Algorithms for implementing network services with a guaranteed quality of service in an SDN-NFV environment

Cédric Morin

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Cédric MORIN

Algorithms for implementing network services with a guaranteed quality of service in an SDN-NFV environment

Thèse présentée et soutenue à Cesson Sévigné, le 18 novembre 2020
Unité de recherche : Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)
Thèse N° : 2020IMTA0198

Rapporteurs avant soutenance :

Mme Bernardetta ADDIS	Maître de conférences, Université de Lorraine - LORIA
M Frédéric GIROIRE	Directeur de recherche, CNRS

Composition du Jury :

Président :	M Guillaume PIERRE	Professeur, Université de Rennes 1
Examinateurs :		
	Mme Bernardetta ADDIS	Maître de conférences, Université de Lorraine - LORIA
	M Frédéric GIROIRE	Directeur de recherche, CNRS
	M Jean Louis ROUGIER	Professeur, Telecom Paris
	M Stefano SECCI	Professeur, CNAM
Dir. de thèse :	Mme Géraldine TEXIER	Maître de conférences, IMT Atlantique

Invité(s)

M Gilles DESMANGLES	Ingénieur, TDF

This thesis has been financed by the enterprise TDF (TéléDiffusion de France).

The research presented in this document have been realized in the laboratories of the Institut de Recherche Technologique b<>com (IRT b<>com).

ACKNOWLEDGEMENT

Je tiens tout d'abord à remercier ma directrice de thèse, Géraldine Texier, pour toute l'aide et le soutient qu'elle m'a apporté lors de chacune des étapes de mes travaux, du choix des thématiques abordées jusqu'à la rédaction de nos publications et leur présentation en conférence.

Je voudrais également remercier tous les membres du jury pour avoir accepté d'examiner mes travaux, et en particulier Bernardetta Addis et Frédéric Giroire pour la relecture de mon manuscrit. Leurs commentaires détaillés m'ont permis d'en améliorer grandement la qualité.

Je voudrais aussi remercier Amélie Kerbellec, Christian Nieps, Davy Darche et Gilles Desmangles, de l'entreprise TDF, pour m'avoir suivi tout au long de ma thèse.

Je remercie également tous les membres du laboratoire Connectivité Avancée de l'IRT b<>com dans lequel j'ai eu la chance d'effectuer mes travaux, ainsi que tous mes autres collègues de l'IRT avec lesquels j'ai pu échanger au cours de ces trois années, que ce soit pour le travail ou en dehors. Plus particulièrement, je souhaiterai remercier Cao-Thanh Phan, qui m'a appris – et continue de m'apprendre – beaucoup de choses dans le domaine des réseaux virtualisés. Grâce à ses conseils j'ai pu orienter mes travaux vers des problématiques d'actualité, tout en les ancrant dans le cadre fourni par la normalisation. Je souhaiterai également remercier tous les doctorants et post-doctorants avec qui j'ai partagé ces années de travail : Imad Alawe, Johan Pelay, Luis Suarez, Franck Messaoudi, Salaheddine Zerkane, Mohamed Rahali, Nour Hobloss et Shanay Behrad.

Je voudrais également remercier Christelle Caillouet, avec qui nous avons eu la chance de collaborer pour mettre en place nos modèles d'optimisation.

Enfin, je souhaiterai bien sûr remercier ma famille et mes amis, qui m'ont soutenu durant toute ma thèse, et bien avant.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

RÉSUMÉ EN FRANÇAIS

Depuis le déploiement de la première génération de communication mobile (1G) dans les années 80, les réseaux mobiles sont en constante mutation. Réagissant à l'augmentation du trafic et à la multiplication des applications, les opérateurs ont fait évoluer leurs architectures et les technologies utilisées pour augmenter leurs performances. Aujourd'hui, le trafic réseau continue à croître de manière soutenue. À titre d'exemple, Cisco prévoit une augmentation du trafic IP de 26% par an sur la période 2017-2022 [37]. Afin d'absorber cette augmentation, les opérateurs se tournent vers la nouvelle génération de communication mobile : la 5G.

Au-delà d'une augmentation globale du trafic supporté, cette nouvelle génération doit également permettre de proposer un nouvel ensemble de services de communication, dont certains requièrent des contraintes très fortes en termes de qualité de service. La nature de ces contraintes permet de diviser ces nouveaux services en trois grandes catégories [138] : les services nécessitant une bande passante élevée, telle que la vidéo 3D, ceux ayant besoin de communications très fiables à très faible latence, tels que les services liés aux opérations chirurgicales à distances dans le cadre de la santé connectée, et enfin ceux requérant de très nombreuses connexions au réseau, dans le cadre de l'Internet des objets.

Ces différents types de services ont tous des besoins forts en termes de qualité de service, mais ces différentes contraintes ne s'appliquent pas nécessairement simultanément à chaque service : un service de vidéo 3D pourra s'accommoder d'une latence relativement importante, alors qu'un objet connecté va émettre peu de données. Plutôt que de s'appuyer sur un réseau unique et monolithique pour mettre en place tous ces services, il pourrait donc être plus approprié d'avoir plusieurs réseaux, chacun optimisé pour supporter une catégorie de service réseau exposant un certain ensemble de contraintes. Des réseaux physiques dédiés seraient trop coûteux, peu flexibles, et complexes à maintenir. La 5G mise donc sur des réseaux virtuels, obtenus à partir d'un réseau physique unique. Ces différents réseaux virtuels sont appelés *slices* [56]. Le slicing se fonde sur l'isolation des ressources, assurant que les actions des différents utilisateurs du réseau ne s'impactent pas mutuellement. Pour comprendre comment ces réseaux virtuels flexibles sont obtenus à partir d'un réseau physique unique, il faut détailler l'élément de base qui compose chaque slice : le service réseau.

Le service réseau est formé de trois types d'éléments : deux points de terminaison (une entrée et une sortie), un ensemble de fonctions réseau nécessaires au bon fonctionnement du service, et des connexions réseau reliant les différentes fonctions ainsi que les terminaisons. Le slicing doit donc à la fois prendre en compte les connexions et les fonctions réseau. Deux concepts sont utilisés pour cela, respectivement les réseaux définis par programmation (SDN) [46] et la virtualisation des fonctions réseau (NFV) [114].

SDN est une nouvelle architecture de gestion des ressources réseau qui consiste à séparer le plan de contrôle, qui établit les règles de routage, du plan de données, qui les exécute et transfert effectivement le trafic. Ces deux plans sont habituellement colocalisés dans les

routeurs. Avec SDN, le plan de données reste distribué et se compose d'un ensemble de commutateurs. Le plan de contrôle, quant à lui, est logiquement centralisé en un nouvel élément : le contrôleur. Ce dernier abstrait la complexité du réseau et offre aux applications clientes des services de connectivité via son interface nord. SDN permet l'accélération du développement, de la mise en place et de la mise à jour de ces services de connectivité, ainsi qu'une utilisation des ressources optimisée via sa gestion centralisée. Alliant automatisation et vision globale, SDN permet également le passage à l'échelle réactif des connexions en fonction de la charge de trafic.

Alors que SDN se charge des connexions dans le réseau, NFV va se concentrer sur les fonctions réseau. Traditionnellement, ces fonctions sont des éléments physiques spécialisés, ce qui rigidifie le réseau et limite les avantages attendus de SDN. NFV propose de remplacer ces éléments par des serveurs génériques. Les fonctions réseau sont alors implémentées sous forme de logiciels, et appelées fonctions réseau virtualisées (Virtual Network Function (VNF)), déployables sur ces serveurs. Ceci permet une plus grande agilité dans la gestion des fonctions en autorisant un déploiement et un passage à l'échelle automatisés, ainsi qu'une grande facilité de mise à jour et d'implémentation de nouvelles fonctions. Pour profiter pleinement de ces avantages, un système de gestion et d'orchestration automatisé du cycle de vie des VNFs est nécessaire. Dans ce domaine, l'European Telecommunications Standards Institute (ETSI) a proposé une architecture dédiée : ETSI NFV MANO [66]. Un orchestrateur, le Network Functions Virtualisation Orchestrator (NFVO), ordonne le déploiement des services réseau, composés de graphes de VNFs. Afin de les déployer, il s'appuie sur les informations topologiques remontées par les gestionnaires d'infrastructures (Virtualized Infrastructure Managers (VIMs)), qui proposent une vision des ressources disponibles. Une fois les ressources sélectionnées, le NFVO délègue aux gestionnaires de fonctions virtualisées (Virtual Network Function Managers (VNFMs)) le soin de gérer le cycle de vie des VNFs, alors que lui-même se charge de la gestion des cycles de vie des services réseau dans leur ensemble. La gestion du cycle de vie inclus nécessairement l'instanciation et la terminaison de l'élément, ainsi que la mise à jour ou le passage à l'échelle si besoin. En tant que gestionnaire des services réseau, le NFVO a pour clients les gestionnaires de slices.

Afin de tirer le meilleur parti de la virtualisation et de l'automatisation, SDN et NFV peuvent être appliqués conjointement : les VIMs de ETSI NFV MANO peuvent être pourvus de contrôleurs SDN pour gérer les ressources des liens de leurs domaines. Cette virtualisation complète des ressources du réseau permet de mettre en œuvre le slicing de façon efficace.

Bien que SDN et NFV présentent de multiples avantages, plusieurs difficultés subsistent pour pouvoir pleinement les exploiter. Dans cette thèse, nous nous intéressons à la mise en place d'un service réseau de bout en bout. Ce processus débute lorsqu'un gestionnaire de slices émet une requête de création de service réseau vers le NFVO, et s'achève lorsque le service est effectivement installé dans le réseau. Nous pouvons distinguer trois étapes importantes lors de ce processus. Tout d'abord, si le propriétaire du NFVO ne possède pas suffisamment de ressources via ses propres VIMs pour installer le service, alors il doit acheter des ressources supplémentaires auprès de fournisseurs tiers, typiquement des opérateurs cloud publics. La difficulté consiste à obtenir les ressources nécessaires au prix le plus bas

possible, en choisissant la combinaison d'offres la plus avantageuse. Afin d'optimiser le coût global, le propriétaire du NFVO peut considérer ses besoins en ressources sur une période donnée, plutôt qu'acheter des ressources pour chaque service réseau séparément. Une fois les ressources obtenues, la deuxième étape consiste à placer la chaîne de fonctions constituant le service réseau sur ces ressources. Ici se pose une double difficulté : au niveau du NFVO tout d'abord, le placement des services réseau doit être effectué à la volée, sans connaissance des requêtes à venir. Dans un contexte de ressources limitées, le NFVO doit donc être muni d'une stratégie visant à maximiser la quantité de services réseau admis sur le long terme. Au niveau des VIMs ensuite, ceux n'appartenant pas au même propriétaire que le NFVO peuvent refuser de transmettre au NFVO la représentation exacte de leur infrastructure, sur laquelle se fonde les méthodes de placement habituelles. Une autre représentation des ressources doit donc être envisagée. Finalement, une fois le placement décidé par le NFVO et communiqué aux VIMs, ces derniers doivent configurer le réseau pour installer le nouveau service. S'appuyant sur les capacités offertes par l'architecture SDN, ils doivent être capables d'offrir aux services une bande passante et une latence garanties. Nous proposons ici une contribution pour chacune de ces trois étapes de la mise en place d'un service réseau.

Tout d'abord, considérons un aspect fondamental de l'établissement d'un service réseau : la mise en place d'une connectivité avec une qualité de service garantie, sans fonction réseau supplémentaire. Elle est nécessaire pour deux raisons : elle permet à l'opérateur réseau de respecter les engagements de qualité de service pris auprès de son client, et elle permet également l'isolation du trafic des différents utilisateurs, ce qui est fondamental dans le slicing. La mise en place de cette connexion passe par trois étapes : déterminer dans le réseau un chemin compatible avec les besoins exprimés, mettre en place ce chemin, et s'assurer du maintien des garanties au cours du temps.

Les protocoles traditionnels permettant de gérer la qualité de service dans un réseau, tel que DiffServ, sont soit complexes à mettre en œuvre, soit basé sur l'overprovisionning, une stratégie coûteuse en ressources. SDN permet d'envisager de nouvelles approches. Dans cette architecture, il est aisé pour le contrôleur centralisé de déterminer un chemin satisfaisant et de le mettre en place. C'est la troisième étape, le maintien de la qualité de service au cours du temps, qui constitue une difficulté. Les propositions avancées dans la littérature peuvent majoritairement se diviser en deux catégories : réactives et proactives. Une fois le chemin mis en place, les solutions réactives [15] surveillent en permanence le réseau pour s'assurer que les contraintes des différents chemins sont toujours respectées. Si ce n'est pas le cas, le contrôleur doit réagir, typiquement en déplaçant un ou plusieurs chemins. Cette méthode souffre de plusieurs inconvénients. Tout d'abord, la qualité de service n'est pas garantie à tout instant. Ensuite, la quantité de données de surveillance est importante, ce qui impacte les ressources de calcul des commutateurs chargés de les générer, ainsi que les ressources de bande passante des liens chargés de les transporter. Enfin, les chemins peuvent être modifiés fréquemment, ce qui peut mener à des pertes de paquets et à une surcharge du contrôleur chargé de les calculer. Inversement, les solutions proactives se basent sur les files d'attente pour garantir le partage des ressources. Bien que le contrôleur soit toujours en charge de diriger les flux au

travers du réseau, et de leur assigner des files d'attentes spécifiques, il n'est donc plus responsable du maintien de la qualité de service au cours du temps. La mise en œuvre des stratégies proactives peut être soit statique, soit dynamique. Dans le cas statique [50], un certain nombre de files d'attente sont prédéfinies dans chaque commutateur, et les flux sont redirigés vers la file correcte en fonction de leur importance. Deux flux de même importance peuvent cependant interférer entre eux. Ce problème ne se pose pas dans le cas dynamique, où les files d'attente sont créées à la volée, en fonction des besoins (jusqu'à une file d'attente par flux [73]. Bien que cela permette une gestion beaucoup plus fine de la qualité de service, le nombre de files d'attente limité au sein des commutateurs empêche le passage à l'échelle de cette stratégie.

L'objectif est ici de conserver les avantages des deux catégories de solutions, en évitant leurs inconvénients : il faut que la solution soit proactive, flexible et peu génératrice de données de surveillance. C'est l'objet de notre première contribution architecturale [113]. Constatant que les limitations des solutions proactives proviennent du rôle proéminent joué par le contrôleur, notre solution va viser à le décharger d'une partie de ses fonctions. La première étape consiste à déporter le calcul des chemins. Pour ce faire, un nouveau module est ajouté au plan de contrôle : SDN Traffic Engineering Management (STEM). Celui-ci est chargé d'orchestrer le traitement d'une requête de connexion émise par une application. Lors de la réception d'une telle requête, STEM la redirige vers un élément de calcul de chemin dédié : le Path Computation Element (PCE). La communication entre les deux modules est assurée par le protocole standard Path Computation Element Protocol (PCEP), ce qui permet de connecter tout type de PCE de façon transparente.

Le PCE reçoit des informations sur la topologie et les ressources totales du réseau en interrogeant le contrôleur, qui les obtient via une surveillance régulière. Il faut noter que cette surveillance concerne uniquement la topologie, et non les flux utilisateurs comme dans les solutions de mise en place de la qualité de service citées plus haut. Les évènements topologiques, tels que l'ajout d'un commutateur ou la rupture d'un lien, sont très rares comparés aux fluctuations de trafic, et génèrent très peu de trafic de surveillance. Afin de déterminer un chemin répondant aux contraintes d'une requête, le PCE peut utiliser un algorithme de chemin multi-contraint. Ici, nous avons choisi d'implémenter un algorithme proposé dans la littérature : SAMCRA [152]. Lorsqu'un chemin valide est sélectionné, le PCE le transmet au module STEM, et garde en mémoire les ressources utilisées dans la topologie. Cela assure qu'il n'utilisera pas deux fois les mêmes ressources : la qualité de services est ainsi garantie au niveau du plan de contrôle.

Reste maintenant à appliquer ce chemin au niveau du plan de données. Pour cela, STEM va générer un ensemble de règles qui seront ensuite transmises au contrôleur, pour être appliquées dans le réseau. Ces règles vont d'abord viser à établir un tunnel MPLS entre les deux extrémités de la connexion souhaitée. En entrée de ce tunnel, un mécanisme de contrôle d'admission est installé dans le commutateur, et paramétré par le contrôleur. Ce mécanisme classe le trafic en fonction de sa priorité, paquet par paquet, et note cette classification dans l'en-tête MPLS. Par exemple, on peut définir trois niveaux de priorité : *élevé* pour le trafic faisant partie d'une connexion à bande passante garantie, *normal* pour un trafic sans garantie (type *Best Effort*), et *bas* pour un trafic ayant dépassé le débit qui lui était alloué. Ce système de classification peut être affiné pour inclure plus de niveaux de priorité. Enfin, le paquet est

envoyé dans le tunnel, et sera pris en charge au niveau de chaque commutateur par la file d'attente correspondant à sa priorité. Pour ce faire, les commutateurs doivent mettre en œuvre une file d'attente par niveau de priorité (trois dans notre exemple). Cette solution offre plusieurs avantages : aucun trafic de surveillance n'est requis car le contrôle du trafic est déporté dans le réseau sous la forme des files d'attente et du contrôle d'accès, la solution est proactive grâce aux files d'attente, et la solution est flexible grâce au contrôle d'accès, qui peut finement classifier la priorité de chaque paquet. Enfin, il est possible de n'utiliser qu'un nombre réduit de files d'attente sans crainte que les flux prioritaires n'interfèrent entre eux, grâce à la gestion efficace des ressources au niveau du plan de contrôle. En effet, le PCE prend garde à ne pas attribuer aux liens plus de trafic prioritaire qu'ils ne peuvent en écouler.

Pour valider cette contribution architecturale nous avons mis en place une expérimentation en utilisant un switch physique PICA8 compatible avec le protocole OpenFlow souvent utilisé en SDN pour assurer les communications entre le plan de données et le plan de contrôle. Nous avons pu valider le respect de la qualité de service promise pour chaque flux, et ce sans qu'aucune donnée de surveillance propre à la gestion de la qualité de service ne soit échangée entre le plan de contrôle et le plan de données.

Après avoir proposé une solution au problème de garantie de qualité de service sur les connexions, abordons la prochaine étape pour obtenir un service réseau complet : l'ajout de VNFs. Le NFVO doit choisir comment placer chaque service réseau dans les ressources mises à sa disposition, en fonction des contraintes de ces services.

Le problème de placement de chaînes de VNFs est un sujet ayant fait l'objet de nombreuses contributions dans la littérature scientifique [72], qu'il est possible de classer en catégories, selon le type de placement et l'objectif d'optimisation. Il existe deux types de placement : offline et online. Le placement offline suppose que toutes les requêtes devant être placées sont connues en avance, et sont toutes placées simultanément. Le placement obtenu peut ainsi être optimal, car il n'existe aucune inconnue. Cependant, il faut au préalable avoir une connaissance totale des requêtes. Le placement online, au contraire, consiste à placer les requêtes une à une dans le réseau, sans connaissance préalable des requêtes futures. Ce placement a l'avantage de représenter la situation réelle à laquelle est confronté un opérateur devant satisfaire les requêtes de ses clients. Cependant, le placement final reste sous-optimal, car l'algorithme n'a jamais accès à toutes les informations. Une fois la famille de placement choisie, les possibilités d'optimisation sont multiples. On peut ici détailler deux grandes catégories d'objectifs souvent étudiés : économiser les ressources des nœuds d'une part, et des liens d'autres parts. Ces deux objectifs peuvent souvent être déclinés en minimisation des coûts et de la consommation énergétique. Le principe permettant l'économie de ressources au niveau des nœuds est la consolidation : lorsqu'un service réseau est installé, il est possible qu'il n'utilise pas entièrement les ressources qui lui sont allouées. Afin d'augmenter ce taux d'utilisation, il est possible de mettre en commun ces ressources avec d'autres services. Les ressources des liens, quant à elles, sont économisées en réalisant des chaînes plus courtes. Les deux objectifs sont donc souvent en conflit, le premier incitant à aller chercher des ressources spécifiques, le second encourageant à utiliser les ressources les plus proches. Dans un contexte virtualisé,

la consolidation souffre de plus de deux limitations dont l'impact sur la qualité du placement reste peu étudié. D'une part, au niveau du NFVO et des VNFMs, les services réseau peuvent passer à l'échelle s'ils sont peu utilisés, réduisant au passage les ressources mobilisées pour eux. D'autre part, au niveau des VIMs, l'infrastructure présentée au NFVO peut être adaptée en fonction de l'usage constaté des ressources : si les ressources réservées sont rarement entièrement utilisées, le VIM peut décider de pratiquer du surbooking. Ainsi, les ressources perçues comme étant perdues au niveau du NFVO sont en fait purement fictives.

Dans cette contribution nous avons choisi de nous intéresser au cas du placement online, car il représente la réalité du problème rencontré par le NFVO. Si le placement offline est également intéressant, par exemple pour réoptimiser le placement à intervalle de temps régulier, le placement online reste donc indispensable. Sur cette base, nous avons choisi notre objectif d'optimisation spécifiquement pour pallier à la principale faiblesse du placement online : le fait que les requêtes futures lui sont inconnues. L'objectif va donc être de parvenir à placer un maximum de requêtes dans le réseau sur le long terme, sans pouvoir planifier exactement l'usage des ressources. Pour effectuer notre placement nous considérons deux éléments : d'une part, comme nous l'avons évoqué, la 5G doit être adaptée à un ensemble de services, et notamment à des services à très faible latence. Un tel service ne peut pas être placé librement dans le réseau : ses contraintes de latence nous imposent de placer les VNFs qui le composent proches des points de terminaison. D'autre part, les ressources dans le réseau ne sont pas réparties de façon homogène : chaque nœud du réseau n'est pas directement connecté à un vaste datacenter. Schématiquement, on peut considérer que les nœuds centraux ont davantage de ressources que les nœuds de bordure, mais ces derniers, plus nombreux, sont en moyenne plus proches des utilisateurs finaux. Partant de ces deux constats, nous proposons un modèle visant à placer les chaînes de VNFs formant les services réseau de façon à épargner au maximum les ressources là où elles sont rares, afin de mieux les préserver pour des services dont les contraintes seraient plus fortes, et qui n'auraient pas d'autre choix que d'utiliser ces ressources de proximité pour satisfaire leurs contraintes [112].

Afin de valider l'efficacité de notre approche, nous comparons la quantité de services réseau acceptés par notre modèle avec celle obtenue par un modèle ne cherchant à optimiser que la consommation de bande passante, tout en respectant toujours les contraintes exprimées. La mesure est effectuée lorsque la quantité totale de services réseau installés se stabilise, ce qui correspond à un régime en charge du réseau. Pour effectuer nos tests, nous nous intéressons à quatre paramètres : la structure de la topologie, la concentration des ressources dans cette topologie, la force des contraintes de délai exprimées par les requêtes, et la durée de vie des services (qui impacte directement le taux d'occupation du réseau).

Nous mesurons tout d'abord l'influence des deux premiers paramètres, liés à la topologie. On distingue deux types de structures pour les topologies : les topologies *flat*, où tous les nœuds ont une centralité comparable, et qui correspondent à des réseaux vastes, d'échelle nationale par exemple, et les topologies *edge*, représentant les réseaux à une échelle plus locale, où les nœuds ont des centralités bien distinctes. Typiquement, les topologies edge présentent trois catégories de nœuds, des plus centraux aux moins centraux : nœuds de cœur, d'agrégation et d'accès. Le paramètre de concentration traduit quant à lui la répartition des

ressources dans le réseau : dans un scénario à concentration faible, tous les nœuds présentent les mêmes ressources. Avec une concentration moyenne, la quantité de ressource augmente avec la centralité. Cette hétérogénéité est encore accentuée dans les scénarios à concentration forte. Les résultats montrent que la structure de la topologie n'influence pas les performances du modèle. En revanche, la concentration des ressources a un impact important. Quelle que soit la concentration, on note tout d'abord que notre modèle affiche une performance au moins aussi bonne que le modèle de comparaison. Ensuite, pour les scénarios à concentration faible, la quantité de requêtes acceptées peut aller de +5% à +7%, et de +15% à +22% dans les scénarios à concentration forte. Comme nous l'avons détaillé, notre modèle cherche à économiser les ressources là où elles sont rares. Cette notion de rareté est peu présente dans les scénarios à faible concentration, puisqu'initialement tous les nœuds ont les mêmes ressources. Des disparités locales peuvent toutefois apparaître au cours du temps, expliquant la performance obtenue par notre modèle. Dans les scénarios à forte concentration, les plus proches de la réalité, la disparité des ressources est forte, ce qui explique les meilleures performances.

Cependant, dans tous les scénarios on observe que, dans le pire cas, notre modèle présente des performances comparables à celle du modèle de référence. Pour expliquer cela il nous faut analyser l'influence des deux autres paramètres : la force des contraintes de latence et la durée de vie des services dans le réseau. On observe que les performances les plus élevées sont obtenues lorsque les contraintes de latence sont en moyenne fortes ou modérées, et que la durée de vie des services dans le réseau entraîne moins de 70% de saturation. Lorsque les contraintes de latence sont faibles, les requêtes peuvent être placées en tout point du réseau, et économiser les ressources localement ne présente pas d'intérêt. Cependant, en réalité, on s'attend à ce qu'une part des services présente des contraintes de latences fortes, notamment dans le cadre de la 5G. Concernant la durée de vie des services, si le réseau est entièrement saturé alors aucun modèle de placement ne peut présenter de meilleure performance qu'un autre, dans la mesure où la consolidation des ressources n'entre pas en jeu. Mais cette situation serait le résultat d'un mauvais dimensionnement du réseau en amont.

En conclusion, notre stratégie de placement présente des performances bien supérieures à celle du modèle de référence dans les scénarios les plus fidèles à la réalité. Cependant, notre système, comme tous les systèmes proposés dans la littérature, repose sur le fait que les VIMs transmettent au NFVO une vision exacte de leurs ressources. Cela n'est pas nécessairement vrai si le NFVO et les VIMs n'appartiennent pas à la même entité.

Le contexte multi-propriétaires, où les entités qui interagissent n'appartiennent pas toutes au même propriétaire, commence à être abordé dans la normalisation [54] (travaux en cours) et dans certains projets menés par des industriels [69], car il correspond à une situation concrète à laquelle les acteurs seront de plus en plus confrontés avec le déploiement généralisé de réseaux virtualisés : les VIMs doivent pouvoir garder les détails de leurs topologies secrets, tout en fournissant assez d'informations au NFVO pour vendre leurs ressources, ce qui reste leur but premier.

Dans les réseaux traditionnels, une telle situation se résout généralement en effectuant une abstraction du réseau et de ses ressources. En raison de son importance en pratique,

cette thématique est le sujet de nombreuses contributions scientifiques [151]. On distingue trois grandes méthodes d'abstraction : la *compaction*, qui rassemble l'intégralité de la topologie en un nœud unique, le *full mesh*, qui consiste à connecter directement entre eux tous les nœuds de bordure du réseau, et l'*étoile*, qui connecte tous les nœuds de bordure du réseau à un nœud central virtuel, le nucleus. Parmi ces trois méthodes permettant de cacher efficacement les détails internes d'une topologie, la plus étudiée est le full mesh, car elle peut offrir l'avantage majeur de ne pas perdre d'informations utiles pour le routage. Il faut cependant noter qu'aucune des solutions proposées ne prend en compte les ressources des nœuds, puisque leur intérêt dans le contexte des réseaux n'est apparu qu'avec l'avènement, relativement récent, de NFV. Lorsque ces ressources sont intégrées, le full mesh perd son avantage, et d'autres pistes doivent être explorées pour représenter efficacement les ressources.

Pour apporter une réponse à ce problème nous proposons une abstraction en étoile. Grâce à son nœud central, celle-ci nous semble adaptée pour représenter les ressources des nœuds. Dans notre méthode, les ressources des nœuds de bordures de l'étoile sont fixées égales à celles des nœuds de bordure de la topologie réelle. Le nucleus, quant à lui, rassemble les ressources de tous les autres nœuds. En ce qui concerne les liens, nous avons procédé comme suit : le nucleus est tout d'abord identifié au nœud de la topologie réelle ayant la plus grande centralité. Ensuite, pour un nœud de bordure donné et une métrique donnée, le chemin le plus court reliant le nœud central à ce nœud de bordure est déterminé, via un algorithme de Dijkstra pondéré. Le score obtenu détermine la valeur de la métrique pour la branche de l'étoile reliant le nœud central à ce nœud de bordure. La procédure est répétée pour chaque métrique et pour chaque nœud de bordure.

Afin d'évaluer l'efficacité de notre abstraction, nous avons conduit une série de tests comparant la performance du modèle holistique contre le même modèle avec un niveau d'abstraction. Cette performance est mesurée en termes de nombre cumulé de requêtes admises dans le réseau au cours du temps. Les résultats montrent que notre stratégie d'abstraction présente de très bonnes performances, plaçant entre 92% et 98% de requêtes dans le réseau comparé au même modèle sans abstraction [112]. De plus, cette stratégie peut également être utilisée par le NFVO lorsque les VIMs lui fournissent des topologies détaillées afin de réduire le temps de calcul, car nous constatons expérimentalement que cette stratégie réduit très fortement l'impact de la taille de la topologie sur le temps d'exécution.

Dans la précédente contribution, nous nous sommes donc intéressés au cas multi-propriétaires, où les VIMs n'appartiennent pas nécessairement au même acteur que le NFVO. Avant de placer ses services sur les ressources gérées par ces VIMs, le propriétaire du NFVO a donc dû les acquérir, en payant pour cela un certain coût. Les VIMs réels, que sont les opérateurs de cloud publics tels qu'Azure ou AWS, présentent de vastes choix d'offres pour obtenir ces ressources, et il peut être difficile de trouver la meilleure combinaison en termes de coût.

Bien que plusieurs contributions scientifiques aient abordé l'optimisation des coûts et des prix dans le contexte NFV-MANO, aucune, à notre connaissance, n'a considéré ce problème particulier. Parmi les contributions dans ce domaine, on peut notamment citer l'optimisation des offres proposées par les VIMs au NFVO [144] ou l'optimisation du gain obtenu par le NFVO

lors de la vente de ses services réseau [83].

Notre troisième contribution porte sur la création d'un modèle visant à déterminer la combinaison d'offres commerciales permettant d'obtenir une quantité minimale donnée de ressources au cours d'une période définie, et ce au coût le plus faible [111]. Pour cela, nous développons une représentation générique des offres commerciales des différents clouds publics, en distinguant un prix fixe payé en avance, et un prix variable payé uniquement si la ressource est utilisée. En plus de ces offres publiques, nous considérons aussi le cas où le propriétaire du NFVO possède également son propre datacenter, typiquement de taille limitée. La quantité de ressources à réserver à tout instant doit être déduite à partir de prédictions de trafic.

Afin de mesurer l'intérêt d'appliquer notre modèle, nous l'évaluons en nous basant sur des données de charge de trafic réelles de la ville de Milan [16], courant de novembre 2013 au 1^{er} janvier 2014. Ce trafic présente une double périodicité : quotidienne, avec un trafic élevé le jour et bas la nuit, et hebdomadaire, avec un trafic plus élevé en semaine que le week-end. Il présente également un pic important au 1^{er} janvier. Les offres commerciales envisagées pour réserver les ressources nécessaires sont celles proposées par AWS, l'un des acteurs majeurs des services de cloud public. On en distingue trois types : *réservées* (la ressources est prépayée sur toute la période), *à la demande* (la ressource n'est payée que lorsqu'elle est réellement utilisée), et *planifiées* (la ressource est prépayée, mais seulement sur une plage horaire quotidienne définie, identique pour chaque journée. Ici on choisit la plage horaire 8h00-20h00, qui concentre la grande majorité du trafic.). Le coût horaire d'une offre réservée est égal à celui d'une offre planifiée, et inférieur au coût horaire d'une offre à la demande. L'objectif de notre modèle étant de combiner plusieurs offres pour obtenir le prix le plus bas possible, nous l'évaluons en lui proposant un panel de plus en plus large d'offres à exploiter. Notre prix de référence (100%) sera le prix obtenu en combinant les trois offres publiques. Nous considérons tout d'abord un cas simple, ou seules les offres réservées sont disponibles. Cela correspond à la situation traditionnelle des opérateurs réseaux, qui doivent dimensionner leur infrastructure pour faire face au pic de trafic, ici le pic du 1^{er} janvier. En conséquence, la grande majorité des ressources, prépayées, sont la plupart du temps inutilisées, ce qui conduit à un prix élevé (291%). Nous passons ensuite à une deuxième stratégie naïve : se limiter aux offres à la demande. Cette stratégie est l'exact opposé de la première : cette fois, l'opérateur suit entièrement le paradigme du cloud stipulant que les ressources sont accessibles à la volée, de façon totalement dynamique et sans planification. Le coût total est nettement inférieur à celui obtenu avec la première stratégie (128%), car aucune ressource n'est perdue, ce qui compense le coût horaire supérieur. Cependant le coût total peut être réduit en combinant ces deux offres, et c'est ici qu'intervient notre modèle. On constate cependant que le gain est assez faible (121%), car l'offre réservée reste peu attractive, du fait de la quantité de ressources perdues. En ajoutant l'offre planifiée, beaucoup plus adaptée au profil du trafic, le coût diminue cette fois significativement (100%).

