Université du Québec à Montréal
Laboratoire de Téléinformatique

Traffic Enginneering dans l'interconnexion de SANs

Spécifications

Benoît CHRISTOPHE
13 mai 2003

Révision : 2.1

Aperçu du document

Ce document présente l'architecture, la spécification et la conception des différentes classes, interfaces sur lesquelles nous allons nous baser pour déployer un module de « traffic engineering » au travers de SANs.
Ce document présente l'architecture de façon globale puis plus en profondeur. De plus, ce document présente les fonctionnalités des différents modules que nous intégrerons dans notre projet. Je présente également le diagramme de classes composant le projet, ainsi que des diagrammes de séquence permettant de visualiser les différentes situations dans lesquelles nous pouvons nous retrouver.
Enfin, ce document présente les différents environnements de programmation que j'utiliserai ainsi que les équipements CISCO avec lesquels je travaillerai.

Statut du document

Version : 2.1

Ajout par rapport aux versions précédentes :

- orientation totale des communications entre les diverses entités vers Web Services
- définition d'un Workflow de Web Services
- technologies à mettre en place pour gérer ce Workflow

1 Introduction

1.1 Objectif

L'objectif de ce système est de réaliser un module capable de fournir de la meilleure façon, des demandes de services émis par des clients avec des disponibilités réseaux. Le module de « traffic engineering » fonctionne comme suit : des clients font des demandes de services avec un horaire précis, une durée donnée, une bande passante voulue et une qualité de services attendue. Notre module va périodiquement regarder si de nouvelles requêtes ont été ajoutées. Si c'est le cas, notre module se charge de redéfinir le réseau, afin d'utiliser au mieux les ressources disponibles en incluant les nouvelles requêtes clientes. Une fois que la requête cliente est prise en compte, notre système assure que la requête cliente bénéficiera du service qu'elle souhaite.
Exemple d'une demande de service à un moment donné :

Le but du client est d'utiliser un service de 7h à 18h avec une bande passante ayant une largeur de bande d'au moins 45Mb/s (T3), avec une qualité de services maximale (qui permette d'assurer au client que sa demande une fois prise en compte sera traitée sans être préemptée par une autre demande cliente).

Nous avons actuellement une bande passante disponible d'une largeur d'environ 50 Mb/s.
Néanmoins, nous ne disposons pas de ces 50Mb/s pour le temps demandé par le client.
Il nous faut donc, pour prendre en compte la demande du client, réorganiser les différentes requêtes qui sont déjà prise en compte, afin de voir si il est possible d'allouer 45Mb/s au client pendant le temps qu'il désire. Si notre module ne peut trouver de solutions avec la configuration actuelle, il devra calculer quelle commande de ressources nous serons amené à faire afin de prendre en compte la requête du client.

Notre système sera basé sur une architecture utilisant les « Web Services ». En effet, nous allons devoir communiquer avec des entités hétérogènes, n'ayant pas le même protocole de communication, ne connaissant pas les différentes interfaces que chaque tiers a développé. De plus, nous allons essayer de faire communiquer ces Web Services entre eux. Nous créerons ainsi un Workflow de services web.

1.2 Notions et termes

SAN : Storage Area Network, réseau basé sur des connexions Fibre Channel, dédié aux stockage de données.

iSCSI : protocole utilisé dans l'interconnexion de SANs

FCIP : protocole utilisé dans l'interconnexion de SANs

WSDL : Web Services Description Language

Workflow de « Services Web » : entité qui va nous permettre de décrire et de contrôler les communications entre « Web Services »

WSCL : Web Services Conversation Language : langage permettant d'orchestrer les différentes interactions entre des « Web Services ».

1.3 Connaissances employées

- Web Services
- XML
- Java
- Algorithmes d'ordonnancement
- Techniques de « traffic engineering »

1.4 Répartition du travail

Nombre de développeur : 1
Les différentes étapes constituant le projet sont décrites comme ci-dessous :

- Implémentation des différentes interfaces utilisant des Web Services
  - Spécifications à fournir
  - Implémentation à réaliser
  - Documentation à réaliser
- Implémentation du module « admission control »
  - Spécifications des méthodes à fournir
  - Implémentation à réaliser
  - Documentation à réaliser
- Implémentation d'une interface cliente
  - Spécifications des méthodes à fournir
  - Implémentation à réaliser (WebServices)
  - Documentation à réaliser

2 Exigences et contraintes

2.1 Cas d'usage

Notre projet s'inscrit dans un cadre bien précis de prise en compte de requêtes clientes :

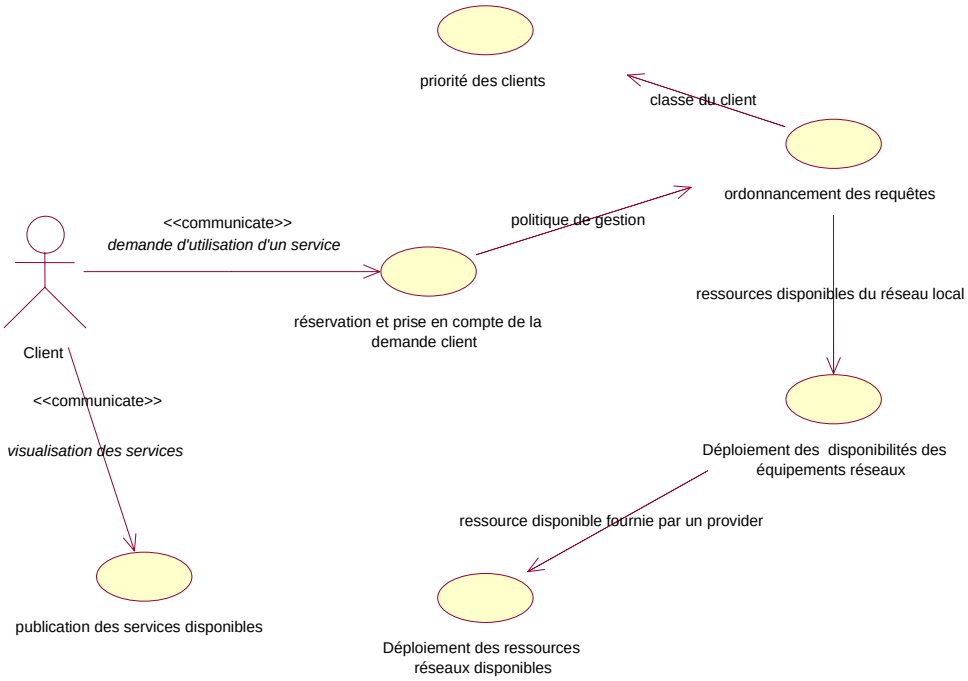


Figure 1 : cas d'usage

2.1.1 L'acteur

Notre système aura un seul type d'acteur en la présence d'un client qui va d'abord regarder les services que nous pouvons offrir puis qui va émettre des requêtes pour utiliser un service.

2.1.2 Les différents états

Ce cas d'usage montre les différentes étapes de notre système. Une fois la requête du client émise, il nous faut la prendre en compte dans un serveur LDAP. Ensuite, nous devons tenir une politique de gestion afin de savoir comment prendre en compte la

requête cliente. Enfin, il nous faut déployer notre réseau afin d'avoir les ressources suffisantes pour traiter toutes les requêtes.

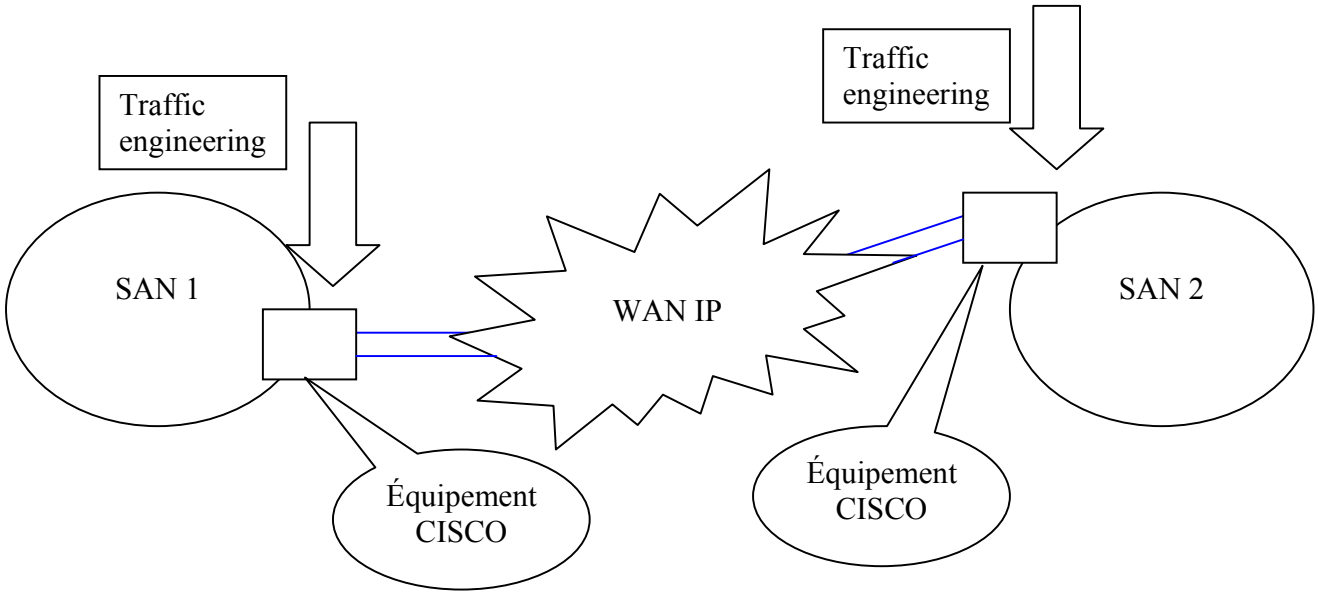
2.2 Contraintes

La communication entre le client, les différentes entités et notre module, doit se faire par « Web Services ». De plus, notre module se décompose en sous modules. Un sous module principal chargé de récupérer les informations données par les composants réseaux via « Web Services » et de les transmettre aux autres sous modules via des « Web Services ». Cela représente un workflow que nous devrons être capable de gérer.

3 Architecture

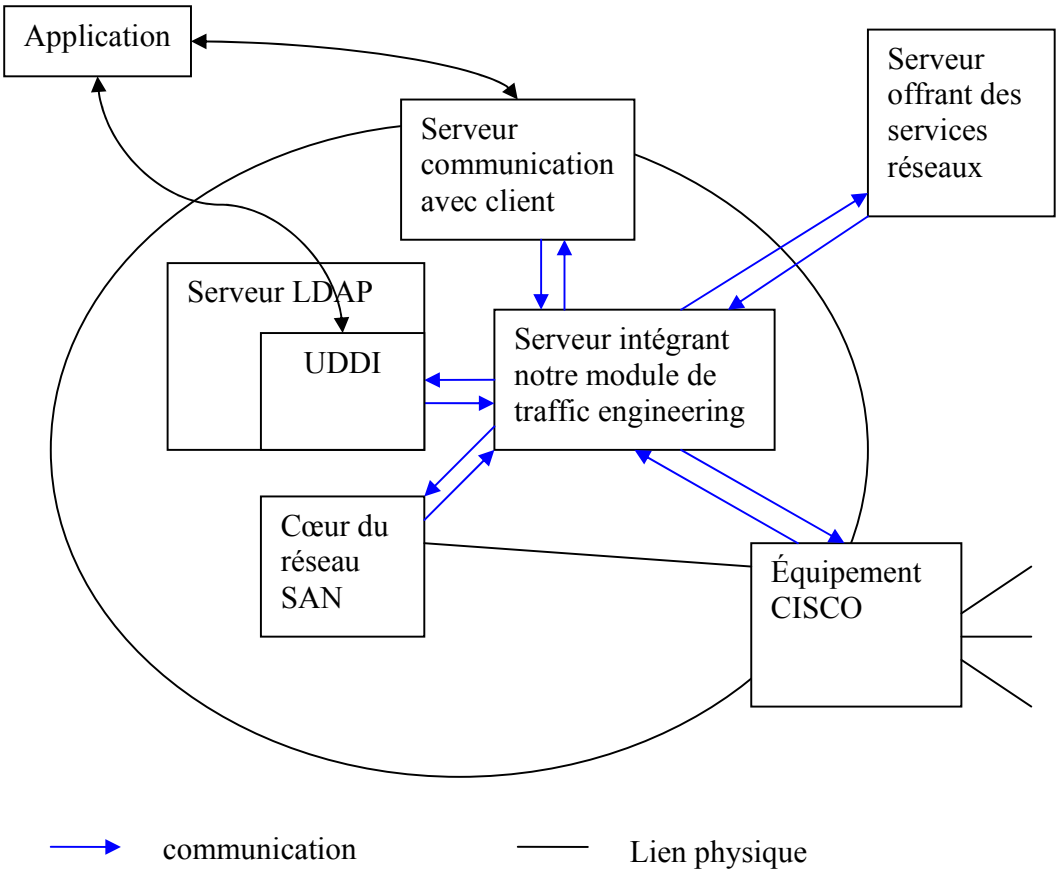
3.1 Vue d'ensemble

Notre projet de « traffic engineering » se situe au niveau des points terminaux de SANs, pour ordonnancer au mieux les requêtes interconnectant les SANs.



