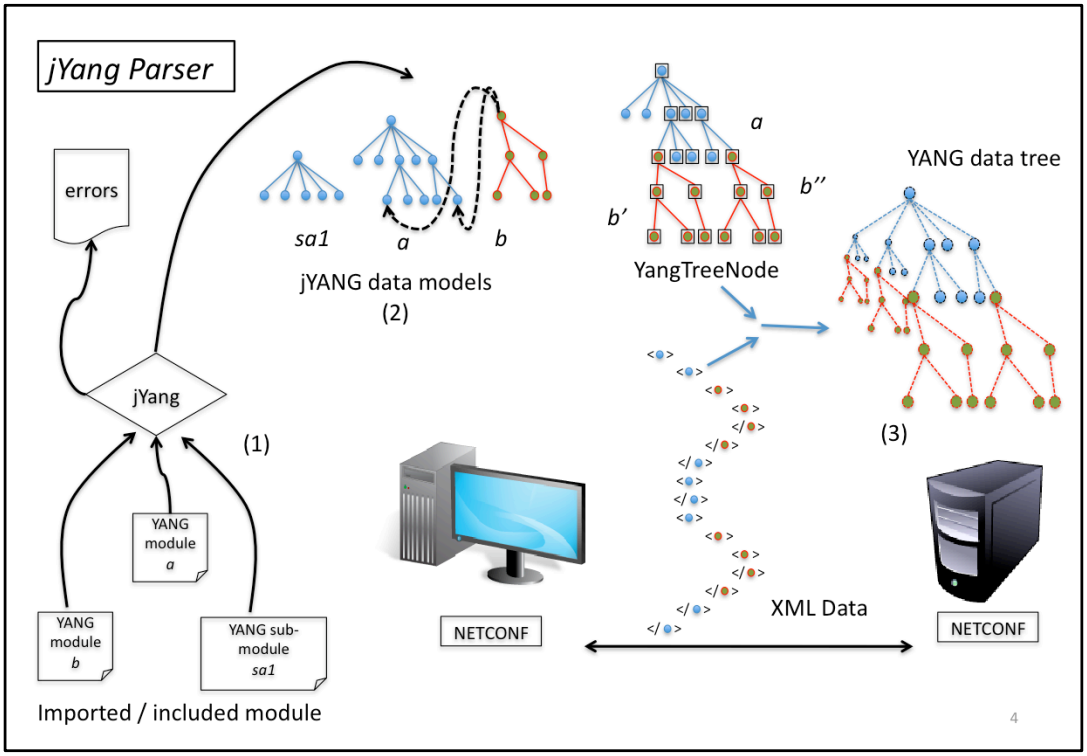
YANG defines a limited set of build-in types such as string, boolean and integer. These basic types can be used to create other types with a "typedef " statement (line 4). A typedef allows to add some constraints on its base type as the length of a string (line 6). Another construct that improves reusability is the "grouping" statement (line 8) that can be used, with a "use" statement at separate places (line 21 for example).

Data models are mainly expressed with the following statements (called datadefs) :

- Leaf : a value of one type.
- Container : a set of datadefs.
- List : an ordered set of entries and all entries are made from the same set of datadefs. One or more datadefs of the set entry must be defined as list key.
- Leaf-list : a list of values of the same type.
- Choice ; an alternative of different cases of datadefs.

The example shows two containers (lines 9 and 14), a list (line 15) and a choice (line 20).

The API we propose reflects the YANG statements hierarchy. For each YANG statement a corresponding java class is available (see class diagram on the right part of the figure). Each java object has getter methods to follow the tree of instances. Hundred java classes were required to represent any YANG model.

