Configuring Network Devices with NETCONF and YANG

Ladislav Lhotka
⟨*lhotka@cesnet.cz*⟩

26 March 2011

Main Topics

❶ NETCONF protocol for remote configuration of network devices and services;

RFC 4741 – Enns, R. (Ed.): *NETCONF Configuration Protocol.* IETF NETCONF WG, December 2006. 95 p.

❷ YANG language for modelling configurations, operational state data, sysadmin commands and notifications;

RFC 6020 – Bjorklund, M. (Ed.): *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF).* IETF NETMOD WG, October 2010. 173 p.

❸ Standardized mapping of YANG data models to DSDL, validation of instance documents.

RFC 6110 – Lhotka, L. (Ed.): *Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content*. IETF NETMOD WG, February 2011. 100 p.

Why All This?

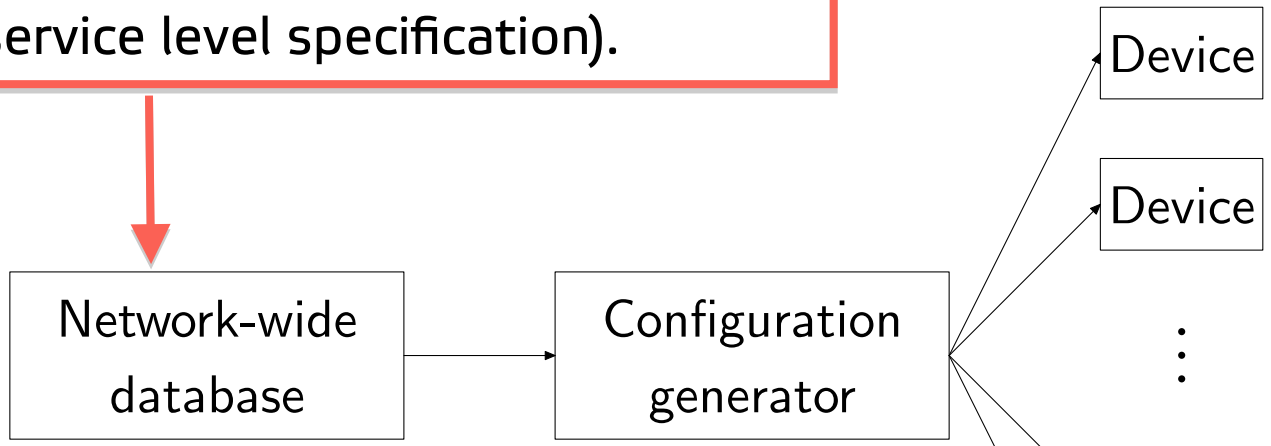
Large heterogeneous networks require automated and reliable management procedures, yet existing frameworks and tools for configuration management are far from satisfactory.

In particular, Simple Network Management Protocol (SNMP) as a *configuration* tool cannot compete with proprietary command-line interfaces. The reasons are both technical and marketing.

Previous IETF efforts to create a protocol-independent data/information modelling language (SMIng) failed, because of many conflicting requirements.

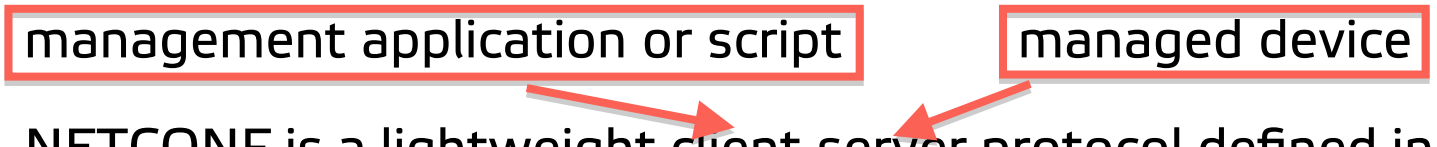
Typical (Large) Network Configuration Workflow

The database is fed with high-level network description (topology, routing and security policies, service level specification).



No reliable method for pushing generated configurations to devices existed before NETCONF.

NETCONF Protocol



NETCONF is a lightweight client-server protocol defined in RFC 4741. It uses XML for both protocol operation and data contents.

NETCONF was modelled after JUNOScript – Juniper routers use the same configuration daemon for both XML and CLI input.

