RAPPORT SDCI

Bureau d'étude

Groupe A1 - SDBD

Christophe CHASSOT
Samir MEDJIAH

PAR
Nathan BALBLANC
Gautier DELORME
Yacine SMINI
Chloé PROUVOST

5ème année Informatique et Réseaux
Janvier 2018

Sommaire

Introduction

Le but de ce projet SDCI est de déployer dynamiquement et de façon transparente des fonctions de réseau virtuelles relevant d'une activité de l'Internet des objets. Ces fonctions permettent de répondre à des besoins fonctionnels ou non d'applications distribuées. Ce déploiement est réalisé en appliquant les concepts et les techniques relevant de la virtualisation de fonctions de réseau (VNF) et des réseaux pilotables par logiciel (SDN).

L'architecture du projet est la suivante :

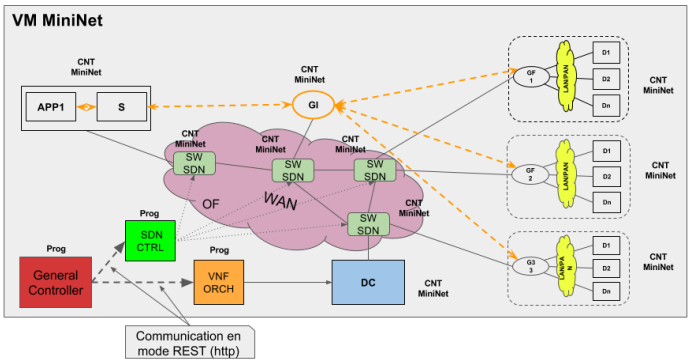

FIGURE 1 – Topologie MiniNet

Si une congestion de flux est présente dans le réseau, une nouvelle gateway intermédiaire doit être créée. Celle-ci sera reliée à une des gateways finales et permettra de décongestionner le réseau. Dans un premier temps le fait de créer une nouvelle gateway intermédiaire se fera manuellement. Dans un second temps cette action se réalisera de façon autonome lorsque le flux dépassera un certain seuil.

Ainsi, nous avons réalisé 9 conteneurs MiniNet, 4 switchs et 5 hosts correspondant aux 3 gateways finales, la gateway intermédiaire et l'appserver.

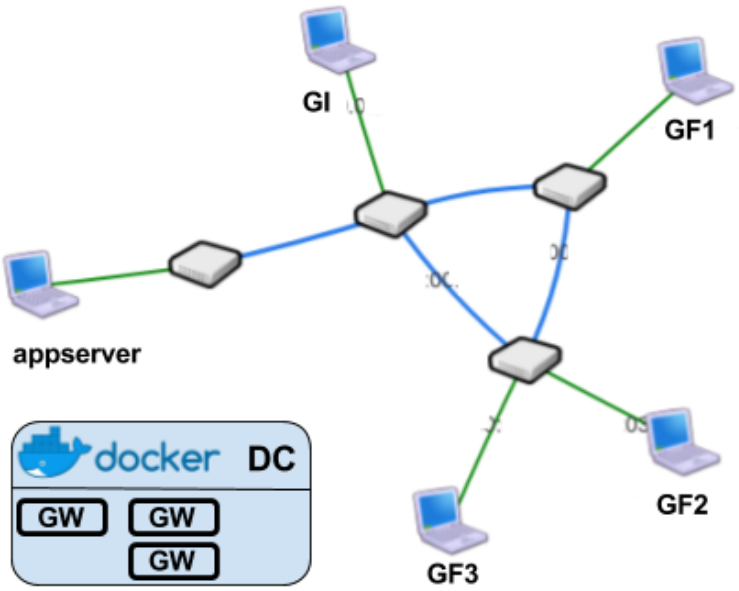

FIGURE 2 – Topologie MiniNet Réalisée

Chapitre 1

Résolution manuelle d'une congestion

1.1 Cas d'utilisation

Cas d'utilisation 1

- Identifiant : USECASEGENERAL_DEPLOY
- Version : 1.0
- Name : General Deployment
- Actors involved : Admin, SDN Controller, VNF Orchestrator
- Description : If a congestion occurred in the network the admin will trigger a VNF deployment (gateway intermediaire deployment) from the General Controller using a REST API endpoint. The General controller will require a VNF deployment on the data center from the VNF orchestrator and a SDN rules update from the SDN Controller. Once the VNF is deployed and the SDN rules updated (rerouting) the congestion issue should be fixed.
- Goal : VNF deployment + SDN rules update
- Triggers : Admin action (API request)
- Pre conditions : Environment setup
- Post conditions : VNF deployed + SDN rules updated
- Post conditions : congestion is gone

Cas d'utilisation 2

- Identifiant : USECASEADAPTATION_NOCONGESTION
- Version : 1.0
- Name : Adaptation no congestion
- Actors involved : Admin, SDN Controller, VNF Orchestrator
- Description : If a low flow is detected in the network (with more than 2

intermediate gateway) the admin will trigger a VNF suppression from the General Controller using a REST API endpoint. The General controller will require a VNF suppression on the data center from the VNF orchestrator and a SDN rules update from the SDN Controller. Once the VNF is removed and the SDN rules updated (rerouting), the issue should be fixed.
- Goal : Adapt network when there is no congestion (delete gateway) + SDN rules update (rerouting)
- Triggers : better use of resources

