Validation de configuration

Introduction

Une configuration est un ensemble de données qui sont les paramètres de fonctionnement d'un équipement ou d'un logiciel. Le protocole Netconf est dédié au transfert de ces données, de manière sure et sécurisée (utilisation de TCP et SSH). Il définit les services de base pour manipuler les configurations (get, get-config, edit-config, copy-config,…) sur un mécanisme d'appel de procédure à distance (RPC). Les données sont structurées sous la forme d'un document XML.

Le langage Yang sert à la description de la structure du document XML et au typage des données. Cette description formalise des concepts en relation avec une structure hiérarchique ou arborescente dans laquelle il existe des nœuds qui contiennent des nœuds et / ou des feuilles. Les données elles mêmes sont dans les feuilles. En plus de la structure, Yang permet de spécifier des contraintes que doivent respecter les données.

La problématique abordée est la suivante : étant donné des modèles de données décrits en Yang et des configurations présentes sur les équipements, mettre en œuvre les moyens nécessaires pour valider les configurations par rapport à leur description. C'est à dire que les configurations aient bien la structure définie en Yang et que les contraintes soient respectées. La configuration à distance, automatisable est sans doute un point important dans l'évolution des réseaux actuels, il est donc capital d'assurer que les configurations respectent les contraintes que l'on leur impose.

Le fait que les contraintes soient sensées garantir un comportement n'est pas pris en compte dans cette validation. Il n'y a pas, dans le contexte actuel, de formalisme permettant d'exprimer un comportement (par ex. avoir le moins de débordement de mémoire possible, ou assurer une bande passante). C'est au concepteur de modèles en Yang de décrire des contraintes sur ses configurations qui garantissent le niveau de fonctionnement voulu.

Une autre problématique est la distribution de la configuration ; les modèles Yang décrivent les configurations des équipements. Or les configurations de différents équipements peuvent être inters dépendantes et il n'apparaît pas clairement une telle notion dans Yang. Il faudrait mettre en œuvre un modèle qui permettent de spécifier des contraintes entre différents équipements en conservant la modélisation avec Yang.

Configuration et graphe de dépendances

Modélisation

Les groupes de travail de l'IETF ; netconf et netmod, considèrent qu'une configuration peut être vue comme un arbre de données (figure 1.a). Certains nœuds y décrivent des contraintes qui dépendent de la valeur d'autres nœuds. Si on isole ces nœuds et que l'on ne conserve que les référencements dus aux contraintes, on obtient un graphe de contraintes (figure 1.b). Ce graphe est orienté ; les sources des arêtes dépendent des valeurs des destinations. Il peut être cyclique.

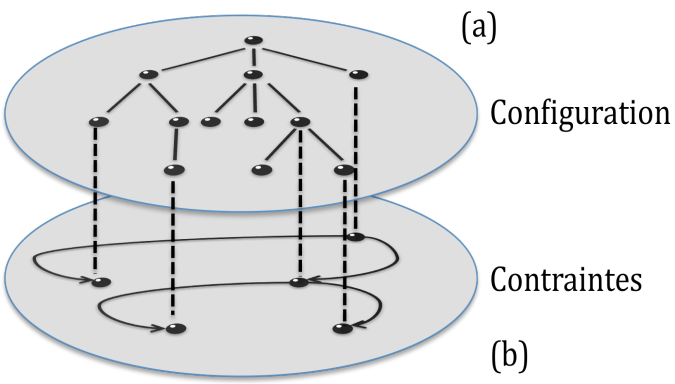

Figure 1

Pour illustrer cette abstraction, le listing ci-dessous est une définition en Yang avec des contraintes.