Après avoir démontré l'intérêt de comparer les offres entre elles, nous nous intéressons à l'opportunité pour l'opérateur réseau de construire son propre datacenter, afin d'obtenir des ressources privées. Du point de vue de notre modèle, cela correspond à une nouvelle offre,

l'offre *privée*, qui a les mêmes caractéristiques que l'offre à la demande, mais avec un coût horaire bien plus faible. De plus, les ressources proposées par cette offre sont en quantité limitée. L'exécution du modèle permet effectivement d'obtenir un coût plus faible qu'avec les seules offres publiques (81%). Ce résultat est indicatif, et dépend fortement de la quantité de ressources privées disponibles. Cependant, ce résultat ne prend pas en compte les investissements consentis par l'opérateur pour acquérir ces ressources. Pour évaluer l'intérêt réel d'obtenir ces ressources, nous analysons le taux d'utilisation des ressources privées. Sur le mois de novembre, celui-ci s'établit à 57% en moyenne. Ce score faible ne s'explique pas uniquement avec l'absence de trafic la nuit, puisque la moyenne en semaine est de 75%. C'est le week-end que les ressources sont sous utilisées (moyenne : 40%) : les ressources obtenues via la réservation planifiée, très utiles en semaine, sont toujours prépayées le week-end, et utilisées préférentiellement par rapport aux ressources privées, dont le coût horaire est très faible mais non nul.

Les résultats obtenus dans cette contribution montrent que notre modèle permet des économies substantielles, comparé à des stratégies naïves. Il montre également que la construction d'un datacenter privé peut se révéler moins rentable qu'espéré, une fois prises en considération les différentes offres publiques concurrentes. Lors de futurs travaux il serait intéressant de se pencher sur la réduction du temps de calcul induit par notre modèle. Ce temps de calcul nous a notamment poussé à sélectionner une seule offre planifiée parmi toutes celles possibles. Prendre en compte toutes ces offres permettrait des réductions de coûts plus importantes encore.

Pour conclure, cette thèse nous a permis d'appréhender les opportunités et les défis que représentent les nouvelles architectures liées à la 5G en abordant les différentes étapes du processus de déploiement d'un service réseau, de la requête par le gestionnaire de slices à son installation dans l'infrastructure, en passant par le problème de l'acquisition des ressources nécessaires à son déploiement.

D'autres défis restent cependant d'actualité. Nous pouvons par exemple citer l'isolation du trafic des différentes VNFs au niveau des hyperviseurs, qui est particulièrement complexe lorsque des interférences entre les ressources physiques se produisent. Alors que notre deuxième contribution met en avant l'importance des ressources de bordure aux yeux de l'opérateur, il pourrait également être intéressant de se placer du point de vue des gestionnaires d'infrastructures, et de se demander en quel emplacement du réseau la construction de datacenters supplémentaires serait la plus judicieuse, notamment pour faire face à la montée des services requérant une très faible latence, ou une très forte bande passante. Enfin, le processus de migration du réseau d'accès radio traditionnel vers une version virtualisée, à l'instar de ce qui est aujourd'hui proposé pour le cœur de réseau, fait aujourd'hui l'objet de nombreuses recherches et analyses. En effet, ce segment du réseau comporte des spécificités : très faibles latences, très forts besoins en bande passante, mais également bénéfices additionnels obtenus via la centralisation des fonctions de planification de l'usage de la ressource radio, tel que la réduction des interférences, qui sont plus difficilement modélisables via les modèles de placement de VNFs traditionnels.

CHAPTER 1

INTRODUCTION

1.1 Motivations

Incoming fifth generation of mobile networks, or 5G, is expected to support the development of a variety of new services, such as massive Internet of Things (IoT), tactile Internet, 3D video streaming, to cite a few. Official organizations tried to establish lists of those potential use case, such as Next Generation Mobile Networks (NGMN) in [115]. However, those new services impose increasing pressure on the network, requesting strong Quality of Service (QoS) guarantees. According to the ITU [138] those requirements fall into three main categories: Enhanced Mobile Broadband, Massive Machine Type Communication and Ultra-reliable and Low Latency Communications. The need for additional broadband is not only due to additional services, but also to the steady increase of devices connected to the network observed year after year by major companies such as Cisco [37].

In this context, network operators have to look toward new paradigms to radically change the way they operate their networks in order to provide the expected performances. To address the wide variety of services that display very strong - sometimes contradictory - needs, foreseen solution is to rely on virtualization, or abstraction, of the resources. This technique allows operators to slice their unique, monolithic, physical network into several virtual ones, each dedicated to serve a given category of services or a given client, with adapted QoS capacities. Each virtual network is called a slice, and this concept is known as slicing. Besides, in order to optimize the usage of the resources, network management should be centralized. Centralization would also help network operators to implement an increasing number of complex services in an automated way.

Networks perform two types of operations: packet transmission and packet treatment. Packet transmission involves links, switches, and routers. It requires “network resources” such as bandwidth, latency or switch Input/Output. Packet treatment is done via middleboxes such as firewalls or encoders that provide specific network functions by analyzing or modifying the packets, and requires “node resources” such as compute, storage or Random Access Memory (RAM). Network operators have to virtualize both of those operations to achieve full virtualization of the network.

Although many solutions may be imagined, two main frameworks emerged today to allow this virtualization: Software Defined Network, which abstracts the network resources, and Network Functions Virtualisation, which abstracts node resources. Software Defined Network (SDN) consists in centralizing the traditionally distributed control plane of the network into a logically centralized controller. Routers are turned into switches that report to the controller whenever they are unable to handle a packet. The controller hides the network complexity from

northbound applications and offer them connectivity services. Network Functions Virtualisation (NFV), on the other hand, is based on the virtualization of network functions. Traditional middleboxes are turned into software applications, called Virtual Network Functions (VNFs), designed to run over multi-purpose servers. In order to perform the deployment and management of the VNFs, NFV needs a dedicated framework: the NFV Management and Orchestration (MANO). It allows the network operator to deploy VNFs wherever they are needed. In addition, it can scale those VNFs to adapt to the load, making them very flexible.

However, the objective of a network operator is neither to provide simple connectivity nor isolated VNFs: both must be combined to form VNF graphs able to deliver full Network Services (NSs). This can be achieved by extending the NFV MANO framework to handle not only node resources, but also network ones. This additional management can rely, for example, on traditional routers and routing protocols. However, in order to fully benefit from the flexibility that NFV promises, the network resource management should be highly flexible too. This can be obtained by implementing the SDN logic into the NFV MANO framework. In this sense, although NFV and SDN are independent concepts, it can be highly effective to use them together.

Resource abstraction and control centralization can help network operators to address the increasing complexity of their networks and the use cases brought by 5G. However, they come with their own challenges. Regarding virtualization, one of the major issue is resource isolation: a slice must not influence the behaviour of other ones. The SDN architecture, envisioned to manage network resources, is still relatively new. Few large scale deployments exist, and research is still ongoing to solve different shortcomings that can be encountered. When it comes to network resources, especially in the relatively new SDN context, improvements still have to be made. Regarding control centralization, algorithms dedicated to finely manage the resources in a timely manner are necessary to benefit from the promise of better resource usage through centralization. Also, to bring end to end services, network operators may have to request resources out of their own pools. This inter domain scenario is a complex problem that has to be taken into account by the new frameworks, as it raises questions in terms of security and pricing.

1.2 Contributions of the thesis

The objective of this thesis is to address some of the difficulties related to resource management in abstracted networks. To articulate our work we consider the challenges raised by the establishment of a Network Service, a graph of VNFs, in an NFV MANO framework working in conjunction with an SDN orchestrator. The contributions of this thesis can be summarized as follow:

— We first focus on the network resource isolation using SDN in order to establish a connectivity service without any additional network function. This contribution has been published in [113]. In this work we develop an architecture based on the SDN framework to guarantee the QoS of the different flows in the network. This effectively divides the network resources, and especially the bandwidth, between the different flows. In addition, other QoS metrics, such as latency, can be guaranteed too. Existing solutions in this

field either overload the central controller with monitoring messages or allow the QoS constraint to be violated from time to time. Our proposition solves these issues by delegating the admission control to the entry switch. This way, the controller keeps full control over the global management of the network, but local packet filtering is realized by the switch. Generally speaking, the idea of delegating local tasks to local equipment while relying on the centralized control for global decisions can greatly improve the flexibility and efficiency of SDN. Our results show that, using no specific monitoring, the QoS is strictly enforced.

— Once connectivity is established we can add network functions along the path. To do so we consider the problem of the placement of a VNF chain in the network, having secured the network resource isolation and relying on existing hypervisors to perform node resource isolation. This contribution, published in [112], is twofold:
  — We first design a placement algorithm whose objective is to place a maximum number of NSs in the network, at runtime. In the 5G context, we consider the placement of both regular and strongly constrained NSs. For the latter, relying only on central datacenters may not be enough, as reaching them may take too much time or consume too much bandwidth. As a consequence, small edge datacenters must be considered too. The strategy we use is to place VNFs in datacenters with abundant resources first, if possible, in order to preserve resources where they are scarce to serve NSs that need them. Extensive tests demonstrate that our algorithm performs much better than an algorithm which only focus on bandwidth fairness and ignore the node resources.
  — We then focus on the multi-tenant scenario. In this document, a tenant refers to an independent entity that manages a set of resources and propose them to its clients. Examples of resources in our context may be: virtual CPUs, virtual machines, full network services ... In this scenario a network operator then has to buy resources from other tenants that own parts of the infrastructure resources to complete its NSs. Tenants are willing to display useful information about their resources, because they want to sell them. However, they may be reluctant to display the full details of their topology. Our idea is to propose an abstraction method to solve this dilemma. Although abstraction techniques have been extensively studied over the years the novelty here is to include node resources in the abstraction. We leverage this abstraction strategy to improve the performances in terms of runtime of our algorithm in the mono-tenant scenario. Experimental results show that our technique is efficient both in terms of placement performances and runtime.
— Finally, we examine the process of node resource reservation a network operator has to go through to actually implement the VNF chain. This contribution has been published in [111]. When the network operator does not own enough resources to host its VNFs it has to buy them from other tenants, typically public clouds. Those tenants display public offers for their resources. To benefit from long term offers, more interesting financially, the network operator has to derive from its past experience a forecast of its needs and of resource prices. Based on those inputs we designed an algorithm able to determine the

best combination of offers, in terms of prices, to match all the network operator needs. We demonstrate through simulation based on real data that using our algorithm leads to important cost savings compared to straightforward approaches. We also extend our tests to estimate the opportunity for a network operator to build its own datacenter, or private cloud.

1.3 Organization of the manuscript

Chapter 2 presents a general introduction of the context of the thesis and a detailed description of the main concepts used in the rest of the manuscript. We first present the different challenges that 5G is expected to bring for network operators. We then described the three concepts that are expected to be used to face these challenges: SDN, NFV, and slicing. We detail their respective features, and how they can be used in conjunction. Finally, we introduce the general use case that will be used as a main thread in the rest of the document.

Chapter 3 presents our first contribution: the enforcement of service QoS through SDN. After a brief introduction we summarize the contributions that already exist in this field, and detail the improvement that can be made. We then expose our proposed architecture, along with its evaluation.

Chapter 4 introduces our second contribution. This chapter first presents an overview of the scientific publications in VNF placement domain, and highlights that few has been done to cover the case in which several actors are involved in the creation of the VNF graph, which constitutes the multi-tenant scenario. We describe first our holistic model to place VNFs, and then, the details of the heuristic used to solve the problems raised by the multi-tenant scenario and the runtime problem of the mono-tenant scenario. After extensive tests of both the holistic model and the heuristic, we present our conclusions and perspectives for further improvements.

Chapter 5 details our third contribution. We first analyse some of the main proposals related to cost savings in the NFV MANO process, and we point out that the resource reservation has not been discussed much in the community. We then describe our algorithm to optimize reservation costs, and we present a series of tests to demonstrate its efficiency against trivial approaches. Lastly, we discuss the opportunity for a network operator to build its own facility to obtain access private resources in a context of concurrence with public offers.

Finally, Chapter 6 presents our conclusions and opens new research axes.

CHAPTER 2

CONTEXT

In this chapter we detail the new performances that 5G is going to require from operator networks, in terms of data rate, latency or connexion rate. Limitations of current network architectures may prevent them from delivering those performances, and the use cases that rely on them. To address those limitations, new paradigms have emerged, or gained renewed attention. These alternatives to traditional architectures include SDN, NFV, and network slicing. SDN is an architecture that separates network control and data planes, and centralizes the control to allow automated, programmable, and optimized management of the network. NFV framework proposes to turn traditional physical network functions, running over dedicated hardware, into virtual network functions running over generic servers, in order to gain flexibility, scalability, and cost reduction. Using SDN and NFV, network slicing is a strategy consisting in providing network services to several clients using multiple specialized virtual networks instead of a unique physical network, aiming at simplifying resource management and QoS enforcement. After introducing those concepts, we present a use case that will drive the explanations of our different contributions through the thesis.

2.1 5G

Mobile networks are in constant evolution. Since the deployment of the first generation (1G) in the 1980s, needs and technologies have continuously evolved, resulting in the development and deployment of new generations of networks. However, current 4G is reaching its limits, confronted with a global increase of the traffic and the apparition of new use cases requiring higher QoS guarantees from the network, leading to the emergence of the next generation: the 5G.

Global Internet Protocol (IP) traffic is growing at a fast pace. According to Cisco projections for the 2017-2022 period it should increase from 122 Exabytes per month in 2017 to 396 Exabytes per month in 2022 (+26% per year) [37]. This augmentation has several causes, but we can highlight two of them: video traffic, as the highest contributor, and Machine to Machine (M2M) communications, as the fastest growing category. Regarding video traffic, and mobile data in general, two factors seem to drive the data consumption: first, devices are more and more high end, hosting an increasing number of data consuming services. Second, the more bandwidth is made available by the network, the more applications tend to consume it, proposing services with higher data consumption such as videos with higher definition. The impact of these factors can be measured by the average amount of data exchanged during a single connection: a 4G connection involves three times more data than a 3G one in average, and is expected to involve three times less than a 5G one.

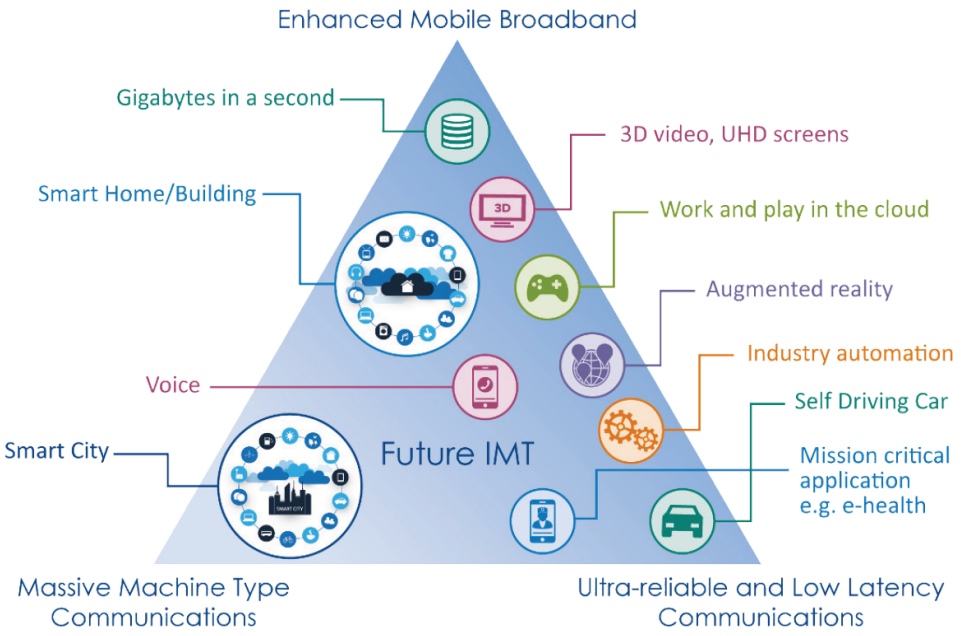


Figure 2.1 – 5G usage scenarios [138]

In addition to this steady increase of traffic the network has to face the arrival of new use cases, some of which are depicted in Figure 2.1. The particularity of those new services is to have very strong requirements in terms of reliability, latency, broadband or amount of connections per second per area.

In order to serve those use cases, 5G will have to display performances radically above 4G ones [11]:

— in normal conditions, the data rate should be 1000 times higher
— minimal latency should drop from 10ms to 1ms
— in spite of the multiplication of devices and the raise of performances, energy consumption and costs should at least not increase, and decrease if possible
— cells should be able to accept tens of thousands of additional connections from low rate devices due to emerging IoT

Those requirements, except for the cost and energy, are designed for extreme situations, and corresponding use cases may require zero, one or eventually two of these capabilities, but not all at once, as depicted Figure 2.1.

To comply with those requirements network operators cannot settle for incremental improvement of 4G technologies, but rather have to rely on new solutions to radically change the way networks are designed and operated, from the air interface [11] to the core. Two of the main paradigm shifts envisioned to build 5G architecture are the virtualization of the resources and the decoupling of data and control planes. Based on those two ideas came two concepts to build future networks architectures: SDN and NFV. Based on those two frameworks, network operators will be able to divide their physical networks into several logical ones, each of them being specialized into one kind of services.

2.2 SDN

In traditional networks, packet routing is handled by specialized hardware: the routers. They work in a distributed fashion, each individual router being responsible for discovering enough information over its neighbours to build and keep up to date its inner routing table, and route the incoming packets accordingly. This is a fully distributed system, with a tight coupling between control and data planes, as both are located in the same physical equipment. This architecture has many advantages, on top of which the scalability, as any new router is responsible for itself. But it also has drawbacks: the distribution may lead to sub optimal exploitation of the resources, the deployment of a new service may require manual intervention on the physical equipment, or even a replacement of the hardware, the variety of routers may lead to different protocols supported, the manual establishment of the services is error prone, and so on.

To face these issues the SDN [46] approach gained momentum over the years. It has three defining characteristics. First, it separates the control plane from the data plane. Second, it consolidates the control plane, logically centralizing it into a single entity called SDN controller. Third, it exposes abstracted network capabilities to northbound applications, hiding the complexity of the network. This concept is not actually new, and the debate over a centralized network versus a distributed one has been going on since the early days of Internet [68]. Today, SDN is pushed by the increasing capacity of network links, the continuous extension and complexification of the networks, the multiplication of service requests, and the apparition of datacenters that bring cheap and nearly infinite compute capacities across the network (primarily on core locations, but gradually gaining the edge), far more abundant than routers ones. Moreover, the incoming of 5G and the new QoS intensive services that it brings encourages an evolution toward a more flexible, programmable way to orchestrate the network.

Authors in [46] provide a quite detailed view of the SDN architecture. However, the Open Networking Foundation (ONF) representation [135] presented in Figure 2.2 is broader and generally preferred. It is composed of three layers. The infrastructure layer handles all data plane operations and is composed of interconnected switches that communicate with the control layer via a dedicated protocol. The application layer contains all the applications that may require network services. They can express their needs through dedicated APIs. The control layer centralizes all control plane operations. It receives applications' requests and translates them into rules applied in the infrastructure layer. To do so, it has to gather the topology information from the switches, and send them the rules they have to apply to manage incoming packets. In practice, the default behaviour of an SDN architecture can be described as follows: when the first packet of a flow enters the network the switch has no specific rule to handle it, as a consequence it matches the last rule of the switch, which consists in wrapping the packet into a specific protocol and sending it to the controller. Based on its view of the network topology, and the various policies that may apply, the controller determines a path for this packet, and issues a set of rules to all switches on this path to actually establish it. Then, subsequent packets will follow the path without any further interaction with the controller.

One of the major challenges of SDN, as any emerging standard, is to be able to interconnect with legacy equipment. Authors in [99] tackle this issue. They postulate that SDN-controlled area should interconnect with legacy area using legacy protocols. To do so they developed an

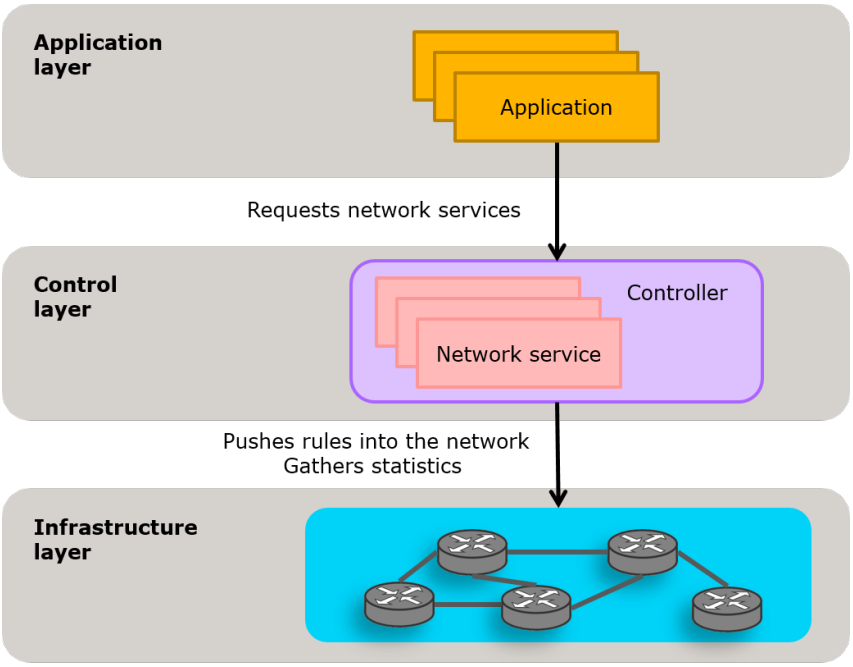


Figure 2.2 – SDN architecture

application running on top of an Open Network Operating System (ONOS) controller able to interpret Border Gateway Protocol (BGP) messages, and keep an updated Routing Information Base (RIB). The controller then creates a full mesh between border switches, and incoming packets are routed to the right direction according to the RIB.

SDN promises many advantages, some of which are common with NFV:

— vendor neutrality: the ONF advocates for the use of open standards for all interfaces of SDN. This openness would allow new vendors to propose their products, and break the traditional vendor dependence in operator networks. The barrier for new actors is even lowered by the simplification of the dataplane equipment, that do not have to implement complex routing protocols.

— programmability: new applications can be deployed in the network via centralized, automated mechanisms relying on SDN programs. It makes the implementation of new services faster, and less prone to errors. Thanks to openness, SDN programs can be written by network managers themselves: operator companies do not have to rely on dedicated teams specialized on each brand of hardware and software. Such automation, however, must be accompanied with many validation and troubleshooting tests. Morover, the SDN automation is also done to hide lower level heterogeneity and complexity. It should however be kept in mind to avoid or understand side effects of the SDN programs [22].

— centralization: the network view and the routing decisions are logically centralized, which allows better use of network resources. This is especially useful for managing large amount of bandwidth, or dealing with ultra low latency applications through optimized, pre-provisioned routes.

— security: although the softwarization raises new threats, the centralized network view allows faster dataplane attacks detection. Equipment failures are also easier to identify.

— scalability: thanks to the multiplication of datacenters, centralized controllers have access to a vast pool of resources, and can manage large networks. Moreover, centralized management allows dynamic load balancing across the network, to relieve highly solicited areas, without violating SLAs.
— reduced costs: both CAPEX and OPEX can be reduced, the first thanks to the vendor neutrality that fosters competition among vendors, and the second thanks to simplified switches and eased management.

From an implementation point of view, SDN landscape has first been dominated by open software and protocols coming from academia or industry research groups. Regarding carrier grade controllers we can cite OpenDaylight (ODL) [104] and ONOS [118]. ODL was initially used for concept demonstration purposes. However, backed by a vast community of developers and industry supports, it reached production-ready state. ONOS, on the other hand, was designed from the beginning for industrial use. It is called “Operating System” because, just as a computer OS, it has a build in finite resource management, isolation between users, capacity to abstract complex resources from the user and security. Back in the time of early SDN controllers all those features were usually optional. Moreover, ONOS is able to manage large scale networks thanks to its distributed core mechanism (see Annex 7.2 for more details). Today, although many controllers emerged, ODL and ONOS are still predominant among SDN users [148], and many other controllers are based on them.

Although SDN implies a centralization of the control plane, it is important to note that this centralization is logical, and not necessarily physical. Early controllers, mainly designed for test and demonstration purposes, were not affected by this distinction as they were composed of a single piece of software, running on a single machine, and then were both physically and logically centralized. Carrier grade controllers cannot afford such simplicity, as physical centralization induces several drawbacks in a carrier-grade applications context, such as a lack of responsiveness, limited scalability and low reliability. To overcome those challenges the control has to be physically distributed, while remaining logically centralized, *i.e.,* the controller may be split into several physical locations while being seen as a single entity by both its clients and its resources. Such distribution implies to maintain the consistency, availability, and partition tolerance of the distributed controller. To address those challenges two main architectures are envisioned: flat and hierarchical. In flat architectures all separated controller instances are identical and have the same level of responsibility. The synchronisation process is carried out in a peer to peer fashion. Hierarchical architectures, on the other hand, propose that higher level controllers manage the synchronisation for lower layer ones. SDN control distribution incentives, challenges, and proposed solutions are described in greater details in Annex 7.2.

SDN controllers require two interfaces, northbound and southbound, and one additional optional east-west interface. The northbound interface connects the controller to its clients, allowing them to request network services via dedicated Application Programming Interfaces (APIs). No standardization exists, but two trends emerge. First, the direct communication with the controller using for example REST APIs. This is the solution chosen by ODL. Second, using an abstraction layer to allow intend-based programming of the network. This solution brings the modelling language closer to application natural language, while solutions like REST stay very

close to network specific terminologies. NeMo project [1], supported by Huawei, explores this strategy. The east-west interface is used by the controller to communicate with its peers. It is not standardized neither. The implementation of this interface is required in case of distribution or replication of the control plane. This aspect will be detailed Section 7.2. The southbound interface connects the controller with its switches. One important goal of SDN is to achieve vendor independence. This implies that any controller from any vendor should be able to operate with any switch of any other vendor. To do so controllers and switches must communicate via an open protocol, and the southbound interface must be standardized.

Although alternatives exist, the OpenFlow [103] protocol is the most popular solution to connect switches with controllers. It was first intended for academic purposes, hence, has many performance shortcomings, such as static header field, which size may become a problem as the number of fields grows [21]. We will explain this protocol in greater details in Section 3.3.1.2. In the infrastructure layer, the most famous switch implementing OpenFlow is the Open vSwitch (OVS) [123]. It suits well to SDN philosophy as it is open, programmable and multi-platform, thus, free from any vendor specific hardware. Through its various versions OVS benefited from many improvements that increased its performances. Internal tests realized in b<>com showed that authors' claim about OVS high performances was justified.

2.3 NFV

Among other improvements, SDN allows an automated, programmable way to handle the connectivity across the network. However, in most of the cases a network service is not a simple connectivity service, but involves also network functions. The multiple benefits expected from SDN cannot be fully exploited unless the network functions management implements equivalent advantages. The NFV concept has been developed to fulfil this objective.

2.3.1 Concept

Today's networks heavily rely on multiple middleboxes, such as firewalls, proxies or load balancers, in order to provide a wide range of services [26]. Those devices are mainly hardware black boxes provided by several vendors, highly specialized and optimized, disseminated across the network. The importance of those elements is reflected in their abundance: in an enterprise network there are present in the same proportion as routers [139]. Despite this central role, middleboxes turn out to be difficult to upgrade, costly to install, and hard to manage. Those drawbacks mainly come from the fact that they consist in proprietary hardware. Installing and upgrading a middlebox then is very costly, since it has to be physically plugged into the network and manually configured. The maintenance also requires both manpower and equipment investment. Moreover, physical specialized boxes cannot be quickly scaled up/down, as a consequence most of those functions are over-provisioned, and still can be overloaded when the traffic reaches unexpected peaks.

To face those problems some approaches have been proposed in the past years, such as

1. NeMo project, `http://www.nemo-project.net/`

in [137] where authors propose to consolidate pools of middleboxes by decoupling control and data planes. The control centralization authorizes partial de-specialization and load balancing to face peak activities. Although this solution acknowledges and addresses middleboxes limitations by introducing additional flexibility, it is mainly an adaptation of the existing system and it does not allow to reach all the capabilities that are expected for 5G networks.

In order to address those issues, a group of industries launched a call for action around a new paradigm in 2012: the NFV [114]. This concept consists in turning the traditional physical middelboxes running over specialized hardware, also referred to as Physical Network Functions (PNFs), into software VNFs running over generic computing devices located either in large central servers (public cloud), edge servers, or even user premises. The NFV call for action has attracted a lot of attention from the research community which analyses the benefits promised by NFV, and identifies the challenges that remain to be solved [158] [109] [114].

A 2016 report on Network Virtualization technologies issued by SDX Central [148] identifies the following points as the main advantages of NFV, ranked by importance according to a survey realized among users and constructors:

— flexibility: this advantage has multiple aspects. First, NFV enables multi-tenancy by allowing several parties to share VNFs. Second, networks can be reconfigured, modified without actually changing any hardware. In the same spirit, new services or equipment updates can be realized without any physical intervention, and with almost no delay. This strongly reduces the time to market for new offers. In addition, a wise management of the resources may also lead to energy savings, ultimately resulting in lower costs.
— Operational Expense (OPEX) cost savings: this advantage mainly results from a highly simplified management relying on centralized orchestration systems that hides complexity and automates service deployments (see Section 2.3.3).
— scalability: resources dedicated to a task can be adapted depending on the workload through automated scaling processes (see Section 2.3.5).
— Capital Expense (CAPEX) cost savings: NFV advocates for breaking vendor dependence and promotes openness by using standardized software over generic servers instead of proprietary software on proprietary hardware. This strategy is likely to open up the concurrence between VNF providers and lower the purchase costs. In the meantime, using multi-purpose servers that can survive NFV updates will be cheaper than running specialized, costly hardware that may soon be outdated.

However, being a new technology NFV still has many challenges to overcome:

— compensate the loss of performances due to the migration from specialized to multi-purpose hardware
— maintain a compatibility with traditional PNFs
— enable the management system to efficiently discover available resources, and allocate these resources optimally
— develop standards to foster the use of open interfaces and software
— solve any security, privacy and isolation issues that may arise from softwarization and multi-tenancy

— present a unified and normalized license model, adapted to NFV use cases [55]

In this section we analyse in details some major aspects of NFV, starting with the key element NFV is meant to provide: Network Services.

2.3.2 Network service

Network Services (NS) represent services that the network operator is supposed to provide to its customer, either using NFV or any traditional system. In NFV those services have a specific structure and, although the functions that compose them are virtualized, they have to be embedded at some point.

2.3.2.1 Network service structure

The exact structure of Network Services may vary depending on the source. In this document we base our works and descriptions on the structure proposed by European Telecommunications Standards Institute (ETSI), which is one of the most detailed to date. The general structure of a NS is described in [66]

Traditional NSs are made of sets of PNFs connected together. Similarly, in NFV a NS is decomposed into a set of VNFs connected together with Virtual Links (VLs). This decomposition is referred to as Virtual Network Function Forwarding Graph (VNFFG). Each VNF provides the services corresponding to a single Network Function (NF). A representation of a Network Service decomposition is depicted in Figure 2.3. While the VNF represents the smallest component from a network point of view, it is further divided into several sub-components called Virtual Network Function Components (VNFCs), which are the smallest components from a resource point of view. Each VNFC runs over one single Virtual Deployment Unit (VDU), and each VDU can host at most one VNFC [64]. A VDU is an object composed of a set of resources required to run software, such as storage and compute. Just as VNFs are chained to form an NS, VNFCs are chained to form a VNF.

Further VNFCs implementation details are out of ETSI's scope, however we may explore some of them to better understand the different VDU offers that can be encountered. There are many different ways to implement software, but some specific designs emerged over time, to avoid the drawbacks of a monolithic application. We can cite for example Model View Controller or Multi-Tier patterns. More recently, the microservices pattern [48] raised a lot of interest in the development community in general, and in b<>com in particular. It consists in dividing an application into a set of microservices, each one focused on a specific task, with a specific context, oblivious of the internal details of the other microservices. They communicate together via APIs agnostic from the programming language, such as Representational State Transfer (REST). Microservices allow any application to be highly available, as they can be replicated, and easy to update, as each one can be modified independently. In this pattern we can either consider VNFCs as microservices, if the structure of the VNF is simple, or we can divide the VNFCs themselves into microservices, if the structure is more complex.