De manière plus précise, nous allons intégrer entre le cœur du réseau SAN et l'équipement CISCO permettant de faire des interconnexions, un module permettant de faire du contrôle d'admission et de l'ordonnancement de demandes de services.

3.2 Vue locale



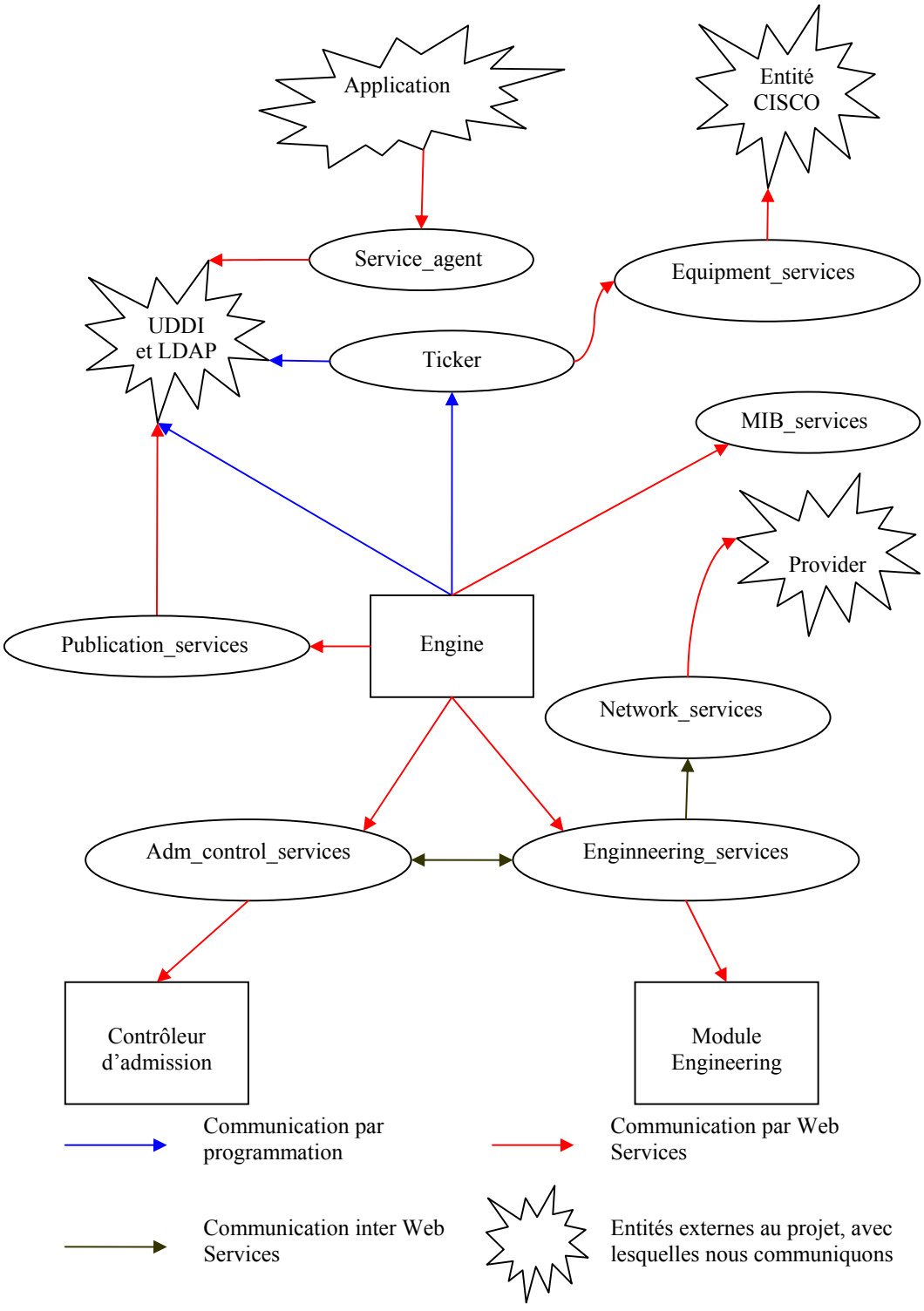
Ce schéma présente les différentes entités qui se trouveront à l'intérieur de chaque SAN et qui communiqueront avec notre module de traffic engineering. Je présente ici le rôle de chacune des entités :

- Serveur LDAP : stockera toutes les informations concernant les demandes de services des clients, les scripts de configuration des équipements CISCO et l'état le plus récent de notre configuration réseau (cette sauvegarde est faite pour accélérer l'analyse périodique de notre réseau)
- Équipement CISCO : permet de réaliser les interconnexions entre des SANs distantes. Les MIBs contenues dans les équipements CISCO nous permettrons de retrouver des informations sur l'état des routes existantes : la métrique, la possibilité d'ouvrir des connexions…
- Serveur offrant des services réseaux : ce serveur est celui à qui notre module devra s'adresser pour des ressources réseaux. Ce serveur nous offrira des services en terme de routes : il pourra nous offrir telle route à telle capacité…
- Annuaire UDDI : nous servira à publier l'état de notre réseau : la capacité que nous pouvons offrir, l'état de notre réseau…
- Serveur communication client : ce serveur va nous servir à communiquer avec les clients désireux d'émettre une requête.

- Module de « traffic engineering » : notre module va communiquer avec les précédentes entités, afin d'avoir une vue de l'état des routes d'interconnexions des SANs

3.3 Architecture du module de « traffic engineering »

Comme le montre la figure 3, nous avons plusieurs communications de nature diverses, entre les différentes entités. Je vais donc expliquer la structure de notre module de « traffic engineering ».


Application
Entité CISCO
Service_agent
Equipment_services
UDDI et LDAP
Ticker
MIB_services
Provider
Publication_services
Engine
Network_services
Adm_control_services
Enginneering_services
Contrôleur d'admission
Module Engineering
Communication par programmation
Communication par Web Services
Communication inter Web Services
Entités externes au projet, avec lesquelles nous communiquons

Ce schéma montre que toutes les communications entre notre module de « traffic engineering » et les entités permettant la collecte d'informations, seront réalisées par le biais de « Web Services ».

Dans ce schéma, plusieurs entités sont présentes. Je vais décrire à présent leur rôle.

- LDAP_services : il s'agit du fichier WSDL proposant des services pour retrouve des renseignements se trouvant sur le serveur LDAP. Celui-ci va publier des services concernant les demandes clients, les scripts CISCO… Nous aurons donc à aller interroger ces services via le protocole SOAP ou http

- MIB_services : il s'agit du fichier WSDL permettant de récupérer des informations MIBs des équipements CISCO. Nous pourrons alors récupérer des informations concernant le trafic

- Network_services : ces « Web Services » nous permettrons d'aller interroger d'autres Web Services qui auront été publiés par un provider. Ils concernent l'allocation de ressources réseaux

- Equipment_services : ces « Web Services » nous serviront à configurer les équipements CISCO sur lesquels nous serons amené à travailler

- Publication_services : par le biais de Web Services, nous publierons l'état de notre réseau et la disponibilité de celui-ci

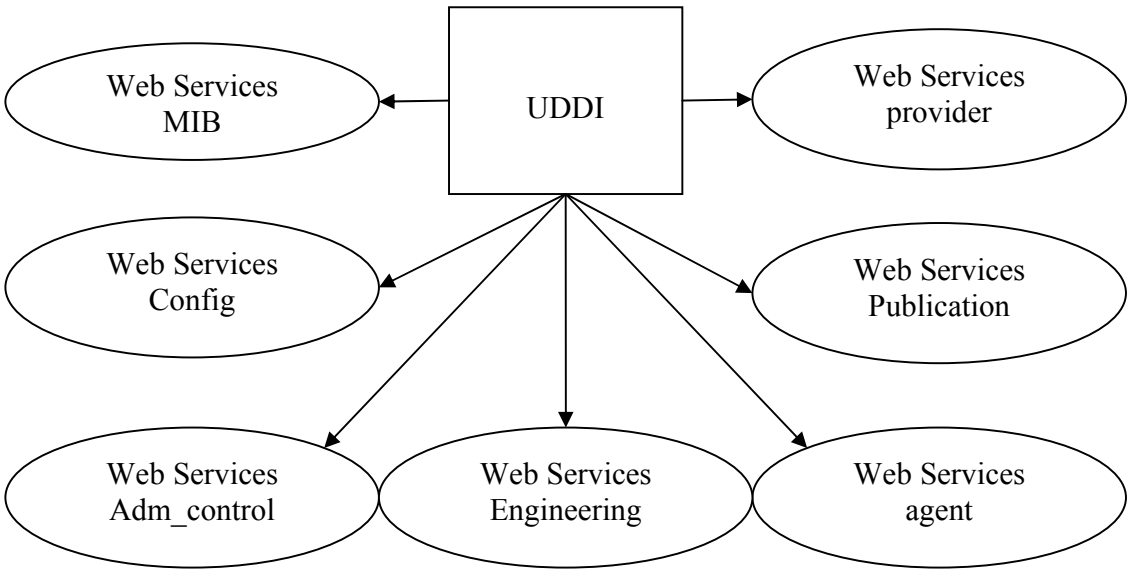
- Service_agent : ces « Web Services » nous serviront à enregistrer une demande de service d'un client ainsi qu'à communiquer avec le client afin que celui-ci puisse voir de quels services nous disposons

- Adm_control_services et Engineering_services : Ces « Web Services » nous servirons pour communiquer entre nos sous modules « Admission_control » et « Engineering » et le module « Engine ».

Ainsi, par l'intermédiaire de « Web Services », le module appelé « Engine » centralisera les données reçues, enverra des demandes d'informations… sans avoir à se préoccuper des différents protocoles à respecter puisse qu 'il ne communiquera jamais de façon directe avec les entités. Puis, il se servira des « Web Services » des interfaces « Adm_control_services » et « Engineering_services » pour envoyer les informations

3.4 Publication des services

Notre système va proposer des services. Nous allons utiliser un registre central qui répertoriera les fichiers WSDL des différentes entités : un annuaire UDDI.