Cas d'utilisation 3
- Identifiant : USECASEGENERAL_MONITOR
- Version : 1.0
- Name : General Monitoring
- Actors involved : Admin, SDN Controller, VNF Orchestrator
- Description : The admin can ask the flow to the SDNControllerAdapter. If he think that the flow is to high he can decide to trigger a VNF deployment.
- Goal : System monitoring
- Triggers : Admin action (API request)
- Pre conditions : Environment setup

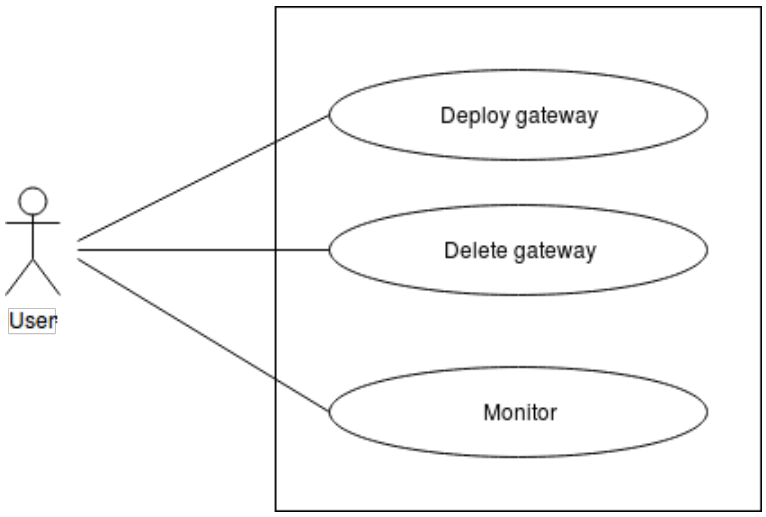

FIGURE 1.1 – Use Case de la résolution manuelle d'une congestion

1.2 Choix d'implémentation

Nous avons fait le choix d'avoir une architecture orientée service, et pour cela avons décidé d'utiliser une API REST. En effet, cette dernière permet de facile-

ment rajouter de nouveaux services à notre application.
Ainsi, l'administrateur pourra, par le biais de Curl, accéder aux endpoints correspondants aux cas d'utilisations définis précédemment. Nous avons fait le choix d'avoir 2 modes d'utilisation pour chacune des actions possibles : un mode libre et un mode guidé.
Par exemple, pour le monitoring, le mode guidé renverra le flux entrant dans la gateway intermédiaire 1, puisque l'objet est de prévenir une éventuelle congestion sur la gateway itermédiaire 1. Si l'administrateur fait le choix du mode libre, il devra fournir comme information l'élément qu'il veut monitorer afin d'obtenir le flux dudit élément.
Si l'administrateur observe une congestion, il peut décider de déployer une nouvelle gateway intermédiaire. Là encore, il dispose de deux modes d'actions. Dans les deux cas, une nouvelle gateway est déployée dans le data center. Dans le mode guidé, une règle SDN est créée et appliquée sur le switch connecté au data center qui redirige le flux prevenant de la gateway finale 1 vers la nouvelle gateway intermédiaire. Dans le mode libre, l'administrateur peut préciser la règle SDN qu'il souhaite appliquer ainsi que le switch sur lequel il compte appliquer cette règle.
Enfin, il est possible qu'après le déploiment de la gateway intermédiaire, le flux redésencde à un seuil qui entraine une sous-utilisation des capacités du réseau. Dans ce cas, l'administrateur a la possibilité de supprimer une règle qu'il renseignera afin d'optimiser l'utilisation des ressources.