Elle définit un container nommé interface (ligne 1). Un container peut être comparé à un objet ayant des relations d'agrégation avec d'autres. Ici, le container interface contient deux objets de type leaf, lignes 2 et 7, qui représentent des objets avec une valeur simple. Le type enumeration du nœud ifType est un entier qui peut prendre deux valeurs (0 et 1 représentant respectivement Ethernet ou atm). Le type int32 du nœud ifMTU est un entier sur 32 bits.

```
(1)  container interface {
(2)    leaf ifType {
(3)      type enumeration {
(4)         enum ethernet;
(5)         enum atm;
(6)      }
(7)     leaf ifMTU {
(8)       type uint32;
(9)     }
(10)  must "ifType != 'ethernet' or " +
(11)  "(ifType = 'ethernet' and " +
(12)  "ifMTU = 1500" ;
(13)  }
```

Le mot clé must définit une contrainte (lignes 10-12) qui doit être valide pour qu'une configuration soit en partie valide (Pour qu'une configuration soit valide, il faut entre autres que toutes les contraintes de ce type soient validées). Cette contrainte est une expression booléenne au format standard XPath, elle est évaluée à vraie si la valeur du nœud ifType n'est pas "ethernet" ou sinon, la valeur du nœud ifMTU doit être 1500.

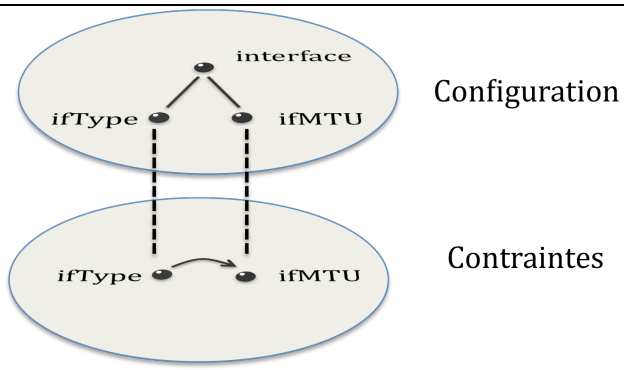

Figure 2

La figure 2 illustre la vue de la configuration et celle des contraintes. Les relations au niveau de la configuration sont liées à la structure des données : le container interface contient deux feuilles ifType et ifMTU. La relation au niveau des contraintes signifie que le nœud ifType dépend de la valeur du nœud ifMTU.

D'autres types de contraintes peuvent être exprimées en Yang :

- Des nœuds de type "leaf " peuvent être spécifiés comme étant "mandatory", c'est à dire qu'une valeur pour ces nœuds doit exister dans une configuration valide. Cette contrainte d'existence génère une dépendance sur le nœud qui le contient ; si ce dernier existe, alors le nœud "leaf" doit exister aussi. Par exemple, si la spécification précédente contenait une telle contrainte (ci-dessous ligne 3), alors on aurait le graphe de la figure 3. La relation de dépendance entre ifType et interface est d'un type différent que celle avec ifMTU ; elle signifie que si le nœud "interface" existe, alors le nœud "ifType" doit exister aussi. Cette relation ne peut pas former de cycle. Cette contrainte peut également être exprimée dans des nœuds d'une type "choice".

```
(1)  container interface {
(2)    leaf ifType {
(3)      mandatory true ;
…
```

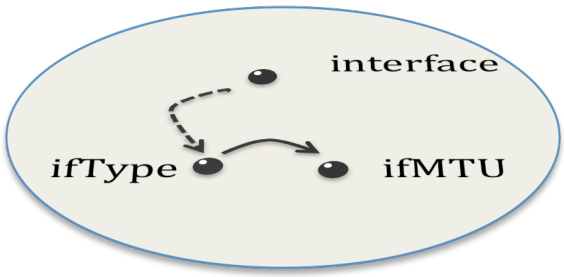

Figure 3

- Il existe un type de donnée, appelé "leafref", dont les valeurs sont celles des références à des nœuds dans une configuration. Par exemple, ci-dessous le nœud "ifTypeRef" (ligne 1 et 2) référence le nœud "IfType", contenu dans le nœud "interface" (expression "path" ligne 3). Si les nœuds référençant doivent exister, alors les référencés le doivent également. La figure 4 illustre ce cas. Il s'agit d'une dépendance du même type que précédemment ; s'il existe un nœud "ifTyperRef", alors le nœud "ifType" doit exister également.

```
(1)  leaf currentType {
(2)    type leafref {
(3)      path "interface/ifType";
…
```