3.5 Fonctionnement du système

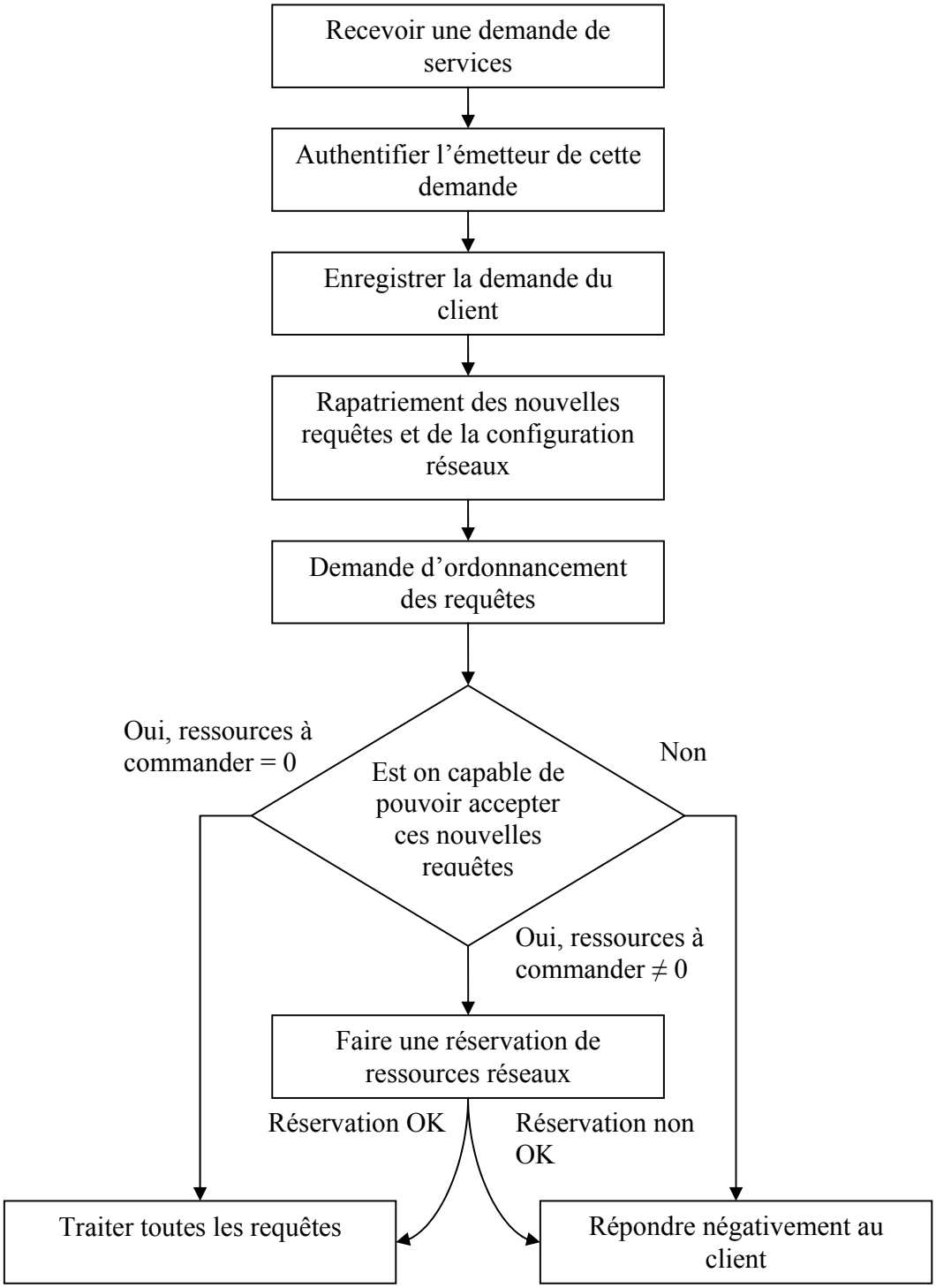
Un utilisateur cherche des services publiés et rendus disponible grâce à un annuaire UDDI. Cet utilisateur trouve la description du service souhaité. Il envoi alors sa demande de service via des « Web Services » en donnant la description de sa demande de service : horaire, temps, bande passante, priorité. Cette demande est acheminée à un contrôleur de service : « Service_agent » qui pourra traiter la requête.

A la réception de la demande de service, le « Service_agent » contrôle l'identité du client puis va inscrire la requête cliente. Pour cela, il utilise des « Web Services » du module « LDAP_services » afin de réaliser ces deux étapes.

Ensuite, le module « Engine » utilise des « Web Services », de façon périodique, afin de regarder si de nouvelles requêtes clientes sont arrivées. Il rapatrie donc toutes les informations de configuration du réseau puis il demande au module « Engineering » (via Web Services) d'ordonnancer les requêtes.

Une fois les calculs effectués par le module « Engineering », ce dernier utilise des Web Services du module « Engineering_services » pour faire une commande de ressources réseaux auprès d'un fournisseur de services. Ensuite, il passe la configuration trouvée au module « Admission_control » (via Web Services). Enfin, il renseigne le module « Engine » en lui donnant la nouvelle configuration réseau, les scripts CISCO…

Ainsi, on peut résumer le fonctionnement du système par le schéma suivant :



4 Les différents modules

Je présente ici les différents modules de l'architecture de notre système. Je commencerai par détailler les modules offrant des « Web Services » et constituant la partie haute de notre architecture (c'est à dire en relation avec le client ou en relation avec notre module « Engine »). Ensuite, je décrirai les modules de couche basse à savoir les modules internes permettant l'ordonnancement et le contrôle des services offerts aux clients.

4.1 Les interfaces « Web Services »

4.1.1 LDAP_services

Ce module offre des « Web Services » nous permettant d'avoir (ou d'ajouter) des informations dans le LDAP. En effet, le LDAP va nous servir à stocker des demandes de services des clients, des scripts de configuration CISCO et une sauvegarde de l'état des routes de notre réseau.

Les différents services que doit offrir ce module sont donc :

- Récupération des requêtes clientes
- Ajout de requêtes clientes
- Récupération de script de configuration CISCO
- Ajout de script de configuration CISCO
- Sauvegarde de l'état du réseau (servira à accélérer la vérification du système qui devra être faite périodiquement)
- Enlever une requête d'un client
- Changer le statut d'une requête cliente (de l'état waiting à l'état OK)

Les différents « Web Services » sont donc :

- get_requests()
- add_request(request : Request)
- remove_request(request : Request)
- get_scripts()
- add_script(script : Script)
- update_network_config(config : Config)
- change_request_status(request : Request)

Avec Request Script et Config différentes structures pouvant être comprise par les « Web Services »

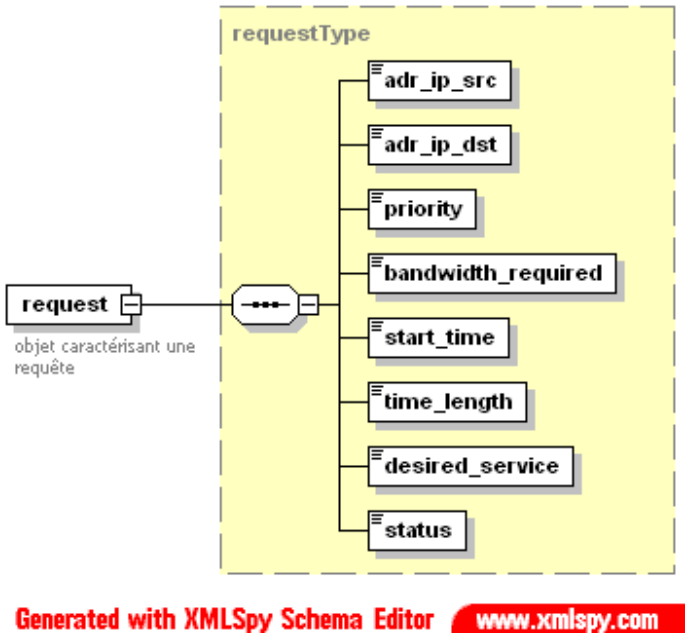
Par exemple, le début du fichier WSDL représentant la description des « Web Services » du LDAP sera :

```
<wsdl:definitions xmlns:ldapsd="http://www.teleinfo.uqam.ca/traffic_eng/services"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.teleinfo.uqam.ca/traffic_eng/services" name="LDAP_services">
    <wsdl:types>
        <xsd:schema targetNamespace="http://www.teleinfo.uqam.ca/traffic_eng/services">
            <xsd:import namespace="http://www.teleinfo.uqam.ca/traffic_eng/services"
schemaLocation="file:///D:/stage/stage/projet/partie-xml/LDAPSchema.xsd"/>
        </xsd:schema>
    </wsdl:types>
…..
```

Ce que l'en-tête du fichier montre est l'inclusion du schéma « LDAPSchema » qui va contenir les définitions des types complexe comme « Request » « Script » ou « Config ».

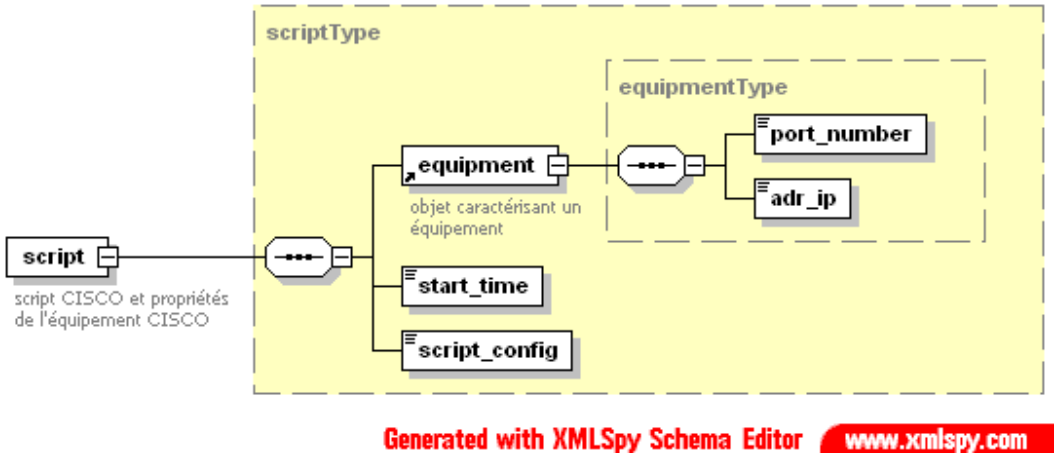
Ainsi, les figures suivantes montrent les définitions des différentes structures :

4.1.1.1 Structure Request

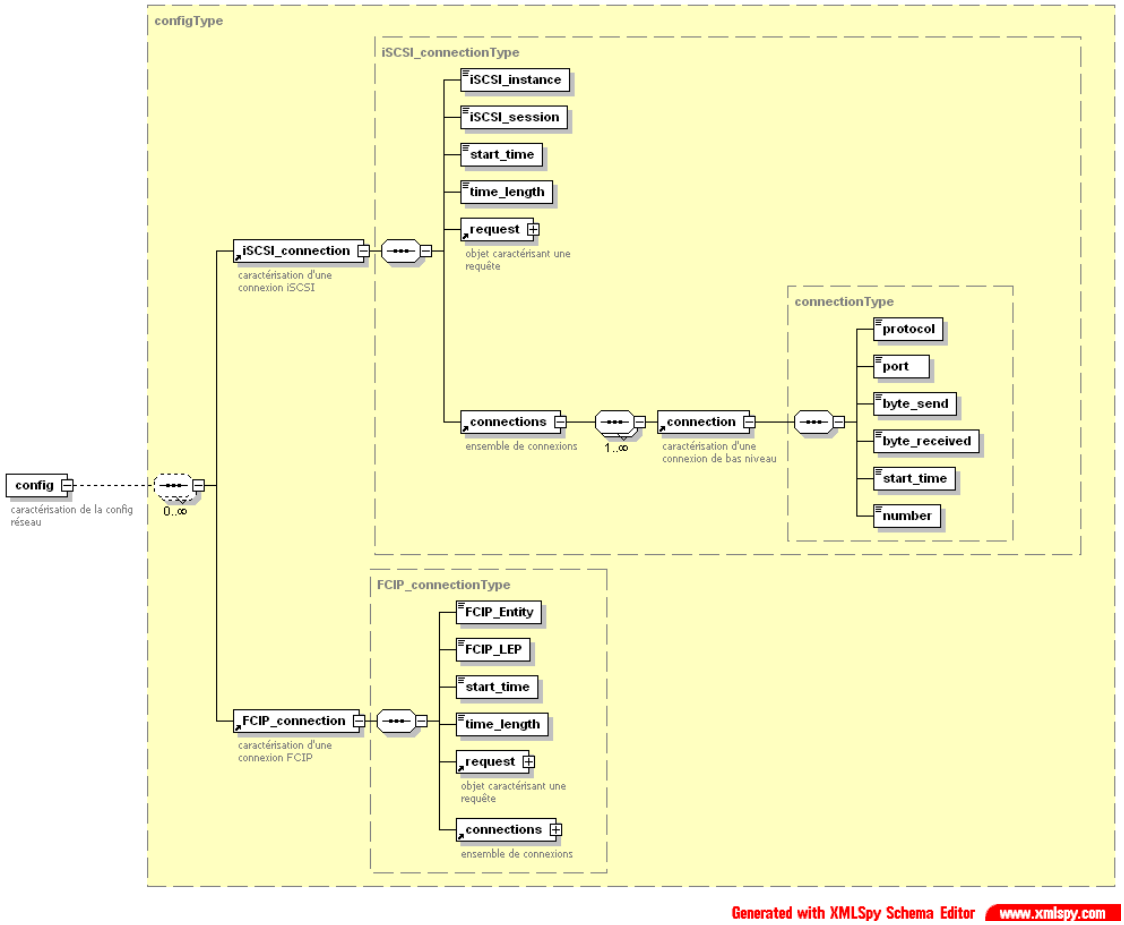


Comme on le verra section 4.3, cette structure est la même que celle définie comme une structure de classe qui nous permettra de traiter par programmation les informations.