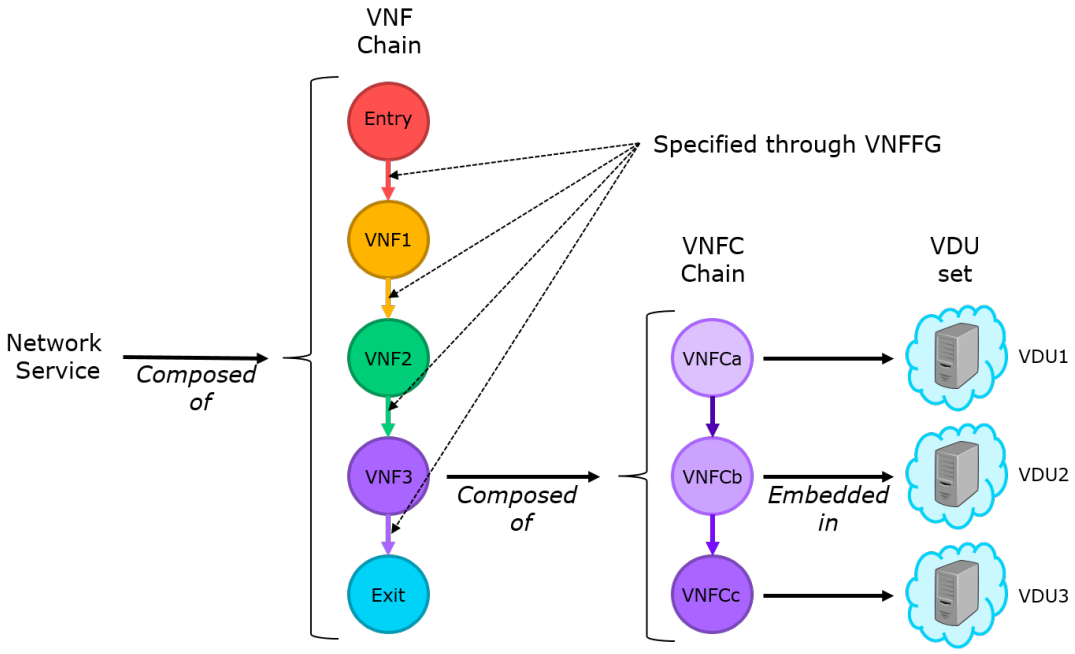


Figure 2.3 – Representation of a network service decomposition

2.3.2.2 Network service embedding

VNFCs must then be installed on VDUs. Although VDUs are often represented as a simple set of raw resources, they must also implement a system to allow the entity responsible for VNFC installation and management to remotely access and use those resources: an Operating System (OS). There are three main ways considered by ETSI to provide such environments [65], that correspond to current virtualization technologies. The first option is to install directly the OS on bare metal. Note that this strategy is sometimes omitted in ETSI architecture representations, as depicted in Figure 2.5, as NFV is mostly envisioned to run on virtual environments. However this first option offers advantages in terms of application runtime. The second option consists in using hypervisors to create Virtual Machines (VMs) from underlying resources, each VM coming with its own OS. Finally, the third option consists in using containers. Containers are built from underlying resources using a Container Infrastructure Service (CIS) that contains a single OS, shared by all the containers.

Some major architecture options are presented in Figure 2.4. Note that the bare metal option is not represented, as it is very close to the VM option, minus the hypervisor. Neither are the VNFs encompassing the VNFCs, for clarity reasons. Figures 2.4a and 2.4b represent respectively VM and container options, with no particular details on the internal implementation of the VNFCs. In those options the VDUs are respectively the VMs and the container.

Figures 2.4c and 2.4d represent the same options, but with microservices implemented. On Figure 2.4d it may seem that the VNFC runs over multiple VDUs. Actually this architecture is inspired by a major container-based virtualization solution called Kubernetes. Kubernetes does not deliver containers directly: it wraps them into pods. The pod is the smallest entity that can be reserved with dedicated resources in Kubernetes, hence it represents here our VDU. The pod may then contain one or several containers. This solution is particularly adapted to

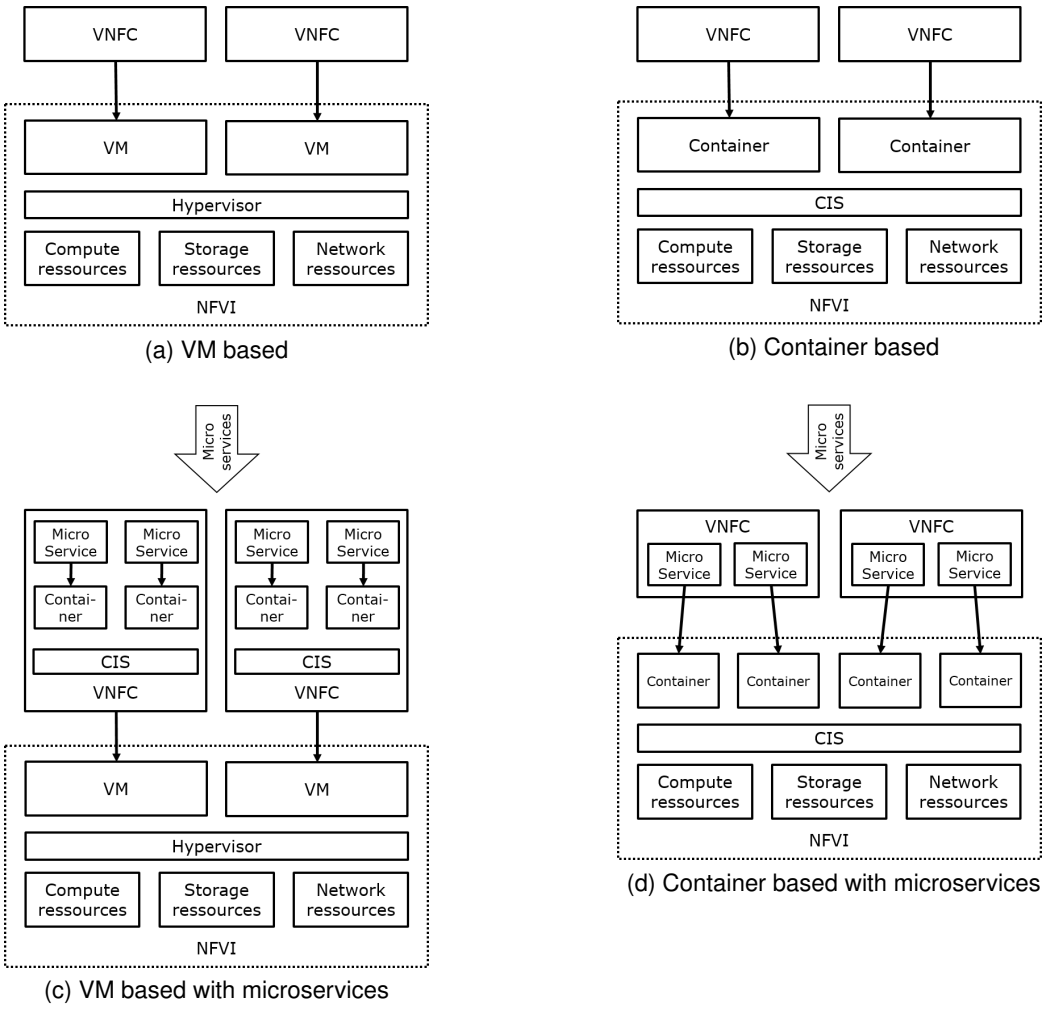


Figure 2.4 – Some envisioned architectures for NFV (derived from [65])

microservices, as containers in the same pod are tightly related to each other. For example, they can communicate over a network internal to the pod. Regarding VMs, such process does not exist, so a VNFC cannot be split over multiple VMs. A solution to implement microservices in this case consists in installing a system of containers within the VM itself, as depicted in Figure 2.4c. This option is not possible for containers, as one container cannot host other containers (or it is at least highly discouraged).

Note that those different architectures can coexist on the same datacenter. Moreover, some alternatives are not represented here. For example, containers in Figure 2.4b can themselves be hosted in VMs.

In order to be deployed and managed in a flexible and automated way, Network Services and VNFs require a dedicated management and orchestration framework, which is a core component of the NFV paradigm. It is responsible for many advantages advertised by NFV, such as flexibility, automation or scaling.

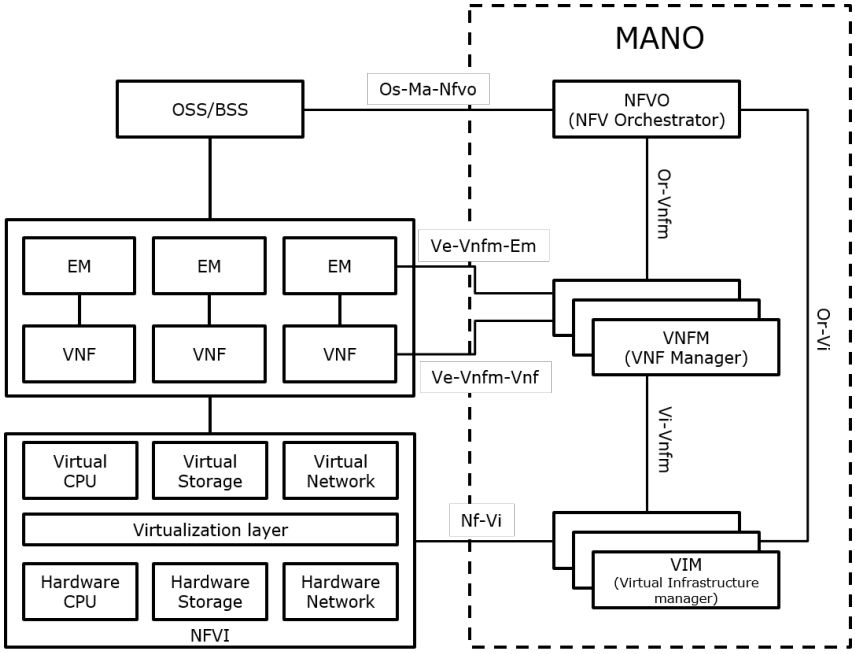


Figure 2.5 – ETSI MANO framework [66]

2.3.3 Management and orchestration

The orchestration and management of VNFs motivated some early research works, such as [39] where authors propose a framework dedicated to the placement of virtual networks. However a consensus emerged toward ETSI framework proposition, namely ETSI NFV MANO [66], presented Figure 2.5.

The ETSI architecture is composed of three main elements: the Network Functions Virtualisation Infrastructure (NFVI), the Operations Support System (OSS)/Business Support System (BSS) and NFV MANO itself. The OSS/BSS issue NF creation requests to MANO, formalized in an Network Service Descriptor (NSD) composed of Virtual Network Function Descriptors (VNFDs), Virtual Network Function Forwarding Graph Descriptors (VNFFGDs) and details about the lifecycle management such as monitoring parameters or scaling policy. Combined together VNFDs and VNFFGDs form VNF graphs (themselves composed of VNFC graphs), and contain all necessary information regarding QoS requirements of the VNFs and the links between them. The NFVI, on the other side, represents the pool of resources on which VNF graphs will be installed. The physical, hardware resources may be abstracted through a virtualization layer - whose main component is usually an hypervisor - and presented to MANO. The role of MANO is to use NFVI resources to fulfill OSS/BSS NS requests, and manage the lifecycle of those NSs, which includes creation, destruction, monitoring, and scaling.

The MANO framework is composed of three entities: the Network Functions Virtualisation Orchestrator (NFVO), the Virtual Network Function Managers (VNFMs) and the Virtualized Infrastructure Managers (VIMs). Upon reception of an NS request from the OSS/BSS, the NFVO has to decide where to place and how to connect the different VNFCs using the resources presented by the different VIMs. Regarding this placement problem, the literature (see Section 4.1) implicitly considers that all the VNFCs that compose a VNF are automatically placed

on the same VIM, so the VNFC placement problem is turned into a VNF placement problem. From an algorithmic point of view it is equivalent. Once the placement is decided the NFVO solicits the VNFMs to create the corresponding VNFs. To do so it provides them the necessary resources that have been obtained from the VIM. VNFMs are then responsible for the lifecycle management of their respective VNFs. When a VNFM needs to access VIM resources, for VNF creation or any other lifecycle management action, it either accesses it directly via the VNFM-VIM interface (direct mode) or indirectly via the NFVO (indirect mode) [63]. In both cases the VNFM first has to obtain the authorization from the NFVO to access those resources through a granting process. VIMs manage underlying NFVIs, and provide the NFVO with updated vision of their resources. They consist either in network capacities (virtual links) or node capacities. When a VIM only manages network resources it is actually called a Wide Area Network (WAN) Infrastructure Manager (WIM) and can then be implemented as an SDN controller compliant with VIM-NFVO and VIM-VNFM interfaces as defined by MANO.

According to ETSI [61] [63], the NFVO obtains resources (VDUs) from the VIM through reservation. A reservation request issued by the NFVO contains either raw resources, VMs, or both. In addition it defines a period of time during which the reservation is active. If the resources are available the VIM responds with a reservation ID. It is this ID that the NFVO may grant to the VNFM when it requests resources during the life cycle management of the VNF, more precisely during the installation and scaling steps. A reservation guarantees the availability of the resource. However, reservations may also be done on the fly, in reaction to an NS installation request. In this case the NFVO is not certain to obtain resources in time. To manage the the resource granting process MANO uses the permitted allowance mechanism. Permitted allowance is an internal NFVO process. Following the network operator policy, the NFVO may define for any resource-consuming entity (depending on the granularity: NS, VNF, consumer, group of consumer...) a maximum amount of resource that this entity can consume. The current maximal consumption is deduced from the resources granted in the past. Whenever a resource granting request would result in a violation of the permitted allowance, the NFVO rejects it.

From an implementation point of view, ETSI MANO has motivated several projects, each one using a different combination of software to implement the three entities [108].

2.3.4 Mono-tenancy, multi-tenancy, and federation

The different blocks that compose ETSI MANO framework may not all be owned by the same administrative entity. For example, the NFVO may be controlled by a network operator, along with a set of VIMs that manage this operator's network, but the NFVO may also be connected to VIMs of other operators, willing to sell part of their capacity, or public clouds operators, acting as VIMs and selling compute and storage resources. In this document, we refer to an entity owning one or several blocks of the MANO system as a tenant. We call the architecture mono-tenant when the NFVO and the VIMs are operated by the same operator, and multi-tenant when the NFVO solicits VIMs belonging to other operators to implement a part or all of the NSs on their infrastructure.

The multi-tenancy is a specific case of a broader concept: the federation. MANO federation

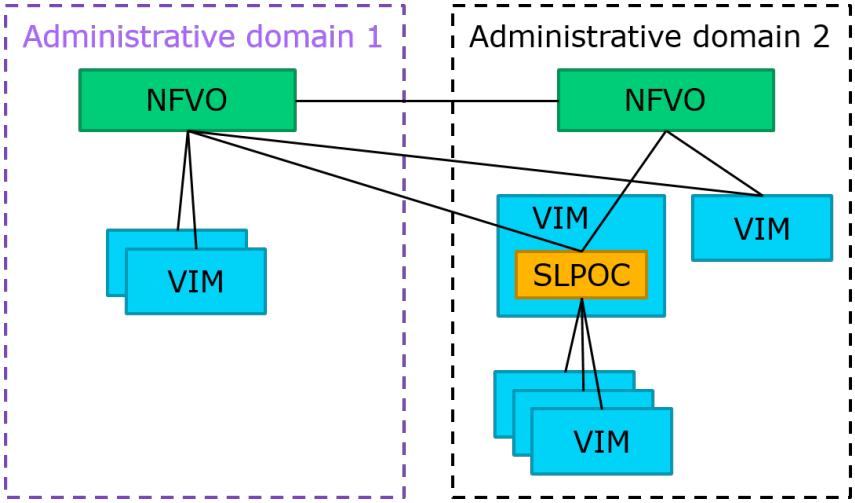


Figure 2.6 – MANO federations

has been studied by many projects, such as 5G-TRANSFORMER[2] or 5G-EXCHANGE[3], in which b<>com was involved. We can distinguish two possibilities of federation, possibly occurring at the same time [58]:

— NFVO-NFVO: the NFVO turns the NS into a composite NS, divided into nested NSs. Each nested NS may then be deployed either by the NFVO itself, or delegated to another NFVO.

— NFVO-VIM: the NFVO requests resources of VIMs belonging to other tenants to implement its NS. ETSI defined two functions to differentiate two cases: Single Logical Point Of Contact (SLPOC) and Multiple Logical Points Of Contact (MLPOC).The former refers to the case where multiple VIMs are aggregated to offer a single resource view to the NFVO. The latter refers to the case where each VIM is connected independently to the NFVO.

In this thesis we focus on the NFVO-VIM federation, or multi-tenancy, where VIMs and NFVO do not necessarily belong to the same tenant. Such situation generates specific problematic that will be developed in Sections 4 and 5. The federation possibilities are represented together Figure 2.6. The MLPOC function is not represented as it is the default case where multiple VIMs are connected to the NFVO. Additional federations can be formed at OSS/BSS level. This aspect will be detailed in Section 2.4.

2.3.5 Scaling

Among all the possibilities offered by NFV automated scaling is very promising in terms of peak traffic handling and resource savings. The general concept consists in identifying Key Performance Indicators (KPIs) in the different VNFs composing an NS. VNFMs are then charged with monitoring those KPIs and, when a given threshold is crossed, they can perform a scaling of the VNFs. They can inform the NFVO. It can then decide to perform scaling in order to adapt the reserved resources to the workload [3]. The scaling may be horizontal (scale out / scale in)

2. 5g-transformer, `http://5g-transformer.eu/`
3. 5GEx, `https://5g-ppp.eu/5gex/`

or vertical (scale up / scale down).

When the decision is made the NFVO reserves resources if it is a scale out, and delegates the scaling itself to the VNFM. Freed resources can be reallocated for other purposes, or servers can be shut down to save energy. This is a major advantage compared to traditional physical middleboxes, that have to be sized to handle peak traffic and cannot serve other purposes, leaving resources idle most of the time.

As explained in Section 2.3.1, ETSI specifications stipulate that NSs are composed of VNFs chained together via VLs, themselves composed of VNFCs also chained together via VLs. VNFCs are hosted on VDUs that represent physical resources. Consequently, the capacity of an NS ultimately depends on the VDUs its components are instantiated on. Based on this idea, the scaling system acts at two levels: NSs and VNFs.

At NS level, each VNFD presents several profiles, each one allowing a specific minimum and maximum number of VNF instances to be created. A given set of exact number of instances for each profile is called an instantiation level. The NFVO may collect KPI measurement from the VNFMs, and potentially from other sources, to monitor the performances and the resource consumption of the NS. When the NFVO considers that the current instantiation level is no longer optimal to support the Network Service, it can move to another instantiation level through scaling. These automatic scaling procedures are driven by the auto-scaling rules provided in the NSD. Note that the KPI analysis and scaling decision may be outsourced to a specialized component connected to the NFVO, as proposed in [3]. Once the scaling is decided, the scaling enforcement of each individual VNF is delegated to the corresponding VNFM.

At VNF level the pattern is similar. Within a VNFD each VNFC presents a list of of VDUs profiles. A VDU profile is defined by the resource it provides, namely compute resources defined by the Virtual Compute Descriptor (VCD) and the storage resources defined by the Virtual Storage Descriptor (VSD). For each profile the maximum and minimum number of VDU instances that can be used simultaneously is defined, and a combination of specific number of instances for each profile of VDU forms an instantiation level for the VNF. At runtime, the VNFM may decide to move from the current instantiation level to another one, based on its auto scale parameter and the KPI measurements. Although the VNFM does not require the assistance of the NFVO to perform this scaling, it may optionally request its services to reserve/release the resources needed for/freed by the scaling process. The VNFM should also notify the NFVO of scaling events. Typically, when the VNFM reaches the maximum instantiation level for a given VNF (*i.e.,* the one offering the largest amount of resources), the NFVO may decide to start a scaling procedure at NS level. The scaling system at VNF level is summarized Figure 2.7 using the example of a virtual Mobility Management Entity (vMME) scaling possibilities. The VNF here is composed of a unique load balancer that distributes the workload over several workers backed by one or two databases. Note that the scaling also includes the possibility to scale the VLs using various flavours corresponding to different QoS levels, such as bandwidth capacities.

2.3.6 SDN and NFV

While SDN virtualizes network resources NFV virtualizes the node ones. Although the two concepts are separated and can work without each other they share many similarities both

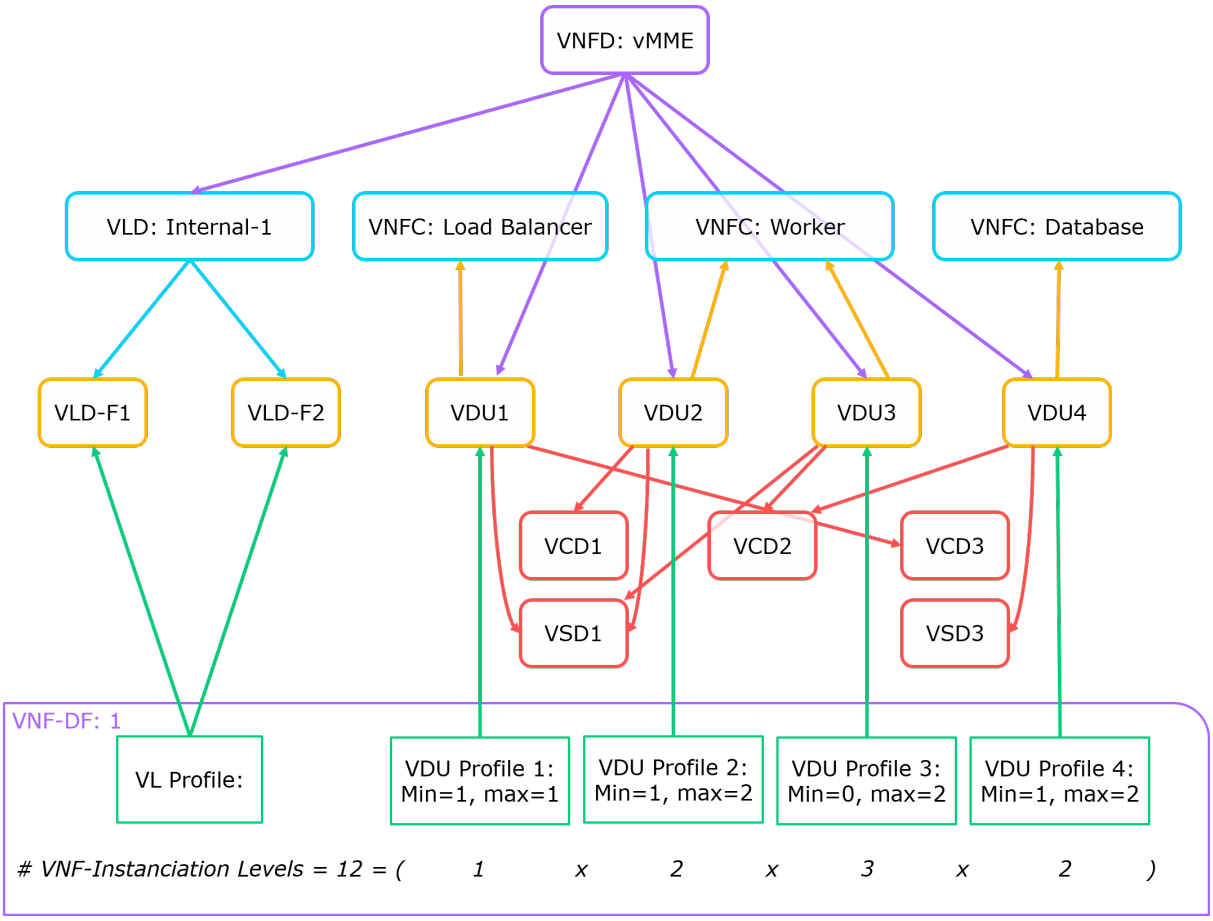


Figure 2.7 – VNF scaling possibilities example

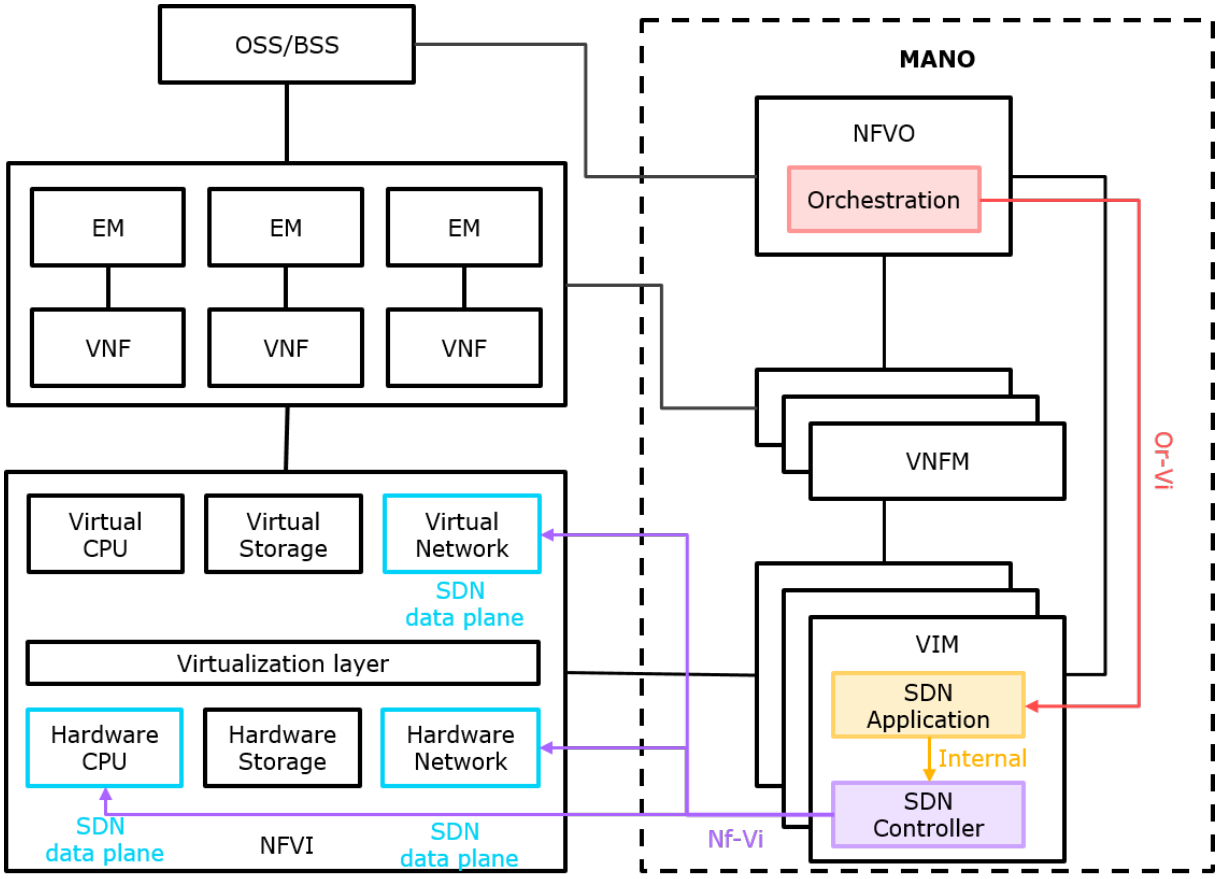


Figure 2.8 – Architectural framework combining SDN and NFV

in terms of objective and characteristics. Both promote centralized, programmable, automated control plane and open, virtualized, adaptable data plane. Without NFV, SDN enhances network operators routing management, but paths have to go through the fixed locations of the requested PNFs to complete the network service. Without SDN, NFV can deploy VNFs on demand wherever resources are available and manage automatically their whole life cycles, but path establishments would remain subject to routers own decisions.

As a consequence, merging SDN and NFV into a single architecture seems natural to fully benefit from their capacities. The most common design patterns to integrate them both within an architectural framework have been studied by ETSI and summarized in [60]. They are based on NFV MANO framework in which SDN is incorporated. The question is then to identify in which of the NFV MANO components the three SDN layers are located (plus an orchestration component which gather monitoring information from either the controller or the application layers). ETSI makes an exhaustive list of the possibilities, however many of them are highly unlikely in practice, either requiring specific prerequisites or relying on interfaces that do not exist in the NFV MANO framework. In Figure 2.8 we represent one of the most likely organization of an SDN-NFV framework that fully takes advantage of the two systems without specific modifications.

In this framework SDN controller is a tool used by the VIM to control parts of the NFVI: the physical switches (network hardware), the virtual switches installed on computers (compute hardware) and the switches of virtual networks, seen as physical ones. Several distinct

controllers can be part of the same VIM, for example to segment the management of different technologies within the resource pool managed by the VIM. The metrics provided by the controller or the application are used both internally by the VIM and externally by the NFVO, which obtains them via its legacy interface with the VIM. In the remaining of this thesis we will use this framework as a reference when merging SDN and NFV concepts. From an implementation point of view, this framework is also the one that Telefonica is planning to use in its SDN-NFV based solution UNICA [41].

2.4 Network slicing

5G is expected to bring new use cases with strong QoS needs. The requirements can vary a lot depending on the use case, possibly involving very low latency, very high throughput or massive amount of connections. To face this diversity the traditional monolithic network is ill-adapted. Resource management, already challenging today, will become unbearable with traditional QoS management systems, especially with the expected evolution of the traffic load [37]. SDN and NFV, through their abstraction abilities, present an unique opportunity for network operators to abstract their unique, monolithic networks into several virtual networks, each one specialized into one category of services and isolated from the others. This concept is called network slicing, and virtual networks are referred to as slices. A slice may be installed over various administrative or technical domains, but presents a unified end to end service to its customer.

Network slicing has been subject to various attempts of normalization, and the definition may vary depending on the source. In 2017, ETSI issued a report [56] listing major contributions in this domain. Although details may vary, all propositions are very similar. In this document we describe and use the 3GPP proposal [1]. This approach is well documented, has already covered many technical aspects of the slicing and fits well within ETSI NFV MANO architectural framework.

A network slice is based on one or several network slice subnets. While the network slice offers end to end network services, such as a full mobile network, slice subnets may provide more specialized services, such as a RAN, a transport network or an Evolved Packet Core (EPC). The slice selects the slice subnets that suits its needs. For example, a slice dedicated to IoT use cases will probably be built over slice subnets offering LoRa or Sigfox access. Slice subnets are nested: one subnet may be the aggregation of several others. On top of the slices, communication services exploit slice abilities to deliver services to final user, such as a given amount of VoIP connections over a given area. An overview of the slicing system is presented in Figure 2.9 (for sake of clarity slice subnet nesting is not represented). To summarize, from the client point of view a slice represents a high level virtual network able to deliver a set of end to end services, while from the network point of view slicing involves a resource isolation challenge.

To gather network services, slice subnets have to rely on some network orchestration framework, typically the NFV MANO framework enhanced with SDN capabilities. Such combination is, for example, proposed in [121]. Figure 2.10 represents 3GPP slice management system

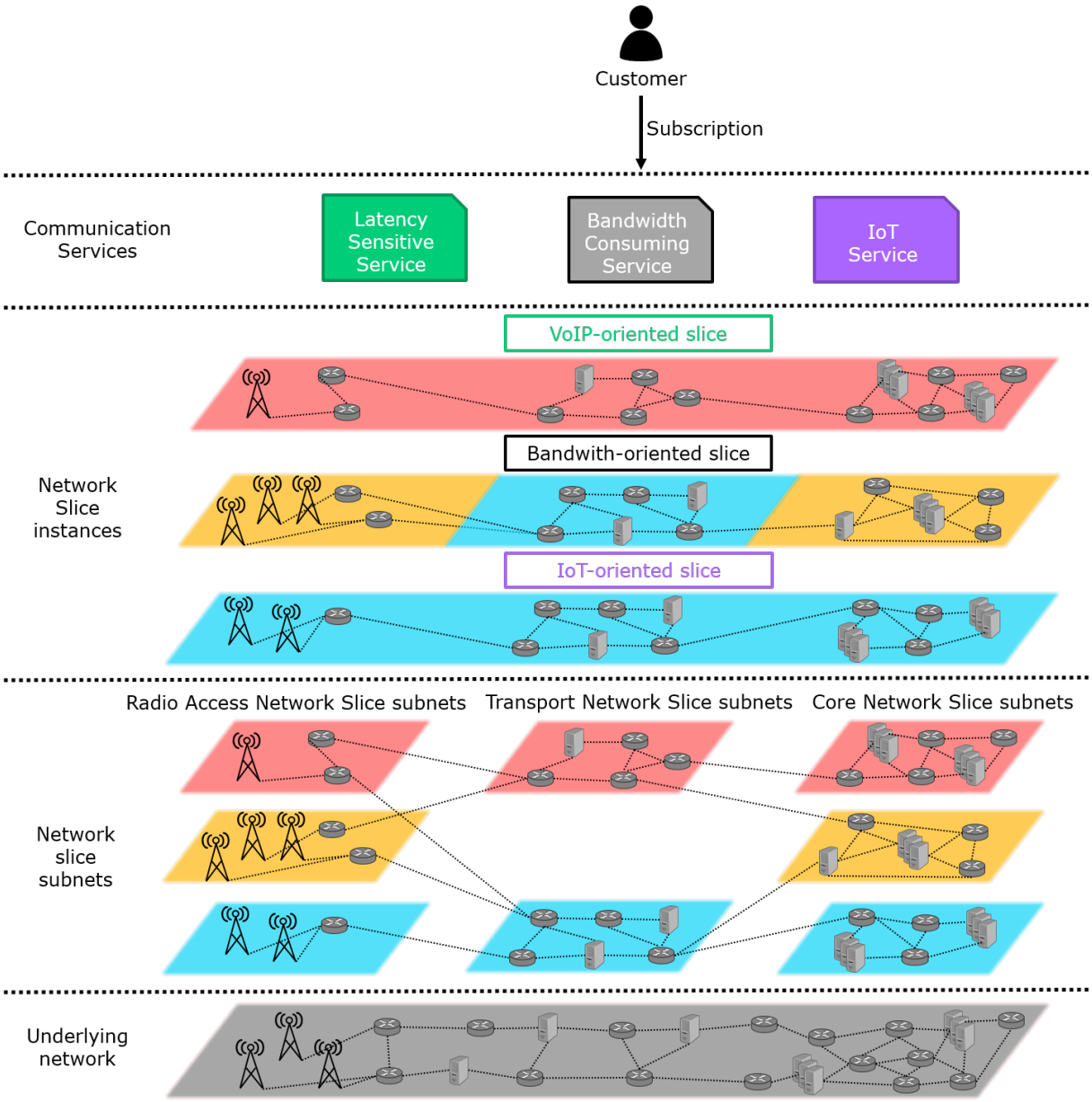


Figure 2.9 – Example of slicing for mobile network VoIP, data and video services

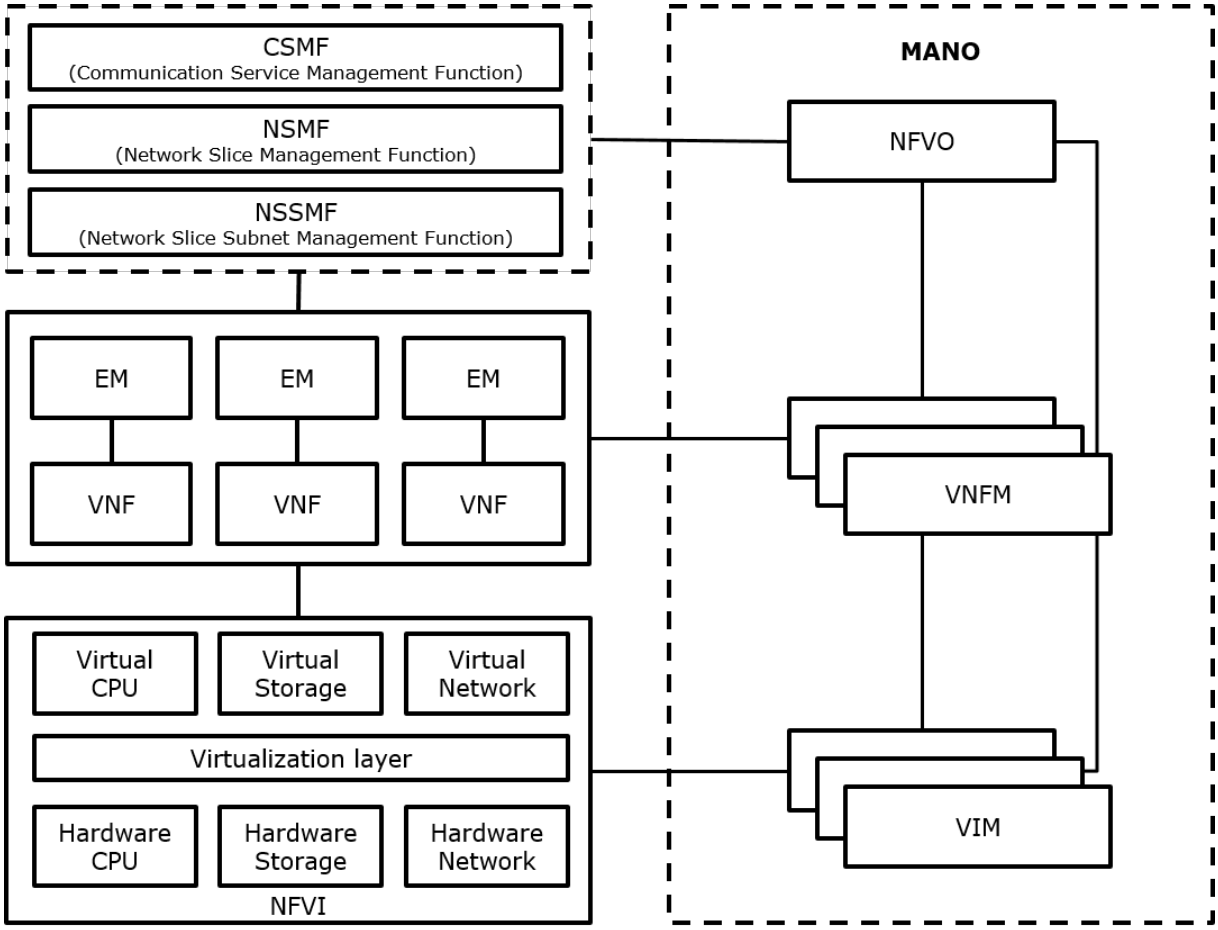


Figure 2.10 – Network slice management in an NFV framework [56]

interfaced with NFV MANO. Slicing does not imply any major modification to the MANO framework, however it emphasizes the need for a strong isolation of the NSs in terms of performance, resiliency, security, privacy, and management.