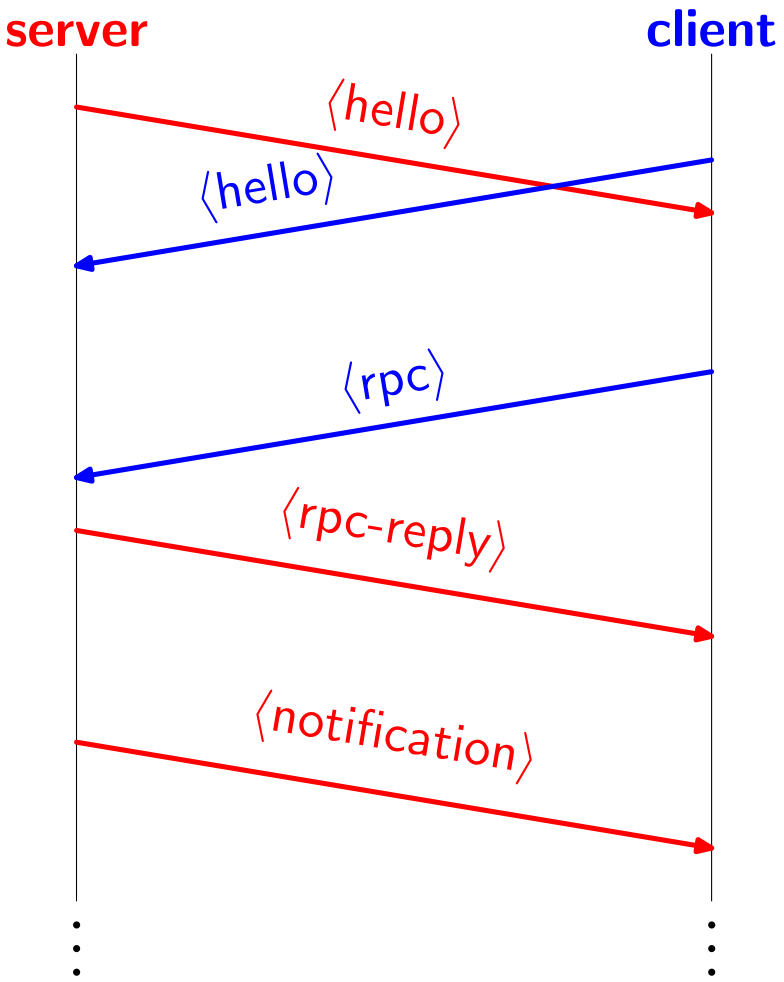
NETCONF can be used with *multiple transport protocols*, the following four have been defined so far: **SSH** (RFC 4742, mandatory), **SOAP** (RFC 4743), **BEEP** (RFC 4744) and **TLS** (RFC 5539).

NETCONF uses a *document-oriented* approach: configuration and operational state data are represented as a structured document.

Devices must have `running` configuration datastore, may have others, such as `candidate`.

Management operations are implemented as *remote procedure calls (RPC)*.

NETCONF Session



hello messages are exchanged at the beginning of each session, both parties send the protocol version, the server also advertizes supported data models and prescribes `session-id`.

Several **rpc** methods are an integral part of the protocol but new methods may be defined, too, e.g. in YANG data models.

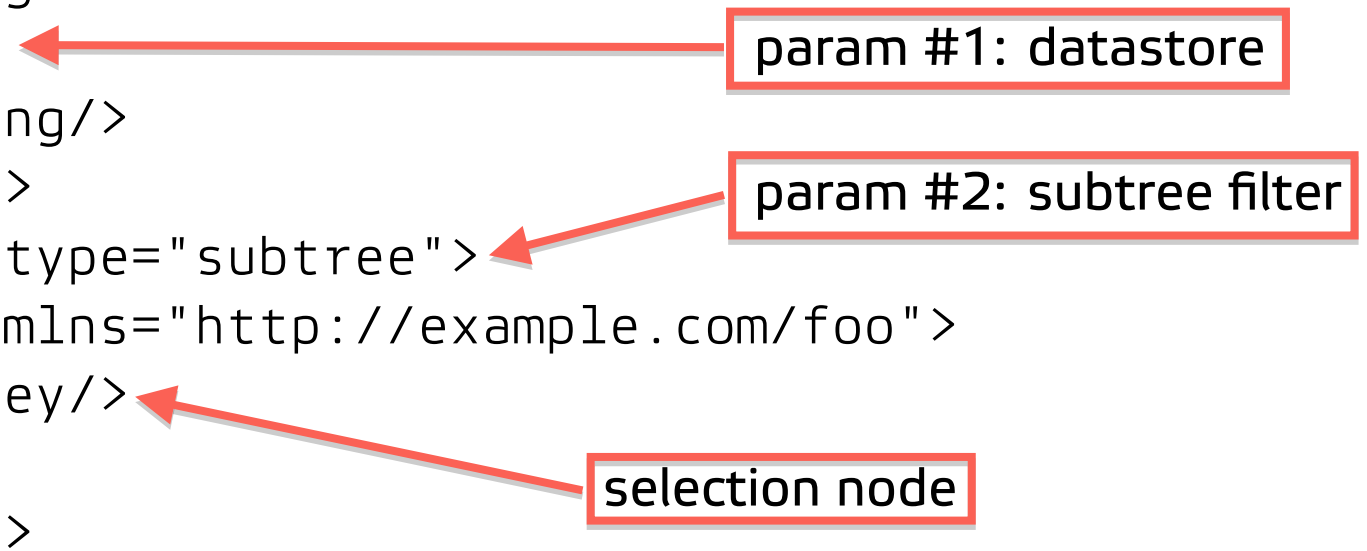
A **notification** is an unsolicited message from the server, clients may subscribe to selected channels. Notifications are defined in RFC 5277.