4.1.1.2 Structure Script



4.1.1.3 Structure Config



Ce schéma n'a pas été déployé totalement pour plus de visibilité. Aussi, on voit qu'une structure « Config » possède plusieurs types complexes internes : « iSCSI_connection », « FCIP_connection », « connection », « request ».

4.1.2 MIB_services

Ce module offre des « Web Services » nous permettant de recevoir des informations concernant l'état actuel des routes utilisées par les différents protocoles permettant L'interconnexion de SANs.

Ces « Web Services » doivent être basés sur les protocoles de communication d'interconnexion SAN. On doit donc retrouver toutes les différentes entités présentes dans le protocole de communication iSCSI et dans le protocole de communication FCIP.

Ainsi, les « Web Services » dont nous aurons besoin seront :

- Pour une communication basée sur le protocole iSCSI :
    - Web Services servant pour les statistiques du réseau :
        - Retrouver toutes les instances iSCSI
        - Retrouver les sessions iSCSI associées à une instance
        - Retrouver les connexions TCP associées à une instance et une session
        - Retrouver la date d'ouverture de la connexion TCP
        - Retrouver la bande passante utilisée par cette connexion TCP
        - Retrouver le nombre de bytes envoyés et reçus par cette connexion TCP
    - Web Services servant pour la prise en compte de nouvelles requêtes :
        - Récupérer les nœuds iSCSI existants
        - Retrouver le nombre de connexions possible à allouer pour chaque nœud iSCSI
        - Retrouver le nombre de bytes maximum que l'on peut transférer pour une nouvelle ouverture de connexion sur un nœud donné
- Pour une communication basée sur le protocole FCIP :
    - Web Services servant pour les statistiques du réseau :
        - Retrouver toutes les entités FCIP
        - Retrouver tous les points terminaux associés à ces entités
        - Retrouver toutes les connexions TCP associées à un point terminal
        - Retrouver la date d'ouverture de la connexion TCP

        - Retrouver la bande passante utilisée par cette connexion TCP
        - Retrouver le nombre de bytes envoyés et reçus par cette connexion TCP

    - Web Services servant pour la prise en compte de nouvelles requêtes :

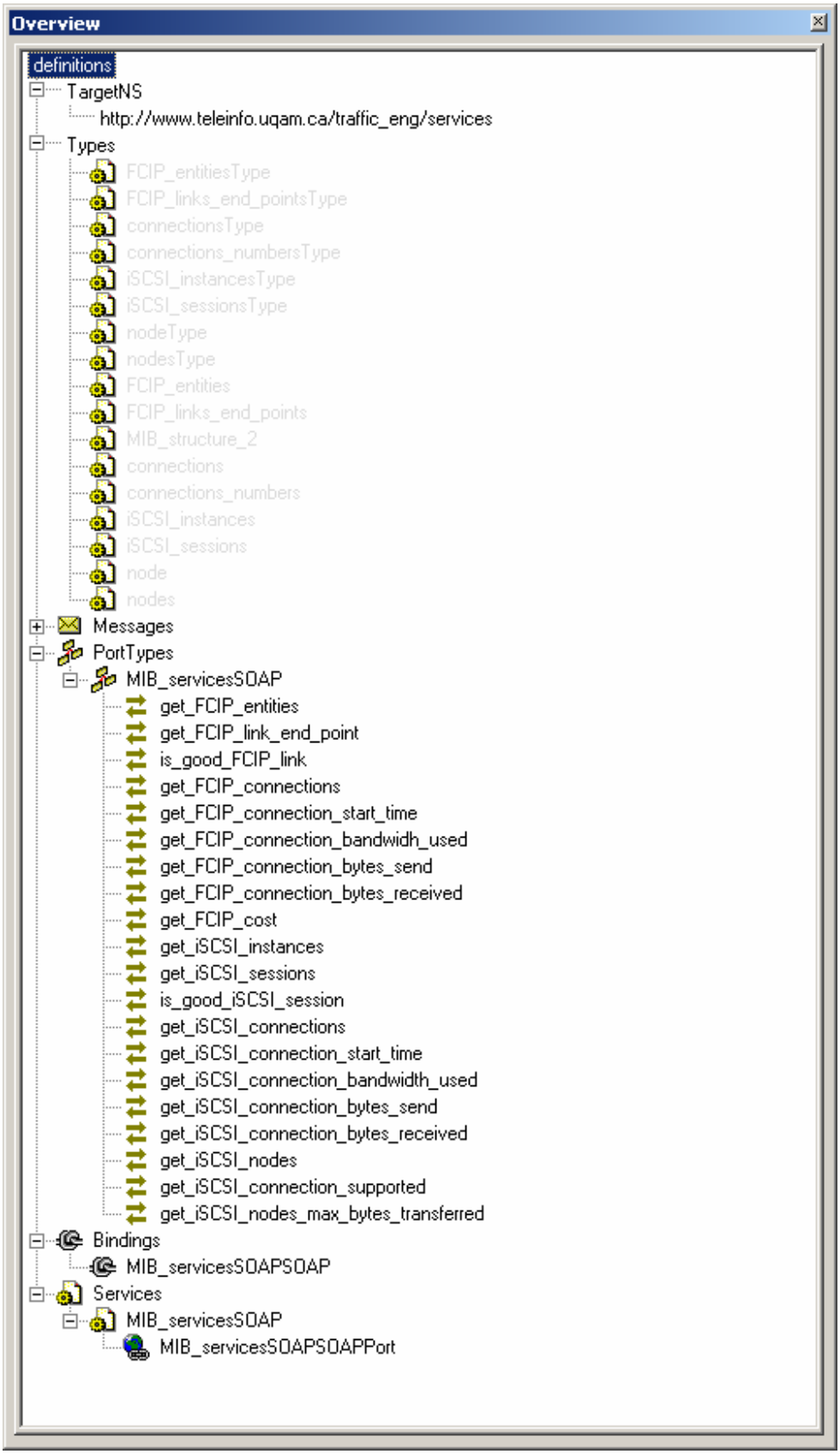
        - Retrouver le coût de l'établissement d'une nouvelle interconnexion de SAN par le protocole FCIP

Ces « Web Services » vont se servir des MIBs présentes dans les équipements CISCO pour aller chercher ces informations. A partir de ces informations, notre programme interne d'ordonnancement de tâches pourra tirer des statistiques, notamment sur le débit de transfert de données.

Les principales MIBs utilisées seront donc les MIBs suivantes :

- iSCSI_MIB
- FCIP_MIB
- TCP_MIB

Je ne décrirai pas les fonctions comme pour le module LDAP_services du fait de leur nombre. Néanmoins, voici un extrait du fichier WSDL correspondant à la description des « Web Services » du module MIB_services.


Overview
definitions
TargetNS
http://www.teleinfo.uqam.ca/traffic_eng/services
Types
FCIP_entitiesType
FCIP_links_end_pointsType
connectionsType
connections_numbersType
iSCSI_instancesType
iSCSI_sessionsType
nodeType
nodesType
FCIP_entities
FCIP_links_end_points
MIB_structure_2
connections
connections_numbers
iSCSI_instances
iSCSI_sessions
node
nodes
Messages
PortTypes
MIB_servicesSOAP
get_FCIP_entities
get_FCIP_link_end_point
is_good_FCIP_link
get_FCIP_connections
get_FCIP_connection_start_time
get_FCIP_connection_bandwidh_used
get_FCIP_connection_bytes_send
get_FCIP_connection_bytes_received
get_FCIP_cost
get_iSCSI_instances
get_iSCSI_sessions
is_good_iSCSI_session
get_iSCSI_connections
get_iSCSI_connection_start_time
get_iSCSI_connection_bandwidth_used
get_iSCSI_connection_bytes_send
get_iSCSI_connection_bytes_received
get_iSCSI_nodes
get_iSCSI_connection_supported
get_iSCSI_nodes_max_bytes_transferred
Bindings
MIB_servicesSOAPSOAP
Services
MIB_servicesSOAP
MIB_servicesSOAPSOAPPort

4.1.3 Network_services

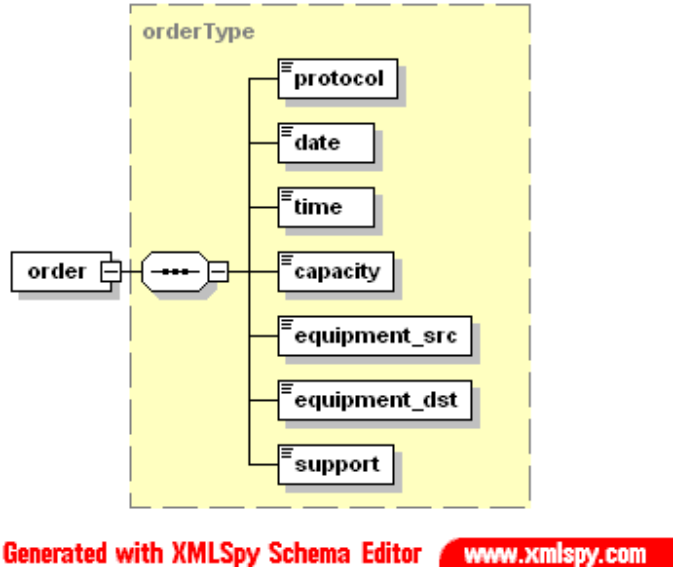
Ce module est chargé de nous apporter des informations sur les capacités réseaux qu'un provider peut nous allouer. Il se charge également de passer un ordre de commande de ressources réseaux. Les fonctionnalités que ce module doit offrir sont donc :

- Demande d'informations sur les routes pouvant nous être allouées
- Envoi d'un bon de commande en ressources réseaux : précision du type de support, du type de connexion, de l'adresse source et de destination, de la date de l'allocation, du temps.

On aura donc deux « Web Services » du type :

- get_available_resources()
- order_available_resources(order : Order)

Là encore, la structure Order sera analysable par le « Web Services » car elle sera définie dans le fichier WSDL du module « Network_services »



Là encore, cette structure est la même que celle qui sera définie section 4.3

4.1.4 Equipment_services

Ce module sera chargé de configurer les équipements CISCO afin de prendre en compte les décisions prises par notre programme d'ordonnancement de tâches.

Ce module devra fournir le service suivant :

- Configurer les équipements CISCO

Cela va donc se traduire par le service suivant :

- Set_config(script : String)

Où le paramètre « script » est une chaîne de caractères.

4.1.5 Publish_services

Ce module va assurer la communication via des « Web Services » entre notre programme et le serveur UDDI.

En effet, une fois que des requêtes clientes ont été prises en compte, l'état du réseau a changé. De même, plus de services sont disponibles. Il faut donc publier ce nouvel état.

Ainsi, notre module sera chargé de fournir les services suivants :

- Publier une configuration réseau initiale
- Mettre à jour la configuration de notre réseau SAN

Soit comme traduction au niveau opération :

- publish_disponibilities(initial_config : Config)
- update_disponibilities(new_config : Config)

Ainsi, lors de chaque mise à jour de l'état du réseau, on renseigne les clients.

4.1.6 Service_agent

Ce module va nous servir à réaliser la communication entre le client et notre programme. D'une part ce module montrera au client les services dont nous disposons actuellement. D'autre part ce module sera chargé de recevoir des requêtes clients, d'appeler le serveur LDAP pour une authentification, de dire au serveur LDAP si la requête peut être ajoutée et de confirmer au client l'état de sa requête.