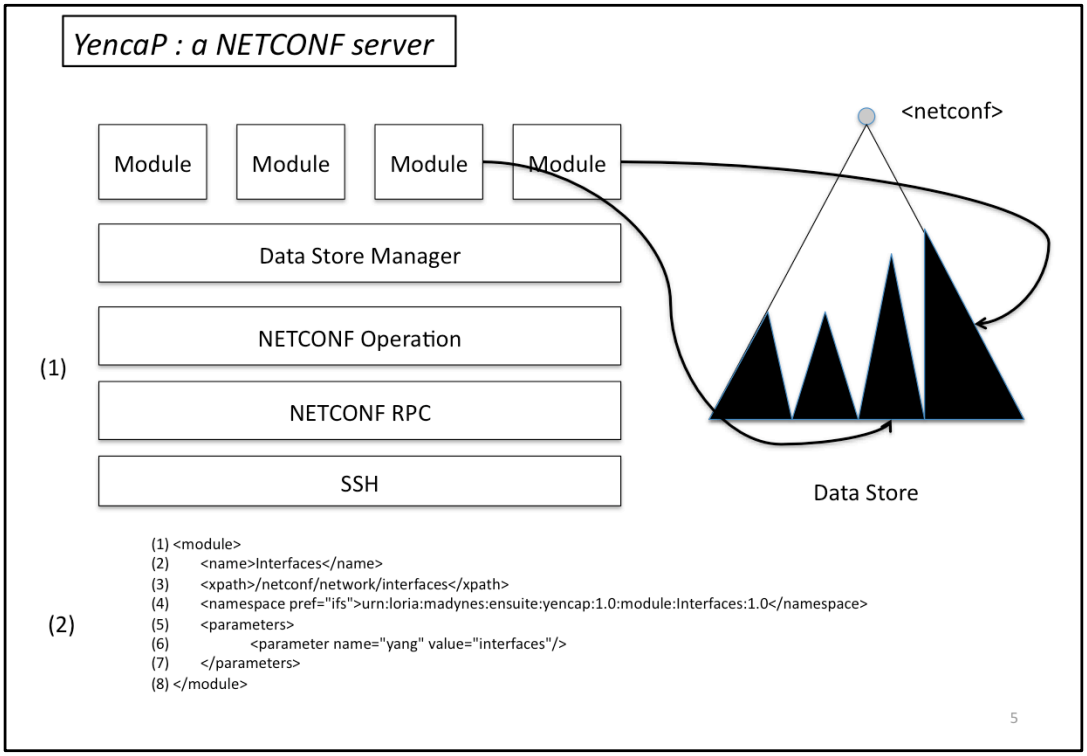


jYang is an open source parser for the YANG language. It is written in java with the javaCC library (https://javacc.dev.java.net/). The parser supports multiple modules inclusion (part (1) of the figure).

Full syntax checking is done and some constraints are checked. YANG allows data modelers to express static constraints like default values, key indexing or sub-typing that can be checked during the parsing phase. Also, YANG allows dynamic constraints on data values, like the conditional presence of data depending on other data values or hosting device capabilities, that obviously can not be checked at parsing time.

When no error is encountered the YANG data model is represented by the YangTreeNode object. This tree, built from java instances, is an interpretation of YANG data model where only YANG statements describing data values are present. For example on the part (2) of the figure, suppose the module *b* is only a grouping statement that is used by some statements in the module *a* (those with a dashed arrow). Then the final YangTreeNode reflects this by a tree with a full copy of all needed statements, called *b'* and *b''*. In the same way, when there are choice statements in the YANG data model then all cases are represented. This is not the true for type definitions because a type is not a value. In our example the submodule sa1 is only a typedef collection and so there is no copy of them in the final tree.

The YangTreeNode is used to match the YANG data tree with raw NETCONF XML data. The YANG data tree is the hierarchy of all configuration data values in a NETCONF server. The matching process takes place in the configuration manager when receiving NETCONF responses. The part (3) of the figure shows that the YANG data tree has more nodes than the YangTreeNode and these extra nodes have a common pattern. This is the case when a YANG list or leaf-list is defined and when the YANG data tree has several entries. At the opposite some data could be optional depending of the device itself, or when the YANG data model has choice statements.