The main challenges that network slicing is confronted with today are isolation and responsiveness [121]. Those objectives are challenged respectively by the sharing of a common underlying infrastructure and the multiplication of orchestration and abstraction levels. Regarding the first point, our first contribution presented in Chapter 3 focuses on traffic performance isolation. Regarding the second point, efficient algorithms for resource allocation are part of the solution. A proposition on this topic is made in Chapter 4.

In addition to those challenges some questions remain open regarding the exact characteristics of the slices. For example, although it is certain that the core should be sliced, it is still not fully clear whether the slice should go down to the User Equipment (UE) or if it should be interrupted at some point in the Radio Access Network (RAN), considering that full slicing will have a negative impact on the rare radio resources and the stringent timings that characterize this part of the network [32].

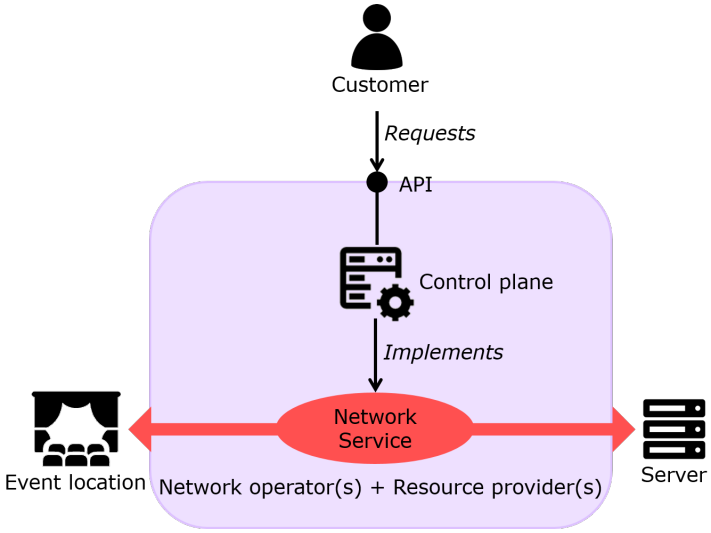


Figure 2.11 – Studied use case

2.5 Use case

Our work follows one main thread: the establishment of an end to end network service between two entities. It could be, for example, two servers for a data transfer, or one event location streaming a video to a remote server for further usage. The main idea is that the network operator's client should be able, through a dedicated API, to request the desired network service, and this service should then be handled automatically and rapidly delivered. This request also includes a Service Level Agreement (SLA) that specifies the expected QoS or Quality of Experience (QoE). This process is illustrated in Figure 2.11.

In this thesis, we suppose that the client expresses its needs with a very low level language, defining, for example, the required latency between two points, the minimal amount of required bandwidth or the quantity of CPUs necessary to run a VNF. As we evoked earlier, this might not be the case: the client may formalize its requests with a higher language involving, for example, a guaranteed number of connections within a given geographical area, an average availability of the service or QoE metrics. In this case, we suppose that the network operator possesses dedicated tools to convert those high level requests into low level ones.

In the remaining of the thesis, we will first focus on the design of a customized SDN architecture that delegates local policy enforcement to the switch in order to establish a simple connectivity service with guaranteed QoS that does not require constant monitoring from the controller (Chapter 3). Then, we will move to the realization of a VNF chain placement algorithm in order to enrich this simple service with network functions to turn it into a full network service. This algorithm will focus on optimizing the placement on edge resource, scarce but well-suited to host latency-constrained services. A heuristic will be introduced to tackle the placement problem over multiple domains (Chapter 4). Having decided where to place the network functions we will focus on the process of resource acquisition run by the network operator to be able to actually install the network services into the network. The core of this problem resides in the design of an algorithm able to satisfy the resource needs at the lowest cost, based on public and private cloud commercial offers (Chapter 5).

CHAPTER 3

QOS ENFORCEMENT MECHANISM IN AN SDN ARCHITECTURE

In this first contribution we explore the bandwidth reservation scenario raised by network service providers such as Telediffusion de France (TDF). Clients request a given amount of bandwidth for a restricted period of time to connect two locations. This situation may occur when a client wants to organize a punctual massive data transfer between two data centers, or when a client such as a stadium, a theatre or any actor of the events industry requires a temporary connection between its location and a data center. As implemented by TDF, clients must precise a Committed Information Rate (CIR), which will be their guaranteed amount of available bandwidth at any time (in the rest of the Chapter we use CIR or Guaranteed Bit Rate (GBR) without distinction). They also indicate a Peak Information Rate (PIR) that represents the maximum bandwidth they may reach (typically the size of the physical link), but which may not be guaranteed any time. They can also express latency constraints. The establishment of this simple connectivity service is our first step toward a full network service. It is represented Figure 3.1. However, guaranteeing a constant QoS with an SDN centralized control represents a challenge. Either the enforcement has to be loose, leading to potential QoS violations, or the controller must constantly pool traffic metrics to monitor resource consumption, and apply corrections if necessary, generating an important traffic load in the control channel and additional computations at controller's level.

To provide such a service, we introduce our solution: the SDN Traffic Engineering Management (STEM) module. In an SDN network, this module provides on the fly bandwidth allocation for users. The process is automated and does not require an operator's direct intervention. Network resources are managed by a Path Computation Element (PCE) located in the control plane. The PCE memorizes allocated resources, and does not have to poll the network equipment to gather information. To enforce the bandwidth allocation policy, STEM uses a fixed number of queues and the meter tool. IP packet headers are not modified. The rest of the chapter is organized as follows. Section 3.1 presents the related works on QoS solutions in SDN networks. Section 3.2 details our solution based on Multi-protocol Label Switching (MPLS). Its implementation and its validation through experimentation are reported in Section 3.3. Finally, Section 3.4 draws a general conclusion.

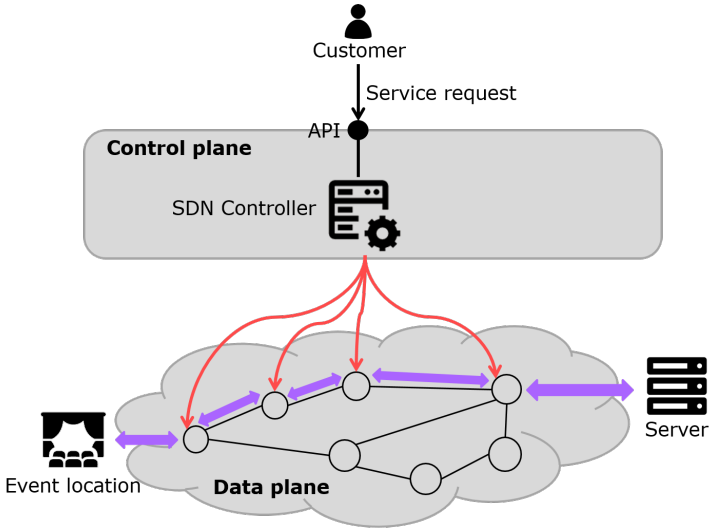


Figure 3.1 – Establishment of a connectivity service

3.1 Existing QoS enforcement methods

New 5G standards such as high peak data rate, high user experienced data rate or very low latency will impose a fine and dynamic control of QoS parameters (bandwidth, latency, loss or jitter). Legacy networks rely on protocols such as DiffServ to apply QoS management in the network. However, those protocols are either fine grained but require complex implementations, or simple to deploy but coarse grained, and are not adapted to the new requirements. Consequently, industry and research started to focus on SDN to handle this problem [85].

The QoS enforcement can be divided into three actions: a request reception (the network service indicates the QoS parameters required by the new flow), the path selection (the control layer determines the most suitable path for the flow) and the path enforcement (the switches or routers are configured to handle packets with respect to their priority and latency constraints).

The selection of a path satisfying QoS constraints forces the control plane to maintain an accurate representation of the network available resources. This is true both to place a new flow into the network, but also to check that existing flows still receive the QoS they need. Indeed, upon traffic changes (*e.g.,* the apparition of concurrent flows), the respect of the QoS guarantees might result in flow re-routing to take into account the variation of the available resources, as detailed in [15]. The proposed approaches are either reactive or proactive. Reactive approaches are based on monitoring to adapt the flow distribution to QoS shortage detection. In [38], authors consider that QoS packets must not be dropped at all, and force network equipment to send a warning when such event occurs: this is an approach based on event triggering, which avoids periodic monitoring. This approach is efficient, but we have to wait to loose a packet to react, although the QoS is potentially already violated and will continue to be so until the controller finds a new path. To reduce this duration, [160] uses path diversity: several paths are pre-calculated for a given flow, so that it can quickly switch from one to another when QoS is no longer respected. However this strategy leads either to over-provisioning or to non guaranteed QoS. Proactive approaches can be implemented by resource reservation. Casellas *et al.* [29] use this technique to allocate optical wavelengths in optical networks, which cannot be done using statistics. This method can be extended to other QoS parameters.

Although the QoS is always respected, provided that the affected wavelengths deliver enough bandwidth, the unused reserved resources are lost.

Reactive solutions are highly volatile since traffic may vary a lot and provoke constant rerouting. To protect path with guaranteed QoS several solutions propose to use queuing mechanisms to prioritize traffic. Queues allow to shape and prioritize traffic, to share the bandwidth and control the latency. The most straightforward option is to directly map priority flows to a high priority queue, and best effort traffic to a lower priority one [50] [136] [154]. [7] refines the process by creating four queues: one for best effort and three for QoS flows, with different levels of priority depending on the status of the flow. Priority queues may be dimensioned to match the quantity of traffic announced by the priority flows, or left un-dimensioned. However, if one flow emits more traffic either the others in the queue or all flows with equal or lower priority are left with no bandwidth, depending whether the queue is dimensioned or not.

To adapt to flow workloads and avoid unexpected congestion, many solutions rely on network statistic gathering techniques to build, and periodically update, their knowledge on both the current network traffic and consumed resources [89] [149] [25] [159]. However, the constant fluctuation of the traffic load implies frequent updates of the statistics. First, this leads to an important augmentation of control traffic. Second, it generates a heavy workload for the controller and requires a lot of computational power. Such aspects are not taken into account, as most of the experiments in the literature consider only small topologies, with reduced number of flows. However, in [43] the authors point out that, in a production network, an excess of statistic polling could dramatically overload the control plane (both switch CPU and controller), and increase the response time of the switch. Third, the reaction time of the system does not guarantee that QoS contracts will be respected anytime, especially between two polls (in particular if the controller is overloaded). To mitigate this issue [102] implements a threshold in the switch. When a link occupancy exceeds this threshold an alarm is sent to the controller which in reaction adapts some of the routes. Although it efficiently solves the statistic pooling problem the controller may still have to recompute large numbers of routes and the solution may still suffer either over-provisioning if the threshold is too low or QoS violations if it is too high.

Presented solutions use pre-defined queues: they are pushed once and for all in the network equipment, and are not modified afterwards. This reduces the flexibility of the QoS offer, since the number of queues is limited. To solve this, some solutions dynamically create queues depending on the needs [73] [149]. Although it enables to control the QoS with a very fine grain, it includes two major drawbacks. First, in real switches the number of available queues is limited. For instance, the PICA8 P-3295 [1] has only 8 priority queues per port, which is not enough to handle all the flows going through a port of a production network equipment. Second, the queue creation mechanism is associated to switch configuration, not to flow rules configuration. For this reason the timescales to create a flow rule and a queue are different, as specified for example in the OpenFlow (OF) documentation [116]. Adding queue creation to flow establishment time will considerably slow down the process.

Monitoring traffic and delays are induced by the SDN concept itself, and the separation of the control and data plane. To address these issues authors in [161] propose DIFANE. They

1. http://www.pica8.com/documents/ pica8-datasheet-48x1gbe-p3290-p3295-v1.9.pdf

design an algorithm to choose in the data plane authoritative switches to store pre-computed rules. Those rules are updated by the controller when a new host enters or exits the network. Regular switches query authoritative ones when they need information to handle a flow. Although this approach is very efficient to handle micro flows, it partially breaks the SDN concept as the management of the network is not fully handled by the controller anymore: although the rules are pre-computed at control plane level the controller does not know which rule will be used or not, hence, the resource consumption is not regulated. To mitigate this problem DEVOFLOW [43] proposes an intermediate approach: the data plane is responsible for the management of small flows, that have limited impact on the resource consumption, while larger flows (also called elephant flows) are managed by the controller. Statistics are aggregated to limit monitoring traffic. Although the impact is mitigated, this solution still implies that the controller abandons part of its authority over the data plane.

To bring flexibility without having to create a lot of queues and to reduce monitoring traffic without giving up the control over the network [93] proposes a solution based on the meter tool. A meter is a switch element that can measure and control the rate of packets [2]. For each flow, one meter is placed in the entry point of the network. The meter counts the IP packets, and alters the Differentiated Service Code Point (DSCP) field in their IP header. The new DSCP can take 3 values: a priority value (if the flow respects its allocated bandwidth), a non-priority value (if the flow exceeds its allocated bandwidth), or a default value (if the flow did not require any specific bandwidth, *e.g.,* Best Effort flows). The main drawback is that the original DSCP value is lost during the transfer, which may impact the treatment of the packet after it exits the network.

We propose a solution to guarantee QoS anytime and provide bandwidth allocation across an SDN network involving the cooperation of the control plane with a PCE to identify suitable paths and reserving the resources in the data plane. To address scalability issues, we use a predefined number of queues, and avoid the constant polling required by statistics gathering thanks to the memorization of allocated resources. Flexibility is provided by traffic classification using the meter tool, as in [93]. In order to preserve all the fields of the transported IP packet, the QoS information will be stored in a MPLS header, not in the IP header itself. This solution effectively moves a local enforcement of a global policy into a local element, the switch, allowing the controller to save its resource for the management of global policies.

3.2 Delegation of the local policy enforcement to the switch

We aim at enforcing QoS policies with the following objectives. The northbound application specifies the constraints and bandwidth value it requests. QoS must be guaranteed anytime. Network resources should be fully exploited, reducing wasted bandwidth to the minimum. The control plane must not be flooded by information. Queues are not dynamically created: they are a feature of the equipment connected to the network. Their number depends on the precision requested by the QoS policy and is limited by the equipment capabilities.

2. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf

To meet these objectives the solution must address resource management and packet handling. We propose the architecture depicted in Figure 3.2. It mostly impacts the control plane and the data plane denoted as the infrastructure layer.

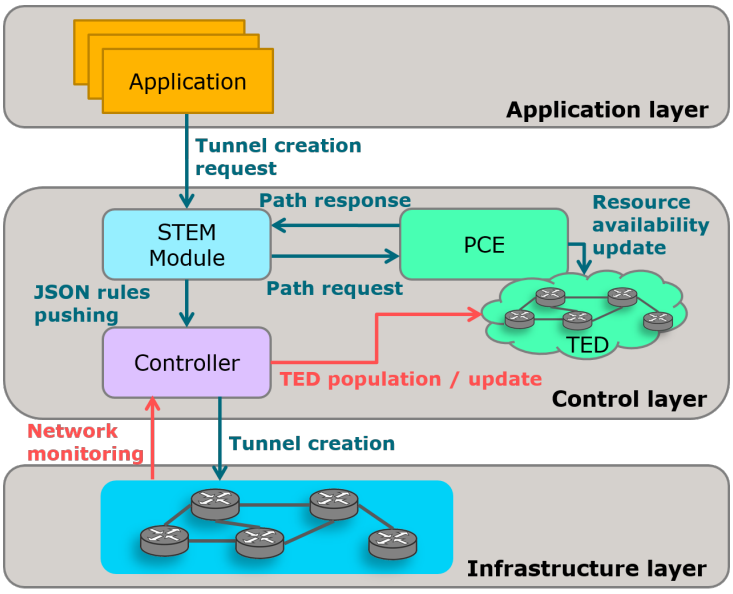


Figure 3.2 – Architecture overview

In the application layer, a northbound application emits tunnel creation requests. These requests include all the necessary information to identify a flow, typically the five-tuple given by the source IP address, the destination IP address, the source port, the destination port, and the upper layer protocol. However, other information (*e.g.,* the DSCP, the Ethernet address or Virtual Local Area Network (VLAN) identifiers) can be used. The request also specifies the desired QoS parameters. Currently the supported parameters are the bandwidth, the hop count or the cost, but bounds on latency or packet losses may be added afterwards.

The requests are handled by the control layer, composed of three modules: an SDN controller, a PCE and our STEM that materializes our strategy of policy enforcement offloading from the control plane to the data plane by designing the appropriate rules for the switches. STEM receives tunnel creation requests, interprets them, and sends them to the PCE. The PCE calculates a suitable path across the network to satisfy the request, and returns it to the STEM module. STEM translates the path into flow rules transmitted to the controller. The SDN controller gathers the network topology information and enforces the flow rules in the data plane switches, two basic features provided by any major SDN controller solution. These commands are received and interpreted by the network equipment, such as switches or routers.

3.2.1 The PCE

Widely studied and standardized [95] in its stateless version, the PCE is used in production networks to calculate routes, possibly under multiple constraints. A PCE receives path computation requests from a Path Computation Client (PCC) through Path Computation Element Protocol (PCEP). To be able to compute paths, the PCE must know the topology and the current load on each links of the network and populates its Traffic Engineering Database (TED)

accordingly. In SDN, the best way to acquire the network topology is to interconnect the PCE and the controller. There are several ways to achieve this goal, depending on the architecture in which the PCE is inserted. The PCE and the controller interconnection can be performed in three ways [28]: either integrated, or external to the controller, or the PCE is seen as an application. As shown in Figure 3.2 the PCE here is used as an application, from the legacy SDN controller perspective. The STEM module plays the role of PCC, and also interacts with the legacy SDN controller as an application (Still, both the PCE and the STEM module are part of the control layer, as they are purely devoted to the control of the network). This configuration has been chosen for various reasons:

— Simplicity: the controller does not have to implement PCEP.
— Encapsulation: since the controller does not implement new modules (*e.g.,* PCE or PCC) the application can use almost any controller out of the box, with only small adjustments at northbound interface level. For the same reason the PCE can be changed or updated very easily.
— Security: as a consequence of a better isolation, separating the different components of the control layer is a good practice to enhance system security [133].
— Maintainability: the separation of the possibly complex controller from STEM simplifies maintenance and reduces potential interference or update problems.

Instead of using flow statistic polling, the control layer relies on a stateful PCE [42] to secure flow reserved resources. A stateless PCE populates its TED and, then, performs path computations based on it, without recording the resource changes. As a consequence, resources in use can be reallocated several times, thus, QoS may no longer be ensured. On the contrary, a stateful PCE keeps track of the allocated resources by recording the computed routes and the associated QoS requirements. This mechanism guarantees that the same resource will not be allocated twice at control plane level (unless over-booking is explicitly allowed).

The PCE does not rely on a specific algorithm to perform its path calculation. The choice is determined by the number of additive QoS constraints, *e.g.,* cost, hop count, latency or loss (which can be reduced to an additive constraint through a logarithmic transformation). We implemented in the PCE the SAMCRA algorithm [152], used when multiple constraints are involved, and a Dijkstra shortest path routing otherwise.

The SAMCRA algorithm can be seen as a Dijkstra with multiple metrics. Its principle can be summarized as follows:

— SAMCRA starts from the source node of the path.
— It explores one random neighbouring node of the source node, making the first subpath. This first subpath P has a length L. There are many different ways to define the path length, here we chose the one proposed by SAMCRA authors: $L(P) = max_{1 \leq i \leq m}[\frac{w_i(P)}{L_i}]$, where $w_i(P)$ is the value of metric i for the subpath P, and L_i is the maximum acceptable value for this metric.
— The algorithm keep exploring the graph. In each iteration, it explores one random neighbouring node of the subpath that has the lowest length.
— When the destination node is reached by the path with the lowest length, the algorithm stops.

— The algorithm also includes two mechanisms to reduce the complexity of the search: path dominance and look ahead.

3.2.2 The SDN Traffic Engineering Management (STEM) module

The STEM module exposes a northbound interface to the applications that want to reserve bandwidth. When a request is received, the module turns it into an appropriate PCEP request and submits it to the PCE. Once the path is computed, it creates the appropriate set of flow rules and sends it to the controller. STEM module does not only creates the direct IP path from source to destination: it also handles the way back and the corresponding Address Resolution Protocol (ARP) tunnels, required for IP communication. Other automatic services can be added, depending on the needs.

The STEM module creates tunnels called Label Switched Paths (LSP) between two points in the network using MPLS, a protocol widely used to forward traffic in legacy networks and offering a traffic engineering functionality. Due to the reduced size of its header, MPLS packets can be treated very rapidly in switches and routers. The MPLS header is formed by a stack of labels. Each entry is composed by a label (L) field (20 bits), a Traffic Control (TC) field (3 bits), a bottom-of-stack field (1 bit) and a time to live field (8 bits). This header is placed between the level 2 and 3 headers (Ethernet and IP in this case). MPLS encapsulates the IP packets, which allows to transport them without alteration. In particular, the TC field can be used to store the packet priority, the DSCP field doesn't have to be modified.

STEM designs rules to enforce the QoS policy using priority queues and meters. First, we set the packet priority (in the TC field) of each QoS flow. To do so, a meter is created in the network entry equipment to count the corresponding packets. When the bit rate exceeds the defined GBR, the meter triggers an action to increment the MPLS packet TC field, reducing it's priority. Best Effort packets have a medium priority. Then, we assign to the packets a label that identifies the flow they belong to. Label and TC settings only occurs in the entry switch. Other switches simply match the label and TC fields to select the output port and queue respectively. We fixed the number of priority values (High, medium and low), requiring only 3 queues in each port. However, additional priority levels might be added if more queues are available, for further latency and loss management. It is to be noted that, as opposed to queues, meters can be created on the fly at the same timescale as flow rules. Moreover, only one meter per flow is requested, not one per flow and per switch.

An example of this strategy is represented in Figure 3.3. Here we use 3 queues, associated with three TC values where TC values for high, medium, and low priorities respectively are 1, 2 and 3. We feature 3 flows entering the network: F1 (1 Gbps reserved, 2 Gbps input), F2 (Best Effort, 2 Gbps input) and F3 (2 Gbps reserved, 2 Gbps input). At the input none are labelled, as they are still IP flows. Entry switches apply them correct labels and TC fields, following the rules previously pushed by the SDN controller: F1 reserved traffic receives high priority, while its exceeding traffic receives low priority. F2 best effort traffic receives medium priority. As F3 stays within its reservation boundaries, all its traffic receives high priority. When they all reach an inner switch that does not have enough output to transmit all of the traffic, the lowest priority traffic; which falls in the lowest priority queue, is dropped. Here, it is the 1 Gbps out-of-reservation F1

traffic. Note that, for sake of clarity, the figure represents two layers of switches. However, the same situation may occur with only one switch, the three flows entering via different ports and trying to exit by a single one.

Considering PIR, the solution to ensure that the connection can, under ideal conditions (*i.e.,* no other traffic), reach a given bandwidth, is to create our MPLS tunnel using exclusively links exposing a total bandwidth at least equal to the requested PIR.

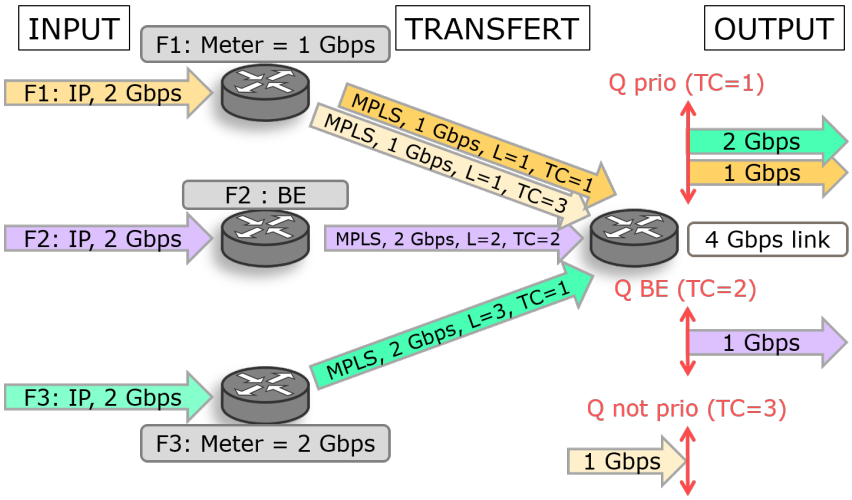


Figure 3.3 – Example of local QoS policy enforcement using the meter tool

Using our strategy, QoS flows are ensured to always receive their GBR, and unused bandwidth is dynamically redistributed to flows that could make use of it. It is relevant with one of our goals: avoid to waste resources. However, this strategy may be harmful from an isolation and security point of view. Regarding isolation, it implies that the workload of a client may impact another one, as it influences the amount of additional bandwidth available (although GBR is always respected). This breach in isolation may create a security problem: a malicious user may analyse the amount of additional bandwidth available through time to deduce the workload pattern of the global network, and of specific tenants, provided it manages to obtain additional information about the nature of the concurrent flows. To solve both the isolation and security issues, the network operator may decide to drop the packets when they exceed the GBR instead of decreasing their priority. Changing this policy doesn't require any modification of our architecture or implementation besides changing the action in the controller flow rules. This is a classical dilemma between strong isolation and optimal resource usage, and it is up to the network operator to define its own policy.

Regarding Best Effort, we cannot apply such dropping strategy to strictly enforce isolation, as Best Effort traffic is, by nature, not isolated from other Best Effort flows. In order to mitigate the security issue caused by this lack of isolation, other strategies such as obfuscation of the sharing technique should be implemented, but this is out of scope of this contribution.

3.3 Implementation and experimentation

3.3.1 Implementation

We implemented a platform to evaluate our bandwidth management solution for SDN networks based on MPLS tunneling and the meter tool. The application layer is simply implemented by a REST client. The implementation of other layers requires the support of OpenFlow, MPLS, queues, and meter. It turned out to be complicated, especially regarding the meter tool.

3.3.1.1 Control plane

In the control layer, the SDN controller is an unmodified ODL controller Lithium SR4 using OpenFlow 1.3 (OF13) to communicate with the data plane [3]. The ODL controller is open source, production-ready and offers features such as rule persistence and dynamic topology update that are promising for further development.

The PCE is based on Netphony PCE [4]. We modified the original software to fully support resource reservation (stateful PCE). The PCEP protocol has been extended to request several different metrics such as cost or latency.The PCE is not stateful active yet: it cannot propose general network optimization.

For our experimentations we do not take into account the requested PIR, as it does not raise any technical challenge. Enforcing this parameter would only require to change the SAMCRA algorithm used in the PCE, so that the path research process discards the links having a total bandwidth lower than the requested PIR.

3.3.1.2 OpenFlow

The protocol we use to carry the communications between data and control planes is Openflow [103]. We chose this protocol because it is widely spread in the scientific community. For this reason it is already built in most of the SDN controllers, including ODL, and in the main software switches used to test SDN architectures, such as OVS [5] and CPqD [6]. We here provide a brief description of OpenFlow rule mechanism, specified in [120]. Advanced features, such as learning rules (rules that automatically generate other rules), are not detailed in this manuscript.

The OpenFlow system is based on tables, each one containing a set of flow rules. Flow rules are composed of three main fields: match, priority, and instructions. The **match field** contains assertions that a packet has to match in order to be treated by the flow rules, such as IP source or destination. A packet can only match flow rules of the table it is currently in (packets all start in Table 0). If a packet matches several flow rules, then, the flow rule with higher **priority field** prevails. Once the packet is assigned to the correct flow rule, actions contained in the **instructions field** can be performed. It may, for example, include modifications

3. https://www.opendaylight.org/
4. https://github.com/telefonicaid/netphony-pce
5. http://openvswitch.org/
6. https://github.com/CPqD/ofsoftswitch13

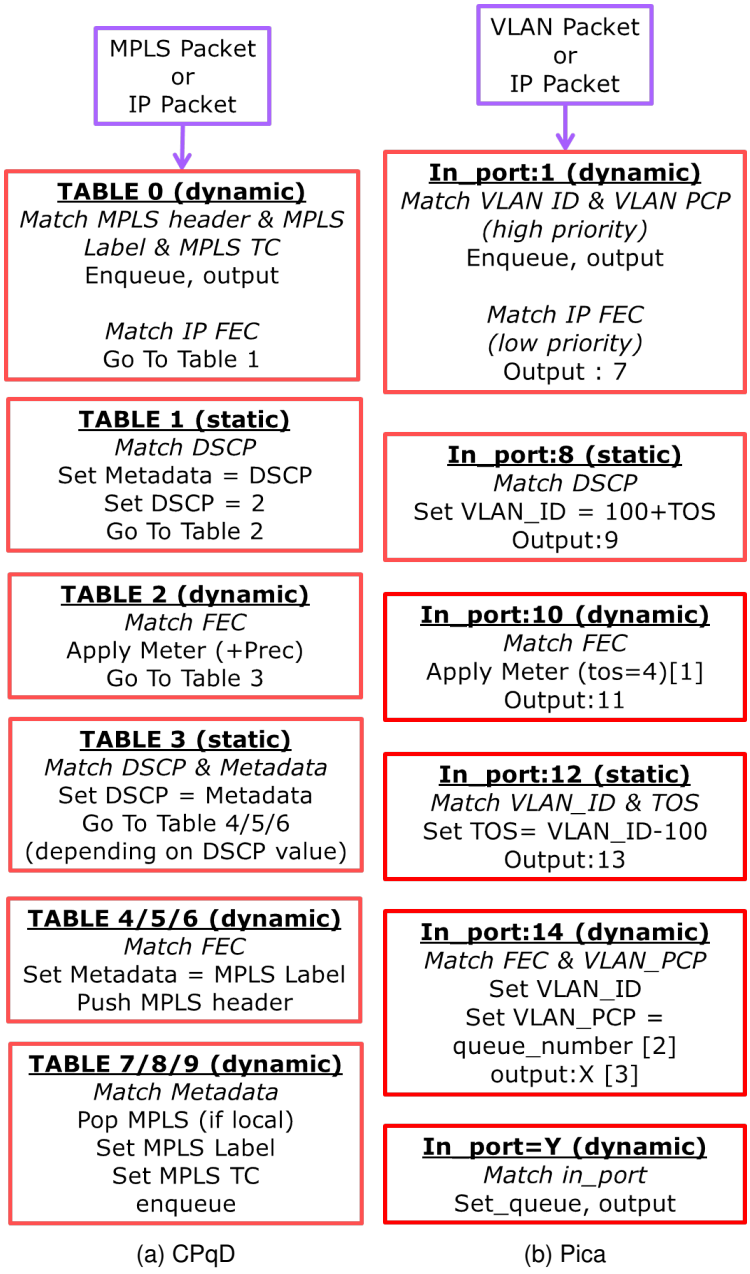
to a given field of the packet header or the assignment of the packet to a given queue. Finally the output action is applied: the packet can either be dropped, sent to the controller, sent to an exit port of the switch or to another table.