Les différents services que nous devons utiliser sont donc :

- Ajouter des requêtes clients
- Montrer les disponibilités que nous pouvons offrir

En effet, la fonction d'ajout de requêtes pourra se charger de récupérer une requête cliente, de l'envoyer au LDAP pour authentification et de renvoyer un message au client caractérisant la prise en compte de sa requête.
Nous aurons donc deux services :

- add_request(request : Request)
- show_disponibilities()

La encore, Request sera une structure définie par un schéma XML.

4.1.7 Adm_control_services et Engineering_services

Ces modules vont permettre d'établir directement des communications entre le contrôleur d'admission et le module « Engineering ». Ainsi, la remontée d'informations n'ira pas jusqu'au module « Engine ». Il y aura donc moins de messages envoyés à travers le réseau. De plus, ces deux modules vont décharger le travail du module « Engine ».

4.2 Les modules internes

Les modules interne sont composés des modules suivants :

- Engine : module principal chargé :
  - de la collecte d'informations réseaux par « Web Services »
  - du formatage de ces données en utilisant les différentes structures définies
  - de l'appel des différents autres modules internes

- Admission_Control : module chargé de contrôler si les exigences des requêtes clientes prises en compte sont bien respectées

- Engineering : module chargé du ré ordonnancement des tâches dans le cas où le contrôleur d'admission perçoit un problème ou dans le cas où de nouvelles requêtes sont à prendre en compte

- Ticker : module se lançant de façon périodique qui va nous permettre de tester l'état du réseau, de trouver de nouvelles requêtes, de lancer de nouveaux scripts de configuration.

4.2.1 Le module Engine

Ce module représente le cœur de notre projet. Il est chargé de rapatrier des informations réseaux, d'appeler les différents modules de vérification et d'ordonnancement de tâches (via des Web Services), de lancer le module « Ticker »et de passer des commandes au module « Network_services ». Il se charge donc de tout l'aspect communication avec les différents modules.

Les différentes fonctions de ce module devront nous permettre de faire fonctionner notre programme. Ainsi, les fonctions devront couvrir les points suivants :

- Récupérer l'état actuel du réseau (interrogation des MIBs)
- Récupérer de nouvelles requêtes clientes (interrogation LDAP)
- Formater les données par l'utilisation de structures (définies partie 4.3) (??)
- Appel du module « Admission Control » (via Web Services)
- Passage des données à ce module (via Web Services)
- Appel du module « Engineering » (via Web Services)
- Passage des données à ce module (via Web Services)
- Création du module « Ticker » (création d'un processus permettant de ne pas bloquer le travail du module)
- Réservation des ressources
- Appel du module « Engineering » après prise de connaissance de la réponse à une commande de ressources (via Web Services)
- Modification d'une requête afin de dire qu'elle est prise en compte (passage par le module LDAP_services)
- Suppression d'une requête cliente (passage par le module LDAP_services)

4.2.2 Le module « Admission_control »

Ce module aura une vue globale de l'ensemble des routes de notre réseau inter SANs. Il devra donc analyser les informations qui lui ont été fournies par le module « Engine » afin de savoir si les demandes clientes sont bien respectées. Les différentes fonctions seront donc :

- Charger la configuration courante
- Tester la configuration

4.2.3 Le module « Engineering »

Ce module est chargé de l'ordonnancement des requêtes clientes. Les différentes opérations que ce module devra réaliser sont :

- Charger la configuration courante

- Charger les nouvelles requêtes
- Ordonnancer les requêtes (renvoi d'une structure au module « Engine » vue section 3.5 via Web Services)
- Charger la nouvelle configuration (charger la configuration que le provider peut nous allouer)
- Prendre une décision

4.2.4 Le module « Ticker »

Ce module se servira également de « Web Services » pour communiquer avec différentes entités. En effet, ce module est chargé de « scanner » le serveur LDAP, afin de savoir

- si des requêtes clientes sont en attente d'être prise en compte
- si des scripts de configuration des équipements CISCO sont à lancer

De plus, il est chargé de communiquer par « Web Services » avec le serveur LDAP et les MIBs CISCO pour un test de la configuration courante.

Enfin, il doit communiquer par « Web Services » avec le module « Equipment_services » pour le lancement des scripts CISCO

Ce module sera mis dans un « thread » afin de ne pas gêner le travail du module « Engine ».

4.3 Les structures

Je définie ici les différentes structures au niveau programmation dont les modules internes aurons besoin pour analyser les données collectées par le module « Engine »

4.3.1 Structure Config

Cette structure sert à spécifier une configuration de notre réseau. Dans le projet d'interconnexion de SANs, je me suis basé sur les deux protocoles iSCSI et FCIP pour construire cette structure. La configuration de notre réseau est donc vue comme un ensemble de connexions de type FCIP ou iSCSI.

Cette structure contiendra donc deux attributs :

- config_iSCSI : Vector(Config_iSCSI)
- config_FCIP : Vector(Config_FCIP)

4.3.2 Structure Config_iSCSI

Cette structure est vue comme une instance iSCSI comprenant un ensemble de sessions iSCSI. Elle contient donc deux attributs :

- iSCSI_instance : Double
- iSCSI_sessions : Vector(iSCSI_session)

4.3.3 Structure iSCSI_session

Cette structure va être associée à une requête cliente. Elle va aussi avoir un attribut répertoriant toutes les connexions TCP associées à cette session. De plus, elle possède un identifiant de session. Enfin, une date d'ouverture de session et une durée lui sont associées :

- iSCSI_session : Double
- iSCSI_connection : Vector(Connection)
- request : Request
- start_time : String
- time_length : String

4.3.4 Structure Connection

Cette structure sert à définir une connexion. Dans notre étude, il s'agit de connexions TCP. Cette structure répertoriera les différents éléments caractérisant une connexion TCP :

- protocol : String
- number : long
- port : long
- bytes_send : double
- bytes_received : long
- start_time : String

4.3.5 Structure Config_FCIP

Cette structure matérialise une entité FCIP. Cette entité est vue comme un ensemble de points terminaux. Elle possède également un identifiant :

- FCIP_entity : Double
- FCIP_link_end_point : Vector(FCIP_LEP)

4.3.6 Structure FCIP_LEP

Cette structure est associée à une requête cliente. Elle comprend donc toutes les connexions TCP associées à la requête cliente. De plus, elle possède un identifiant, une date de création et une durée d'ouverture :

- FCIP_LEP_number : Double
- FCIP_connections : Vector(Connection)
- request : Request
- start_time : String
- time_length : String

4.3.7 Structure Request

Cette structure caractérise une requête cliente : les adresses source et de destination, la qualité de services, la bande passante voulue, l'heure de début du service, le temps du service et le service désiré. Les attributs seront :

- adresse_ip_src : String
- adresse_ip_dst : String
- priority : int
- bandwidth_required : Double
- start_time : String
- time_length : String
- desired_service : String
- status : String

4.3.8 Structure Decision

Cette structure est utilisée par le module « Engineering » pour renvoyer au module principal « Engine » le résultat de son calcul (booléen oui ou non) ainsi qu'une structure de bon de commande :

- state : boolean
- order : Order

4.3.9 Structure Order

Cette structure est utilisée par le module « Engineering ». Elle contient les attributs nécessaires pour que le module principal « Engine » puisse faire une commande de ressources réseaux auprès d'un provider. Ces attributs sont les suivants :

- protocol : String
- date : String
- time : String

- capacity : String
- equipment_src : String
- equipment_dst : String
- support : String

Afin de visualiser les différentes dépendances entre les structures, je fourni en annexe, un diagramme de dépendance entre les structures.

5 Notion de Workflow

Comme on la vu dans la section 4.1, il y a diverses entités proposant des « Web Services ». Certaines d'entre elles communiquent via des « Web Services ». Il est donc nécessaire de connaître les interactions qui vont entraîner l'appel d'un « Web Service » par un autre « Web Service ».

En effet, en contrôlant les interactions entre « Web Services », nous empêchons notre programme de faire des appels à nos « Web Services » avant que ceux ci aient terminé d'exécuter un calcul…

Il nous faut donc avoir une vue globale de notre système afin de pouvoir parfaitement gérer les transactions entre « Web Services ».

Dans notre cas, plusieurs interactions entre « Web Services » devront être contrôlées :

- Attendre que le module « Engine » ait reçu toutes les données qu'il attend avant de faire appel aux « Web Services » des modules « Admission_control » et « Engineering »

- Attendre que le module « Admission Control » ait fini de faire la vérification d'une nouvelle configuration réseau pour ensuite donner les informations de configuration au module « Engine »

- Attente du « Time out » du module « Ticker » pour relancer une évaluation de l'état de notre réseau.

Afin de réaliser le contrôle de flux des interactions entre « Web Services », nous allons nous servir (à confirmer) de la technologie WSCL développée par Hewlett Packard permettant de définir les interactions entre Web Services.

Un document WSCL est constitué de trois blocs :
- Les schémas XML des documents échangés durant l'interaction
- La description des interactions
- La description des transactions permettant le passage d'une interaction à une autre

Les documents WSCL pourront être enregistré dans un annuaire UDDI.

Il faudra donc définir toutes les interactions entre les « Web Services » pour produire un document WSCL.

6 Tables du LDAP

Les tables que le LDAP doit contenir, concernent :

- Les requêtes clientes
- Les informations de configuration CISCO
- L'état du réseau

6.1 Table de requêtes clientes

La table concernant les requêtes clients aura les mêmes champs qui ont été défini dans le fichier WSDL du module « LDAP_services »

- L'adresse source
- L'adresse de destination
- La qualité de service demandée
- La bande passante demandée pour le service
- Le temps de service demandé
- L'heure de départ du service demandé

6.2 Table des scripts CISCO

Concernant le stockage des scripts de configuration CISCO, il va nous falloir une table de données contenant les informations suivantes :

- Heure de lancement
- Script de configuration
- Équipement choisi
- Port choisi

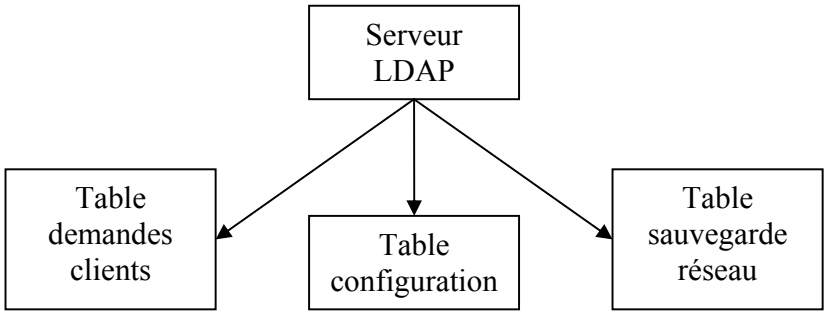
6.3 Table de sauvegarde de l'état du réseau

Concernant la sauvegarde courante du réseau, il nous faut sauvegarder les différentes connexions iSCSI et FCIP :

- Config_iSCSI
- Config_FCIP

6.4 Schématisation

Ainsi, le serveur LDAP est schématisé de la façon suivante :



Avec pour la table des demandes clients :

Nom client	Service demandé	Date de début	Durée	Priorité	Bande Passante	…
Client 1	Service 1	17052003	12	4	100 Mo	

De même pour la table de configuration des équipements CISCO :

Nom Équipement	Adresse IP	VSAN	Port	Date	Script	…
CISCO 5420 SN	132.168.137.45	2	12	17052003	…	