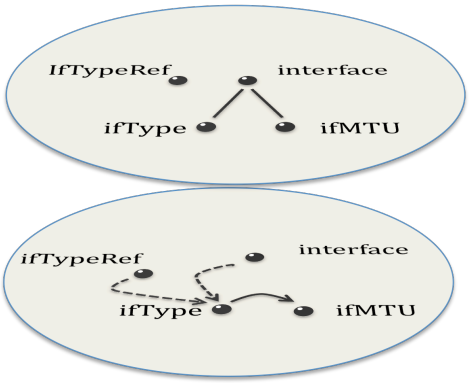

Figure 4

- L'existence d'un nœud peut dépendre d'une condition booléenne du même genre que celle vue dans l'exemple avec la contrainte "must". Cette contrainte, appelée "when" doit être validée lorsque le nœud existe dans la configuration.
- On peut également avoir une contrainte d'existence de certains nœuds liés à la présence d'une caractéristique annoncée par le serveur.

- Il existe en Yang des types de nœuds dont les valeurs sont des listes, soit de valeurs d'un même type (nœud "leaf-list"), soit de valeurs d'un même ensemble de nœuds (nœud "list"). On peut imposer un nombre minimum et maximum d'occurrences dans ces types de nœuds. Si on spécifie un nombre minimum supérieur à zéro, alors la présence du nœud implique la présence d'autres nœuds.

Validations des contraintes

Pour valider les contraintes d'une configuration, il faut tout d'abord extraire les graphes de dépendances des spécifications Yang comme on vient de le montrer. Ensuite, il faut récupérer les données de la configuration qui sont nécessaires à sa validation et enfin faire les évaluations.

Par exemple, si on a la configuration suivante ou arbre de données :

```
<interface>
      <name>lo0</name>
      <ifType>ethernet</ifType>
      <ifMTU>1500</ifMTU>
</interface>
<ifTypeRef>ethernet</ifTypeRef>
```

Le graphe (figure 4) indique que la présence d'un nœud « interface » doit entrainer la présence d'un nœud itType, de même pour la présence d'un nœud "ifTypeRef". Il suffira de ne faire qu'une vérification pour valider ces deux contraintes. Pour la contrainte entre ifType et ifMTU, elle devra être vérifiée avec les valeurs dans les nœuds correspondants.

Les contraintes doivent être validées à différents niveaux :

- Dans les requêtes RPC contenant des données de configuration (edit ou copy config). Dans la figure 5.a, les données de configuration transportées par Netconf dans une requête de type edit-config doivent tout d'abord être validées isolément. Pour cela, figure 5.b, on doit isoler les contraintes pour pouvoir les valider. Si cette validation est correcte, les données de la requêtes vont pouvoir être rajoutées à (ou vont modifier) la configuration existante (figure 5.c), qui est censée être valide.