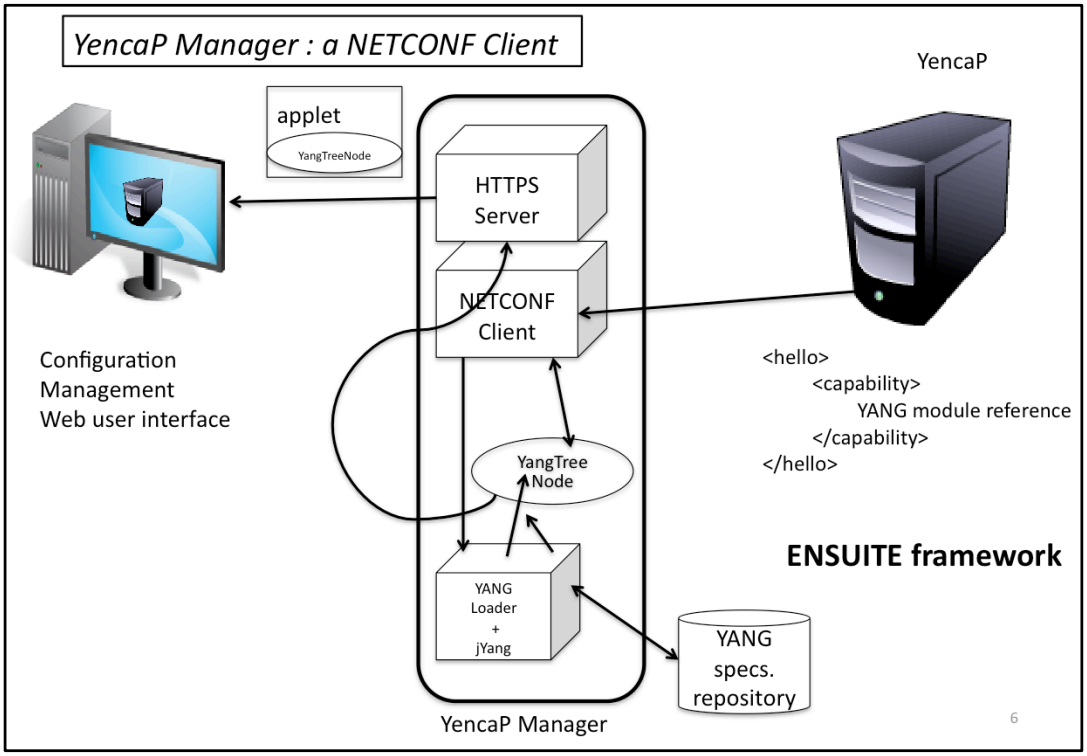
YencaP [8] is an open source implementation of the NETCONF server side. The bottom part of its architecture is an SSH layer. The RPC layer implements the RPC mechanism with <rpc> and <rpc-reply> primitives that carry basic operations on top of NETCONF operation layer like <get>, <get-config> or <edit-config> (notifications are planned). The Data store Manager layer entity is responsible for maintaining a virtual database of configuration (and state) data and provides a read / write access to these data. The Data Store, illustrated in the figure, can be seen as a global XML data tree where each module has its place.

When bootstrapping, the Data Store Manager looks for modules through a local configuration file and dynamically loads them. A module is a piece of code that accesses specific configuration and state information with an interface compliant with NETCONF operations. For example there are modules for network interfaces, system, protocols like RIP or OLSR...

Although YencaP was written before YANG, the module concept of the former fits well with the YANG one. For each module, the configuration file contains some directives as for the interface module in the part 2 of the figure. The location of data module in the Data Store is made by giving a path (an Xpath expression) from the root <netconf> node to the root node of the module. For example, at the line 3, the interfaces module is localized with the "/netconf/network/interfaces" expression and it maintains data under the <interfaces> node.

The integration of YANG into YencaP is made in the configuration file through the addition of a parameter with the "yang" name attribute and the module name as a value attribute (line 6). For each module inside the YencaP server there is one and only one associate YANG module. One can see on the figure that there is a <namespace> markup at line 4. This name space is needed inside NETCONF requests to distinguish its XML naming. We choose that it must be the same as the one defined in the corresponding YANG module.