Basic RPC Operations

`<get-config>`	Retrieve all or part of the contents of a configuration datastore.
`<get>`	Retrieve all or part of the *running* configuration datastore **and** operational state data.
`<edit-config>`	Modify all or part of a configuration datastore.
`<copy-config>`	Copy an entire configuration datastore.
`<delete-config>`	Delete a configuration datastore.
`<lock>`	Request a lock of an entire configuration datastore (partial lock $\rightarrow$ RFC 5717).
`<unlock>`	Release a previously acquired lock.
`<close-session>`	Request graceful termination of a session.
`<kill-session>`	Force the termination of a session.

Example: Querying a Configuration Datastore

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <doc xmlns="http://example.com/foo">
        <dopey/>
      </doc>
    </filter>
  </get-config>
</rpc>
```



It is also possible to use XPath for selecting parts of the data, but this is an optional capability.

YANG Data Modelling Language

Design objectives:

- Apart from the schema and data types, it is also important to define the *semantics* of the modelled information.
- Data model *readability* is the priority, preferences of model writers and software developers are secondary.
- The data modelling framework has to be modular and capable of accommodating a potentially large set of modules that will be developed over a long period of time.

The top-level unit of organization is the *module*, which may be subdivided into *submodules*. A particular data model consists of one or more YANG modules.

Each module defines a namespace (also shared by submodules) for its data nodes, data types and other classes of objects. A module can import global definitions from other modules.

Two Equivalent Syntaxes

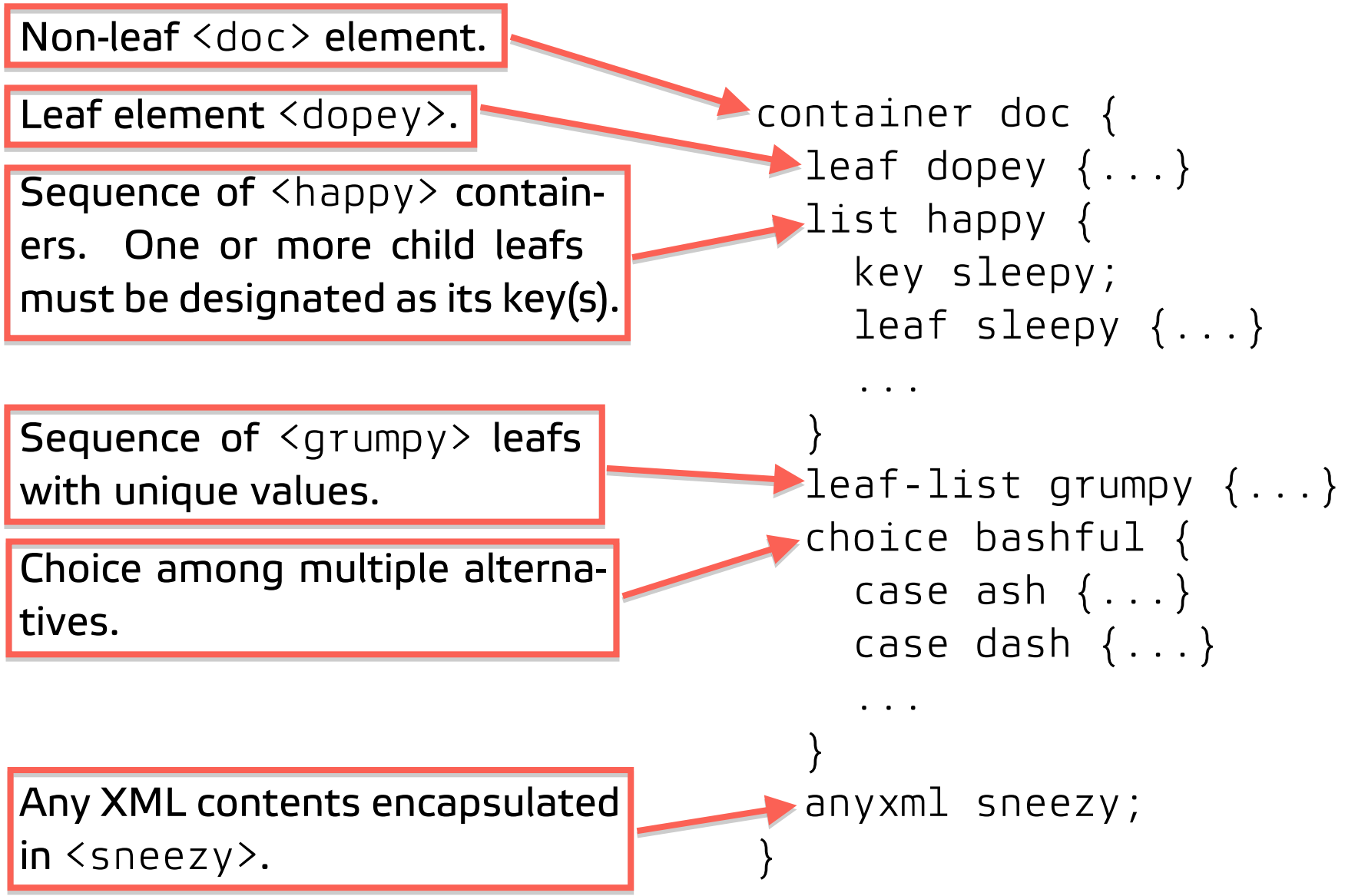
The principal syntax is compact:

```
module foo {
  namespace "http://example.com/foo";
  prefix foo;
  ...
}
```

An alternative XML syntax, named **YIN**, is also available:

```
<module name="foo"
        xmlns="urn:ietf:params:xml:ns:yang:yin:1">
  <namespace uri="http://example.com/foo"/>
  <prefix value="foo"/>
  ...
</module>
```

Schema and Data Nodes



Data Types

Most YANG data types have their counterparts in W3C XML Schema Type Library, although their names mostly follow the SNMP/SMI tradition:

`int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, `int64`, `uint64`, `string`, `boolean`, `binary`.

The `decimal64` type is similar to XSD `decimal` but allways with 64 bits (XSD `totalDigits` = 19) and the number of fraction digits has to be specified.

The `leafref` type is the basic instrument for *linking*, mainly constrained to contain a value of a list key.

Leafs of the `instance-identifier` type contain a simplified XPath expression.

Derived Types

YANG supports subtyping: a new type may be defined by restricting a built-in (or another derived) type. For example:

```
typedef hour {
  type uint8 {
    range "0..23";
  }
}
```

Names of derived types belong to the namespace of the defining module.

RFC 6021 contains two YANG modules with essential type libraries:

- `ietf-yang-types` – generally useful types
- `ietf-inet-types` – Internet-specific types

Extraordinary Features of YANG

1. Data models address several different types of instance documents: configuration datastores and operational state data, standard NETCONF messages, but also new RPC operations and notifications defined by the data model.
2. Definitions of data nodes may also contain:
   - **must** statement specifying semantic rules and other conditions,
   - **default** statement specifying default values of *leaf* nodes,
   - **when** statement that makes the parent data node conditionally valid, depending on the contents of a particular instance document,
   - **description** statement with *normative* information concerning semantics of the parent node.

3. The **augment** statement allows for adding data nodes to any place in an existing module. For example, this is how a module for generic network interfaces can be augmented with Ethernet-specific stuff:

```
import interfaces {
  prefix "if";
}
...
augment "/if:interfaces/if:interface" {
  when "if:type = 'ethernet'";
  container ethernet {
    leaf duplex {
      ...
    }
  }
}
```

This way, modules may be laid out in the “Russian doll” style and still be extensible.

Mapping YANG to DSDL

This work has been done in parallel to the development of YANG.

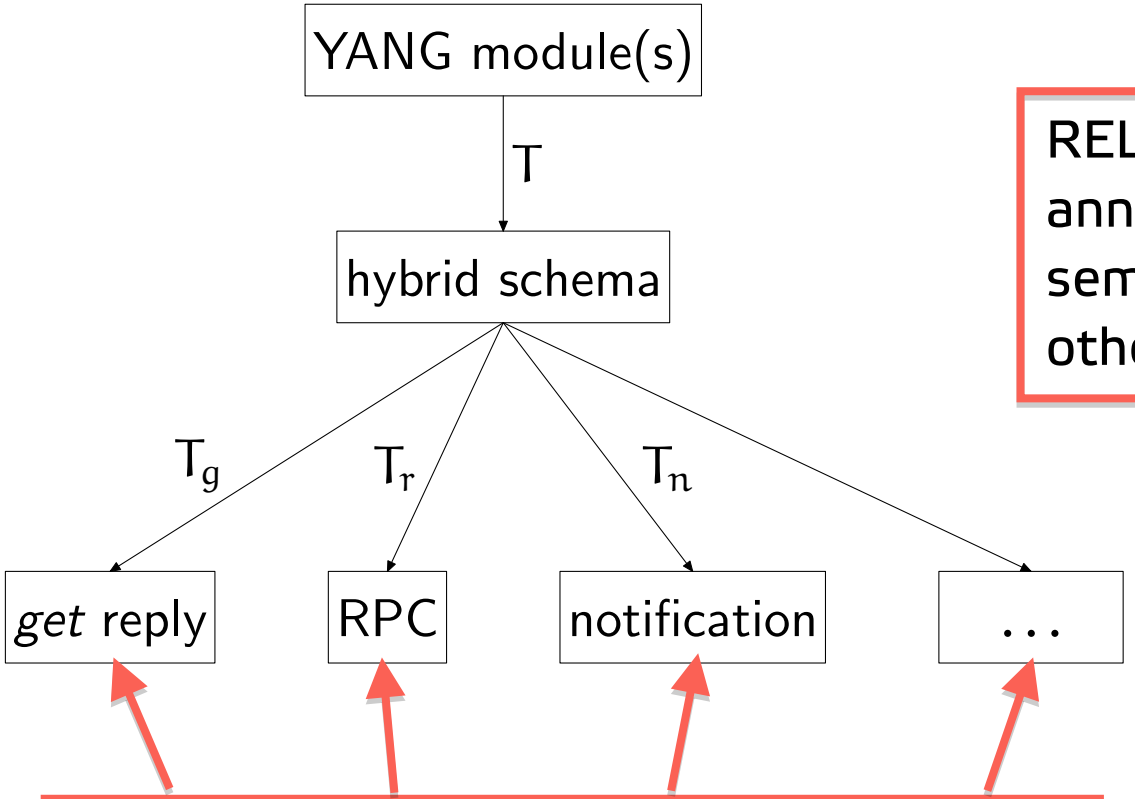
Objectives:

- leverage the existing XML tools for validating various NETCONF instance documents,
- facilitate exchange of data models and schemas within IETF or elsewhere.

Input: a data model in the form of one or more YANG modules + additional information, and the target document type, such as a complete datastore or reply to `get-config`.

Output: three schemas (RELAX NG, Schematron and DSRL) thay may be used for validating an instance document of the given type.

Structure of the Mapping



RELAX NG syntax with annotations representing semantic constraints and other information.

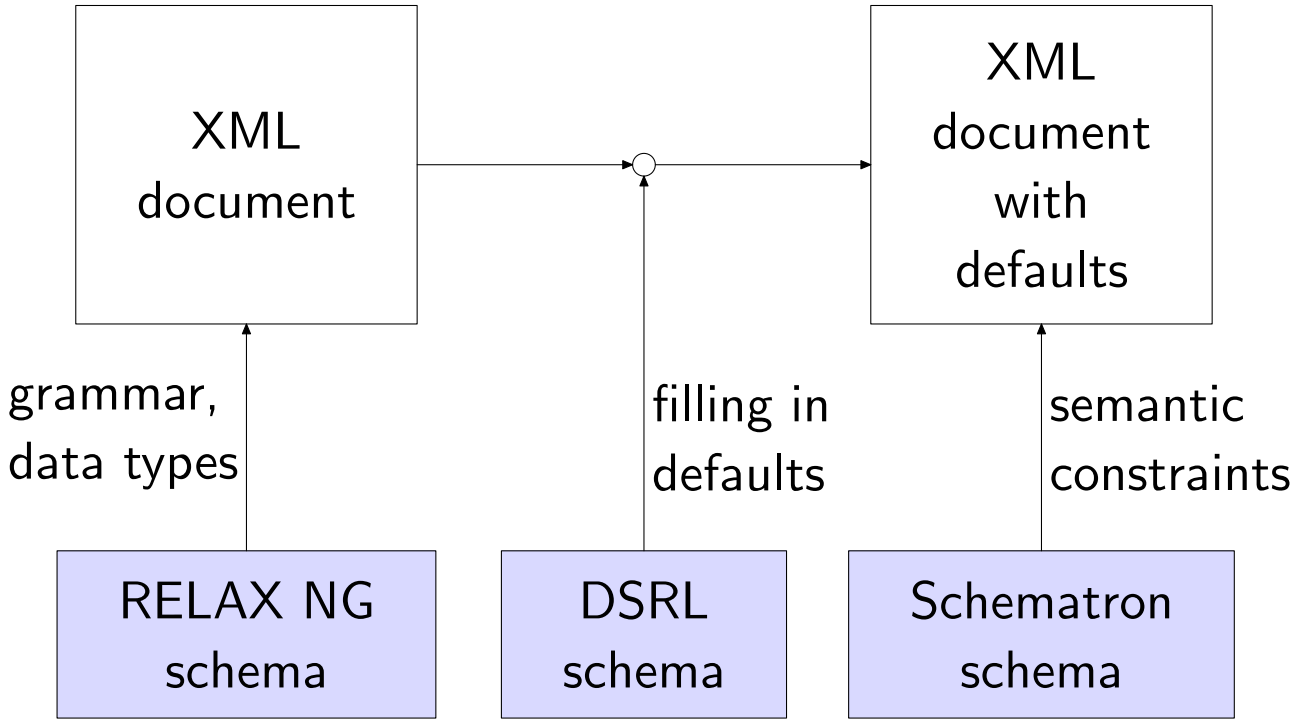
Each box represents three schemas:

- RELAX NG – grammar and datatypes,
- Schematron – semantic constraints,
- DSRL – default contents.

The Difficult Parts

1. Mapping of YANG *groupings* (analogy of RELAX NG named pattern definitions):
   - Names of groupings have a namespace and may also be scoped within the schema hierarchy. In contrast, RELAX NG named pattern definitions are always top-level and share a flat namespace inside a grammar. Therefore, the names of YANG groupings (as well as derived types) must be mangled.
   - Names of data nodes defined inside a grouping acquire the namespace of the module where the grouping is *used*. RELAX NG also supports this "chameleon behaviour" but requires a special arangement of the schema.