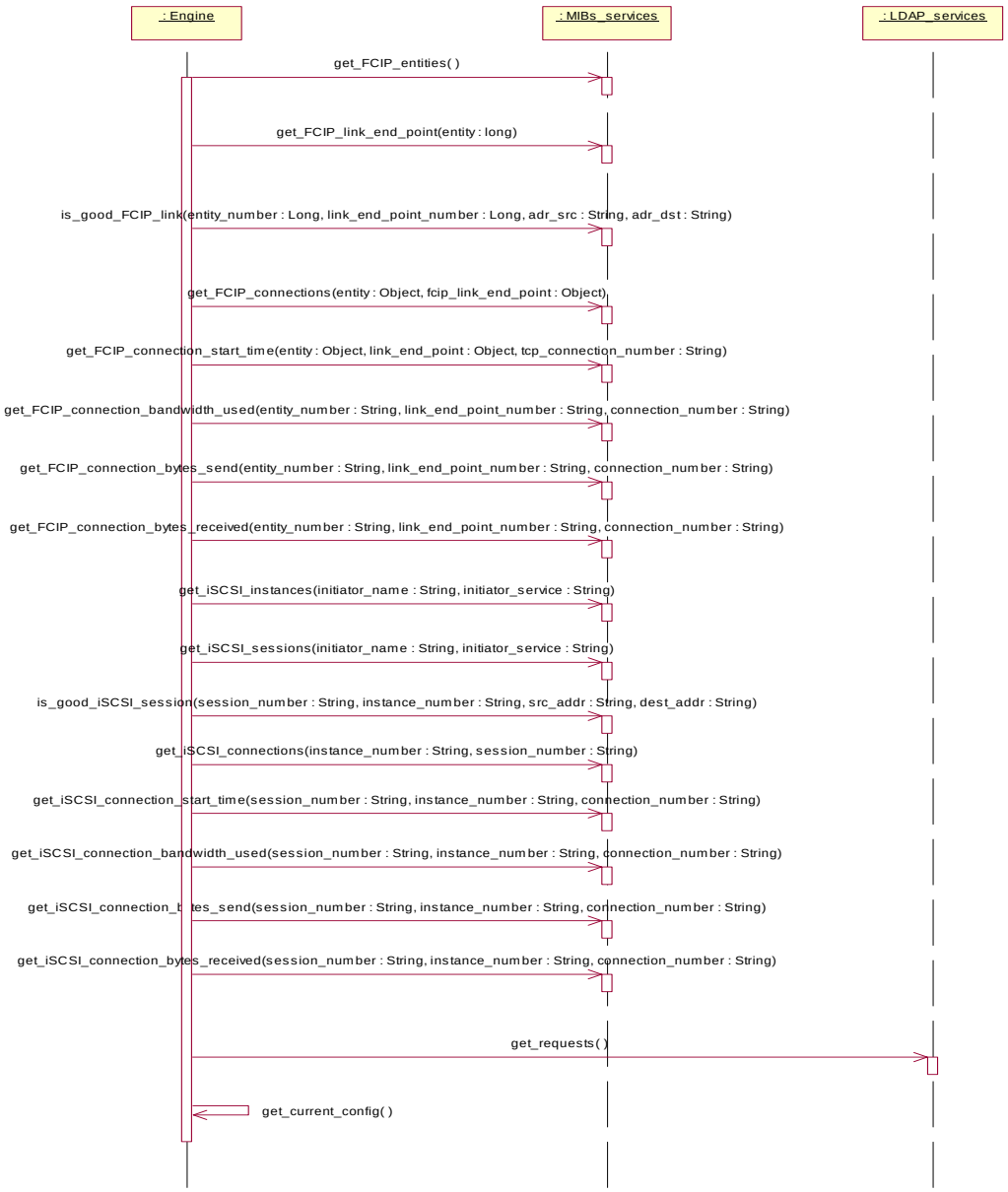
Enfin, pour la sauvegarde de l'état du réseau :

Type de connexion	Instance	Session	Connexion_TCP	Date ouverture	Durée	…
iSCSI	1	3	12	17052003	3600	

7 Diagrammes de communication

Je présente ici différents diagrammes de séquences qui se passeront entre les différents modules.

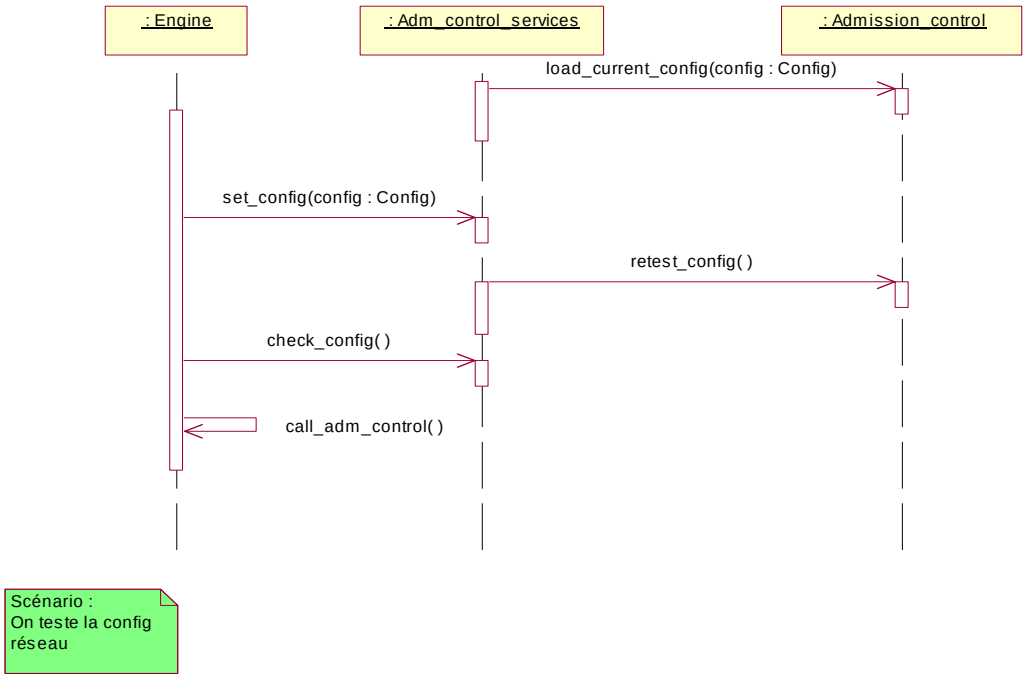
7.1 Collecte d'informations



Ce schéma traduit le fait qu'en lançant la fonction get_current_config(), le module « Engine » fera appel à tous les « Web Services » permettant de donner un état du réseau contenus dans les modules « MIB_services » et « LDAP_services ».

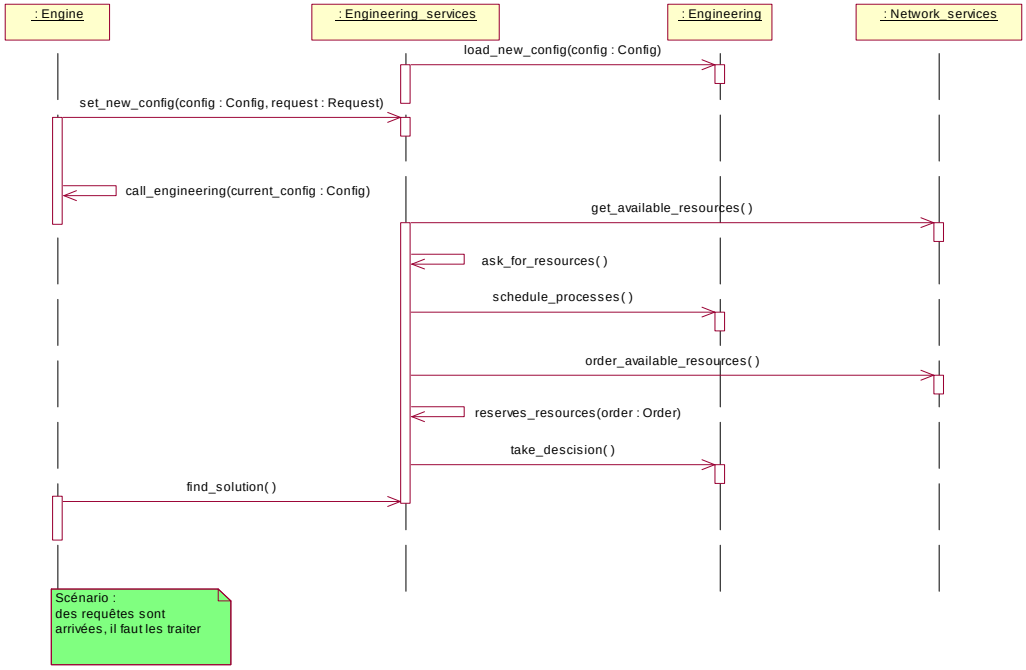
7.2 Envoi des informations aux modules internes

7.2.1 « Engine » vers « Admission_control »



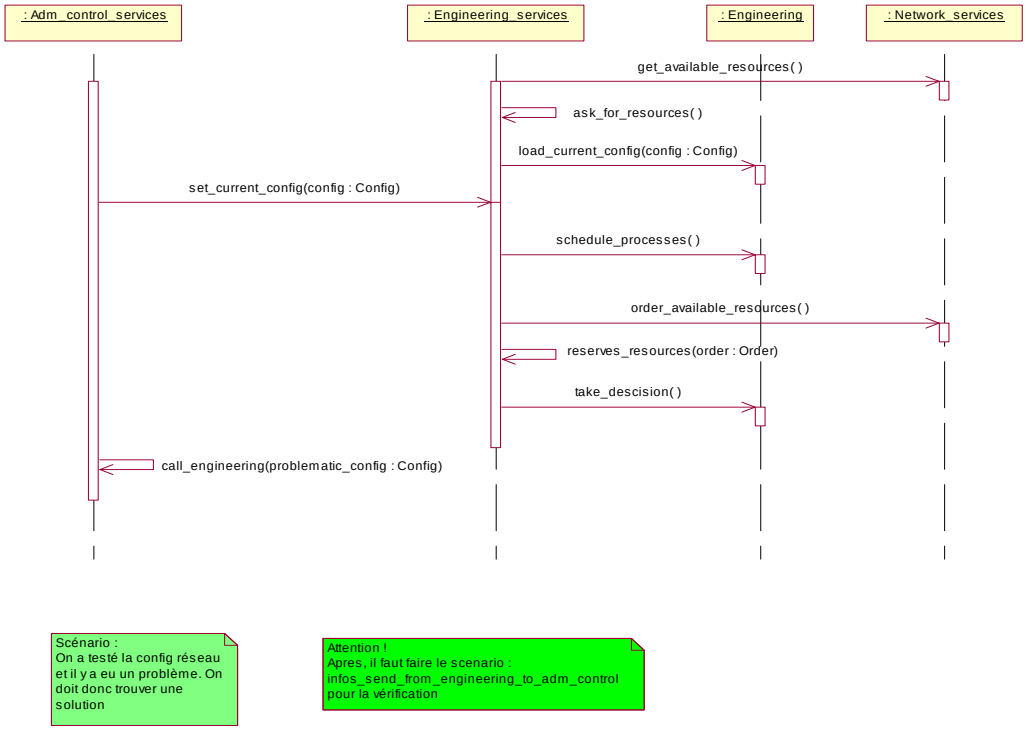
Ce diagramme vient à la suite du premier. Une fois les informations collectées par le module « Engine », celui-ci les passe au module « Admission_control ». Ce dernier va donc tester la configuration.

7.2.2 « Engine » vers « Engineering »



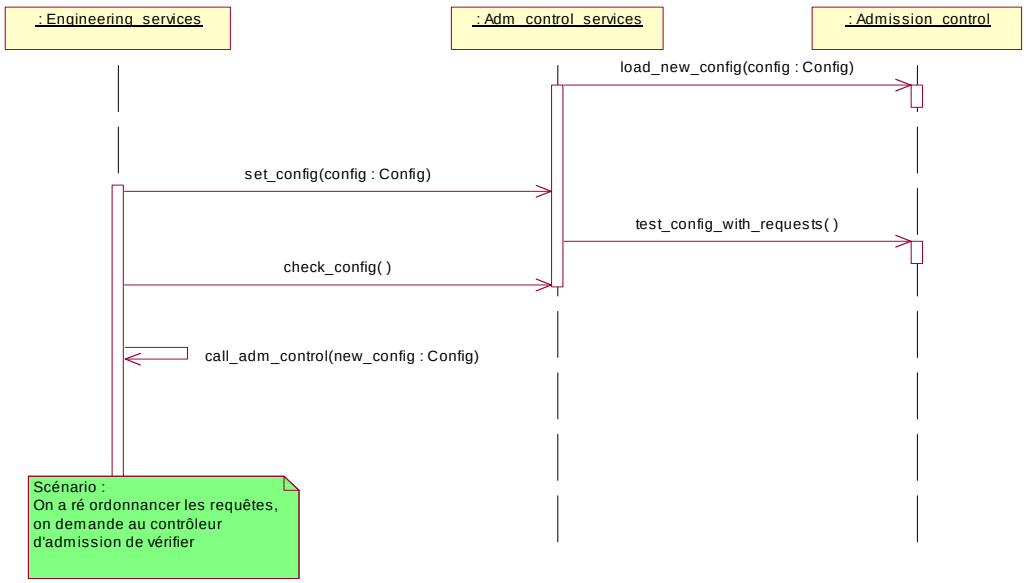
Le diagramme précédent se produit lorsque des requêtes clientes attendent d'être acceptées. Une fois les informations collectées par le module « Engine », celui-ci envoi les informations, via des Web Services, au module « Engineering ».