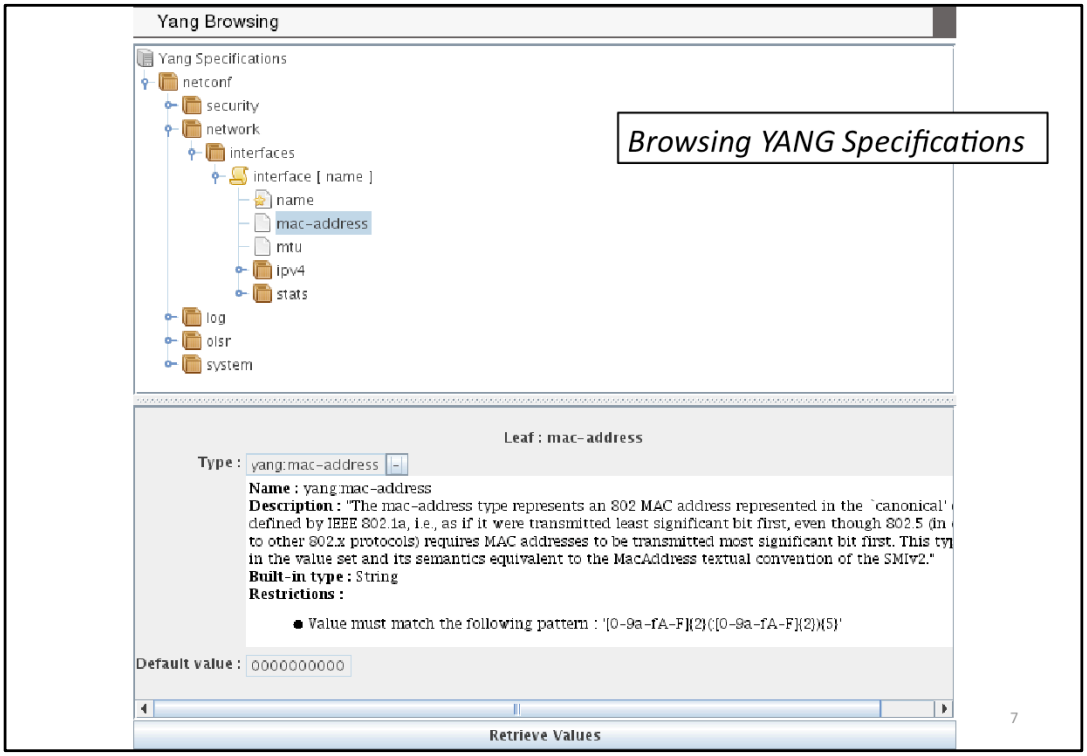
Thus, one part of the Data Store is managed by the Data Store Manager (white on the figure) and the other parts are managed by the modules (black sub trees). This enables modularity of the server without increasing the complexity of the Data Store Manager.



The YencaP Manager, on the center part in the figure, is an open source NETCONF client application that can send queries and receive responses with any NETCONF-compliant server. The NETCONF client can have several NETCONF sessions with several servers at one time. Each session is initialized by the HTTPS server inside the YencaP Manager when a user opens an HTTPS session. There is a one to one mapping between HTTPS and NETCONF sessions. The couple (YencaP / YencaP Manager) forms the ENSUITE framework.

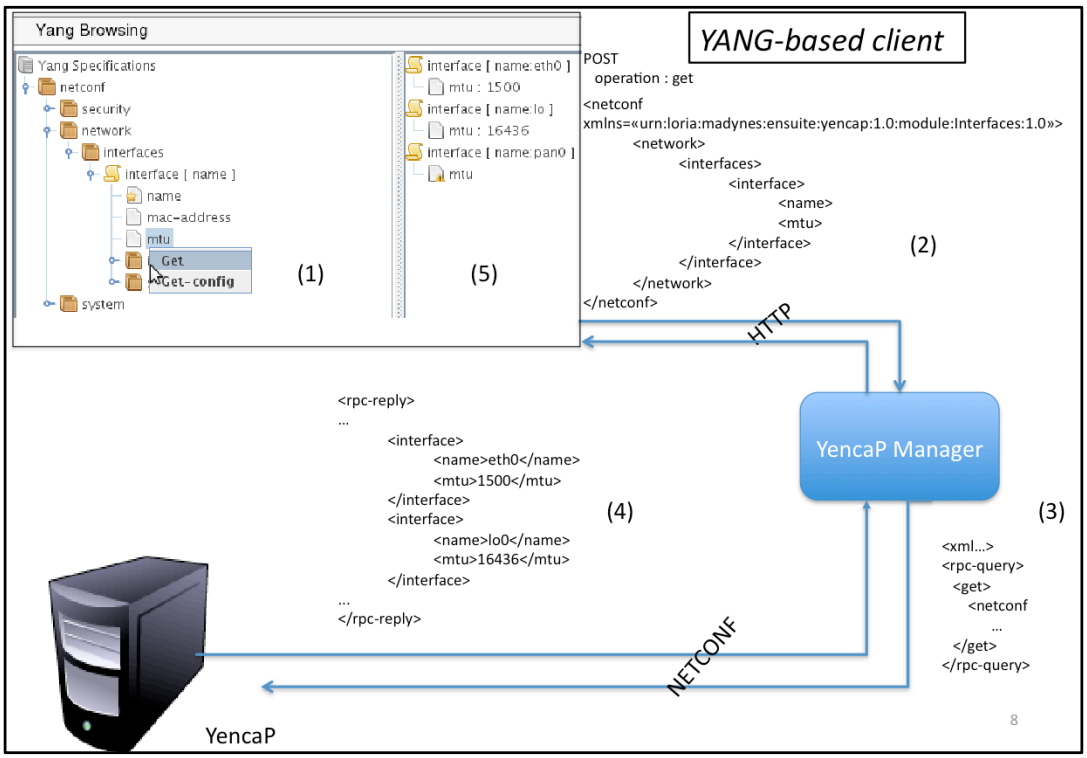
On the right part of the figure, we extends YencaP to give it the possibility of announcing which YANG modules it implements as a capability in its standard hello message (together with version and revision information). This was easily realized with the information in the configuration file shown in the previous figure 5. On the YencaP Manager side, a YANG loader is used when such a capability is detected. We do not constrain the YencaP Manager to solely work with YANG but to accept servers that are YANG enabled or not. The YANG loader gets the specification from an external repository and builds a specific YangTreeNode for the data model maintained by the server. The YANG loader is a java program that uses jYang to dynamically parse YANG data model. We assume that the YencaP Manager discovers servers without knowledge of their configuration and thus must be able to dynamically load and parse any YANG model. It is also necessary to create the root node in the YangTreeNode as a virtual netconf container. The YANG specification repository is shown as an external element of the YencaP Manager as it should be a global repository implemented for example as a web service.