Among all the possible actions that can be performed via the instruction field one possibility is to apply a meter. A meter is an object that can be explicitly created within the switch. When called by an instruction, its purpose is to count the packets that match the flow rule per unit of time. It may then apply specific actions on those packets when they cross defined thresholds. Meter can only apply two types of actions: drop and DSCP-remark. Drop results in dropping all packets that cross the defined threshold, effectively implementing a traffic shaper. DSCP-remark increments the Precedence bits of the IP DSCP field, which indicates that the packet priority decreases. As several thresholds can be specified, packets can lose several levels of priority.

3.3.1.3 Data plane

In the literature, the data plane usually integrates a network of OVS. Although these virtual switches are very efficient, they do not implement meter yet, which excludes them from our platform implementation. The OpenFlow SoftSwitch (also named CPqD) implements meter and presents two advantages. First, it is implemented in Mininet, allowing to test the solution on different topologies. Second, it implements all the rules needed for our solution to encapsulate IP packets in MPLS packets and to set the label and TC fields in the MPLS header. The main difficulty is that a meter can only increment the IP DSCP field. The solution used by STEM is presented in Figure 3.4a. The tables that contains the higher number of rules (the ones that have to match the 64 possible values of DSCP) are static: they are pushed once and for all when the switch is connected to the network, and do not have to be pushed for each flow afterwards. Unfortunately, the CPqD switch is not able to push the MPLS header and to set the MPLS fields value in the same rule. If it was possible (as in OVS), Tables 4/7, 5/8 and 6/9 presented in Figure 3.4a could be merged. These rules have been tested. The MPLS headers were successfully set. However, CpqD's queuing mechanism does not perform well, and it is impossible to carry experimentation on bandwidth reservation.

For this reason we used a Pica8 switch P-3295 with Linux System Version 2.6.4 and OVS/OF Version 2.6.4. Pica8 is a traditional material L2/L3 switch that implements OpenFlow protocol, to act as an OVS, and both meter and queues. However, OpenFlow is only implemented at software level, and hardly reflected at hardware level, where all packets are processed. In particular, the switch only has one physical Ternary Content Addressable Memory (TCAM) and cannot implement more than one table. To overcome this limitation, the switch merges the rules to fit in the unique hardware table when multiple tables are used at the software level. Such operation is impossible if a rule updates flow parameters matched by another rule. In this case the resulting hardware rule is erratic, and most of the time results in dropping the packet. For the same reason, Pica8 does not implement the bridge system: bridges declared at software level are ignored at hardware level. In [93], authors dealt with these issues by using multiple switches, each switch playing the role of one table. Moreover, P-3295 does not implements MPLS treatment at hardware level, but only at software level, which strongly im-



[1] misbehaviour: the meter does not increment precedency but re-write the DSCP field
[2] bug: queues re-write the vlan_pcp. It must be set equal to queue number
[3] bug: If traffic is aggregated Pica8 queue misbehaves. Each flow is directed to a different port, then re-aggregated before entering the queue

Figure 3.4 – OpenFlow rules implemented in CPqD switch and Pica switch

pacts performances over 10Mb of traffic. Last, having no tables implies having no metadatas, which are necessary in our implementation, as shown in Figure 3.4.

To overcome these limitations, we emulated multiple tables by encapsulating the IP packet. MPLS encapsulation cannot be used for this purpose because, when a packet is encapsulated in MPLS, it cannot be matched based on its IP fields anymore. We envisioned to carry out the experiment using VLANs: the IP packet is encapsulated in a VLAN when entering the switch. The VLAN ID field represents the metadata, while the VLAN Priority Code Point (PCP) field represents the table. Once the packet is treated, the VLAN PCP is incremented and the packet is resubmitted to its entry port to be treated again as a new packet. In addition to their "normal" matching options, all the rules will also match the PCP field. The advantage of this solution is that it uses only one port and provides an equivalent to the missing metadata mechanism. The drawback is that VLAN cannot be used by the incoming flows.

Due to the Pica8 limitations regarding resubmit action, the VLAN solution couldn't be used to replace all the tables: "Table 3" couldn't match the new Type of Service (ToS) field resulting of the meter action. We tried to bypass this by using several ports to act as different tables. The input port acts as a table identifier. The main drawback of this solution is that multiple ports are used to perform a single set of actions. However, this solution works well on Pica8 and was used to carry out the experiment, in parallel with the VLAN encapsulation to emulate metadata retention. Eventually, MPLS was replaced by VLAN in the experiment due to MPLS lack of efficiency evoked above. This has two main drawbacks in real applications: the encapsulated traffic cannot be a VLAN itself, and the number of tunnels is reduced as there are less available VLAN IDs than possible MPLS Labels.

3.3.2 Experimentation

We implemented a platform with one Pica8 P-3295 switch, three clients (C1, C2, and C3) and one server. From one physical switch we created the equivalent of a 3 switch topology, as shown in Figure 3.5. The corresponding rules are presented Figure 3.4b. In order to assess the efficiency of our bandwidth management mechanism, we designed the following scenario. We generated traffic with the Iperf tool according to the following pattern: between time t=0s to 60s, C1 generates 100Mb of Best Effort traffic; between t=20s and 80s, C2 generates 100Mb of priority 2 traffic with a requested bandwidth of 30Mb; between t=40s and t=100s, C3 generates 100Mb of priority 1 traffic, with a requested bandwidth of 15Mb. The ARP messages corresponding to these traffics are carried in separated tunnels, with priority 3 (the highest).

Results are presented in Figure 3.6 and demonstrate the efficiency of the STEM module. We observe that, at the beginning, only the Best Effort traffic is present and occupies the entire link bandwidth. But, upon the C2 traffic arrival, 30Mb of priority traffic are transmitted, while the remaining 70Mb are blocked. This is normal since they have lower priority than the best effort. The same situation occurs when C3 starts emitting. At t=60, when C1's traffic stops, the exceeding traffics of C2 and C3 compete for the extra bandwidth (55Mbit/s), with equal priority, which results in fair division of the resource(27.5Mbit/s each). In the end, only C3 remains, and uses the whole link capacity.

The flows behave accordingly to the QoS policy. In particular, the link is always used at full

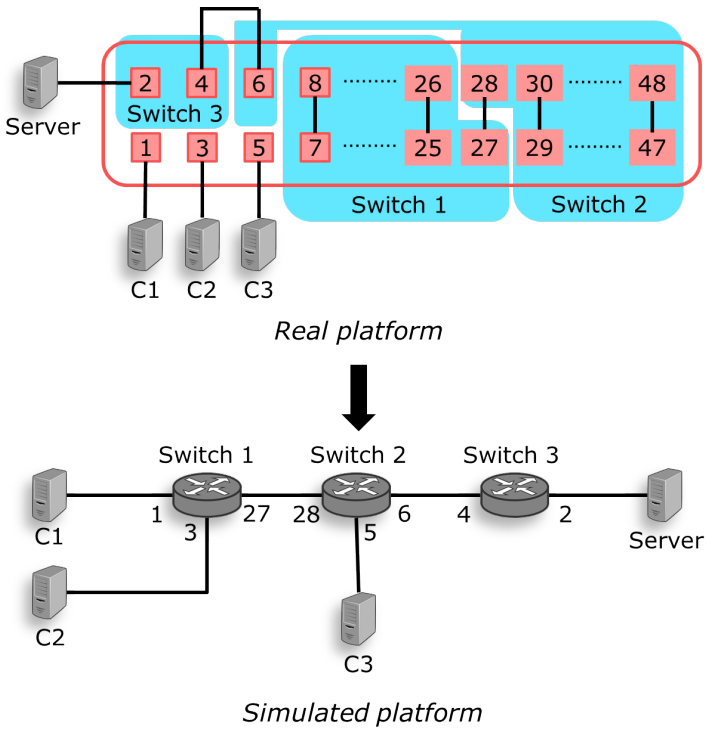


Figure 3.5 – Test platform

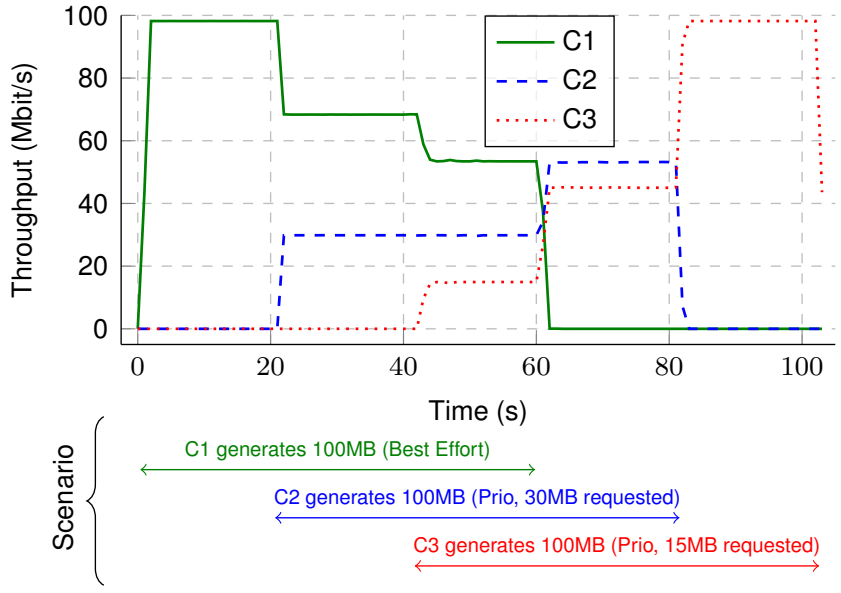


Figure 3.6 – Observed throughput for each client

capacity, even if the bandwidth is never fully reserved: no capacity is lost.

3.4 Conclusion

We presented a full solution to provide guaranteed bandwidth to flows in an SDN network that respects the integrity of the transported packets. The combined use of a stateful PCE and of the meter tool allows network clients to reserve flexible amounts of bandwidth, guaranteed anytime, without constantly polling the network to gather flow statistics. We implemented our solution in a platform and conducted experiments to show that the solution can be deployed in an SDN network. We exposed our difficulty to find a proper equipment to support the different flow rules required by this solution. This shows that improvement are needed at hardware and software levels for OpenFlow-capable devices. Although meter is a known tool in OpenFlow, it is not widely supported yet, and it would be interesting to fully integrate it in OVS in order to enforce QoS requirements in SDN networks. Since the end of our work the drop action has been implemented in the meter tool. However, the DSCP-remark one is still missing in OVS version 2.12.90. At the control plane level, we introduced the module STEM to manage the northbound application requests to forward flows with QoS requirements by delegating traffic engineering decisions to a PCE and the path enforcement to the SDN controller.

This contribution has been published in the 13th International Conference on Network and Service Management (CNSM 2017) [113]. In addition, our architecture has been used as implementation support for two other contributions regarding the placement of VNFs [12] [125].

To go further, an improvement to our system would consist in turning the stateful passive PCE into a stateful active one. Besides accomplishing the same actions as a passive PCE, a stateful PCE is able to suggest global re-optimization of network paths when appropriate (for example, when the rejection rate crosses a given threshold, or periodically). Such improvement would require the development of a re-optimization algorithm, along with extensions to the current implementation of the PCEP protocol. It would improve the resource utilization of the network over time, preventing it from degrading excessively due to successive globally sub-optimal placements.

In this first contribution we provided a solution to efficiently enforce QoS policies along paths connecting endpoints of the network. To move one step further and place full network services we now have to take into account the placement of virtual network functions together with the network paths connecting them.

CHAPTER 4

VNF GRAPH PLACEMENT IN MONO- AND MULTI-TENANT ARCHITECTURES

In the previous chapter, we described an architecture and an algorithm to efficiently use the SDN concept to establish a connectivity service between two end points of the network, providing isolation through guaranteed quality of service. Our objective is now to build a full Network Service based on this first approach. As detailed Section 2.3, in the NFV concept a full Network Service is composed of a VNF graph with specific QoS constraints, both between the VNFs and between the end points. In order to provide such Network Service we have to deploy and maintain the related VNF graph into the network. In the example presented in Section 2.5 in which we want to connect two end points, one streaming a video and the other receiving it, we can expect that the actual Network Service would at least require an encoder, a decoder, and a firewall across the data path. This problem is represented in Figure 4.1.

To efficiently manage this graph we can use the NFV MANO framework enriched with SDN capacities, as presented in Section 2.3.6. In this framework the Virtual Network Function Graph Placement Problem (VNFGPP) is under the responsibility of the NFVO, while the actual instantiation of the graph is delegated to the VIMs. As a consequence, controllers located in the VIMs can use our solution presented in the previous chapter, ignoring the PCE step. This allows the VIMs to finely manage the QoS required by the VNF graph.

Placement algorithms rely on topology information to perform their task. It was the case for SAMCRA in the precedent section, whose TED was populated by the SDN controller. Here, the NFVO relies on the information provided by the VIMs to make a decision. In Section 2.3.4 we introduced the notion of mono- and multi-tenancy, and we precise that an architecture is considered mono-tenant when all its components belong to the same entity (or tenant), and multi-tenant otherwise. Due to its NP-completeness [101] [5] the VNFGPP problem is difficult to solve efficiently in mono-tenant architecture, and multi-tenant architectures introduce additional obstacles as the different actors are not willing to fully share their information. In particular, VIMs may be reluctant to expose the details of their internal topologies to the NFVO. This restriction will have consequences on the VNFG placement.

In addition, in 5G some Network Services are expected to express strong QoS requirements, such as very low latency or high bandwidth. In order to meet those needs, operators cannot only rely on traditional cloud resources, that may be located far away from users [36]. They must also exploit local edge resources. Those scarce and expensive resources call for a wise and adapted management in order to serve a maximum of services.

In this chapter we propose an optimization strategy to maximize the acceptation of new Network Services by reserving in priority resources on links and nodes where they are abundant,

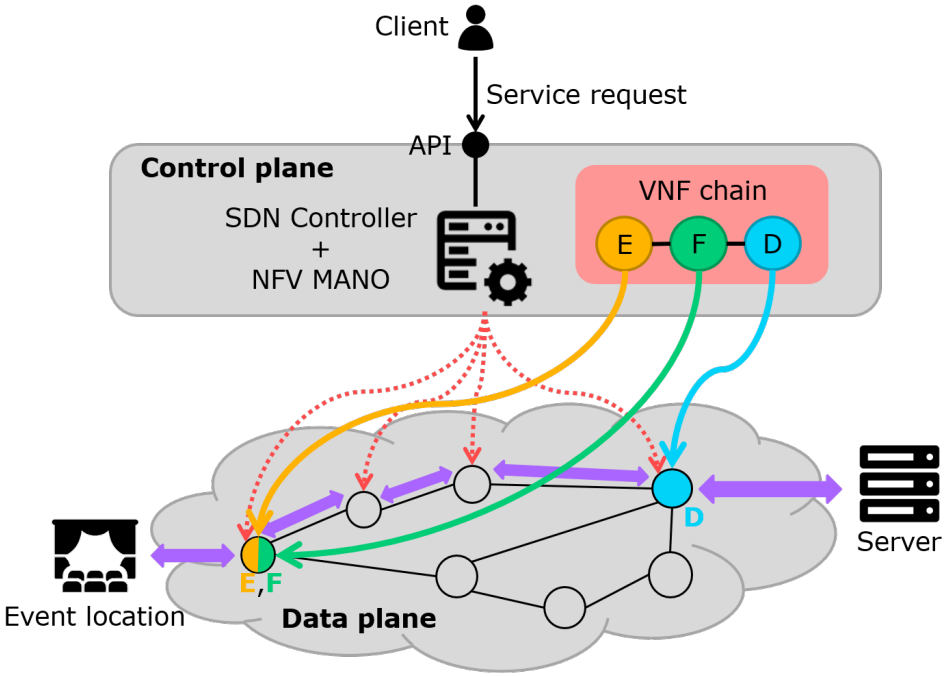


Figure 4.1 – Illustration of a Virtual Network Function Graph (VNFG) implementation in the data plane, orchestrated by the control plane

saving low capacity elements for requests with stronger requirements. We formalize this strategy as an Integer Linear Programming (ILP) problem, and propose an extensive analysis of its performances depending on requests and topology characteristics in a mono-tenant environment. Then, we propose a heuristic based on network abstraction to handle both computational complexity and multi-tenant context challenges. Section 4.1 presents different existing approaches related to the VNFGPP, introducing different optimization objectives and the heuristics used to reach them. We divided our own contribution into two parts: we detail our optimization strategy in a mono-tenant scenario in Section 4.2, and, then, we detail how to adapt this strategy to the multi-tenant case in Section 4.3. Both sections include extensive simulation results to evaluate the efficiency of our approach for different parameters. We conclude and present some perspective in Section 4.4.

4.1 Overview of existing optimization strategies

The first approaches regarding placement of virtualized components in an architecture aimed at efficiently placing VMs in a datacenter. Although this problem may seem close to the VNFGPP, the constraints faced in a closed datacenter and in a telecommunication network are different. In a datacenter, network resources are usually considered infinite: since physical distances between the different elements are very short, and network equipment is highly efficient, latency and loss are ignored most of the time, and bandwidth is considered unlimited. On the contrary, the attention is focused on the nodes of the network, the servers: how to avoid congestion, how to minimize energy consumption, and so on. For example, authors in [157] aim at mitigating the creation of hotspots (overloaded servers) by migrating some of the VMs installed on these machines to less loaded ones. In order to optimize their action,

they migrate in priority the VMs that consume a lot of resources while being the least heavy possible. This way they alleviate the hotspot and do not overload any node. [142] tackles the same problem, additionally considering that switches can become hotspots too, but this is the only network-related consideration they have. [107] tries to reduce the latency between VMs by locating VMs that communicate frequently close to each other. They do not evaluate the impact on the consumed bandwidth.

Generally speaking, due to the extremely low latency and loss within a datacenter, VM placement does not take into account network related QoS metrics, which are crucial when it comes to VNF chains placement in geographically extended networks. However, VM placement and VNF chain placement are complementary: the VNFGPP (and all its variations that can be found in the literature) selects a node - that may represent, for example, a datacenter - on which the VNFs are to be installed. However, since the node is abstracted by the VIM, the algorithm does not have precise information about this node, and the exact placement within the node is under the responsibility of a VM placement algorithm, such as [157], [142] or [107]. Consequently both VNFGPP and VM placement algorithms are both used by different entities to provide the final exact placement.

The emergence of the NFV concept has brought a lot of interest in the VNFGPP [72]. Although the global concept remains the same, each different contribution has focused on a different aspect of the problem. We can first divide the solutions into two distinct families: online and offline. Online solutions place the chains one by one, following the reception of the requests, while offline solutions place the full request set at once. The placement strategy must then choose one objective, *i.e.,* which aspect of he problem should be optimized. Finally, as the VNF placement problem has been proven NP-complete [101] [5] most of the contributions in this domain propose either an heuristic or an exact method to reduce the computational cost of the placement.

The choices made by the different contributions we analysed - regarding family, optimization, computational time reduction - are summarized in Table 4.1. It can be noted that these choices are not always independent: the optimization choice often drives the heuristic. The different entries of the table are explained into details in the rest of the section, along with some of the references.

4.1.1 Family of problems

We distinguish two families of placement strategies: online and offline.

Online strategies consist in placing each service as soon as the related request arrives, which corresponds to the real situation faced by a network operator. The waiting time of the request is thus minimized. However, as future request details are unknown, and past requests already instantiated in the network are usually never modified, the placement is always sub-optimal. Hence, future requests risk to be rejected, due to inadequate placement choices for earlier services.

Offline solutions, on the other hand, suppose that the whole set of requests is known, and services can be placed all at once. They provide better results than online strategies, because they have more data to perform their optimization. However, they all have a major drawback:

Table 4.1 – VNF chain placement contributions

Family	Online	[143] [51] [130] [128] [105] [34]
	Offline	[4] [33] [94] [14] [80] [101] [131] [124] [23] [110] [81] [14] [10]
Objective	VNF consolidation	[33] [145] [101] [124] [147] [98] [40] [81]
	Links metrics	[80] [51]
	Costs minimization	[14] [94] [128]
	Problem-specific	[23] [131] [27] [9] [110]
	Spare rare resources	[34] [143] [14] [17]
	Joint	[105] [130]
Computation time reduction method	VNFs first	[33] [94] [147] [128] [34] [162] [94] [140]
	Paths first	[51] [9]
	ILP-specific methods	[80] [98] [100] [10]
	Problem-specific	[5] [145] [14]

they require a method to obtain the set of requests, and few papers explicit how they manage to do it. The few ones who provide an explanation rely on batch processes: requests are stored during a given time (*e.g.,* one hour) and, then, served all at once [100] [162]. This system suffers two weaknesses. First, the storing process induces an additional delay (as service establishment is much less sensitive to delay than the service itself the impact is actually reduced). Second, since batches do not gather all the requests submitted to the network the placement remains suboptimal, it is a mitigation process. Authors of [100] and [162] do not compare the performances of different batch sizes to support their choice.

Authors of [147] design their algorithm so that it can adapt to both situations. They first design a classical ILP for the offline situation, and then propose an heuritic than can easily be adapted to handle requests one at a time. In order to reduce the blocking probability, the placement is assisted by a forecast mechanism based on Fourier-Series. At regular time intervals (called maintenance times), the forecast algorithm is executed, and preallocates some VNFs according to the expected request arrival.

We estimate that the additional delay in the establishment of a service induced by the offline placement strategy, due to the batching process, goes against the dynamicity promised by NFV. Moreover, the lack of clear analysis regarding the benefits of batching the requests, together with the additional computing complexity it incurs, encourages us to consider an online approach. To be valid, such approach must focus on optimizing the placement for maximizing to probability to accept unknown future requests. We however consider offline strategies useful in the context of re-optimization: they could be applied over a network with already installed chains of VNFs (installed through an online process), in order to optimize the overall placement.

4.1.2 Optimization options

The VNF chain placement problem can be considered from several points of view. Being either online or offline, each contribution focus on a specific aspect of this problem, and a particular optimization objective.

Many contributions aim at consolidating the VNF usage. This strategy consists in serving multiple Network Services with the same VNFs, in order to use VNFs at full capacity and waste a minimum amount of resources. This objective is usually a tradeoff between node resources and network resources. If few VNFs are instantiated the consolidation is very strong as each VNF will serve many flows, however the flows are likely to make longer detours to reach those rare VNFs. This impacts bandwidth consumption and latency. As opposed, offering many VNFs would increase the chances for the flow to find a suitable VNF around the optimal path between its end points, but the average load of each VNF will be lower, and resources could be wasted. In [81], authors also consider the consolidation of link usage. The reason is that the energy cost of a link can be divided into two parts: a fix cost due to the link being on, and a variable cost proportional to the bandwidth transmitted over that link. The fix cost can be saved if the link is not used at all, and switched off. Through minimizing the amount of VNFs in the network, the final objective of consolidation is either to reduce the costs, the energy consumption or both. This objective is entirely valid when it comes to PNFs: those functions are specialized to do a specific task, and have a fix amount of resources. Consequently, any unused resource cannot be reallocated to another task and is wasted. On the other hand, VNFs can scale with the workload, as detailed in Section 2.3.5. Consequently, unused resources can be released, and potentially used for another task. Although consolidation would still save resources, as scaling does not allow to exactly fit the load due to the granular nature of the VMs hosting the VNFs, the impact is likely to be reduced. This aspect is not analysed in the different contributions listed here, although in [81] authors consider VNFs small enough to be viewed as single levels of scaling. Besides allowing VNF sharing, authors of [10] tackles two other interesting aspects of the placement problem: anti-affinity rules and partial orders. Anti-affinity rules stipulates that two VNFs (resp. virtual links) must not be located in the same compute node (resp. must not share the same links). Affinity has been designed mainly to allow resiliency enforcement. In this contributions, authors only consider VNF affinity, which is a first step for resiliency. Partial order suggests that flow packets have to go through a given set of VNFs, but that the crossing order may or may not be important. When order constraint is used only when needed, the overall placement quality increases.

As we mentioned, consolidating VNFs creates a tradeoff between node and edge resource consumption, and most of the contributions aiming at VNF consolidation also consider bandwidth consumption. However, if VNFs consolidation is not part of the objective, then the problem may focus on optimizing link-related metrics exclusively. [80], for example, aims at minimizing the total bandwidth used by the services. Authors of [51], on the other hand, try to minimize the delay of each service. This last approach is not common, as delay is usually seen as a constraint, and not as an objective. Indeed, some services do not require reduced delay, and providing them with low latency connections may induced additional costs.

Both VNF consolidation and link usage optimization ultimately aim at reduce the cost of

the service (eventually through energy savings). While those papers reduce the cost by limiting the resource waste, other pursue the same objective by trying to minimize directly the installation cost. They consider that resource prices may vary depending on which node or link is considered. Their objective is then to find the best tradeoff between the amount of resources used to offer a service and the price of these resources, as reaching resources that exhibit the best tariffs may induce additional travel or reduced consolidation, hence additional link or node resource consumption.

Besides those classical objectives, some other contributions tackled very specific versions of the problem. [23] and [131] focus on security VNFs (e.g., firewall, deep packet inspection ...) required not by the client, but by the network owner itself, either to protect its own network or to provide basic security to its clients. Consequently, those VNFs can be placed anywhere in the network, because they are not related to a specific flow. Their objective it to use a minimum of VNFs to inspect all the packets. [23] first places a VNF on the node with highest centrality, and repeats the process until all flows cross a VNF. [131] has a similar approach, except that it first places the VNF on nodes that are crossed by a maximum number of flows. [9] focuses on multicast, each packet having to go through the same VNF chain, with the same source, but multiple destination. It ends up with the same dilemma as consolidation papers, opposing node and link resources, and uses a tree to solve it. Authors in [27] propose a load balancer system to share the workload between VNFs. Authors of [110] focus especially on the cohabitation of PNFs and VNFs. In their model, a Network Service does not consume directly CPU or RAM to fulfil its needs, but units of service instead. On a given node, these units of service can either be obtained by consuming generic CPU and RAM resources (this corresponds to the installation of a VNF), or by consuming directly units of this specific service provided by the node (which means that a PNF is installed here). Although this type of consideration for hybrid deployment is uncommon in the literature, the problem of cohabitation between traditional PNFs and new VNFs is a key challenge for NFV deployment.

Some contributions pursue an objective similar to ours, sparing resources where they are scarce and using them where they are abundant, but using different heuristics. [34] tackles an issue that is very close to VNF chain placement Virtual Network Embedding (VNE), and can be seen as an early version of the VNF placement problem. This problem consists in mapping a requested abstract network over a substrate network, taking into consideration both node and link resources. To accelerate the problem resolution, authors separate node and link placement. [143] places the VNFs as close as possible to the source as long as the datacenter resources are below an evolutive threshold. They do not consider that a VNF chain may have a destination, and their solution tends to over-utilize border datacenters as long as they are above the threshold. [14] aims at the same objective, but in the RAN, where delay constraints are very strong. Authors in [17] consider a much more abstract approach. They use a 2-tier architecture, where clouds are labelled either as edge (small) or public (large). Then, all the similar clouds are merged together, and the VNFs are placed in one of the two abstractions. This approach cannot be applied to large networks, where clouds cannot be considered as collocated.

Taking a step behind, some contributions propose to merge the chain creation with the chain placement, which can be referred to as the joint optimization problem. In MANO such algorithm

is not straightforward to implement, as chain creation is handled by the OSS/BSS (or, more precisely, the slice manager, if this architecture is implemented) while the chain placement is performed by the NFVO, and those two entities are not designed to work together regarding resource optimization. Taking into consideration all the possible chains that offer a service leads to intractable complexity, so [105] ends up splitting the problem: the algorithm first selects the chain that induces the lowest data rate, and, then, places it into the network. Similarly, in their heuristic of the joint optimization problem authors in [130] separate the decomposition and the placement. They associate a cost to each decomposition computed depending on the characteristics of the chain. The algorithm chooses the chain with the lowest cost. In the end for both papers the problem is not actually a joint optimization when it comes to the heuristic, which has a positive side effect: it is easier to implement in MANO.

4.1.3 Reduction of computational time

Due to the NP-completeness of this problem [101] [5], most of the contributions related to the VNF chain placement probem propose a heuristic adapted to their specific situations. We detail some of the most recent contributions in this section.

A first strategy consists in prioritizing the VNF placement over the path placement. Authors of [94] base their approach on a simplified model which separates the different steps : first they determine the instances location, second they allocate the request, third they trace the routes. A first set of solutions is generated through a fast greedy process. Then, they apply a variant of the simulated annealing technique to perform an optimization. They create neighbours to each solution, found by randomly modifying current allocations. To increase the convergence, the process does not rely on pure randomness: the instances are removed by pack and the relocation takes into account the impact on the objectives (weighted randomness). In [147], authors use an affiliation-aware VNF placement: VNF chains are merged together based on their source-destination tuple. This provides a new chain that includes all the VNFs requested by the merged chains. Identical VNFs can then be merged together. This strategy efficiently enforces their node consolidation objective, and reduces the computational time. In [128], the resolution is separated into three steps. First, for each VNF a set of nodes able to support it is computed (candidate nodes). Then, VNFs are sorted based on their number of candidate nodes. Finally, they are placed, starting by the ones having the less candidate nodes. More radically, [140] ignores network constraints.

On the contrary, [9] places the path first, and, then, instantiates the VNFs over it. Similarly, in [4] authors perform a two step optimization using ILP hierarchical objectives. The path optimization objective is prioritized, and the VNF placement comes second. On an alternative version of the problem, [51] considers that the VNFs are already placed, and represent constraints. They focus on the path placement.

When the proposed ILP is compatible, the reduction of computational time can be obtained through ILP specific methods. For example, [80] [98] and [100] are based on techniques related to linear programming to speed up the resolution: [80] blends all the metrics into one (bandwidth), and applies column generation to strongly reduce the amount of variables of the problem. [98] applies simulated annealing to obtain the same result. In an extension of their

work that also considers link consolidation [81], the authors of [80] refine their algorithm. The heuristic first splits the problem into three parts: a routing module, that computes the best route for each request at a time using a weighted Dijkstra that consolidates link usage, a service chain placement module that places all the VNFs at once over the computed paths, and an energy saving module. This last module calls the two others, deleting the least used link at each iteration, hence forcing the placement to let it free. Authors further accelerate the chain placement through column generation technique. [100] iterates over the problem by strengthening/loosening the constraints in order to manipulate the solution space. In [10], authors perform a four steps heuristic. First, they solve the Linear Programming (LP) relaxation of their ILP. Second, they round the path variables to obtain the likely optimal paths for their placement. Third, they optimally place the VNFs over those paths. Finally, they perform an online placement for the requests that couldn't be placed using the first three steps.

Finally, some other contributions rely on strategies that do not fall into any of the categories listed above. In these cases, authors often tackle a specific version of the placement problem, and exploit this specificity to reduce computational time. [5] considers that each demand requires a single network function (or that all VNFs of the chain can be merged into one), thus, strongly reducing the amount of variables. [145] uses a tree search to find the best placement. Because the construction of the tree is intractable they rely on a Monte Carlo search tree to explore only the more promising nodes. [14] perform a classical placement, but in the RAN, where delay constraints are very strong. Leveraging on those constraints, the authors design an heuristic that defines, for each VNF of the chain, the reachable area where it can be placed. Because delays are strong those areas are tight, which allows them to scale. To spare scarce resources they suppose that large datacenters are far from the antennas, and force the VNFs to be placed in the farthest possible reachable area. Their approach is efficient, but only because the RAN is strongly constrained, else the areas defined by delay boundaries would be too large to be useful, eventually encompassing the whole network.

4.1.4 The multi-tenancy scenario

All the contributions listed in this section implicitly make the hypothesis that their algorithms have access to exact information related to the infrastructure (topology, resources) to perform their placement. This assumption is probably correct in a mono-tenant architecture, as defined in Section 2.3.4, but not in the multi-tenant scenario. We found no contribution tackling this case.

4.2 VNFG placement in a mono-tenant architecture

In this chapter we propose an online algorithm to solve the VNFGPP problem. The objective of our algorithm is to place a maximum number of requests over time. To do so, for each placement we try to spare resources where they are scarce. This strategy tends to maintain resources all across the network, which enables us to serve strongly constrained requests that can only reach a small subset of the network nodes. In order to address large topologies we develop a heuristic based on network division and abstraction. We leverage this heuristic to

tackle an essential aspect that has been ignored by contributions cited above: the VNFGPP in a multi-tenant environment (see Section 2.3.4). Finally, we perform extensive evaluations to study the behaviour of our algorithms for various types of requests and topologies.