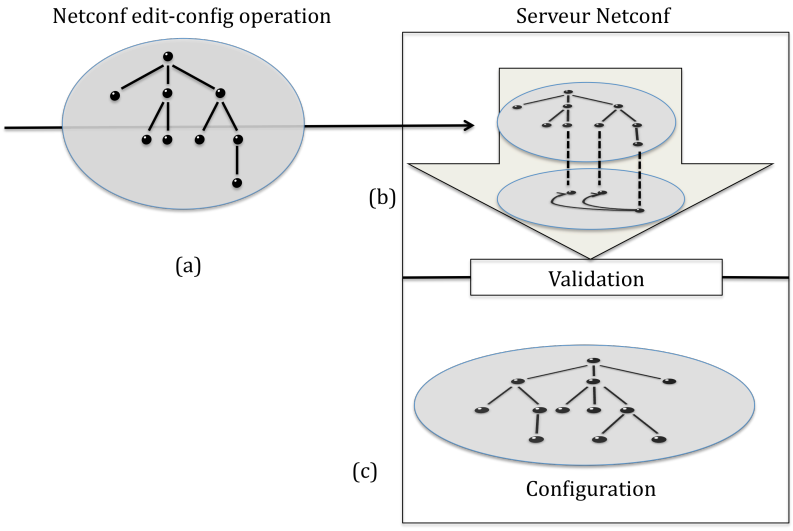

Figure 5

- Lors de l'édition proprement dite, où le serveur doit modifier la configuration existante suivant les données de la requête et également ajouter les nouvelles contraintes (figure 6).
  La configuration résultante et ses contraintes doivent être validées. Les nouvelles contraintes peuvent former des graphes de dépendances disjoints du graphe existant, mais également rajouter des nœuds et des relations sur le graphe initial. Dans ce dernier cas, l'ensemble des contraintes directement ou indirectement concernées doivent être validées.

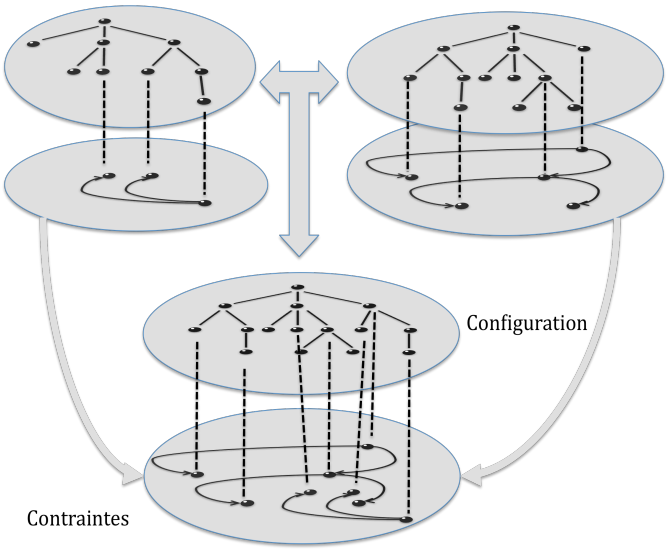

Figure 6

- Enfin, un serveur Netconf peut avoir différentes configurations ; une configuration de démarrage (dite « startup »), une en cours de fonctionnement (« running ») et d'autres en attente de passer en fonctionnement (« candidate »). Ces configurations sont à valider lorsqu'elles sont manipulées par l'opération "edit-config" vue précédemment mais aussi les opérations "copy-config" pour copier des configurations sur le serveur, et "commit" pour faire passer une configuration de « candidate » à « running » ou « startup ».

Distribution de la configuration

Dans le modèle actuel, une configuration est liée à un équipement, or la configuration d'un équipement peut dépendre de celle d'un ou plusieurs autres.

Pour modéliser cela avec Yang, il faudrait pouvoir établir des configurations avec des dépendances portant sur plusieurs équipements. La validation de ces dépendances devra alors être faite de manière distribuée, où chaque équipement gèrera sa configuration et où les dépendances inter équipements seront gérées par un serveur Netconf intermédiaire. La figure 7 illustre ceci avec un exemple dans lequel deux configurations d'équipements (que nous appellerons des configurations locales) ont une dépendance au niveau d'un nœud. Le serveur intermédiaire devra avoir une vue de la configuration globale et gérer uniquement les dépendances globales.