Once a HTTPS session is opened the user can ask for the configuration of a YANG enabled device. In doing so it receives a java applet that contains the YangTreeNode for this server only (left part of the figure). The applet will be loaded by the web interface to provide the user with a graphical interface representing the configuration.



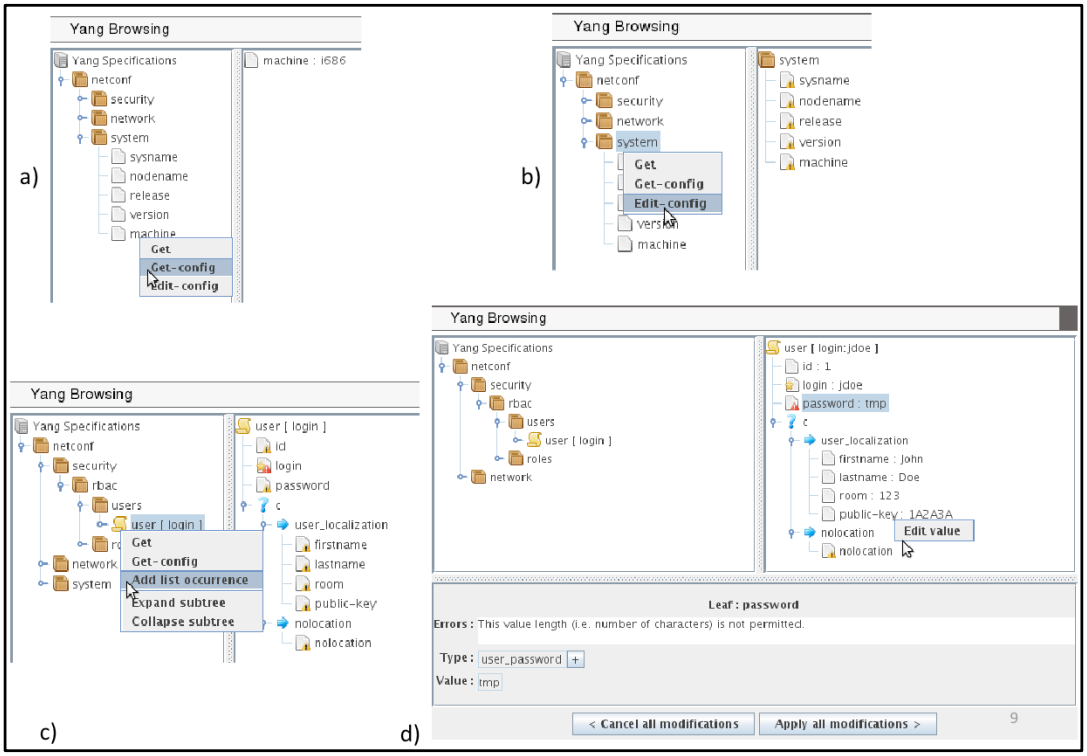
The figure shows the applet part of the web interface displayed when a user is connected for the configuration of a device. This first view can be used as a YANG specification browser looking like a file system browser (we use the swing Jtree interface). The tree view matches well with YANG because it defines a schema tree. Specific icons are used to distinguish containers, lists, keys or leaf nodes.