2. The **augment** statements from all input modules have to be collected in advance as a set of "patches" and these are applied at appropriate locations of the schema tree.

Validation of Instance Documents



Implementation

The mapping of YANG data models to DSDL schemas is implemented in the *pyang* tool (see “Resources”):

1. Transformation of input YANG modules to the hybrid schema is implemented in Python.
2. XSL Transformations (EXSLT) are used for generating the validating schemas (RELAX NG, Schematron and DSRL) from the hybrid schema.

The *pyang* distribution also contains the free ISO Schematron implementation by Rick Jelliffe.

As there is no implementation of DSRL so far, I wrote an XSLT stylesheet that translates the required subset of DSRL (`default-content`) to XSLT 1.0.

A shell script executing the full validation procedure is also included.

Future Work

The NETMOD WG was rechartered in 2010, the main emphasis is now on "bootstrapping" the development of YANG data models in other working groups. NETMOD WG itself will develop the following core modules:

- essential system data,
- framework for configuration of interfaces (`draft-bjorklund-netmod-interfaces-cfg-00`),
- framework for configuration of routing (`draft-lhotka-netmod-routing-cfg-00`).

An extensive YANG data model is already being developed by the IPFIX WG: `draft-ietf-ipfix-configuration-model-09`.

Feedback from module developers may lead to some updates of the YANG language. For instance, a set of XPath extension functions in support of special YANG data types would be quite useful.

Resources

Software:

- **Netopeer** – NETCONF implementation in C
  `http://code.google.com/p/netopeer`
- **pyang** – YANG tools (Python, XSLT)
  `http://code.google.com/p/pyang`
- **Yuma** – NETCONF and YANG development toolkit
  `http://sourceforge.net/projects/yuma`

Documentation:

- `http://www.netconfcentral.org`
- `http://www.yang-central.org`