1.3 Diagramme de classe

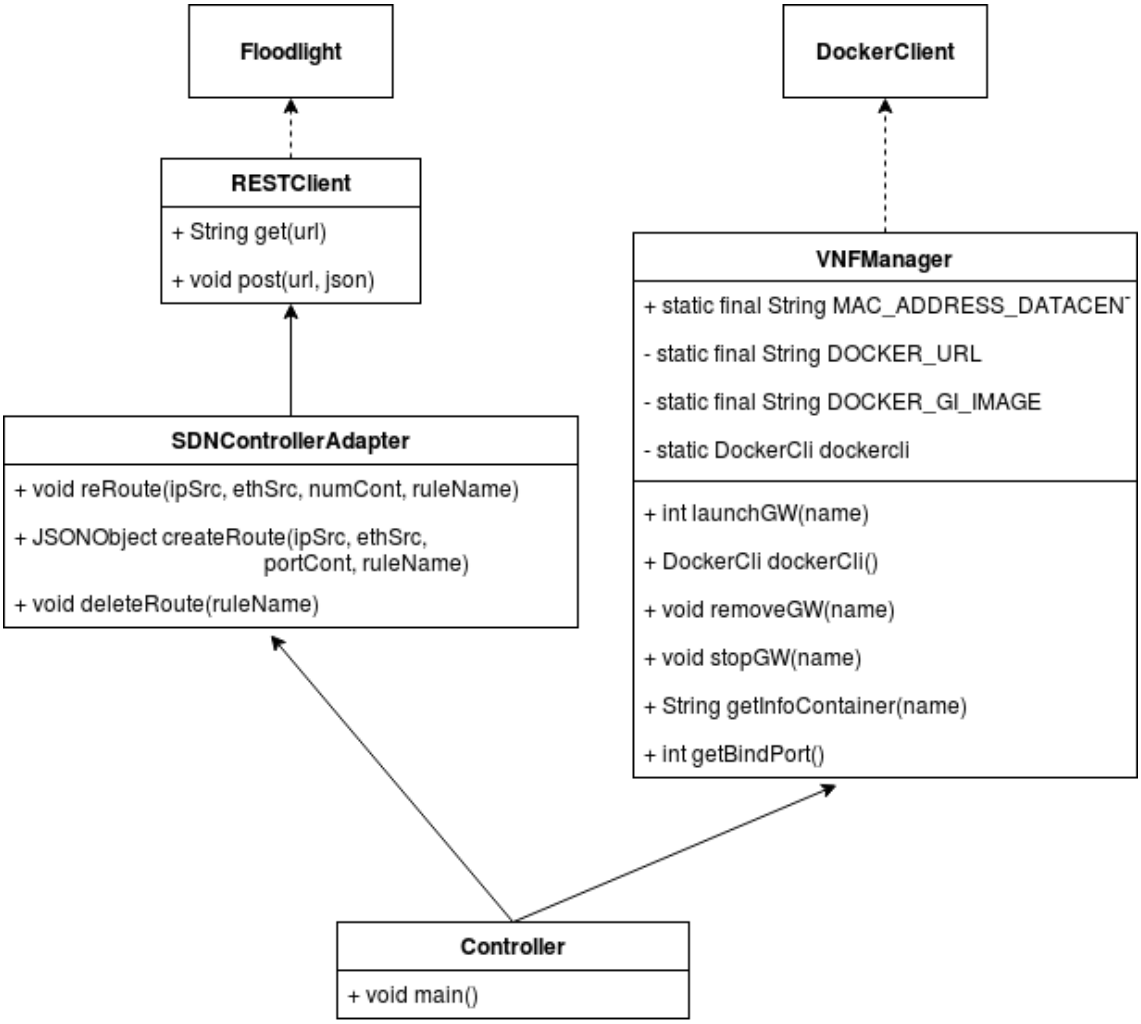

FIGURE 1.2 – Diagramme de classe de la résolution manuelle d'une congestion

En plus de Floodlight et de Docker notre conception possède 4 entités : Un controller, un VNFManager, un SDNControlelrAdapter et un RESTClient.

1.4 Diagramme de structure composite

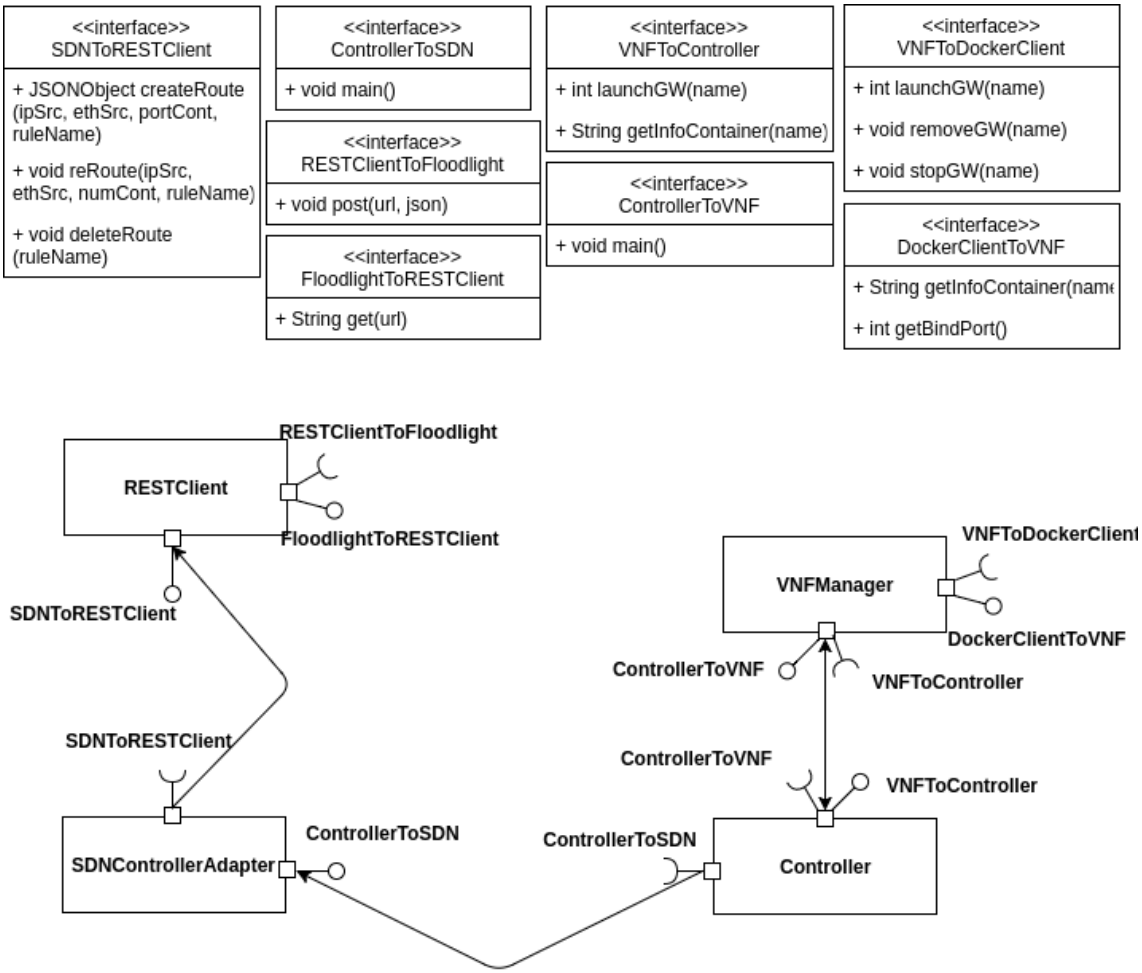

FIGURE 1.3 – Diagramme de structure composite de la résolution manuelle d'une congestion

- Le controller est relié au VNFManager et au SDNControllerAdapter. Il ordonne la création ou la supression de gateways intermédiaires.
- Le VNFManager, relié au DockerClient et au controller, permet de créer ces nouvelles gateways (Docker) dans le Datacenter (Docker), de les supprimer, et d'obtenir des informations sur ces gateways.
- Le SDNControllerAdapter, relié au controller et à un RESTClient, permet de rediriger les routes du réseau lors de la création d'une nouvelle gateway.
- Un RESTClient permettant de faire le lien entre le SDNController et Floodlight.