When selecting a node, the lower part of the applet shows details of its YANG specification, such as the type of a leaf and constraints such as default value or range intervals. A leaf type is always at least of a built-in type (string, int8,…) and can be refined by other types with added constraints. It can also use an existing type (e.g. mac-address). The interface allows the user to see either the refined or the built-in type of data.



The same applet is also used for sending requests and receiving responses. A request is made when the right mouse button is pressed on a YANG node (part 1 of the figure). The request is built from the root node to the selected one. At this step, the applet is vertically separated to show on the right the result of the request. The resulting XML document is sent inside an HTTP POST request (part 2). A specific header called “operation” is added to specify which NETCONF operation must be performed on the server (get, get-config or edit-config). The figure shows a get request on the leaf mtu inside the interface list. Note that the key of the list is added to the request while it is not explicitly requested. This is because further requests on this list (and especially on list entries) will need the key.

When the request is received by the YencaP Manager, it is encapsulated by the rpc and filter mechanisms and sent like a valid NETCONF request (part 3). From this step we are independent of any YANG concern because we are in a full NETCONF session. The part (4) shows the NETCONF response to the request that will be forwarded by the Yencap Manager while adding HTTP headers. The response is displayed part (5) when it is received by the applet.



The figure depicts some functionalities of the applet related to NETCONF get and edit config operations.

Part (a) is a simple access to a leaf in a container and where we get the response of a get-config operation.

Part (b) is an edit-config for a container. When editing a container, its components are listed with a warning until a correct value is given. The applet shows warning messages and have editing functionalities (as in part d) to set values on edited data.

Part (c) is an edit-config for a list. A list is edited entry by entry (called a list occurrence) and one can see an empty list entry ready to be filled. Note that a little star is on the “login” leaf because it is the key of the list and so its value must be set.

Part (d) illustrates the choice representation and its edition. A choice case (“user-localization” or “nolocation”) can only be edited if all of its components are set. On the same part the leaf called “password” is marked because the value is not long enough. This is an example of dynamic constraint one can check with the tool. Range values of integer or float, pattern matching of string are also checked.

Conclusions and future works

- jYang: a YANG parser
- ENSUITE framework : YANG enable
  - YencaP : server announces
  - YencaP Manager : YANG view applet
- YencaP : agent builder
- YANG constraints

We provide two contributions to the network configuration domain. The first one is a YANG parser and semantic checker close to the actual version of the draft definition of YANG. The second contribution is the support within the ENSUITE framework of YANG data models both on the server and the client side. The server can announce which YANG data models it implements and the client has a GUI to handle them and their instances.

The server has to be extended to accept user defined operations and send notifications as there are YANG statements to define new operations and notifications.

We plan to design an other backend for jYang to YANG validation constraints inside YencaP. Such validation will be done by generated code from YANG data models. The constraints we aim are default values, must and presence conditions, references between values, length or pattern matching. YencaP will check its configuration data and notify a manager if any constraints are not validated.

Références

[1] R. Enns
NETCONF Configuration Protocol, RFC 4741, December 2006

[2] M. Bjorklund
YANG - A data modeling language for NETCONF
draft-ietf-netmod-yang-07, Network Working Group, Internet-Draft, 13 July 2009

[3] K. McCloghrie, D. Perkins, J. Schoenwaelder.
Structure of Management Information Version 2 (SMIv2), RFC 2578, April 1999.

[4] Case, J., Fedor, M., Schoffstall, M. and J. Davin,
Simple Network Management Protocol, RFC 1157, May 1990.

[5] Huiyang Cui; Bin Zhang; Guohui Li; Xuesong Gao; Yan Li
Contrast Analysis of NETCONF Modeling Languages: XML Schema, Relax NG and YANG
International Conference on Communication Software and Network, 2009, ICCSN'09, 27-28 Feb 2009 Page(s):322 - 326

[6] Hui Xu; Debao Xiao
Data modeling for NETCONF-based network management: XML schema or YANG
11th IEEE International Conference on Communication Technology, 2008, ICCT 2008, 10-12 Nov 2008 Page(s):561 – 564

[7] J. Schoenwaelder
Common YANG Data Types
draft-ietf-netmod-yang-types-03, Network Working Group, Internet-Draft, 13 Mai 2009

[8] V. Cridlig; R. State
YencaP Documentation
Technical Report, 2005, 25 Pages,http://hal.inria.fr/inria-00000804/fr