7.2.3 « Admission_control » vers « Engineering »



Le diagramme précédent traduit le fait qu'un problème est survenu lors du test de la configuration réseau effectuée par le contrôleur d'admission. Ce dernier envoi donc la configuration qui lui a posée problème au module « Engineering » via des Web Services. Ainsi, une communication directe est établie entre ces deux modules et le module « Engine » n'a pas à intervenir. Cela réduit donc le nombre de messages à échanger.

7.2.4 « Engineering » vers « Admission_control »



Le diagramme précédent est réalisé à chaque fois que le module « Engineering » doit ré ordonnancer des processus. En effet, après avoir ré ordonnancer, il est obligé (pour une question d'assurance) de passer la nouvelle configuration réseau au contrôleur d'admission, afin que celui-ci regarde si la configuration est valide. Là encore, la communication entre ces deux modules est directe.

7.3 Prise en compte de nouvelles requêtes clientes

Le schéma précédent concerne uniquement l'ajout de requêtes clientes dans le cas du protocole iSCSI. Le module « Engine » passe tout d'abord la configuration au module

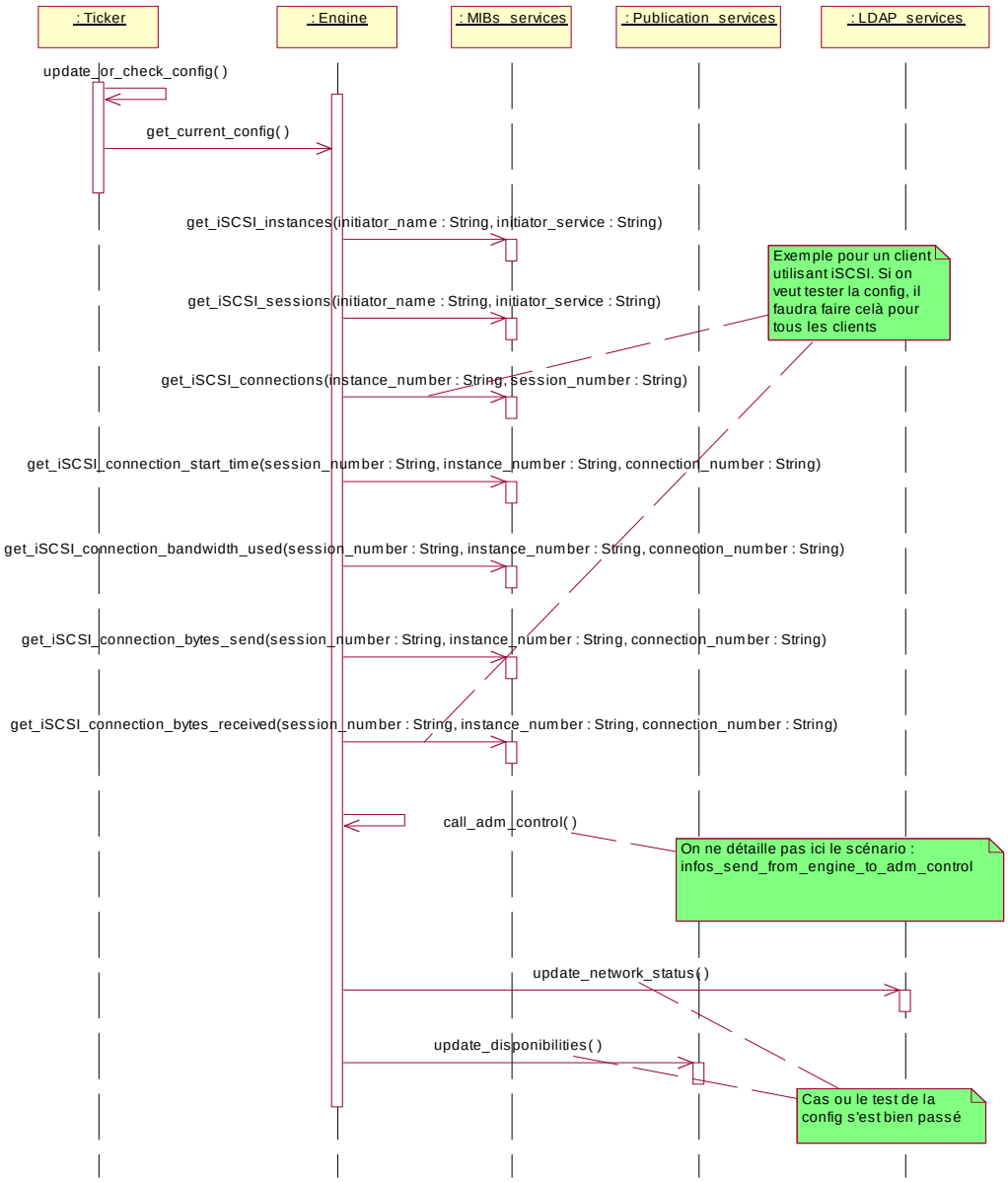
« Admission Control » afin que celui puisse voir si en ajoutant bêtement la requête d'un client, les autres requêtes sont toujours traitées avec la même qualité de services - cette étape peut être supprimée. Néanmoins, elle ferait gagner du temps en calcul car elle permettrait de ne pas toujours réordonnancer pour chaque nouvelle requête client -.

Ensuite, le module « Engineering » prend le relais. Il charge la configuration, les requêtes, les ressources qui sont a priori disponibles. Il ordonnance les processus puis il envoi une réponse au module « Engine ». Celui-ci se charge de réserver des ressources. Ensuite, il rappelle le module « Engineering » afin que celui-ci puisse prendre une décision (il ne peut pas la prendre avant car il ne sait pas si les ressources sont réserver).

Enfin, le module « Engine » rappelle le module « Admission_control » afin que celui-ci vérifie la nouvelle configuration réseau.

Après ces étapes, le module « Engine » se charge de charger les scripts et de changer l'état des requêtes clientes (waiting → OK) ou (waiting → ¬OK) en utilisant les « Web Services » des interfaces vu section 4.1.

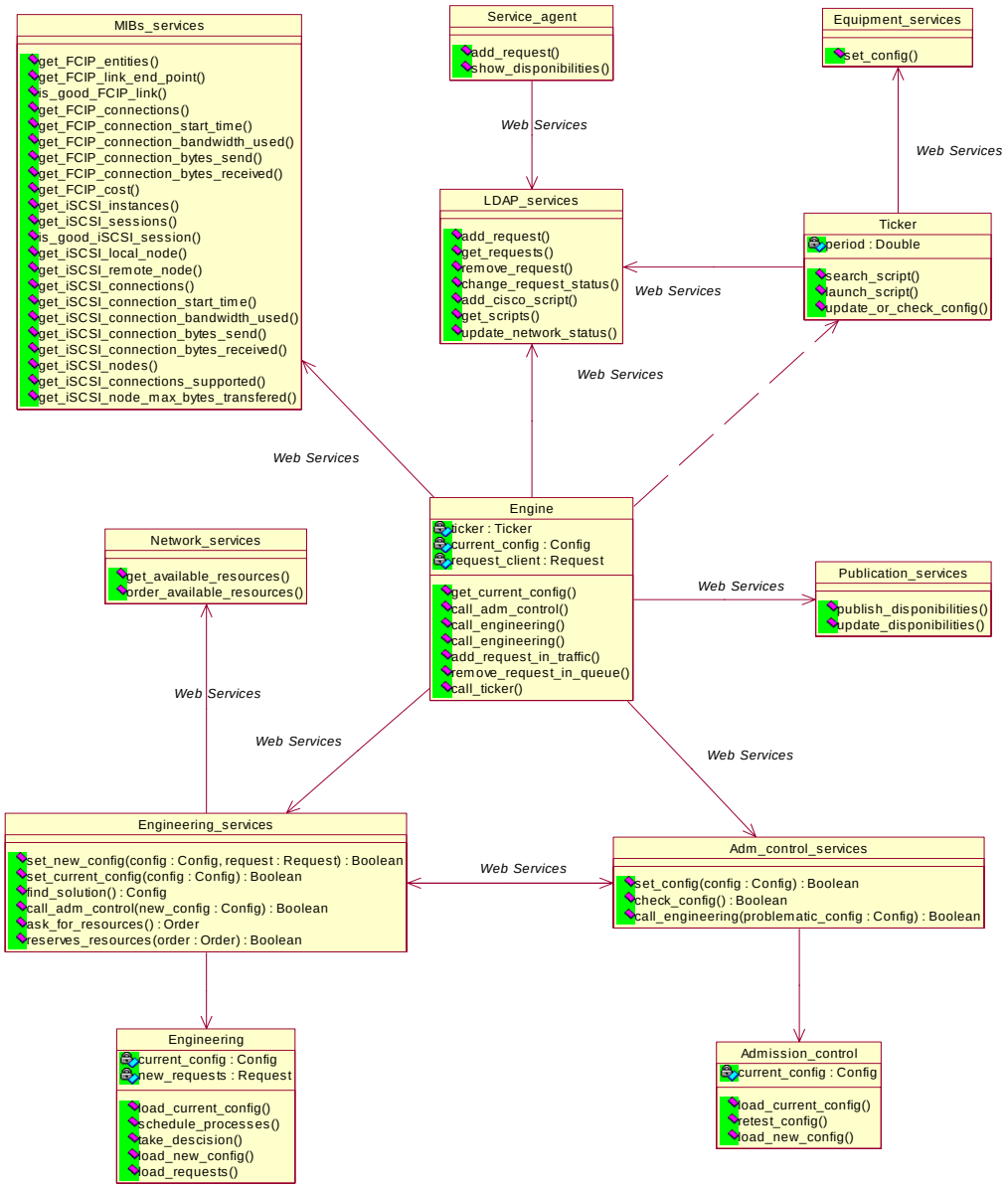
7.4 Vérification de l'état du réseau



Ce schéma présente la façon avec laquelle la vérification de l'état du réseau sera effectuée. Tout d'abord le module « Ticker » dans un « Thread » séparé, demandera périodiquement au module principal « Engine » de collecter des informations sur le réseau. Ensuite, le module principal passera cette configuration au module « Admission_control ». Ce module sera chargé de tester la configuration réseau. Ensuite, le module principal se chargera de mettre à jour les disponibilités de notre réseau, notamment au cas où des problèmes seraient apparus.

8 Diagramme de classes

Je présente ici l'organisation des classes et interfaces de « Web Services ». Pour des raisons de visibilité, le diagramme ne comportera pas les classes représentant les structures de données.



Ces classes reprennent l'ensemble des fonctions qui ont été décrites précédemment (section 4).

9 Équipements

9.1 Cœur du réseau SAN

Famille CISCO 9000 MF

9.2 Frontière du réseau SAN

CISCO 5420 SN pour la gestion du protocole iSCSI
CISCO pour la gestion du protocole FCIP

10 Environnement de programmation

10.1 Environnement

.Net
Java
WSDL
WSCL ?