1.5 Diagramme de séquence

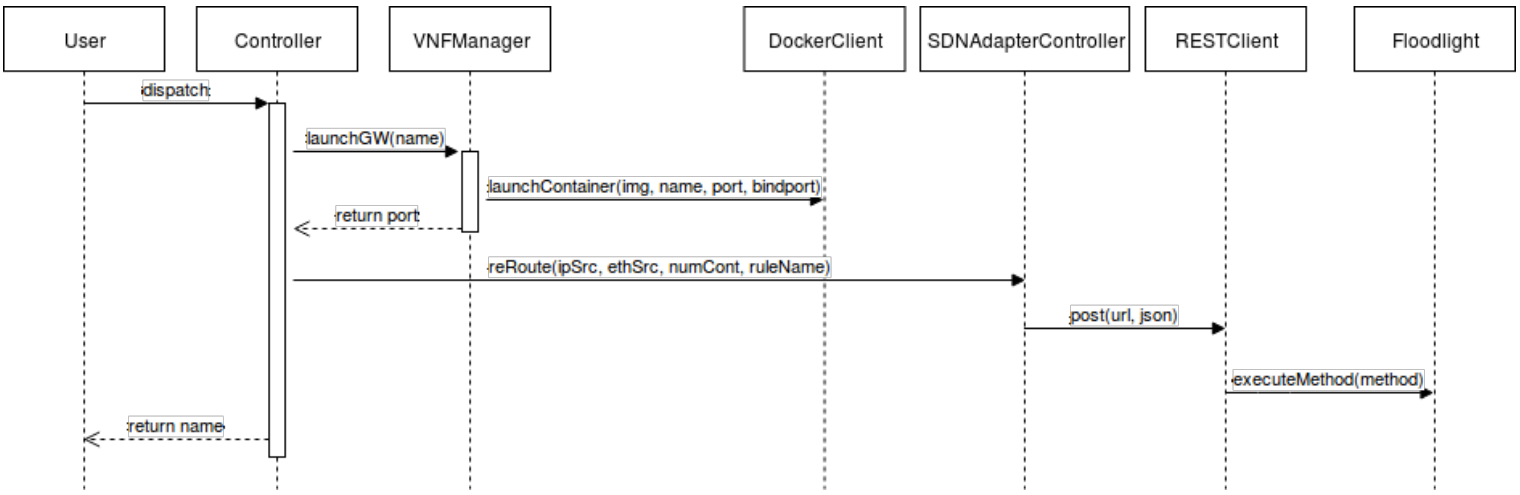

FIGURE 1.4 – Diagramme de séquence de la résolution manuelle d'une congestion

Par le biais du controleur, l'utilisateur peut déclencher la création d'une nouvelle gateway intermédiaire pour décongestionner le réseau.
Le VNFManager se charge de créer la gateway par le biais de la méthode launchGW().
Le numero de port de la nouvelle gateway sera renvoyé au controller.
Le controller ordonne ensuite au SDNControllerAdapter de rediriger les routes du réseau en prenant en compte la nouvelle gateway. La méthode reRoute() du SDNControllerAdapter appelle la méthode post() de la classe RESTClient.
Une fois la gateway déployée, son id est renvoyé à l'utilisateur.

Chapitre 2

Surveillance et résolution automatique d'une congestion

2.1 Cas d'utilisation

Cas d'utilisation 1

- Identifiant : USECASEGENERAL_DEPLOY
- Version : 1.0
- Name : General Deployment
- Actors involved :FlowGestion, SDN Controller, VNF Orchestrator
- Description : A thread will continuously monitor the network's flow. If a congestion is detected in the network (i.e if the flow is above a designated value), the thread will require a VNF deployment on the data center from the VNF orchestrator and a SDN rules update from the SDN Controller using a REST API endpoint. Once the VNF is deployed and the SDN rules updated (rerouting) the congestion issue should be fixed.
- Goal : System monitoring + VNF deployment + SDN rules update
- Triggers : Admin action (API request)
- Pre conditions : Environment setup
- Post conditions : VNF deployed + SDN rules updated

Cas d'utilisation 2

- Identifiant : USECASEADAPTATION_NOCONGESTION
- Version : 1.0
- Name : Adaptation no congestion
- Actors involved : Admin, SDN Controller, VNF Orchestrator
- Description : A thread will continuously monitor the network's flow. If a low flow is detected in the network (with more than 2 intermediate gate-

way) the thread will require a VNF suppression on the data center from the VNF orchestrator and a SDN rules update from the SDN Controller using a REST API endpoint. Once the VNF is removed and the SDN rules updated (rerouting), the issue should be fixed.
- Goal : System monitoring + Adapt network when there is no congestion (delete gateway) + SDN rules update (rerouting)
- Triggers : better use of resources