4.2.1 Problem description

Upon a client's request, the NFVO must place a NS, decomposed in a VNFG, in the network at runtime. This decomposition is done by the OSS/BSS, before being sent to the NFVO. In this contribution the graph is supposed to be a chain, which is consistent with the assumption made by most of the works presented in Section 4.1. Tackling graphs would not affect our general scenario, and would require some minor modifications in our model. The chain is specified by its physical entry and exit points (that can eventually be identical), a succession of VNFs and two types of constraints: on nodes capacities and on links capacities. For constraints on nodes, we consider both Central Processing Unit (CPU) and storage (RAM). Regarding links, we consider two parameters: an additive delay and a min-max bandwidth. Some VNFs (e.g. firewalls, encoders and decoders) may change the QoS requirements on the link (e.g. bandwidth required between VNFs), so the constraints on the link may vary along the VNFG. The required latency may be different between VNFs, as specific timers may exist to complete some local sub-functions. Hence, bandwidth and delay requirements are considered from end-to-end (for the whole service) and locally (between VNFs). This section considers a mono-tenant architecture where the NFVO has a full view over the infrastructure topology to place the VNFs. The NFVO places requests one by one as they arrive without any prior knowledge of future requests.

In this context, our optimization focuses on maximizing the acceptation of new VNFGs. Therefore, we reserve resources in priority on links and nodes where they are abundant, saving them where they are scarce for further requests with potentially stronger requirements. Note that this strategy tends to preserve free resources on most of the nodes of the network. This can be an advantage in the NFV context, since it allows the VNFs to scale up/out with the workload using those resources.

4.2.2 ILP optimization model

In order to find a method to solve the VNFGPP we first note that this problem displays similarities with classical routing problems, so we consider methods that are known to work for these problems. We first tried to adapt the SAMCRA algorithm [152] that we already successfully used in the PCE in the previous Chapter. However, the placement of the VNFs prevented us from implementing an efficient pruning mechanism of the solution space, which is a mandatory element to solve the problem within a reasonable time.

We then considered another option: to use an ILP. It consists in formalizing the problem as a linear objective associated with a set of linear constraints. This method is already used to solve many variations of classical routing problems [127], as well as most of the VNFGPP variations listed in Section 4.1. To formalize the VNFGPP as an ILP we use the notations summarized in Table 4.2. The constraints used in this model are equivalent to those already

Table 4.2 – ILP notations

Name	Description
V	Set of VNFs to be placed
N	Set of nodes of the network
$Succ(v \in V)$	Outgoing neighboring VNF of $v \in V$
$Succ(i \in N)$	Outgoing neighboring nodes of $i \in N$
$B^{v,w}$	Bandwidth requested to connect $v \in V$ and $w \in Succ(v)$
$B_{i,j}$	Bandwidth available on the directed link from $i \in N$ to $j \in Succ(i)$
S^v	RAM requested by $v \in V$
S_i	RAM available at $i \in N$
C^v	CPU requested by $v \in V$
C_i	CPU available at $i \in N$
$D^{v,w}$	Maximum delay between $v \in V$ and $w \in Succ(v)$
$D_{i,j}$	Delay of the directed link from $i \in N$ to $j \in Succ(i)$
T^v	Processing delay of $v \in V$
Δ	Maximum VNFC end-to-end delay
$P(X)$	Price of the resource X

used in the literature, with minor variations coming from the exact definition of the problem. The key difference resides in the objective.

4.2.2.1 Variables

In our situation network conditions and VNFG requirements are known. We are looking for two sets of elements: the set of nodes in which we are going to install our VNFs and the set of links we will use to connect those nodes together. As a VNF is either installed on a given node or not, and a link is either used to connect to VNFs or not, the variables are binary. We define link-related and node-related variables respectively as x (4.1) and y (4.2):

$$x_{i,j}^{v,w} = \begin{cases} 1 & \text{if the traffic sent from } v \in V \text{ to } w \in Succ(v) \\ & \text{uses the link between } i \in N \text{ and } j \in Succ(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

$$y_i^v = \begin{cases} 1 & \text{if } v \in V \text{ is hosted } i \in N \\ 0 & \text{0 otherwise} \end{cases} \tag{4.2}$$

4.2.2.2 Objectives

In this problem we consider three consumable resources: bandwidth, RAM, and CPU. A given placement has a global price that takes into account the consumption of each of these three resources. The sub-price related to a given resource is defined by the amount of this resource used on each element involved in the placement (node or link), and multiplied by

the price of this resource on this specific element. This translates into (4.3), (4.4) and (4.5) respectively for bandwidth, RAM, and CPU:

$$\sum_{(v,w)\in V}\sum_{(i,j)\in N} x_{i,j}^{v,w} B^{v,w} P(B_{i,j}) \tag{4.3}$$

$$\sum_{i}\sum_{v} y_i^v S^v P(S_i) \tag{4.4}$$

$$\sum_{i}\sum_{v} y_i^v C^v * P(C_i) \tag{4.5}$$

We want to place in priority VNFs on nodes/links with abundant resources and to preserve nodes/links with scarce resources for future and more demanding requests. To make it costly to install a VNF/path on a node/link with few remaining resources, we associate a price corresponding to the inverse of the available resources:

$$P(B_{i,j}) = \frac{1}{B_{i,j}}; P(S_i) = \frac{1}{S_i}; P(C_i) = \frac{1}{C_i} \tag{4.6}$$

As we consider an online approach, it is important to highlight that the "available resources" of the n^{th} request are the remaining resources left after the placement of the $(n-1)^{th}$ request.

The optimization of a problem that displays multiple, often contradictory, objectives is a difficult task. An exact solution of such problem consists in finding the Pareto front of the solution space, that gathers all the solutions that are non dominated by other solutions. Finding a good approximation of the Pareto front is the subject of many researches, such as [88] [45] [84] [156].

However in our case having the Pareto front is not really useful, as we still would have to chose one solution among all the ones proposed. Moreover, computing several points of the front would induce additional computing delays, which we would like to avoid. Another approach consists in merging all the objectives into a weighted sum. This process is called scalarization. Here, we perform a scalarization of our costs with custom coefficients α, β, and γ and minimize the result:

$$\min \alpha \sum_{(v,w)\in V}\sum_{(i,j)\in N} x_{i,j}^{v,w}\frac{B^{v,w}}{B_{i,j}} + \beta\sum_{i}\sum_{v} y_i^v \frac{S^v}{S_i} + \gamma\sum_{i}\sum_{v} y_i^v \frac{C^v}{C_i} \tag{4.7}$$

4.2.2.3 Constraints

The VNFG may specify given entry and/or exit to the chain, for example in order to connect two specific locations. If so we have to force the problem to include those locations at the beginning/end of the path. To do so the entry and exit nodes of the chain are represented by two fictive VNFs ($entryVNF$ and $exitVNF$) without any resource needs. We set the entry and exit point with Equations (4.8) and (4.9):

$$y_{entryNode}^{entryVNF} = 1 \tag{4.8}$$

$$y_{exitNode}^{exitVNF} = 1 \tag{4.9}$$

Path continuity is enforced by (4.10) while (4.11) ensures that there are no loops along the path between two consecutive VNFs :

$$\sum_{j \in N} x_{j,i}^{v,w} - \sum_{j \in N} x_{i,j}^{v,w} + y_i^v - y_i^w = 0, \forall i \in N,\ \forall (v,w) \in V \tag{4.10}$$

$$\sum_{j \in N} x_{j,i}^{v,w} + y_i^v \leq 1, \forall i \in N,\ \forall (v,w) \in V \tag{4.11}$$

Each VNF must be placed exactly once:

$$\sum_{i \in N} y_i^v = 1, \forall v \in V \tag{4.12}$$

Embedding must respect VNF constraints regarding link bandwidth capacity (4.13) and delay between each VNF (4.14):

$$\sum_{(v,w) \in V} x_{i,j}^{v,w} B^{v,w} \leq B_{i,j}, \forall (i,j) \in N \tag{4.13}$$

$$\sum_{(i,j) \in N} x_{i,j}^{v,w} D_{i,j} \leq D^{v,w}, \forall (v,w) \in V \tag{4.14}$$

Node CPU (4.15) and RAM (4.16) resources must not be exceeded:

$$\sum_{v \in V} y_i^v C^v \leq C_i, \forall i \in N \tag{4.15}$$

$$\sum_{v \in V} y_i^v S^v \leq S_i, \forall i \in N \tag{4.16}$$

VNFC embedding must respect the maximum end-to-end delay acceptable for the whole NS:

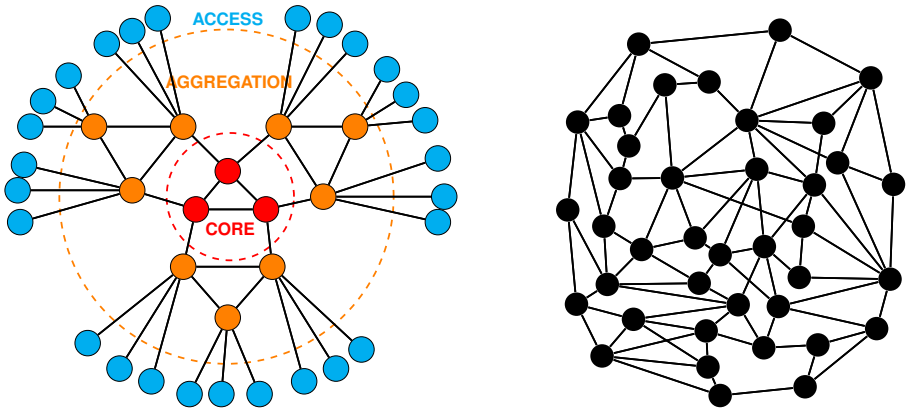
$$\sum_{(v,w) \in V} \sum_{(i,j) \in N} x_{i,j}^{v,w} D_{i,j} + \sum_{v \in V} \sum_{i \in N} y_i^v T^v \leq \Delta \tag{4.17}$$

We now have to evaluate the efficiency of our algorithm against different parameters in order to obtain a representative performance evaluation under different situations and scenario.

4.2.3 Performance analysis

We evaluate the efficiency of our algorithms using the Gurobi solver [74] on a 12 logical cores Intel Xeon E5-2630.

The parameters we can modify to analyse the performances and robustness of our solution fall into two main categories: topology-related parameters, that refer to the structure of the topology, and the resource distribution pattern within this topology, and network service-related parameters, that embrace the various needs a network service can express.



(a) Edge39 (edge topology) (b) Giul39 (flat topology)

Figure 4.2 – Topologies

4.2.3.1 Topology-related parameters

The VNFG placement is primarily constrained by the total amount of resources made available by the network: if the needs expressed by the VNFs exceed the total offer then the service cannot be embedded. However, when we consider strong latency constraints, the localisation of the resources across the topology may also influence the success of the placement. Indeed, if reaching available resources implies to violate the latency budget then the service cannot be implemented neither. We measure the impact of two parameters that may influence resource reachability: the structure of the topology and the resource distribution pattern.

4.2.3.1.1 Topology structure We consider two topology structures: flat and edge. An example of each architecture is presented in Figure 4.2.

Flat topologies are representative of the backbone of large scale networks. They contain only one level of nodes, as all nodes have the same importance, although some are more central than others.

On the other hand, edge topologies represent local or regional network organizations. They are usually modelled using three hierarchical levels of node [4] [17] [41], and are referred to as 3-tier architectures. **Core** nodes are central nodes, that connect the different parts of the network, thus, having the highest centrality. **Aggregation** nodes establish a connection between core nodes and **access** nodes, which are the closest to the end user, but display the lowest centrality, being located at the edge of the network. Note that some publications are also based on a simplified 2-tier architecture, without aggregation layer [17].

To conduct our tests we choose seven topologies, whose characteristics are depicted in Table 4.3. Atlanta, Brain, Cost, Germany50, and Giul39 are available on SDNLib [1] and represent flat topologies, while Edge39 (depicted in Figure 4.2a) and Edge51 are custom topologies that aim to represent a typical edge topology. A graphical representation of those topology is available in Section 7.3. They are similar to the ones used in [143] and [4].

1. http://sndlib.zib.de

Table 4.3 – Topologies characteristics

Topology	Type	#Nodes	#Links	Diameter
Atlanta	Flat	15	22	5
Brain	Edge	161	332	5
Cost	Flat	37	114	8
Edge39	Edge	39	90	6
Edge51	Edge	51	126	6
Germany50	Flat	50	176	9
Giul39	Flat	39	172	6

4.2.3.1.2 Resource distribution When it comes to the placement of VNFs with very strong latency requirement, the location of the resources may be an important factor. In edge topologies, datacenters located on central nodes will typically have much more capacity than those located on access nodes, consequently in our model central nodes will have more resources than edge nodes. In flat topologies the resource distribution will depend on the physical distribution of the datacenters across the area, which can be more random.

We study the impact of resource distribution with three scenarios: low, medium or high concentration. As an example, the resources explicitly set for each scenario for Edge39 topology are represented in red in Table 4.4. Other figures are computed based on explicit ones. “Total” represent the total amount of resources attributed to a category of node, while “Node” indicates the resources of one individual node in each category.

In the *Low concentration* (L) scenario, the nodes and links of all topologies have exactly the same resources: the nodes have 80 CPU units and 100 storage units and the links have 1000 bandwidth units. The total amount of resources in the network is the same for all scenarios. Although this is unlikely to happen in real life, it provides a base to evaluate the importance of the distribution on the performances of the algorithm. The other scenarios focus on the topologies with comparable number of nodes and links: Edge39, Edge51, Giul39 and Germany50. The total amount of resources in the network is the same as in the first scenario but the node resource distribution changes. Nodes are categorized as access, aggregation, and core nodes. This classification is straightforward for Edge39 and Edge51. We classify Giul39 nodes (resp. Germany50) by selecting the same amount of nodes in each category as with Edge39 (resp. Edge51). The nodes with highest betweenness correspond to core nodes as large datacenters are likely to be placed at strategic location in the network. In the *Medium concentration* (M) scenario, each node category has 33% of the network total amount of resources whereas in the *High concentration* (H) scenario core, aggregation, and access nodes detain respectively 60%, 30% and 10% of the total amount of resources.

The percentages exposed in Table 4.4 represent the resource distributions we want to apply in theory to our network, for the different concentration scenarios. They are computed based on the total amount of resources of the Low Concentration scenario. However, as resources at node level can only be integer, the actual resources used for the experiments (CPU and RAM

Table 4.4 – Edge39 resource distribution details

Distribution		Low			Medium			High		
		%	CPU	RAM	%	CPU	RAM	%	CPU	RAM
Core	Total	7.69%	240	300	**33%**	1029	1287	**60%**	1872	2340
(3 nodes)	Node	2.56%	**80**	**100**	11%	343	429	20%	624	780
Aggreg	Total	23.04%	720	900	**33%**	1035	1296	**30%**	936	1170
(9 nodes)	Node	2.56%	**80**	**100**	3.67%	115	144	3.33%	104	130
Access	Total	69.12%	2160	2700	**33%**	1080	1323	**10%**	324	405
(27 nodes)	Node	2.56%	**80**	**100**	1.22%	40	49	0.37%	12	15

columns) may differ slightly from the theoretical resource percentage. For the same reason, the total amount of resources for each scenario is not strictly identical.

The variation of link resource concentration is not analyzed in these experiments, as a multiplication of the experimental parameters may be detrimental to the analyze of the impact of those parameters. Here we focus on the node resources, which distinguish the VNFGPP from traditional routing problems.

4.2.3.2 Network service-related parameters

4.2.3.2.1 VNF delay tolerance The exact end-to-end latency constraint of the Network Service and the latency constraints between consecutive VNFs are likely to influence the global placement of the chains and the exhaustion of edge resources. Since we are not sure of the delays that will be requested by the NS, or the proportion of strongly delay-constrained Network Service in the future, we tested a large range of delay bounds in order to analyze the impact of these constraints on the placement.

Each VNF has an inner processing delay that is randomly chosen between 0 and 100 units (100 units being denoted $maxProcessingDelay$)

We set each link delay to 100 units. Since the studied topologies have different diameters, using the same bounds for each topology may provide results that would be difficult to analyse: a given bound could have no effect on Atlanta but may strongly impact Germany50. To mitigate this issue we determine the delay bounds by introducing a variable that we call delay factor. A delay bound takes into account the topology diameter in order to obtain comparable results between topologies. The delay factor takes the values 50, 100, 150, 200, 300, 400, and 800. For each network service, we randomly select the end-to-end latency constraint Δ and the latencies between VNFs $D^{v,w}$ using respectively (4.18) and (4.19).

$$scaleDelay = \frac{delayFactor * networkDiameter}{2}$$

$$\frac{scaleDelay}{2} \leq \Delta \leq$$
$$numberOfVnfs * (4 * scaleDelay + maxProcessingDelay) \quad (4.18)$$

$$0 \leq D^{v,w} \leq scaleDelay \quad (4.19)$$

The values of the delay factor along with equations (4.18) and (4.19) allow to cover a wide range of cases in terms of maximal number of hops in the topology. For example, a delay factor of 50 corresponds to an average delay between VNFs of one hop, while for a delay factor of 400, it is equivalent to the diameter of the network. Examples of detailed delays and hop counts for topologies Cost and Edge51 are provided in Annex 7.1.

4.2.3.2.2 NS lifespans Network Service lifespans represent the time a Network Service is present in the network. For convenience, the measure of time in our experiment is based on the reception of requests. For example, the first Network Service may indicate a lifespan of 200 requests. When the 200th request is received, the Network Service expires and the reserved resources are released. Similarly to delay tolerance, lifespan values depend on the topology. Indeed, in our model, the resources of the topology are directly proportional to its size. For the same lifespan a small topology such as Atlanta could be saturated while a large one like Brain would have barely dent its resources. Consequently, we need a normalized parameter to measure the influence of lifespan.

For each topology we define the inflexion pointas the number of requests from which the acceptance rates of both our ILP and the baseline algorithm (see 4.2.3.3) become very low, marking the beginning of the saturation of the network, plus 100 requests in order to be sure to reach saturation at this point. We analyse eleven lifespan values for each topology (from 10% to 100% of the inflexion point, plus the infinite lifespan). For each set of parameters we determine the value of the inflexion pointexperimentally. This method is sufficient since the purpose of the inflexion pointis only to allow a good distribution of the lifespan measurements, from low occupation of the network to saturation. A graphical example of the inflexion pointlocation is presented in Figure 4.3.

4.2.3.2.3 Other parameters The value of the following parameters are chosen randomly in consistency with [145], [4], [51], [27] and [124]. Chains comprise between 2 and 5 VNFs. Any VNF consumes between 0 and 10 CPU and storage units, and the link between two consecutive VNFs requests between 1 and 10 bandwidth units. We choose arbitrarily a delay of treatment for each VNF between 0 and 100. Entry and exit points of the chain are randomly chosen in the network, and may eventually be the same. As requests are generated randomly, and potentially with strong delay constraints, some of them may be infeasible, even with the full resources of the network available. As any algorithm would fail to place those requests, they are not adapted to evaluate our model. Consequently, in order to improve the readability of the

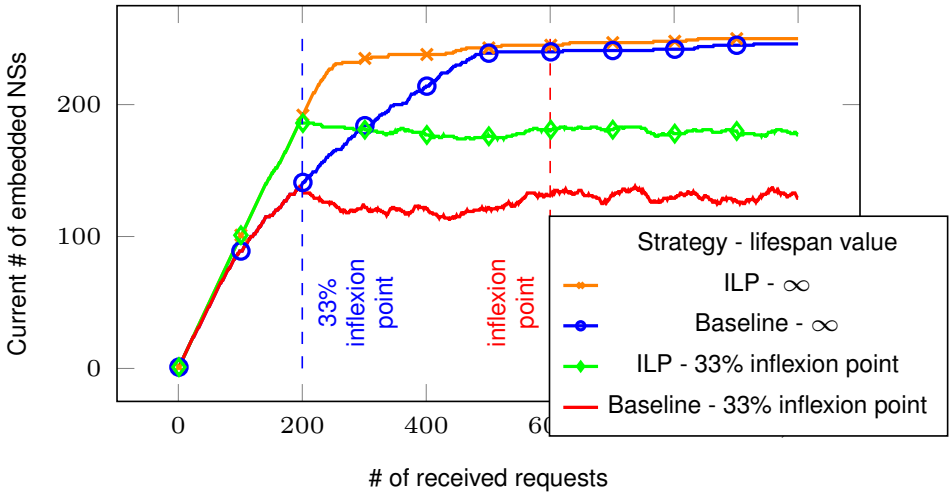


Figure 4.3 – Inflexion point example

results, each generated request is first evaluated to check whether it can be placed in the empty network, and discarded if it is not the case.

4.2.3.3 Baseline algorithm

We evaluate the performance of our ILP by comparing it with a baseline solution that only considers the bandwidth optimisation.

Many computationally optimized solution exist for this problem, some of them especially tailored to increase the computational efficiency of a bandwidth-optimized placement. However, here we do not compare the runtime between the baseline algorithm and our model. Consequently, for convenience we took our ILP with the following coefficients in (4.7): $\alpha = 1$, $\beta = 0$ and $\gamma = 0$. The resulting objective function is:

$$\min \sum_{(v,w)\in V} \sum_{(i,j)\in N} x_{i,j}^{v,w} \frac{B^{v,w}}{B_{i,j}}$$

Another advantage to use a modified version of our model to make the comparisons is that the constraints on the VNF placement are identical. this is important, as those constraints define the exact version of the VNFGPP we are trying to solve.

Measures in this section were obtained as follows: for a given algorithm and set of parameters (topology, lifespan, delay factor) the performance is the averaged number of embedded NSs computed over 2.000 requests after the inflexion point, where the occupancy of the network is stabilized (behaviour at regime).

4.2.3.4 determination of the objective function coefficients

As detailed in Section 4.2.2.2, we scalarize our different objectives into one objective function. When it comes to the experiments, the difficulty lies in the definition of the weights.

Unlike offline approaches, our algorithm solutions quality cannot be measured after one resolution of the problem, as our final objective is to embed a maximum number of requests

in the long run. Intuitively, the weights of the sum have to reflect the resource consumption rate, so that they would focus on saving resources that are consumed faster. Since we cannot predict how resources will be consumed, we first tried to design a flexible system where weights would automatically adjust themselves after each request placement. Unfortunately, we failed at finding useful correlation between optimal weights and resource consumption rates.

Consequently we opted for a static approach. For each topology we set the bandwidth weight to 1, and we vary the other weights independently from 0.1 to 10, and we measure the number of embedded requests after 1000 requests. We repeat this operation ten times for each topology.

We observed that, as long as CPU and RAM weights are not too small compared to bandwidth weight (if they are not below 1) the final results for the different weights are close, and the best combination depends on the topology. We chose the coefficients $\alpha = 1$, $\beta = 4$ and $\gamma = 7$ as they display good performances for all topologies studied.

4.2.3.5 Performances overview

Figure 4.4 provides an overview per topology of the improvement obtained with the ILP compared to the baseline in terms of number of embedded NSs. Each boxplot is composed of 77 results obtained with the 7 values of the delay parameter and the 11 values of the lifespan parameter. In the Low Concentration cases, we observe that our model offers performances similar to the baseline one. As the resources are equally distributed, there is no local scarcity. As our model is designed to optimize the placement to spare resources where they are scarce, but, encountering no such resources, it resumes to the same placement logic as the baseline algorithm, hence leading to similar results. We observe that the higher the concentration the better the improvement. The explanation is identical: the more scarce resources are present in the topology, the more our model's behaviour diverges from the baseline one, the better the placement.

In practice, the High distribution seems more likely to happen as access nodes may be composed of one cabinet, or even one computer, whereas central datacenter resources are often considered as infinite. This confirms our hypothesis: it is crucial to spare nodes with few resources so as not to reject future NSs with strong latency requirements due to a lack of nodes with resources close enough to meet them. The lack of impact of the topology type (flat or edge) can be explained by their small size: the path length from an access node to a central node is quite similar. Lastly, we can see that even for High concentration the median improvement remains under 5%, while the mean improvement lies between 5% and 10%. Top performance can reach 15% to 23%. In order to explain those disparities we will take a closer look to the best performing topology - Edge51 - in Figure 4.5.

4.2.3.6 Detailed performances analysis

Figure 4.5 provides an overview of the improvement obtained with the ILP compared to the baseline in terms of number of embedded NSs for topology Edge51, with the three levels of resource concentration, according to the lifespan of the services in the network and the delay

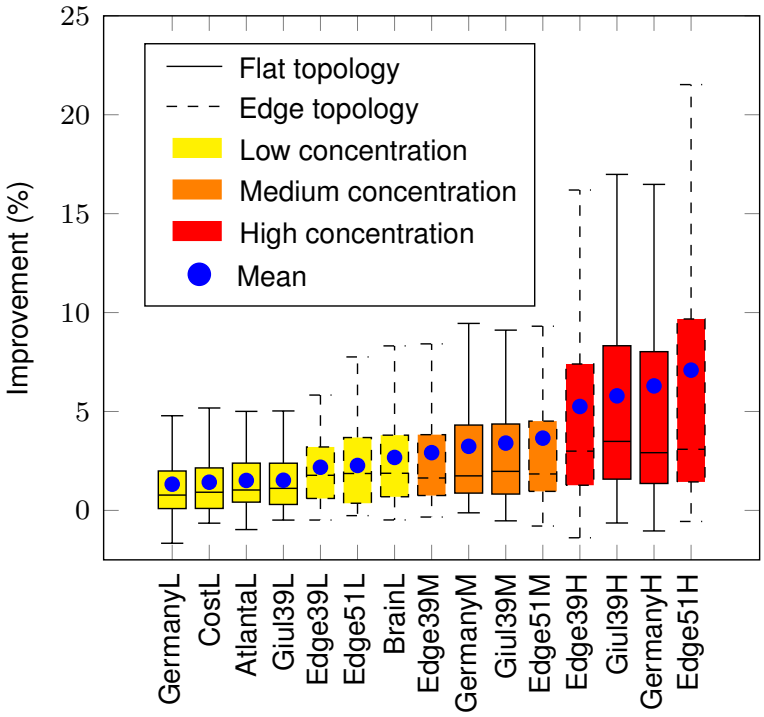


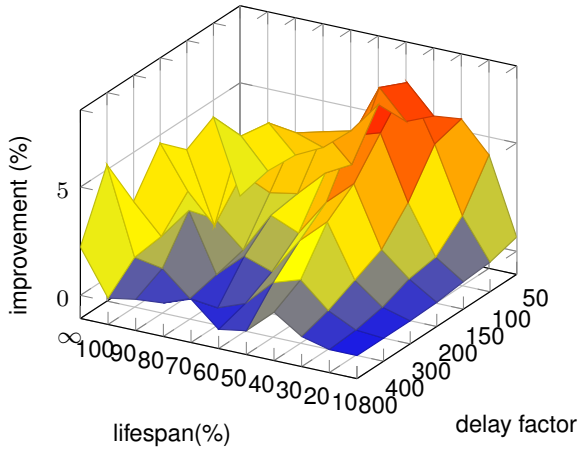
Figure 4.4 – Performance of the ILP compared to a bandwidth-only optimization strategy

factor applied to those services. This figure has the same general aspect for each topology, although scales change. Detailed results for each topology and each resource concentration are provided in Annex 7.4. The best improvement is obtained for delay factors of 50, 100, 150. Higher delay factors lead to very weak latency constraints and the VNFG can involve long detours to take resources in the core nodes when other options are exhausted, reducing the benefit of sparing resources in the access. On the contrary, requests with strong latency constraints benefit from good resource management because they could no longer be accepted in the event of resource depletion in an area.

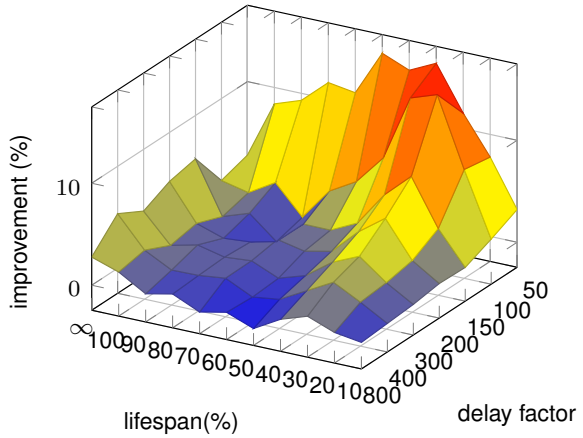
The improvement is maximized for lifespans between 30% and 40% of the number of requests compared to the inflexion point. With lower lifespans the network is underused, and a wise placement is useless since there are much more resources than needed. Higher lifespans saturate the network and only the requests that require small resources are accepted, regardless the placement.

4.3 VNFG placement in a multi-tenant architecture

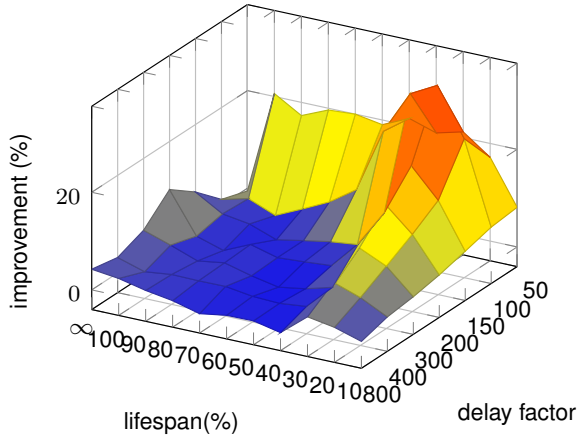
The VNFGPP version presented in Section 4.2.2 requires a detailed knowledge of the network topology and resources, which is impossible in a multi-tenant architecture. When the infrastructure and the orchestration services are owned by different actors, VIMs will be reluctant to disclose confidential network organisation details and will rather present an abstraction of the network to the NFVO. For example, this is the case with UNICA's Virtual Data Center (VDC) concept [41]. Clients that want to use UNICA as an infrastructure platform are provided with one or several VDCs, which are parts of one or several datacenters, logically centralized to



(a) Low



(b) Medium



(c) High

Figure 4.5 – Performance of ILP vs baseline in Edge51 topology

be presented as one datacenter to the client. Thus, clients obtain an abstracted network of datacenters.

Therefore, we introduce a two step computation. First, the NFVO performs the placement over abstracted topologies proposed by the VIMs, resulting in a first placement of the VNFs. Then, each concerned VIM must in turn run the algorithm to determine the final placement within its infrastructure and update its abstracted graph based on resource consumption.

If the VIM is actually a Wide Area Network (WAN), with no resources to place VNFs, it can actually be reduced to an SDN controller and fully use our solution proposed Chapter 3 to compute both the best path through the network using the PCE and, as the SDN controllers in VIMs would do, use the STEM strategy to implement their path.

4.3.1 Clustering

In a multi-tenant architecture, each VIM represents a part of the global NFVO view of the topology, called cluster. Independently, each one of them abstracts its own topology. Note that, rather than performing directly the abstraction process over its topology, a VIM may decide to artificially divide its own network into separated sub-clusters. This choice results from a tradeoff between the confidentiality of the topology and the exactitude of the information provided, enforced respectively by a small and large number of clusters. Exactitude of the information is important because approximative information may result either in missed transactions (the NFVO does not find any way to fulfill the request in the abstracted topology although one exists in the real topology) or failure to comply with SLAs (the abstracted topology advertises capacities that cannot actually be fulfilled). In this contribution the number of clusters is fixed manually by defining the maximum number of nodes in a cluster.

We divide the network into clusters using an algorithm based on the link betweenness centrality metric, that will be referred to as betweenness in the remainder of this document. For each link, betweenness is defined as the number of shortest paths this link is part of, among all shortest paths connecting each pair of nodes of the network [70]. We use this technique because it seemed appropriate to divide the edge networks that we are studying whose links betweennesses fall into clearly separated ranges of values. However other techniques may be applied depending on the topology. For example, for very large graphs, techniques based on nodes spatial localisation (geometric algorithms) such as the one presented in [49] may be more suitable than the ones based on structural properties (betweenness in our case) for complexity reasons.

The algorithm we use to divide the network works as follows. First, we manually define a maximum number of nodes in each cluster. Then, as long as the largest cluster has strictly more nodes than this limit, we recompute edge betweennesses in the largest cluster and we delete the edge with higher score. This process is represented Figure 4.6, with a maximum number of nodes set to five. Once the clustering step is over we obtain a set of separated networks, or clusters, and we can start to abstract each one of them.

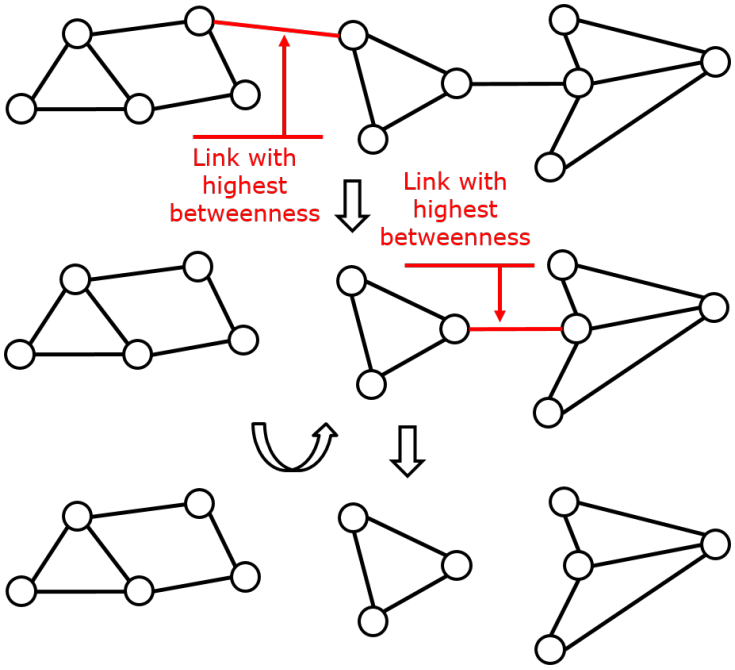


Figure 4.6 – Clustering algorithm

4.3.2 Topology abstraction

Over the years, network topology abstraction has motivated a lot of research works [151]. Those works (such as [96], [91] or [153]) mostly focus on QoS routing over several independent networks in Asynchronous Transfer Mode (ATM) Private Network-to-Network Interface (PNNI) networks [134]. PNNI does not take into account node resources, and the solutions that use it are not intended to place VNFs: they are related to traditional routing problems.