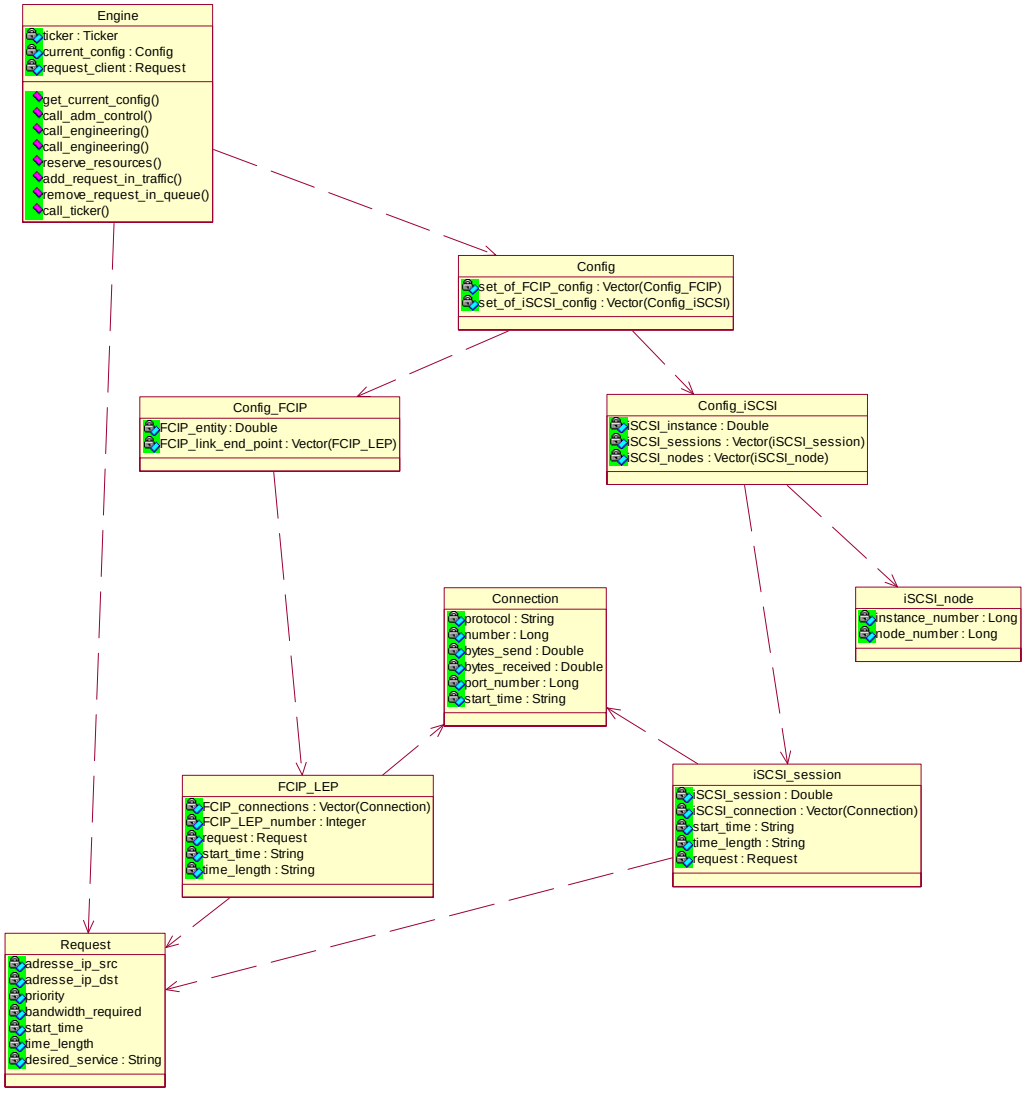
10.2 Outils

XML Spy, OmniOPERA pour la production des fichiers WSDL et XML schéma
JBuilder 8 pour l'interrogation de Web Services par des programmes JAVA

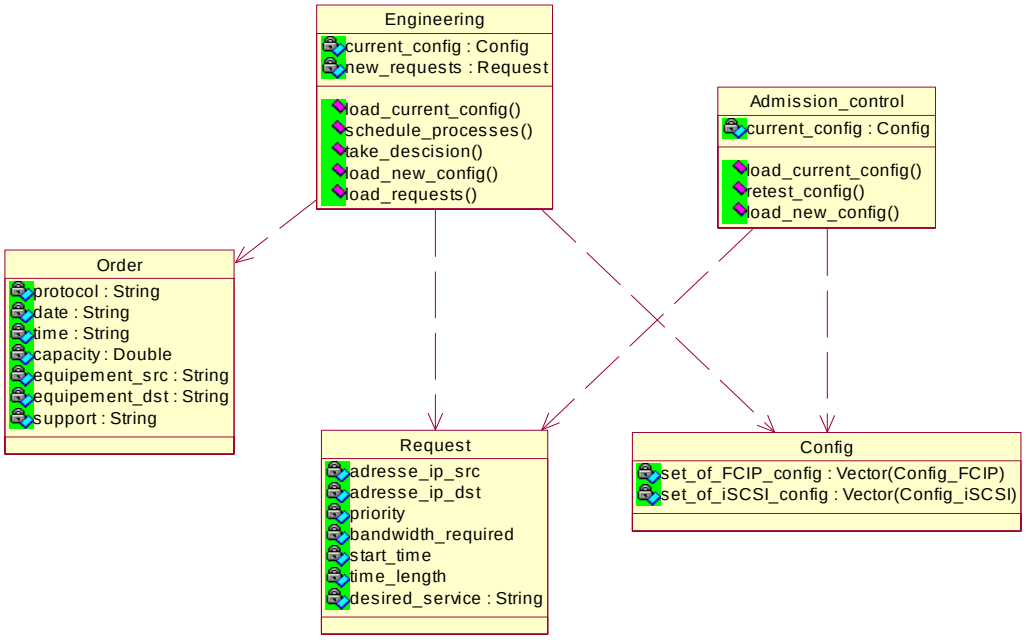
11 Annexes

11.1 Dépendances des classes de notre programme

11.1.1 Module Engine

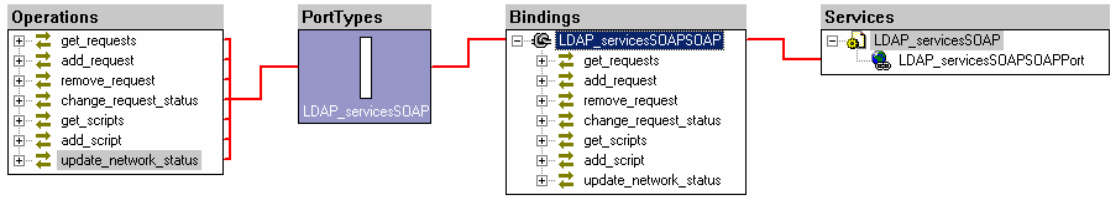


11.1.2 Modules Admission_control et Engineering

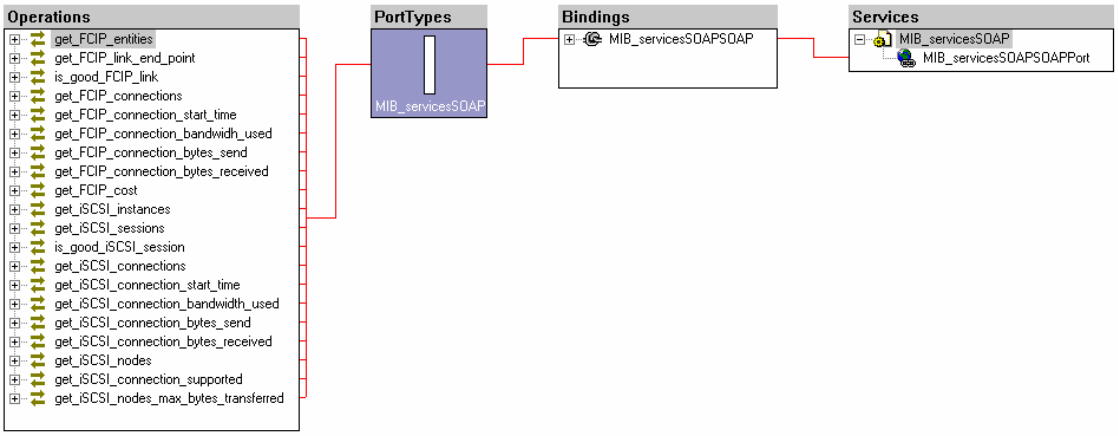


11.2 Fichiers WSDL des entités offrants des Web Services

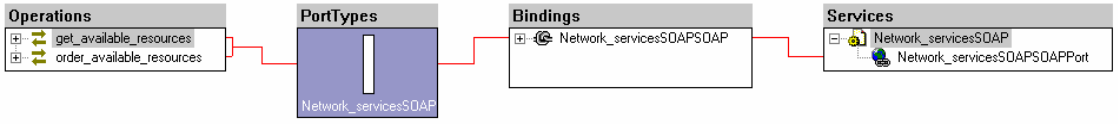
11.2.1 Web Services pour la communication avec le LDAP



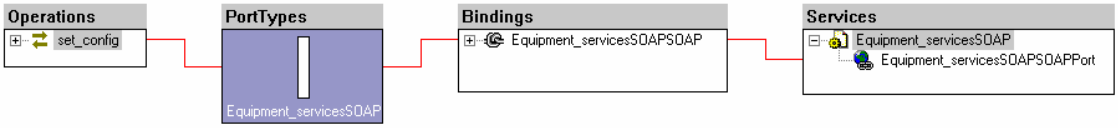
11.2.2 Web Services pour la récupération d'infos MIB



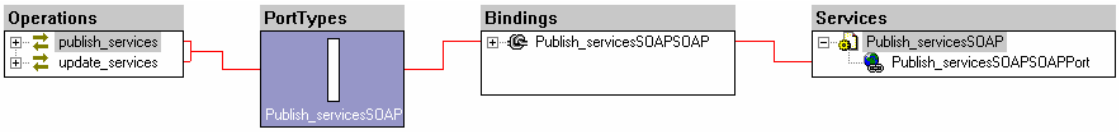
11.2.3 Web Services pour les demandes réseaux



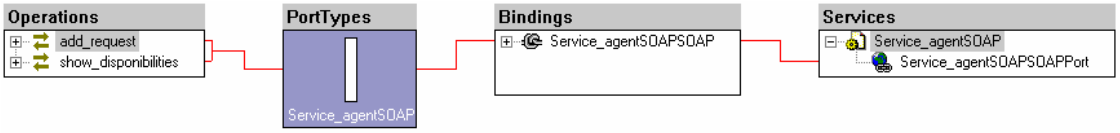
11.2.4 Web Services pour la configuration des CISCO



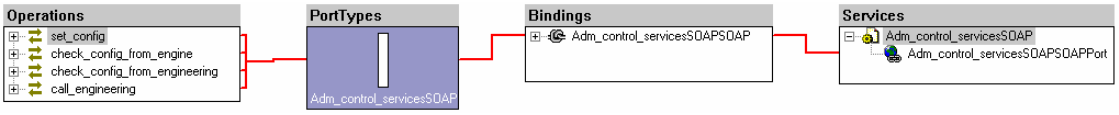
11.2.5 Web Services de publication des services



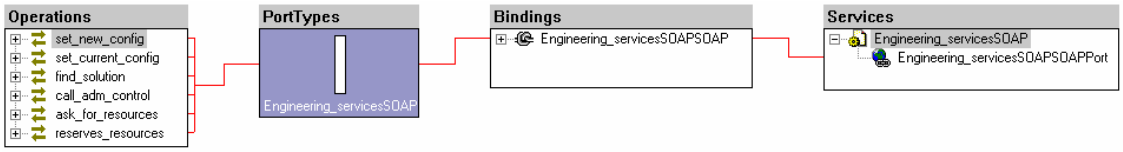
11.2.6 Web Services utilisables par le client



11.2.7 Web Services du module « Adm_control_services »



11.2.8 Web Services du module « Engineering_services »



12 Références

[1] MIB CISCO 5420 SN : http://www.cisco.com/en/US/products/hw/ps4159/ps2160/products_data_sheet09186a00800910f2.html

[2] MIB CISCO 9000 MF : http://www.cisco.com/en/US/products/hw/ps4159/ps4358/products_mib_quick_reference_chapter09186a008014a408.html

[3] WSDL Tutorial : http://www.w3schools.com/wsdl/default.asp

[4] XLANG : http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[5] FCIP : draft-ietf-fcip-fcoverip.txt

[6] FCIP : draft-ietf-ips-fcip-03.txt

[7] iSCSI : draft-ietf-ips-iscsi-20.txt

[8] iSCSI : draft-ietf-ips-iscsi-mib-09.txt

[9] Workflow : Gérer le Workflow : WSFL, XLANG et WSCL : http://www.application-servers.com/publications/chauvet/chap06.pdf

[10] WSCL et UDDI : http://www.uddi.org/pubs/wscl_TN_forUDDI_5_16_011.pdf