Cas d'utilisation 3
- Identifiant : USECASEGENERAL_MONITOR
- Version : 1.0
- Name : General Monitoring
- Actors involved : FlowGestion, SDN Controller, VNF Orchestrator
- Description : A thread will continuously monitor the network's flow. If a congestion is detected in the network (i.e if the flow is above a designated value), the thread will require a VNF deployment on the data center.
- Goal : System monitoring
- Triggers : better use of resources
- Pre conditions : Environment setup

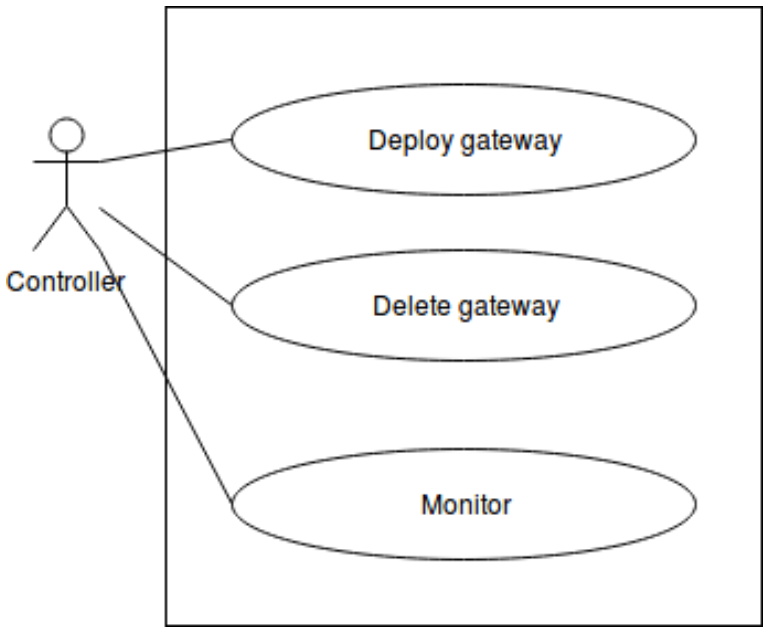

FIGURE 2.1 – Use Case de la résolution automatique d'une congestion

2.2 Positionnement sur l'autonomic computing

2.2.1 Caractéristiques de l'autonomic computing

Il existe 4 grands types de capacités d'auto gestion. Nous allons essayer de nous positionner par rapport à chacune de ses caractéristiques.

FIGURE 2.2 – Les 4 caractéristiques de l'autonomic computing

- Self-configuring est la capacité d'un système à se configurer mais aussi à se reconfigurer pour s'adapter à un environnement qui change dynamiquement. Dans notre cas, cette caractéristique est inhérente à notre application puisque le but est de créer une nouvelle gateway intermédiaire lorsqu'une la gateway intermédiaire 1 est congestionnée.
- Self-optimizing est la capacité d'un système à modifier ses paramètres internes et de mettre en œuvre des actions d'adaptation pour fournir un service optimisé en fonction d'objectifs et de contraintes. Notre application possède cette caractéristique puisqu'elle s'attache à créer une nouvelle gateway intermédiaire lorsqu'elle détecte une congestion au niveau de la gateway intermédaire 1, congestion qui entrainerait une baisse de la QoS. Ainsi, en redirigeant une partie du traffic vers la gateway nouvellement créée, la QoS restera optimal.
- Self-healing est la capacité d'un système à détecter et prévenir une interruption de service. Dans notre cas, nous n'avons pas réellement implémenté quoi que ce soit pour satisfaire cette caractéristique.

- Self-protecting est la capacité d'un système à détecter et prévenir une attaque en provenance de l'environnement. Dans notre cas, nous n'avons pas réellement implémenté quoi que ce soit pour satisfaire cette caractéristique.

2.2.2 La boucle MAPE-K

La boucle MAPE-K est une boucle de contrôle proposée par IBM mettant en jeu une ME (managed entity) qui correspond à l'entité sur laquelle s'appliquent les actions d'adaptation, ici notre réseau, et une AM (Autonomic Manager) qui correspond à l'entité qui élabore les actions à appliquer et les font appliquer, ici notre application.