In order to be efficient, the abstraction must display a tradeoff between the accuracy of the information (to allow the NFVO to compute a correct path) and the confidentiality of internal details. The main abstractions for a network topology are the single node, the star and the full mesh [96], represented Figure 4.7. The single node approach efficiently hides internal details, but fails to provide enough information for the placement decision. The full mesh is a very common abstraction in traditional problems as it allows to keep topology details secrets while providing reliable information for routing [151]. This abstraction can even avoid any loss of information by displaying the full Pareto front between any pair of nodes, using multiple abstract links, at the cost of reduced scaling capacity [67]. However traditional problems focus only on link resources, and do not consider node ones. Problems that consider nodes are very different. When we tried to adapt SAMCRA to VNFG placement we found out that Pareto front was computationally intractable when node resources are involved, mainly due to the impossibility to reduce the search space via subpath domination. Without Pareto front available, the main advantage of the full mesh abstraction does not hold any more. Moreover, representing node resources efficiently using this abstraction is not trivial. Based on those observations, we conclude that the full mesh abstraction is not as adapted to represent node resources as it is in traditional problems. Thus, we opted for the asymmetric weighed star abstraction, as it seems intuitively adapted to represent both node and link resources.

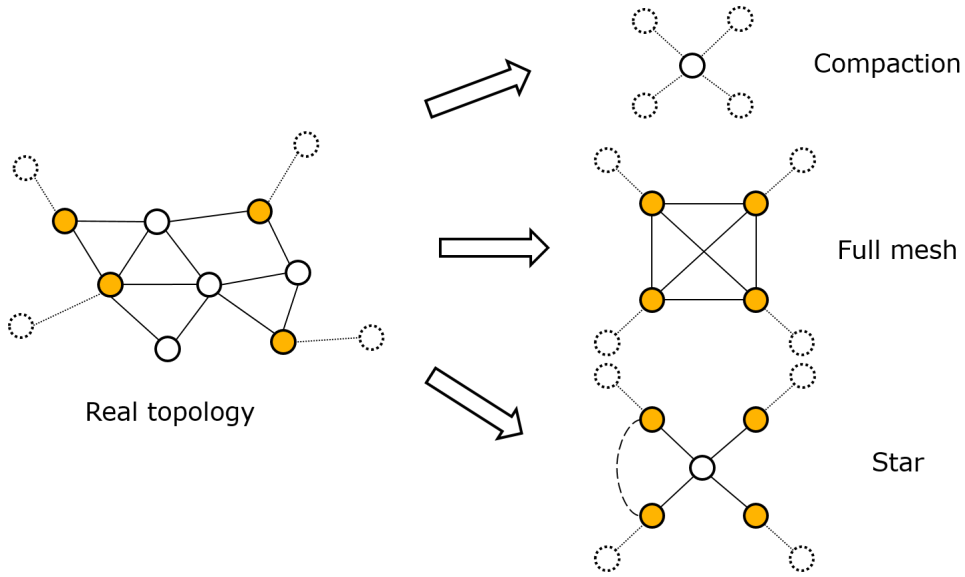
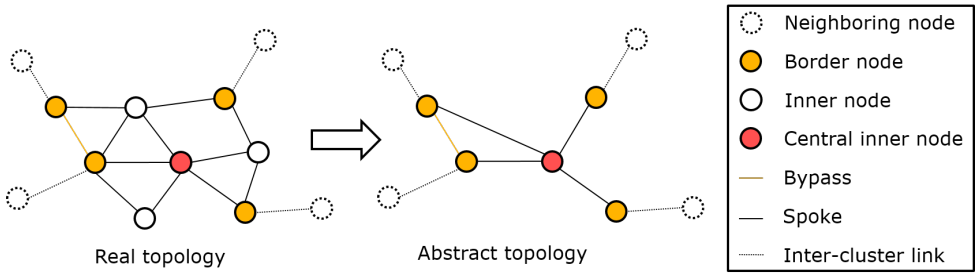


Figure 4.7 – Classical abstractions

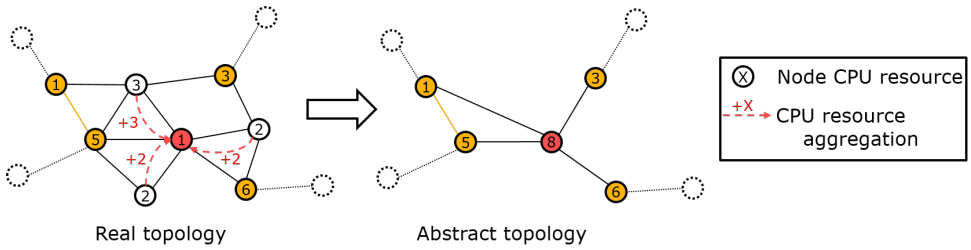
Usually star abstractions are derived from full meshes using various techniques [151]. Since we cannot compute a representative full mesh those techniques do not apply here, so we propose a new way to create the star. Each cluster is abstracted into a star topology that includes an abstracted central node (nucleus) connected via abstracted links (spokes) to the cluster border nodes, themselves connected to neighbouring clusters via real links. To define the nucleus we chose a real node of the cluster, the one with the highest betweenness. It is likely to be, in average, the easiest one to access from any other node in the cluster since it is the one that takes part to the highest number of shortest paths. We then created spokes to connect this node to all the border nodes. In addition to those minimal stars we added some bypasses. Bypasses are extra virtual links that may be added to a star in addition to spokes to reflect a specificity of the network, such as remarkable QoS capacities. In our case, we want to emphasize the fact that a path does not necessarily have to reach the center of a cluster in order to cross it. Typically, in an edge topology, the connection between two neighbouring clusters of aggregation nodes may only require to go through two border nodes of a central cluster. So we keep existing edges between border nodes as bypasses in the star to ease the transition from one cluster to another. The star transformation is illustrated in Figure 4.8a.

Once this topology abstraction is fulfilled, we have to assign resources and QoS parameters to all the elements. Border nodes and bypasses are not modified during the abstraction process, so the values of their resources are identical in the abstracted and original topologies. The central node aggregates the resources of all non-border nodes. We set the parameters of each spoke with the best possible value, determined by running multiple Dijkstra algorithms between the border node of the spoke and the nucleus focusing on one parameter at a time. Delay and bandwidth consumption are minimized, and the path bandwidth is maximized (*i.e.,* we maximize the minimal bandwidth along the path). Despite its simplicity, this aggressive approach has been proved quite effective in [92]. The node and link resource abstraction processes are illustrated respectively in Figures 4.8b and 4.8c.

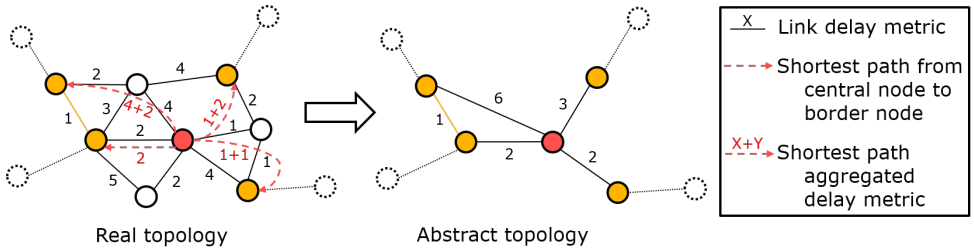
In the mono-tenant problem scenario, resource price is defined as the inverse of this re-



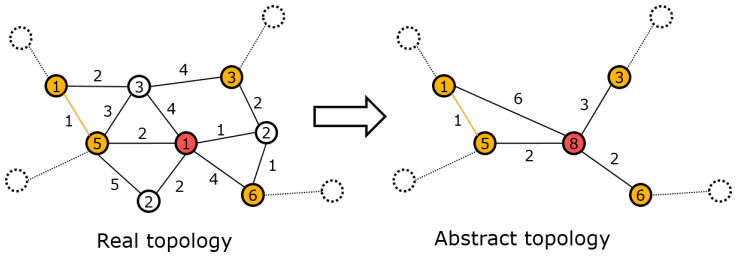
(a) Star creation



(b) Node CPU resource computation



(c) Edge delay metric computation



(d) Final result

Figure 4.8 – Abstraction steps

source, as formalised in Equation 4.6. In the abstracted topology, the nucleus aggregates the resources of all inner nodes, and will typically have more resources than any other node in the star, as in our example in Figure 4.8d. Consequently, VNFs would mainly be placed on the nucleus (and finally in the inner nodes), leaving border nodes underused. However, in our example, at least two border nodes should be more appealing than any inner ones. We mitigate this problem by redefining the price of each resource as the lower price for this resource among the nodes the nucleus represents. In the example Figure 4.8d the price of the resource of the nucleus would be $\frac{1}{3}$, as the inner node with most CPU - hence lower CPU price - has 3 units of this resource, thus, the nucleus will propose 8 units of CPU at a price of $\frac{1}{3}$, instead of $\frac{1}{8}$ in the initial ILP.

Once this process is completed the heuristic can receive placement requests.

4.3.3 Heuristic mechanism

When a placement request is received the NFVO runs the ILP over the abstracted topology, made of abstracted clusters presented by the VIMs. Once the placement is completed it is transmitted to the VIMs that perform a second placement in order to attribute real resources to the services. The combination of those second placements provides the final resource allocation. This process is illustrated in Figure 4.9.

Once the request is embedded VIMs recompute the abstracted resources, as presented in Section 4.3.2, and update the information they expose to the NFVO. The structure of the clusters is not modified, unless new nodes or links are added to the network.

4.3.4 Heuristic performance analysis

4.3.4.1 Heuristic evaluation

To evaluate the performance of our heuristic we compare it to the holistic ILP. The performance reflects the capacity of the heuristic to place a maximum number of NSs in the network.

We ran our evaluation considering a wide range of sizes for edge topologies (based on the Edge39 scheme). We generated topologies, referred to as edgeNCore, composed of N core nodes, N clusters of aggregation nodes interconnected by the core with $2 * N + 1$ aggregation nodes divided into two layers in each cluster, and $4 * N$ access nodes connected to each aggregation node. The total number of nodes for Edge2Core to Edge8Core is indicated in Figure 4.10.

We also study the impact of the number of VNFs in the chain on the runtime. We indicated in Section 4.2.3.2.3 that many publications consider VNF chains of 0 - 10 or 0 - 3 VNFs for their tests. However the number of VNFs in a chain can be much larger, depending on the use case. A full virtual redundant EPC serving a large area would require much more VNFs, for example. Consequently, it is interesting to increase the number of VNFs in our NSs, and check that the algorithm still solves the problem within a reasonable amount of time.

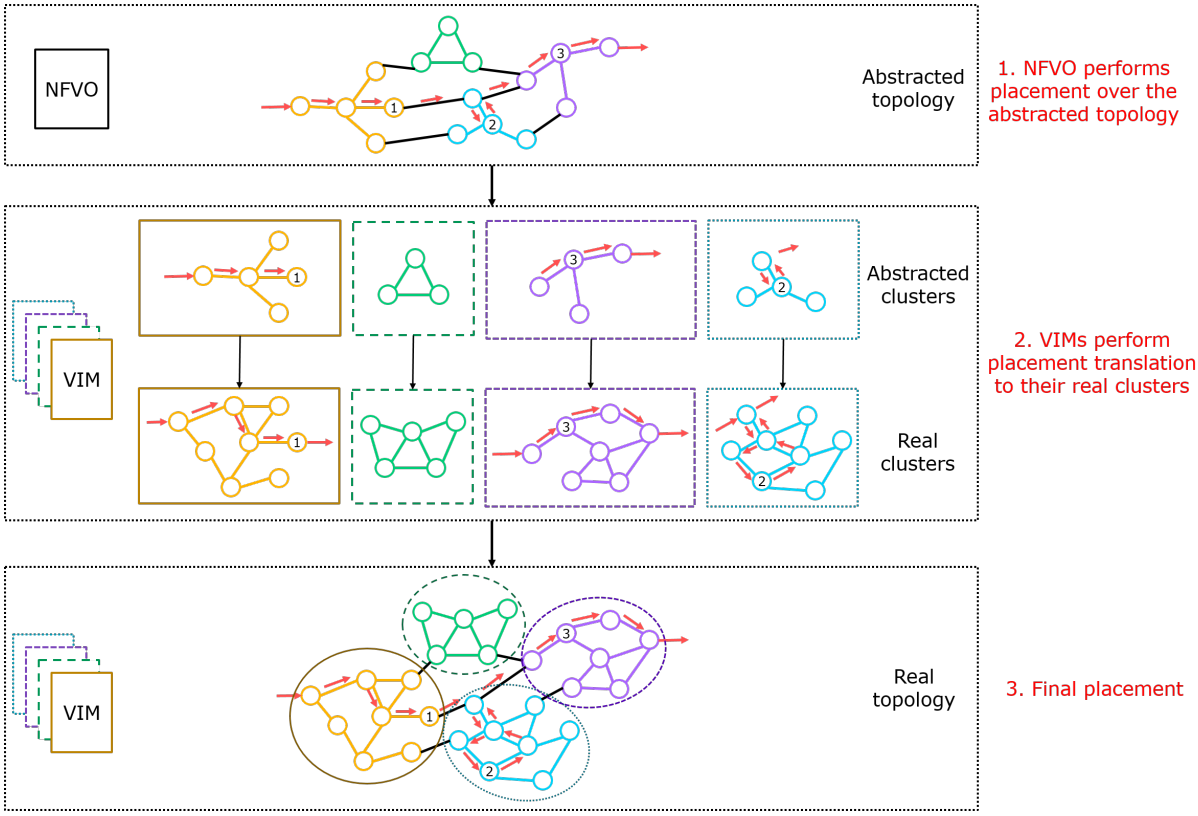


Figure 4.9 – Heuristic example

4.3.4.2 Performance

We first analyse the performance of the heuristic against the holistic ILP. The network is divided into clusters composed of a maximum of 100 elements (nodes and links) and each request tries to place 10 VNFs. Each node has 80 CPU units and 60 storage units, and each link has 800 bandwidth units. The delay factor is 100.

We developed our algorithm in Java, which offers an implementation of the betweenness computation, but only for nodes, not for links. For convenience we used this implementation, and during the cluster creation process we determine the next link to be deleted as follows:

1. We select the node with highest betweenness N_1,
2. Among the neighbours of N_1 we select the one with highest betweenness N_2,
3. The next link to be deleted is the one connecting N_1 and N_2.

Two effects linked to the abstraction technique can influence the heuristic performance. The first is the capacity of the algorithm to find a feasible path when one exists. In our case, this effect is very important when the network is almost empty. In this situation, it is almost certain that a solution exists, but our heuristic may miss it. The reason is that our abstraction are “utopian”: for each metric separately it computes the best possible score, and merges all those scores into a single link. However, there is no guarantee that any real path could provide all those scores at once. Hence, the NFVO may find a suitable path based on the abstracted topologies that cannot be instantiated in reality, while another suitable path may have been successfully implemented. The second effect is the overall quality of the placement over time. This effect is

more important in the end, when many NSs are placed. If the placement is poorly done then the acceptation rate will lower. In the beginning this effect has no impact since few NSs are placed, so the placement quality has limited importance.

To analyse those effects we measure the cumulative acceptation ratio of the heuristic during three phases:

— when the network is empty (the request is the first request received)
— when the network is saturated
— between the first request and the inflexion point (“normal” situation).

The results are presented Figure 4.10. This figure displays the performance of the heuristic compared to the direct application of the model with no abstraction for different topologies. Those results show the efficiency of the heuristic since we achieve a performance ratio between 93% and 96% between the first request and the inflexion point.

When the network is empty, the first request is accepted only 96% of the time due to our aggressive abstraction method that may propose paths that are more appealing than feasible paths but that do not fit in the physical infrastructure. It would be interesting to adapt other abstraction methods presented in Section 4.3.2 to our algorithm and compare their efficiency and reduce the cranckback effect.

Regarding saturation performance, we note that the heuristic performs very well. This tends to show that the overall placement is not perturbed too much by the heuristic, which can be explained since the abstraction method is designed to propose very good paths, sometimes too good to actually exist. We note that for Edge2Core the heuristic performs even better than the ILP. This is actually due to the comparatively very small size of Edge2Core (92 nodes). Because requests are randomly generated, and Edge2Core can only host a few of them, it is possible that, when the network reaches saturation, the request happened to fall right were the heuristic left some resources and the ILP did not. The total number of accepted requests for this topology on saturation when using the ILP is 108, so the difference between the ILP and the heuristic is less fewer than two requests.

4.3.5 Heuristic in a Mono-tenant architecture

4.3.5.1 Motivation

The VNFGPP is NP-complete [33], so computing times can become unbearable when the size of the network or of the VNFG increases. For this reason, all the works presented Section 4.1 propose heuristics to fasten the resolution. However, most of those heuristics are tightly linked to one specific version of the VNFGPP. As we detailed Section 4.1, many different variations of the VNFGPP exist, with their own optimization objectives and constraint specificities. For example, prohibiting to split flows between multiple paths, or to share VNFs between multiple services, potentially makes some heuristics inoperable. Using a heuristic that is highly dependent on some specificities of the constraints of a given situation may be too restrictive in practice. We propose to extend the abstraction technique used in the multi-tenant scenario to define a heuristic that performs well with any form of VNFGPP.

When the topology is too large to apply the ILP, the NFVO artificially divides its topology

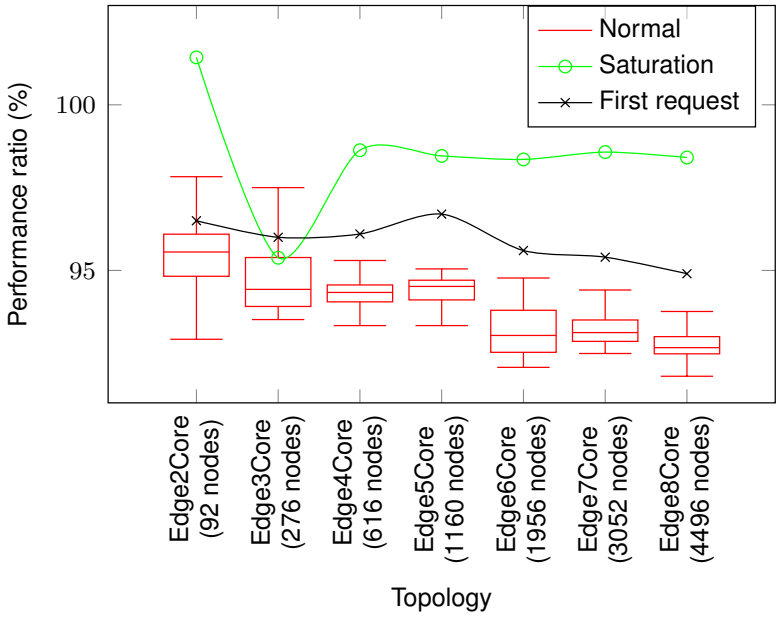
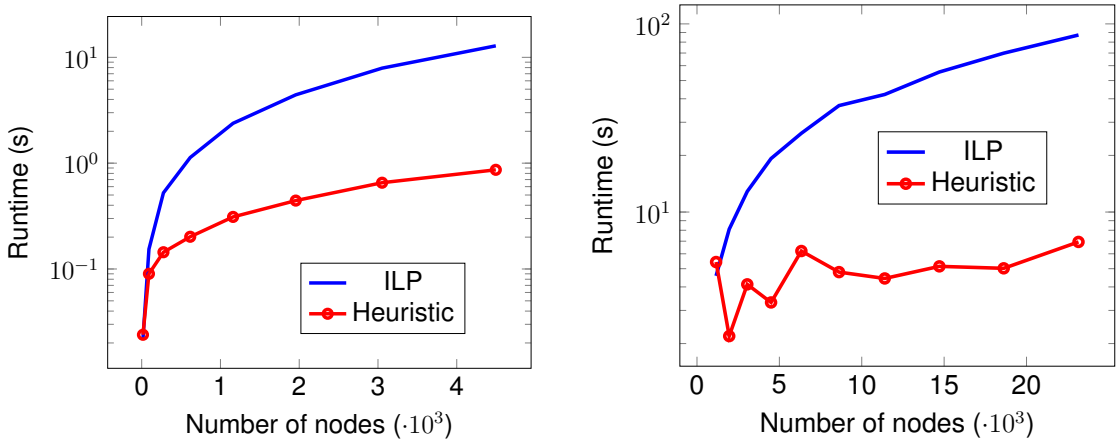


Figure 4.10 – Heuristic cumulative acceptation ratio vs. holistic solution

view into several clusters of limited sizes, using for example the algorithm presented in Section 4.3.1. Note that the abstraction process can be done offline if the topology is not expected to vary quickly over time. The NFVO then internally executes all the steps presented in Section 4.3.1. The execution of the ILP on abstracted topologies gives a first placement solution that associates each cluster with a sub-chain of VNFs to host. The algorithm transforms these results into parameters and executes the ILP a second time on each cluster of the abstract topology. Since the initial ILP is not modified, this heuristic can be applied to any variant of the VNFGPP.

4.3.6 Runtime evaluation

We analyse the computational time of our heuristic versus the holistic ILP. We take into account the time required by the heuristic to reserve and release the resources, as it implies additional computations to keep abstracted graphs up to date. Although the heuristic can run the second step depicted Figure 4.9 in parallel, we use only one thread in all our experiments (note that the solver itself runs in parallel, both for the ILP and the heuristic). Figure 4.11a shows that although runtimes are comparable on small instances, heuristic runtime increases at lower rate while network size grows. In order to emphasize the results we increased the sizes of the networks, we doubled the number of VNFs to be placed and allowed the heuristic to form clusters of up to 3000 nodes. The results, presented in Figure 4.11b, are very similar to the previous ones, pointing out that our heuristic can handle large-scale problems without facing rapidly increasing computation times.



(a) 10 VNFs Cluster max size=100

(b) 20 VNFs Cluster max size=3000

Figure 4.11 – Heuristic vs holistic solution runtime

4.4 Perspective and conclusion

In this contribution we propose an algorithm that solves the Virtual Network Function Chain Placement Problem allowing a fine management of the resources in order to satisfy the greatest number of requests. We show how network resource concentration and request constraints influence the Virtual Network Function Chain placement, unlike the topology structure. For multi-tenant architectures, we propose a method allowing the Virtualized Infrastructure Manager to expose an abstract view of the infrastructure topology. Leveraging on this approach we propose a heuristic to solves the Virtual Network Function Chain Placement Problem in multi-tenant architectures and deal with the computational complexity of the placement, both for mono- and multi-tenant architectures.

This work has been presented in the 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019) [112].

To extend this work it would be interesting to explore other abstraction options and compare the efficiency. Another track would be to design the abstraction recursively and quantify the impact of the size of the clusters to manage very large numbers of VNFs. Being able to place a lot of VNFs at once would allow both offline VNF placement and offline re-optimization of the placement.

Another important aspect of the VNF placement that has not been tackled in this contribution is the placement cost. In this contribution we focused mainly on NSs and VNFs with very strong QoS requirement, and how to manage edge resources to accept a maximum of requests. In this context, it is hard to optimize placement cost as VNFs do not have many placement options. However, for NSs or VNFs with lower constraints - typically, those which are able to reach core nodes - the number of options can be much more important. Thus, optimizing the resource reservation contracts can lead to significant savings. We are going to focus in this topic in the next section.

CHAPTER 5

PUBLIC AND PRIVATE CLOUD RESOURCE ACQUISITION FOR VNF EMBEDDING

In the previous chapter we detailed our solution to solve the VNFGPP, especially in the context of strongly constrained NSs relying on both cloud and edge computing, in mono- and multi-tenant scenarios. The placement choice is driven both by the resource capacities advertised to the NFVO by the VIMs in charge of the management of virtual infrastructures composed of servers and datacenters and by the QoS metrics of the networks that interconnect them.

The next step consists in actually reserving the resources to deploy the Network Services. If the VIM and the NFVO belong to the same tenant, this step can be done directly: the NFVO requests the resources and the VIM grants them, following the mechanism detailed in Section 2.3.3. In the multi-tenant scenario however, the resources are unlikely to be provided for free. While the NFVO can access a view of the resources proposed by other tenants and internally choose where to place its VNF graphs, the network operator would still have to pay (through a dedicated API) so that the NFVO can actually gain access to those resources. ETSI does not detail any negotiation process to buy resources, so we may envision two scenarios : either the buying process is done exclusively offline, and the resources seen by the NFVO are limited to those actually bought by its owner, or the process is done online by a specialized subcomponent of the NFVO, and the reservation process described Section 2.3.3 is adapted to support a payment system. In the rest of this chapter we suppose, for simplicity and without loss of generality, that the resource payment process is entirely managed by the NFVO itself via a dedicated API, although this could be done via other entities managed by the network operator that has access to exactly the same information as the NFVO.

The NFVO must select commercial offers to access resources and embed the Virtual Network Function Components that compose the VNFs. Note that, from the NFVO point of view, VDUs are indivisible. However, the operator that owns the NFVO may decide to position its own VIM between the NFVO and third party VIMs in order to divide the VDUs bought from those third party VIMs into smaller VDUs. For example, a system of containers may be installed on a bought VM, as we detailed in Section 2.3.2. Here we suppose that the NFVO owner is provided with VDUs that will not be further divided, either because they can't be (pods, containers) or because the NFVO owner does not want to perform such division, to spare time, resources or to avoid additional latency for example. For clarity we will suppose in this part that VDUs consist solely in VMs. One VNFC corresponds to exactly one VM running on a server. Each VIM may announce multiple offers from the cloud operator, or from several cloud operators when the VIM is an SLPOC (see Section 2.3.4). The substantial number of offers makes the selection problem more complex. Although the NFVO may reserve the resources on the fly, as a reaction to

an Network Service request, it may be more interesting to secure those resources in advance in order to benefit from lower long-term reservation rates, and to avoid unexpected resource shortage from the cloud provider.

Main cloud operators include Amazon Web Service (AWS), Microsoft Azure or Google Cloud. Although the details of their offers may vary, two main systems can be observed: guaranteed and spot. Clients choosing guaranteed offers have to pay a fix known price to use the resources. Once the price is paid the resources are granted. Regarding spot offers, a varying hidden reserve price is defined for each type of resource by the cloud operator. This price can be higher or lower than the guaranteed price for the same resource. Clients must bid on the price they are willing to pay for the resources, those who make an offer higher than the reserve are granted the resource for the time period, the others have to wait. Since running instances may be interrupted, spot instances are recommended for delay tolerant jobs only, which excludes most of the 5G use cases. Consequently in this contribution we only consider guaranteed offers (detailed in Section 5.2).

In this contribution, we propose an algorithm designed to help a network operator's NFVO to select the best combination of offers (in terms of price) to reserve the VMs needed to support a set of Network Services. The algorithm inputs consist in public offers from different cloud computing providers, as well as traffic load forecasts per VIM and price estimates for the following year from the network operator. To the best of our knowledge, this is the first attempt to address this issue in this context. Such an algorithm allows a network operator to plan its expenses over the next period and pay resource reservations in advance to lower the costs. In addition, we show that it can be used by a network operator to evaluate the utilization rate of a possible future private datacenter.

The chapter is organized as follows: Section 5.1 provides an overview of different works related to cost optimization at different levels of the MANO framework. Section 5.2 introduces our model while Section 5.3 presents both its evaluation in terms of runtime and cost, and how it can be used to assess the relevance for a network operator to invest in a private cloud. We conclude in Section 5.4.

5.1 Cloud resources cost and pricing optimization related problems

Reserving and buying resources from different tenants is a problem that network operators are likely to face with the deployment of an increasing number of virtual networks. Inter-tenant interactions start to be taken into account in the norms [54] (work in progress), including for resource pool sharing. In the same time, projects lead by industrials propose an analyse of the cost of deploying a service over a virtualized infrastructure, considering both the cost of a private infrastructure and the cost of public cloud operator offers [69]. However [69] does not propose any model or algorithm to select the best solution to obtain a given amount of resources for the lowest cost.

To analyse the scope of the different contributions proposed in the scientific literature, we recall the definition of the three actors involved. The cloud provider, represented by the VIM

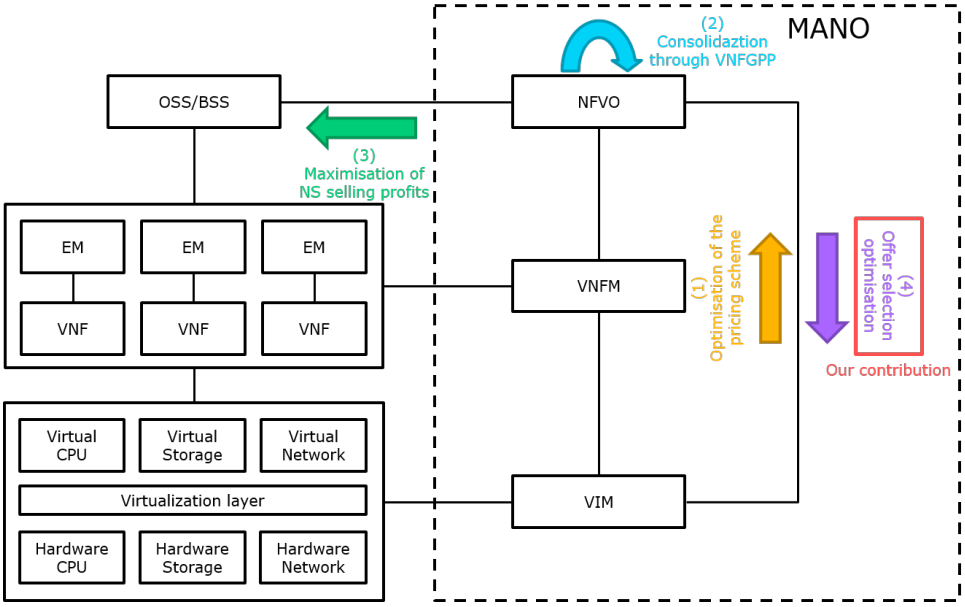


Figure 5.1 – Costs and prices optimization opportunities

in the MANO framework, sells virtual resources, such as VMs. The NFVO uses those VMs to build consistent network services with given features, guaranteed capacity and QoS. The slice provider consumes those Network Services to create a slice, a virtual network with a set of embedded Network Services that can address a family of use cases. The different optimization strategies are summarized Figure 5.1 and detailed in the remaining of the section.

Several papers are analyzing the best pricing scheme to maximize profits from the cloud provider perspective (1). These contributions mainly consider the spot offer, because this type of offer consist in selling resources through a bidding mechanism which can be subject to various optimization strategies. [144] focuses exclusively on spot instances, and aims to optimize cloud provider revenues. Authors in [2] have a similar objective. They are interested in hybrid pricing schemes that present both spot and guaranteed offers. Using game theory and queuing theory, they demonstrate that, in most of the cases, spot offers should not be proposed. Moreover, they show that only a waiting cost threshold determines whether or not a job will try to bid on the spot instance market. In our context of a network operator, a network function that is not started in due time potentially looses all its value (if the service request is cancelled due to the response delay), and potentially discontents many customers. Consequently, this property supports our choice to ignore spot instances. In the context of public clouds with apparent infinite resources, authors in [87] show that a provider may artificially simulate a shortage to maximize their profit. They briefly point out that, for long term jobs, cloud consumers (like NFVOs) should consider reserved options, or even buying their own hardware, but they do not perform any additional analysis.

From the NFVO perspective, the most explored way to reduce costs is through VNFGPP [6, 94, 145, 101, 131]. Although the techniques, context and side objectives may differ, the strategies of those papers follow a similar pattern: in order to reduce the resource cost they consolidate the placement, using each instantiated VNF at the maximum of its capacity. This process is purely internal to the NFVO and does not involve any other actor (2). Authors in [83] adopt another strategy that does not follow MANO architecture, so it is difficult to apply their approach

to our situation. However, if we map what they refer to as a “chunk of network” onto a Network Service, we can use their proposal to address the case where the slice provider has a set of slices to instantiate, and the NFVO disposes of a limited set of Network Services (3). The objective is to maximize the profit of the NFVO using successively price competition, auction and optimisation.

All of those approaches consider interesting and complementary solutions to lower the costs or optimize the revenue for the different actors of the architecture. However, to the best of our knowledge no publication so far investigated the choice that the NFVO has to make when facing different types of commercial offers to buy resources in order to actually deploy the VNFCs requested to run the Network Services (4).

In this paper we propose an ILP that addresses this challenge. We evaluate its computation time against various parameters and we provide cost comparisons of our algorithm with baseline solutions. Finally we suggest that this system can assist the network operator in determining the opportunity to build its own private datacenter.

5.2 Model and problem description

The Network Service provider has a prevision of the traffic that it will have to manage for a given period of time in the future, such as one year, based on past experience. From this estimation, it can deduce the Network Services needed to handle the traffic from its inner network service catalogue, and place them into the network using any version of the VNFGPP, as the ones presented in Section 5.1. This placement takes into account all QoS related constraints, such as delay, loss or amount of available bandwidth along the path. Once every VNFC is assigned to a VIM for any time of the foreseen period, the NFVO can start to evaluate which offers should be selected from each VIM individually. It is this last step that we handle in this chapter. An overview of the inputs and outputs of our model is presented in Figure 5.2.