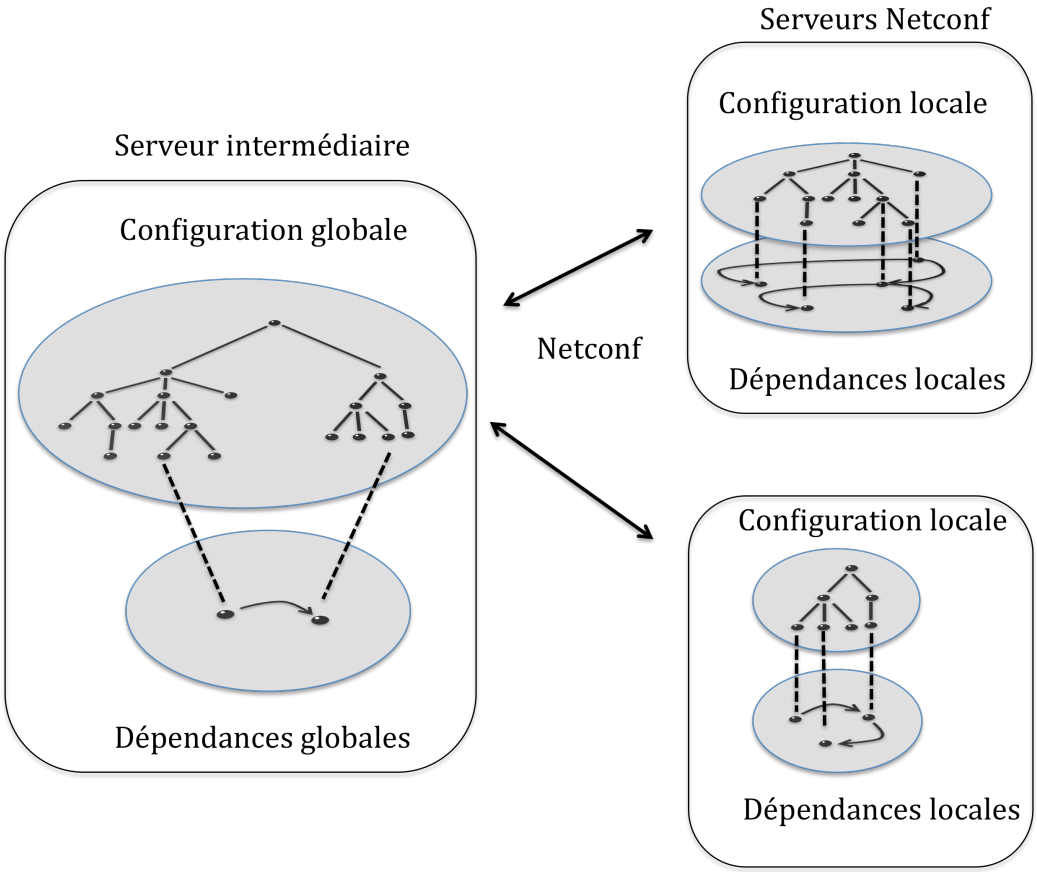

Figure 7

Objectifs

Contraintes et dépendances

Il faut modéliser la traduction des contraintes en Yang vers un graphe de dépendances. Ce graphe doit être suffisamment générique pour que l'on puisse y appliquer des algorithmes éprouvés issus de théories reconnues sur les graphes.

Mécanismes de validation

Une étude doit être menée pour que le mécanisme de validation puisse utiliser les propriétés du graphe de dépendances sans refaire des validations déjà faites. Il faut pour cela établir des règles qui permettent de savoir quelles sont les validations les plus intéressantes à faire ; celles qui valident le plus de contraintes. Il faut également être capable de savoir quelles sont les contraintes à réévaluer lorsque l'on modifie, crée ou supprime des nœuds. Ce travail devra s'appuyer sur une recherche en théorie des graphes.

Contraintes et inter dépendances

Les résultats précédents sur la modélisation des contraintes et leur validation doivent être mis en pratique lorsque l'on voudra gérer les configurations inter dépendantes. De plus, le maintient d'une configuration globale dont une partie des données sont distribuées doit être réalisé de manière sure, transparente et extensible, de sorte qu'il soit aisé d'ajouter ou de supprimer des équipements dans la configuration globale.