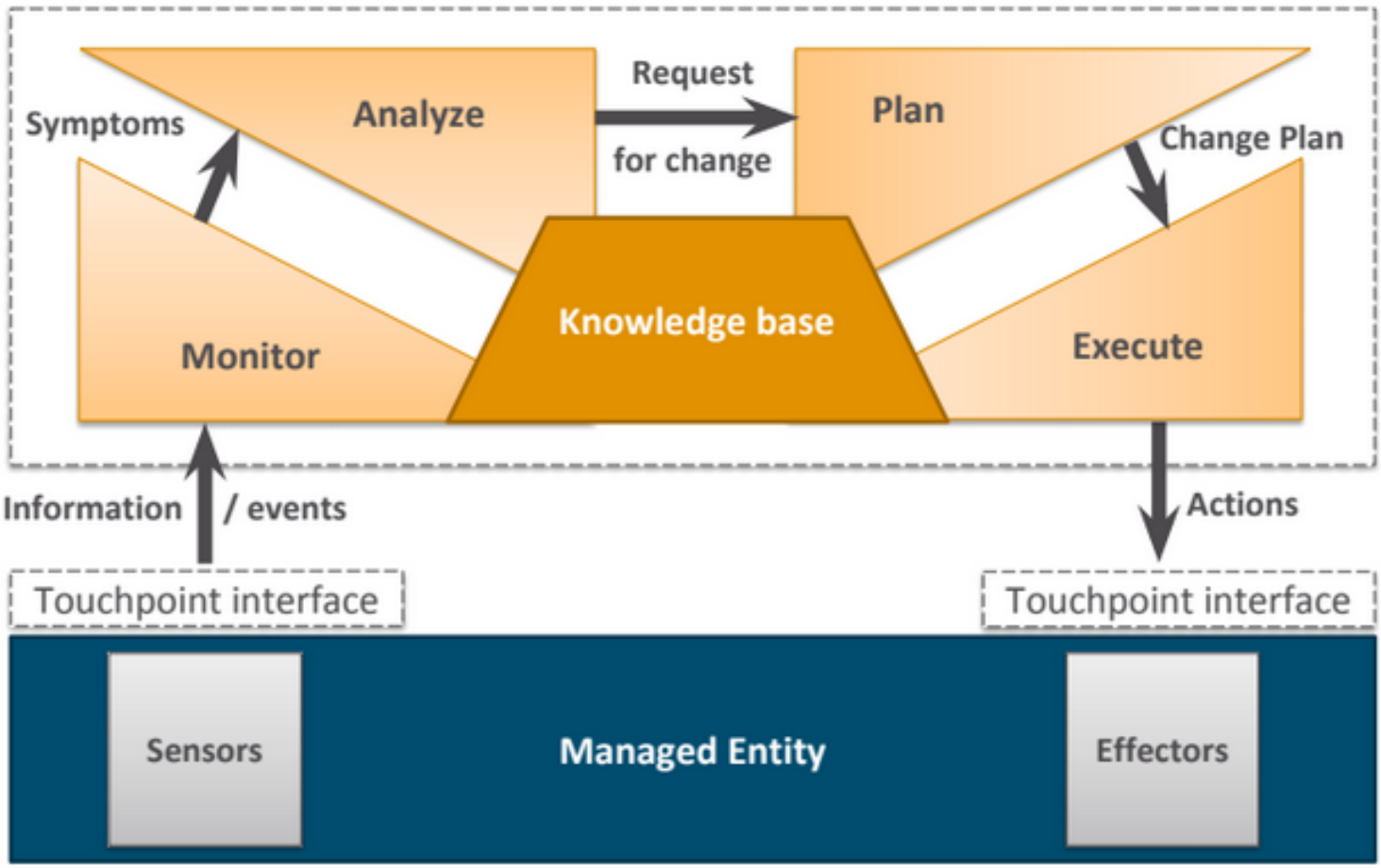

FIGURE 2.3 – La boucle MAPE-K

- Sensors permettent aux AM de récupérer des informations de la part des

ME. Dans notre cas, il s'agit du SDN Controller Floodlight qui renvoie le flux dans le réseau.

- Effectors permettent aux AM de faire appliquer des actions aux ME. Dans notre cas, il s'agit du SDN Controller Floodlight qui permet d'appliquer la nouvelle règle SDN et de Docker Daemon qui permet de créer la gateway intermédiaire.
- Monitor correspond à l'étape où l'AM récupère des informations de la part de la ME via l'interface sensor mais aussi lève de symptômes en cas de non respect des objectifs ou contraintes. Dans notre cas, un thread récupère toutes les 10 secondes le flux entrant dans la gateway intermédiaire 1 et le compare à un seuil donné. Si le flux dépasse le seuil donné, il lève une alerte. Ce choix a été motivé par la facilité d'accès à l'information de flux qui nous permettait de "détecter" la présence d'une congestion. Cependant, après réflexion, nous avons décidé de nous baser sur la QoS et d'établir un seuil sur un pourcentage de la QoS optimale, puisqu'il s'agit ultimement de ce que nous voulons conserver au plus haut niveau.
- Analyze correspond à l'étape d'élaboration d'un diagnostic. Dans notre cas, il s'agit de la comparaison du flux sortant de chaque gateway finale afin d'isoler celle qui émet le plus de flux en la redirigeant vers la gateway intermédiaire précédemment créée.
- Plan correspond à l'étape de définition d'une stratégie visant à prévenir ou corriger un état indésirable ou à réaliser les objectifs à atteindre. Dans notre cas, il s'agit de la préparation de la règle SDN qui servira à décongestionner la gateway intermédiaire 1.
- Execute correspond à l'étape où l'AM donne des ordres d'exécution à la ME via l'interface effector. Dans notre cas, il s'agit de l'application concrète de la règle SDN par Floodlight mais aussi de la création de la gateway intermédiaire par Docker Daemon.

2.3 Diagramme de classe

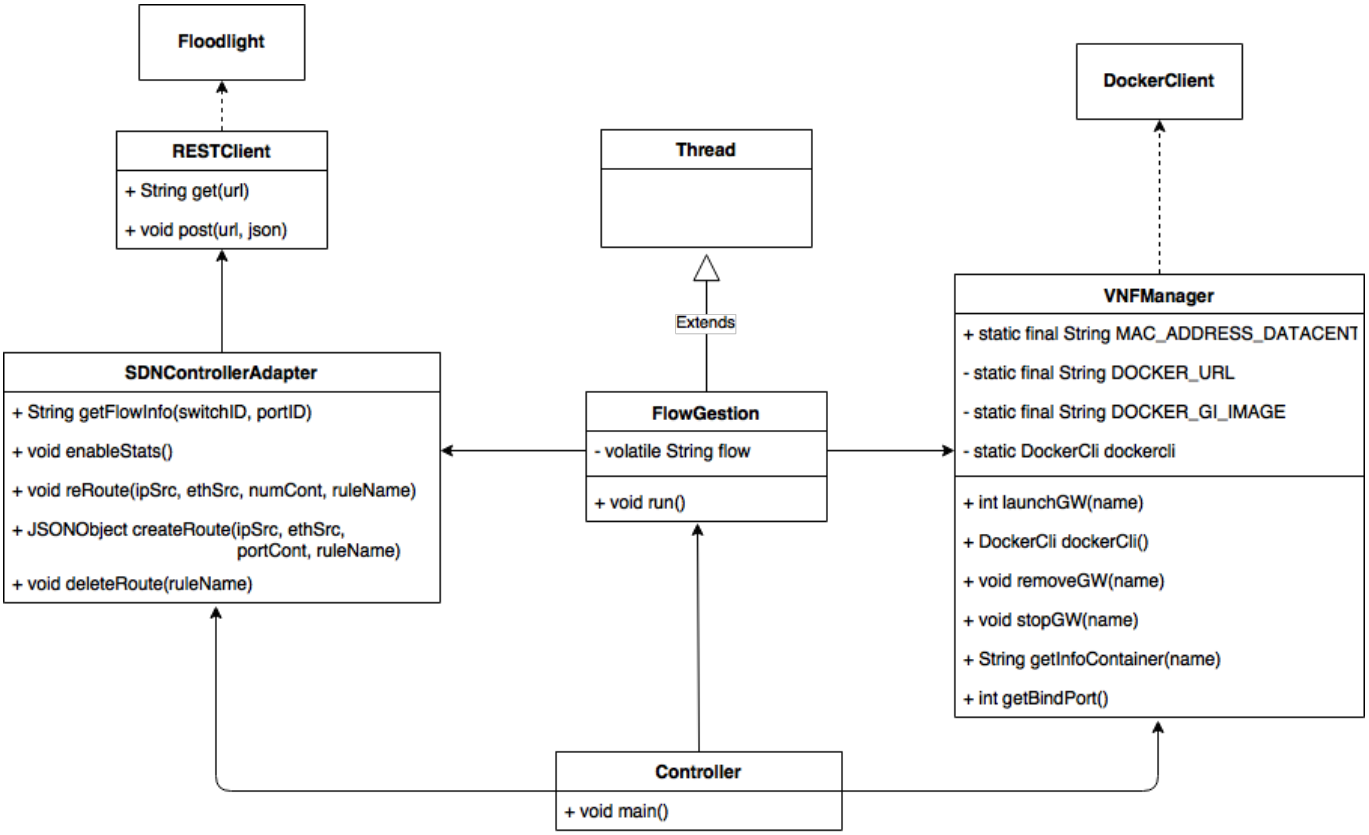

FIGURE 2.4 – Diagramme de classe de la résolution automatique d'une congestion

Pour ce deuxième cas nous avons rajouté une entité dans notre conception : FlowGestion, un thread permettant de surveiller le système et de détecter une congestion.

2.4 Diagramme de structure composite

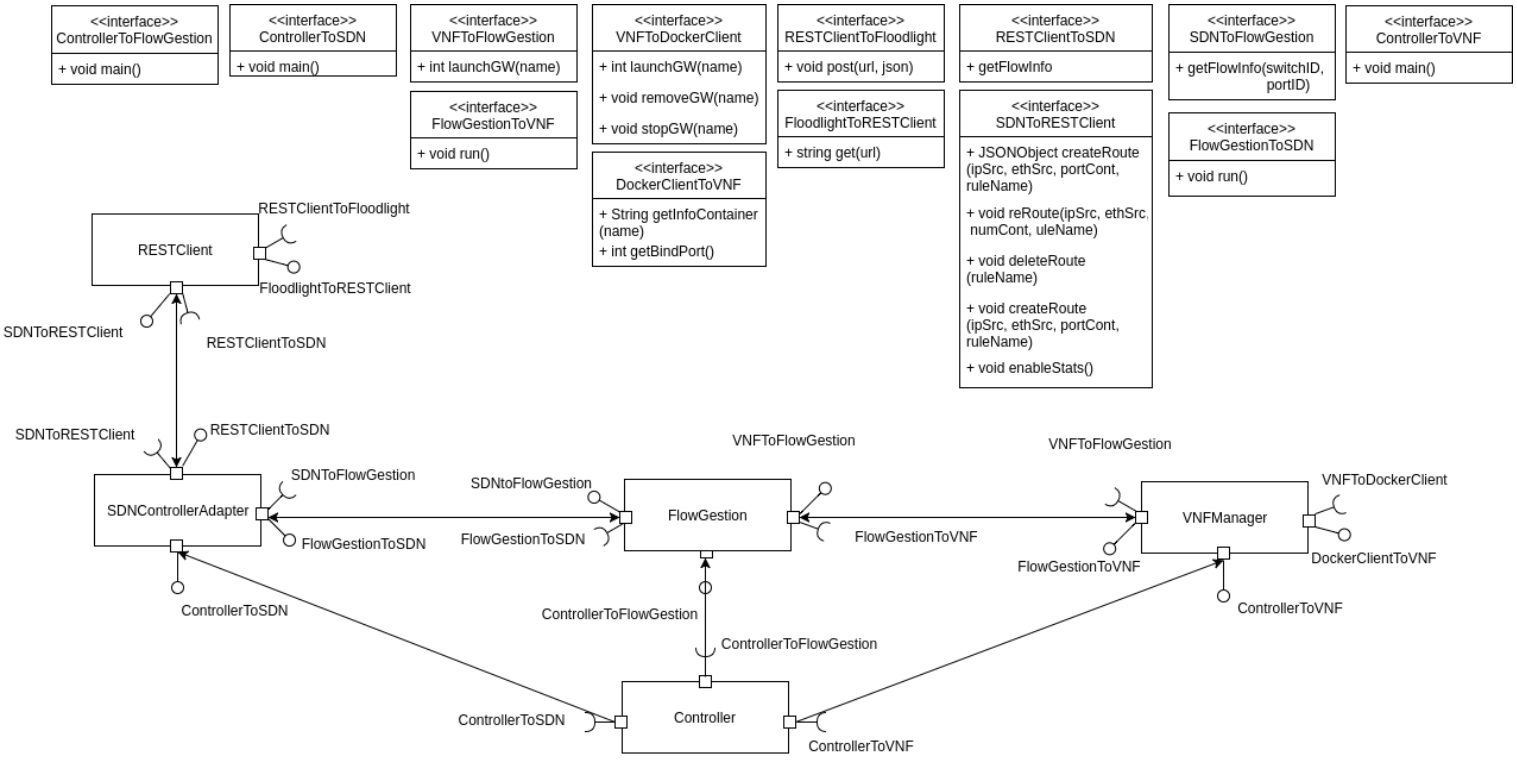

FIGURE 2.5 – Diagramme de structure composite de la résolution automatique d'une congestion

- FlowGestion est relié au controller et au SDNControllerAdapter. Il surveille le réseau en analysant la quantité de flux dans le réseau et en ordonnant la création de gateways intermédiaires au VNFManager.

2.5 Diagramme de séquence

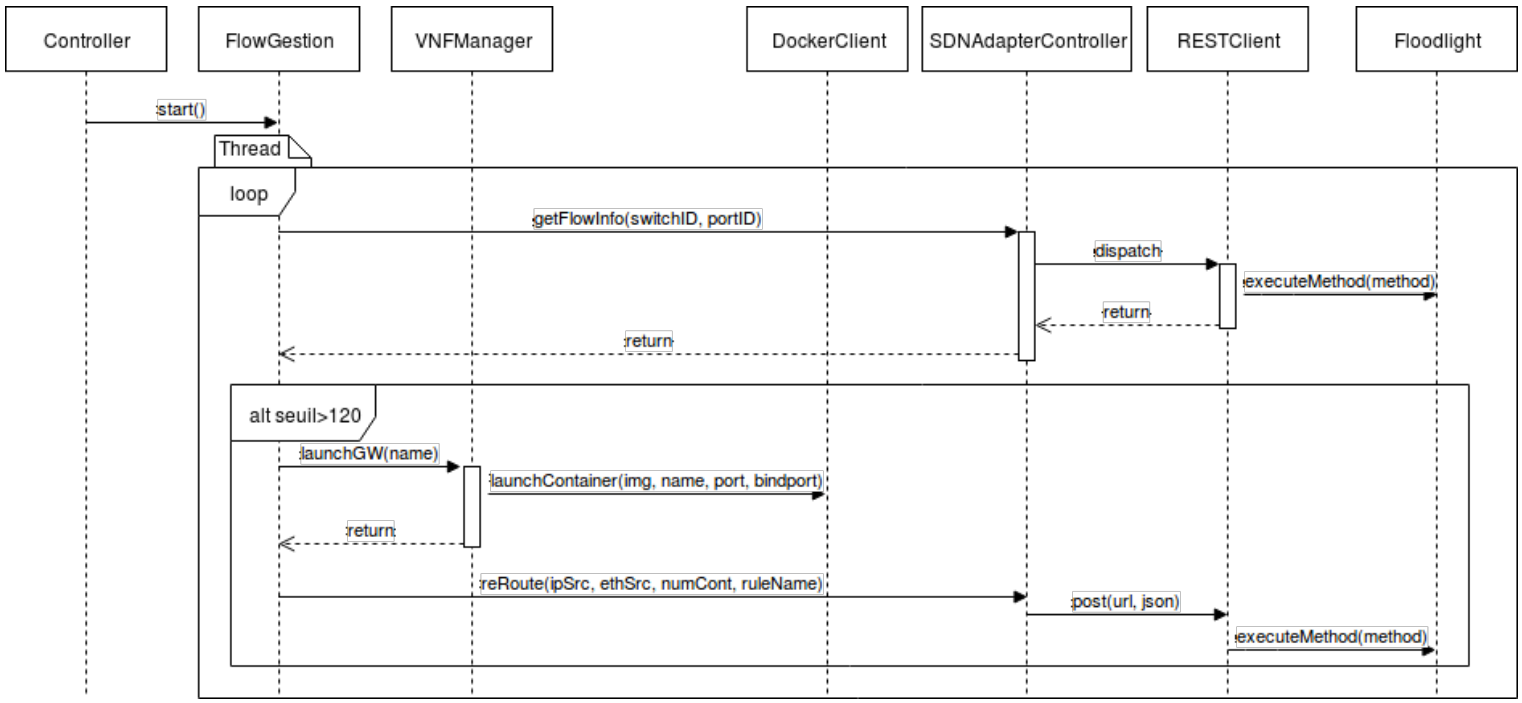

FIGURE 2.6 – Diagramme de séquence de la résolution automatique d'une congestion