5.2.1 Offers description

VIM offers can take many different forms and it would be impossible to consider all of them. We first describe the most generic offer possible. Then, we detail how to express the offers of AWS, one of the leaders of the cloud business, with this template. An offer proposed by a cloud operator represents the pricing of a specific VM template, referred to as flavor, over a given time interval (*i.e.* one or more time slots). Once an offer is paid, the VM of the corresponding flavor can be instantiated during the given time slots. A flavor may have a variety of attributes. Without loss of generality we consider only CPU and RAM . Similarly, we suppose that VNFCs only require CPU and RAM to run. Each flavor may be instantiated using different reservation offers. As evoked in Section 5.1, we only consider guaranteed offers. The most generic form of reservation is composed of a **fixed cost** paid in advance, a **variable cost** paid on a per-use basis and a set of time slots that defines when the reservation can be used. The variable cost might be subject to market fluctuations, and we suppose that the network operator keeps a record of those prices to be able to predict their future variations. We emphasize that an offer is bounded to a specific flavor and, once paid, cannot be used for another one. The

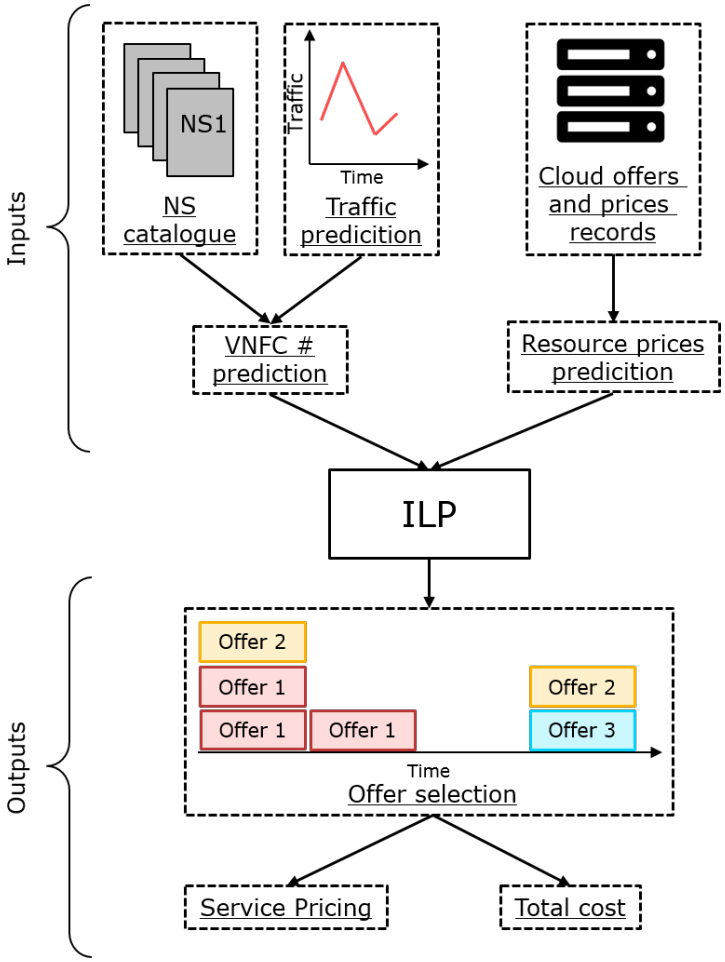


Figure 5.2 – Offer selection model overview

flavor is bounded to a cloud operator (although different cloud operators can happen to deliver the same type of flavors). Our model is designed to handle such a generic reservation offer, but we can easily derive more specific cloud operator related offers. As an example we introduce some AWS tariffs translated in our framework [1]:

— *On demand* offers are purely "pay per use": no time slot restriction and no fixed cost.
— *Reserved* offers have a limited duration (1 or 3 years) and an upfront cost, but the variable cost is null.
— *Scheduled* offers are similar to reserved offers, except that they apply only on given hours within the day. The equivalent hour rate is the same as for reserved offers.

Our algorithm outputs the amount of each offer that should be used at each time slot of the future period.

5.2.2 Notations

Typically a VNFC is running during multiple time slots. From the Network Service consumer point of view it represents one unique VNFC. In our model however, a VNFC is bounded to one unique time slot. Consequently, one VNFC running during N time slots is represented by N VNFCs, each one running during one of the N considered time slots. Moreover, our model does not decide which VNFC should be installed on a specific VM instance. This mapping should be done by a dedicated algorithm, which would typically aim at minimizing the number of migrations (*i.e.,* the number of time a VNFC will be moved from one VM to another).

We remind that, following ETSI MANO framework, a VNFC can be hosted by one VM only (it cannot be splitted over multiple VMs), and one VM can host at most one VNFC at a time even if this VNFC does not consume all the resources.

Notations used in the model are summarized in Table 5.1. In the whole model "operator" refers to a cloud operator.

1. AWS pricing, `https://aws.amazon.com/ec2/pricing/`

Table 5.1 – Notations

Name	Description
V	Set of VNFCs to be placed
T	Set of time slots (of equal length)
V_t	Set of VNFCs to be placed during time slot t
t_v	Time slot $t \in T$ during which VNFC $v \in V$ has to run
C^v	CPU requested by the VNFC $v \in V$
S^v	Storage (RAM) requested by the VNFC $v \in V$
O	Set of operators
F	Set of flavors
F_o	Set of flavors proposed by $o \in O$, $F_o \subseteq F$
R	Set of reservations
R_f	Set of reservations aiming flavor $f \in F$, $R_f \subseteq R$
R_{f_o}	Available reservations for flavor $f_o \in F_o$, $R_{f_o} \subseteq R_f \subseteq R$
τ_r	Set of time slots during which $r \in R$ is active, $\tau_r \subseteq T$
C_f	CPU of flavor $f \in F$
S_f	Storage (RAM) of flavor $f \in F$
C_o	Total amount of CPU that operator $o \in O$ can provide
S_o	Total amount of storage that operator $o \in O$ can provide
$p_{o,f,r,t}$	Variable price for reservation $r \in R_{f_o}$ for time slot $t \in T$
$P_{o,f,r}$	Fixed price for reservation $r \in R_{f_o}$

In order to clarify the notation τ_r, we give here some examples of the its value for the different types of reservation we consider :

— *On demand*: τ_r is composed of only one time slot.
— *Reserved*: τ_r is composed of all the adjacent time slots of the period covered by the reservation. Typically, if we are running our model over a 1 year period, and the *Reserved* offer is AWS 1 year reservation offer, then $\tau_r = T$.
— *Scheduled*: in the AWS case, τ_r is composed of sets of adjacent time slots, which represents all the same period of a day, *e.g.,* from 1am to 6am. In the special case where the sets are adjacent, we are in the *Reserved* case.

We also emphasize that, in the specific context of this model, the term *reservation*, used to refer to the elements of R, is synonym of *offer*. The term offer is actually more accurate, but would have induced some confusion is the notation with the term “operator”. Consequently, On demand offers are also included in the reservation pool, although they are technically never reserved by the NFVO owner, but rather bought when needed. For these specific offers the fixed price is always 0.

5.2.3 ILP model

5.2.3.1 Variables

We introduce three types of variables representing the VM instances (5.1), the VNFC embedding on these instances (5.2) and the fixed costs that have to be paid (5.3). A triplet $[o, f, r]$ defines a reservation of flavor f provided by operator o under reservation tariff r.

$$\phi_{o,f,r,t} = \begin{cases} 0..|V_t| & \text{Number of VMs reserved using } r \in R_{f_o} \text{ at time } t \in \tau_r \\ 0 & \text{if } t \notin \tau_r \end{cases} \tag{5.1}$$

$$x^v_{o,f,r,t} = \begin{cases} 1 & \text{IF VNFC } v \in V \text{ is installed on a VM obtained} \\ & \quad \text{through reservation } r \in R_{f_o} \text{ at time slot } t \in T_v \\ & \quad \text{AND } C^v \leq C_f \\ & \quad \text{AND } S^v \leq S_f \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

$$\Phi_{o,f,r} = 0.. \max_{t \in \tau_r}(|V_t|) \quad \text{Number of reservation } r \in R_{f_o} \text{ to pay in advance} \tag{5.3}$$

5.2.3.2 Objective

The objective of the problem is to minimize the expected cost over the full period. The total cost includes the payment of variable prices only when a VM is actually running, and of fixed prices for the long-term reservations.

$$\min \sum_{o \in O} \sum_{f \in F_o} \sum_{r \in R_{f_o}} \left(\sum_{t \in T} \phi_{o,f,r,t} p_{o,f,r,t} + \Phi_{o,f,r} P_{o,f,r} \right) \tag{5.4}$$

5.2.3.3 Constraints

Each VNFC must be instantiated during its time slot (5.5)

$$\sum_{o \in O} \sum_{f \in F_o} \sum_{r \in R_{f_o}} x^v_{o,f,r,t_v} = 1, \ \forall \ v \in V \tag{5.5}$$

Since each VNFC has to get its own VM their should be at least as much instances as VNFCs (5.6) :

$$\sum_{v \in V_t} x^v_{o,f,r,t_v} \leq \phi_{o,f,r,t}, \ \forall \ (t, o, f, r, v) \in (T, O, F_o, R_f, V) \tag{5.6}$$

An operator cannot offer more CPU (5.7) or RAM (5.8) than its capacity.

$$\sum_{f \in F_o} \sum_{r \in R_f} \phi_{o,f,r,t} C_f \leq C_o, \ \forall \ (t, o) \in (T, O) \tag{5.7}$$

$$\sum_{f \in F_o} \sum_{r \in R_f} \phi_{o,f,r,t} S_f \leq S_o, \ \forall \ (t,o) \in (T,O) \tag{5.8}$$

At any time there must be at least as many fixed costs paid as reservations used (5.9)

$$\phi_{o,f,r,t} \leq \Phi_{o,f,r}, \ \forall \ (o,f,r,t) \in (O,F_o,R_f,\tau_r) \tag{5.9}$$

5.2.4 The licence problem

In addition to the VMs, the cost of the VNF licences is another important aspect that must be taken into account when evaluating the total cost of a service. Reducing it implies to know the licence billing method. This is actually non trivial, since many systems exist and are not standardized [55].

The papers presented in Section 5.1 that tackle this issue use the VNFGPP to try to minimize licence cost and the resource cost all together using consolidation. This logic is directly inspired from traditional networks with physical middleboxes. Middleboxes, just as their licences, should be used at full potential, else they are partially wasted. VNFs, however, have a major property: they can scale up or down, depending on the traffic load, which may reduce considerably the interest of consolidation. Just as licence billing system followed traditional middelboxes logic of exploitation, we may suggest that VNF licences could gradually embrace VNF work flows. Some major actors already issued “pay per use” licences similar to AWS on demand offers [13], and in the future they may produce more complex offers to mirror the ones proposed today for VM reservations. If so, our model could be used to handle licence reservations as well.

5.3 Experimentations and results

We used the Gurobi solver [74], 4 logical cores Intel i5-6200U and 2GB of RAM to evaluate our algorithm.

5.3.1 Parameters

5.3.1.1 Incoming traffic

To simulate the traffic we use the dataset of the City of Milano provided in [16]. For the sake of the example we decided to focus on the deployment of one specific VNF representing a Mobility Management Entity (MME). In 4G networks the MME is a management entity that plays an important role for UE registration and mobility through the network. We extract and aggregate the callIn and callOut activities from the dataset: they correspond to actions that would trigger the MMEs, as they signal the beginning of a communication, initiated respectively by a remote UE and by the current UE. This activity is represented Figure 5.3a. As detailed by the authors of the dataset, the activity correspond to the number of events (such as callIn) that occur in the network, multiplied by an unknown factor to keep network operator real data secret. Figure 5.3b represents in greater details the activity of the first week of November, and

shows that the traffic is cyclic and follows day/night shifts. In addition, the activity over two months (which represents the full dataset) denotes a weekly periodicity. We convert this activity into a number of MMEs that must be provisioned to support the amount of UE registrations directly related to this activity, based on the work presented in [8]: one MME of 2 CPU and 2 GB of RAM to handle 40 requests. As a result, we obtain the required number of MMEs by time slot of 10 minutes for November 2013 (represented in the graphics of Figure 5.5), December 2013 and the 1st of January 2014. Here we suppose that the MME VNF is composed of only one VNFC, which may not be the case in practice but it clarifies our analysis. We also consider that the MME only scales out and does not scale in. This is a strategic choice that has to be made beforehand by the NFVO, our algorithm adapts to the choice by selecting different flavors. We consider that all the MMEs are allocated to the same VIM for the whole city, otherwise the algorithm should be run separately for each VIM.

5.3.1.2 Offers

Regarding cloud offers, we consider two operators: AWS public cloud and a private cloud operated by the network operator itself. Since we only have one type of VNFC to host, only one flavor will be proposed. For this example we chose AWS m5.large (2 CPU 8 RAM) instances, adapted to general purpose computing.

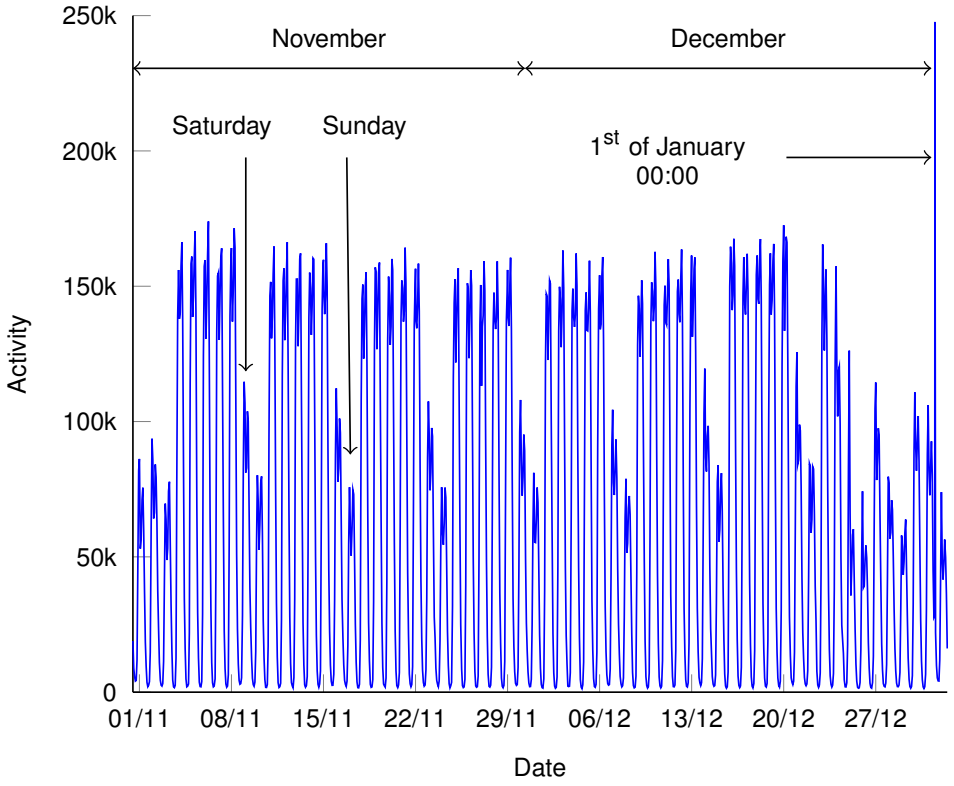
We assume here that, from a client point of view, public cloud resources are infinite. This assumption is consistent with other approaches on the subject [2][87][13]. As a consequence, AWS has enough resources to host all our VNFCs at anytime. We then selected the 3 main AWS offers and their respective tariffs taken from AWS website on the 22/05/2019:

— on Demand (OD): 0.096$ per hour, no fixed cost,
— reserved 1 year (R): 0$ per hour, 501$ fixed cost,
— scheduled daily (S): from 8h00 to 20h00 (daytime) : 0$ per hour, 250.5$ fixed cost.

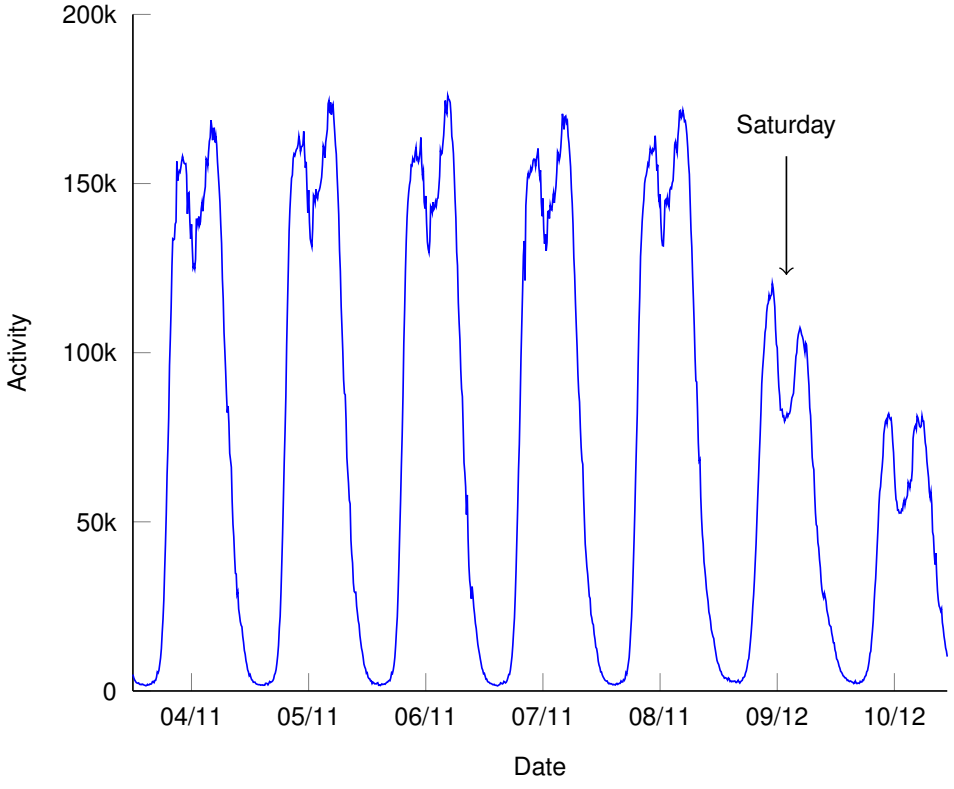
The private cloud, on the contrary, has limited resources. In our scenario we consider the deployment of only one VNF among all the VNFs that a network operator should deploy to provide complete services. Consequently, we attribute to the private datacenter reduced amount of resources, reflecting this idea that only a fraction of the necessary functions has to be deployed. We consider it has 20 CPUs and 80 GB of RAMs. Since it belongs to the network operator, the hourly cost should be marginal, corresponding only to the extra electricity consumption. However, to better estimate the real cost of using the datacenter, we estimated the OPEX cost taking into consideration hardware, staff, and electricity costs. We obtained an approximated hourly rate of 0.012$ (P offer).

5.3.2 Runtime

We analyse the ILP runtime regarding the number of available offers, the length of the foreseen period, and the number of VNFCs to place. Results are presented in Figure 5.4.

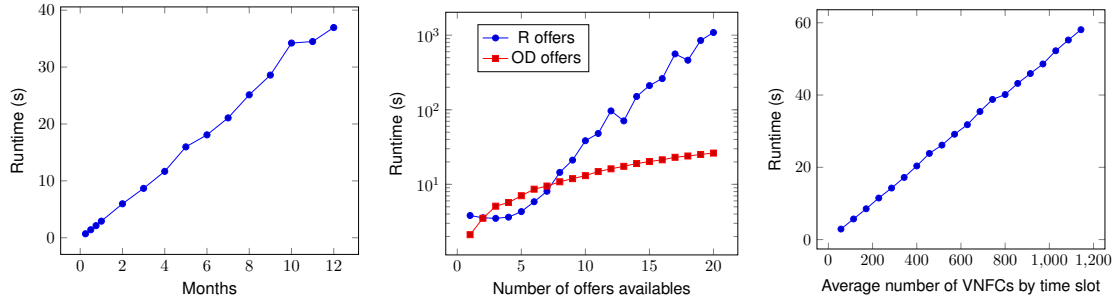


(a) From November 2013 to January 2014



(b) First week of November 2013

Figure 5.3 – Aggregated CallIn and CallOut activities of the city of Milano based on data from [16]



(a) Runtime by length of the period (b) Runtime by number of offers (R or OD) (c) Runtime by number of VNFCs to place

Figure 5.4 – Comparison of ILP runtime with different parameters

5.3.2.1 Length of the period

To produce different period lengths we take one, two, and three weeks of November, the full month, and finally we concatenated from two to twelve times the traffic of the month of November to obtain the equivalent of a year. We propose two offers in the algorithm: R and OD. In Figure 5.4a, we observe that the computation time grows linearly with the length of the period. While other parameters can increase a lot, the length of the period is bounded. Indeed, even if AWS proposes 3 years offers, the precision of the traffic prediction and offer price variation will decrease over time, making long term commitments hazardous.

5.3.2.2 Number of available offers

From a complexity point of view, introducing more cloud operators, more flavors or more offers is equivalent. Therefore, we choose to simply focus on the multiplication of offers in this section. Because R and OD offers have very distinct characteristics we analyze them separately and display the results in Figure 5.4b. Regarding OD offers, we build an offer by randomly taking, for each time slot, a price between 0.060$/h and 0.180$/h, and we propose from 1 to 20 offers. For R offers, we propose the classical OD and R offers, plus between 1 and 20 S offers. S offers have a duration of 4 hours, and start every hour: with 20 of them the full day is covered. We observe that the computation time is linear with the number of OD offers, but exponential with S ones. For the first S offers we observe that the computation is constant. Indeed, first S offers propose to reserve resources during the night (from 0h00 to 4h00 for the fist one, 1h00 to 5h00 for the second one ...), which corresponds to periods with little to no traffic. Consequently reserving resources over those specific periods less interesting than taking R and OD offers, and the ILP quickly discards those nightly reservation offers.

The exponential complexity sets a limit to the capabilities of our model: all possible S offers cannot be considered, a choice has to be made. In our use case, the daily periodicity of the work load makes this task relatively easy, and selecting simply one S offers out of all the possible ones greatly improves the overall cost (see next section).

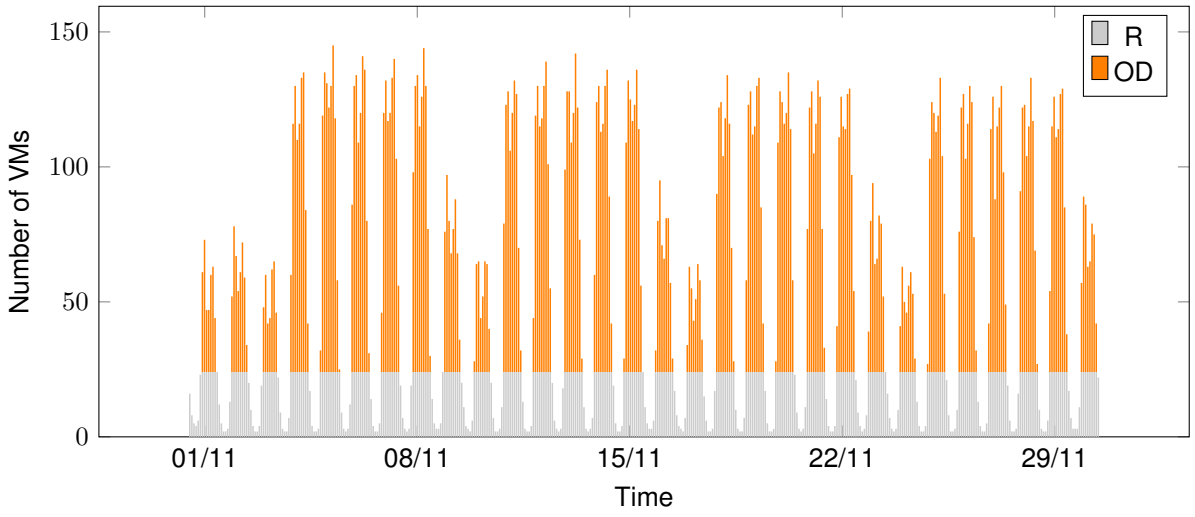
5.3.2.3 Average number of VNFCs by time slot

We also study the behavior of the ILP over the month of November proposing only R and OD offers, and we multiply the number of VNFCs to place at each time slot by 1 (initial situation) up to 20. We observe in Figure 5.4c that the computational time grows linearly with the number of VNFCs, which allows our ILP to be used in large size datacenters on which the network operator may have several VNFCs to embed.

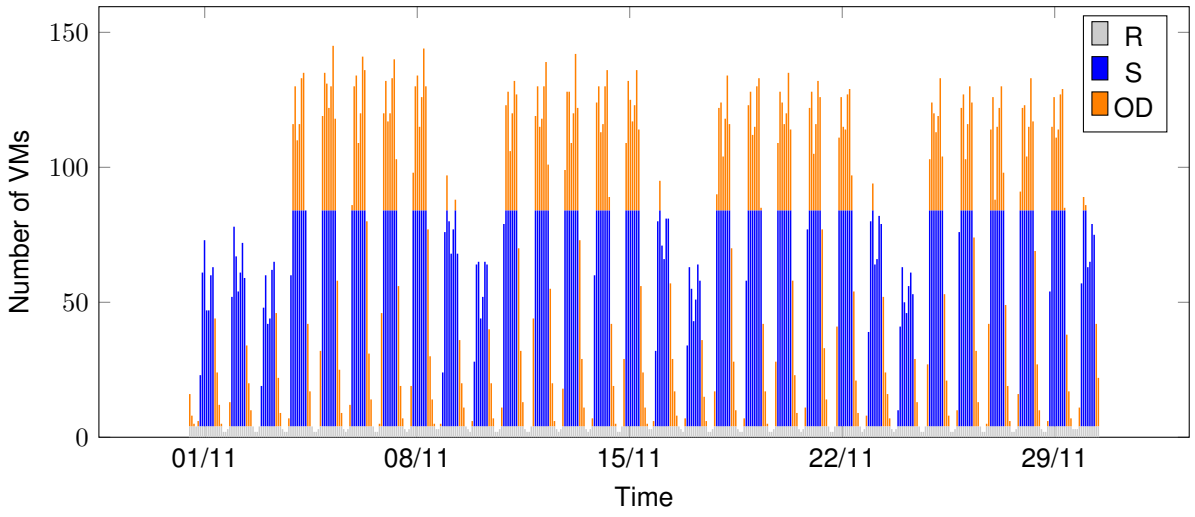
5.3.3 Cost

We evaluate the interest for a network operator to use our algorithm by comparing different reservation strategies. We first suppose that the network operator does not have a private datacenter. When operating their own systems, network operators tend to dimension them not to absorb the average traffic load, but rather to handle peak loads [13], which leads to overprovisioning. Translated directly into AWS language, it would mean taking only R offers. We refer to it as the R strategy. Even if this strategy doesn't seem complicated, choosing an optimal set of reservations when multiple sizes of VNFCs are present already requires some planning. Taking advantage of the flexibility of the cloud, the network operator may decide to buy only OD offers to face the traffic as it comes, using a straightforward OD strategy without any further planning. It could also decide to mix R and OD offers (the $OD + R$ strategy). Lastly, noticing that the traffic strongly follows the night and day cycle, it could opt for a scheduled daytime offer (the $OD + R + S$ strategy). This strategy produces the optimal cost provided by the public cloud, given the offer we chose to focus on, and we base the comparison with all other costs on it (see Figure 5.6). The network operator may wonder what would be the final cost if he was owning its own private datacenter. To give an answer, we introduce an $OD + R + S + P$ strategy considering OD, R, S offers plus a P offer corresponding to the placement in its private datacenter. The results of the strategies involving several types of reservations are displayed in Figure 5.5. In this figure we represent the total cost the NFVO will have to pay over the period depending on the offers considered, as determined by our model. To ease the comparison between the different results, the costs are expressed as a percentage of the case where all the considered public offers are taken into account, but not the private resources. We can note that the R offer is not much used when the S offer is available, as S offer fits much better the workload needs for the same hourly cost. In the $OD + R + S + P$ strategy R offer is actually not used at all because the reduced nightly traffic is absorbed by the P offer.

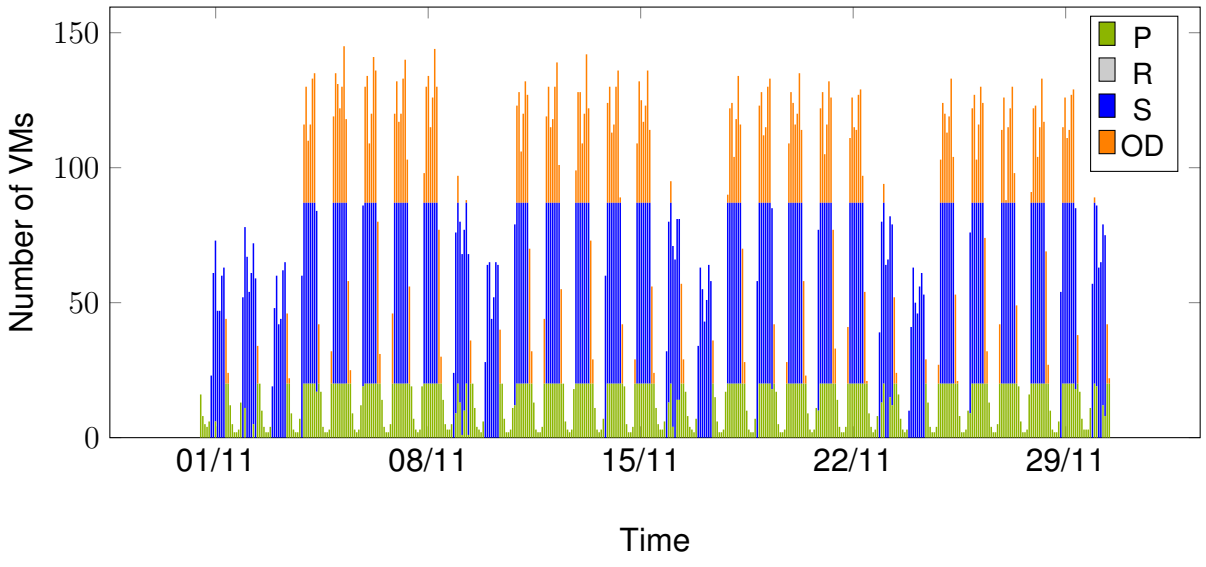
The comparison of the different strategies' costs is provided in Figure 5.6. First of all, we can note that the R strategy performs very bad. This is due to the 1st of January traffic peak that forces the network operator to book a handful of resources that will stay idle the rest of the time. This reflects well the default of the classical over-provisioning strategy. Second, the difference between OD and $OD + R$ is quite small due to the very low traffic at night, which makes reservations quite unattractive. The $OD + R + S$ strategy performs very well because S focuses precisely on peak hours. Thus, using our algorithm to plan in advance the offer selection can bring significant advantages compared to R and OD strategies.



(a) R and OD offers



(b) R, OD and S offers



(c) R, OD, S and P offers

Figure 5.5 – Selected offers through time

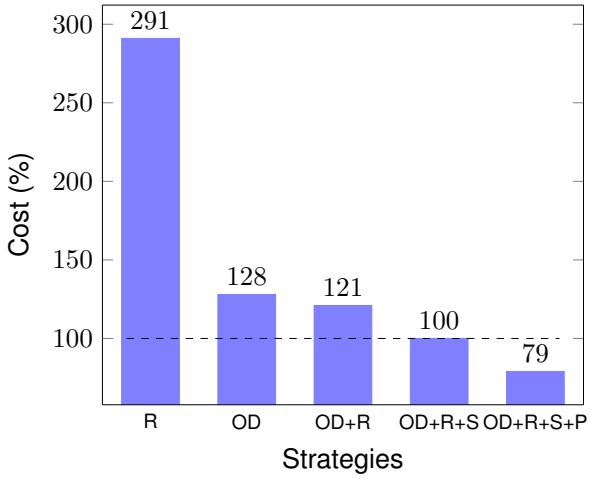


Figure 5.6 – Cost performance using different offers

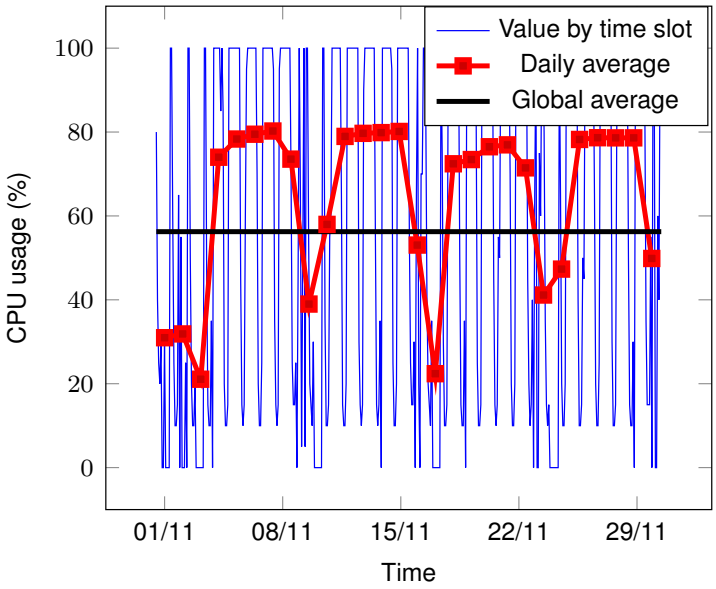


Figure 5.7 – Private cloud CPU utilization through time

5.3.4 Private datacenter utilization

In this section we focus on the situation where a network operator has to decide whether or not to build its own datacenter to absorb a portion of the traffic. In Section 5.3.3 we estimated the benefits in term of costs, however, this does not take into account the global investment required to build the infrastructure. Since such a facility is costly to build, we suppose that a network operator will be interested in knowing whether it will be used at the maximum of its capacity or not, and maybe in predicting the periods when it can sell the unused resources. To analyse this, we take the results of Section 5.3.3 provided by strategy $OD + R + S + P$, and we measure the amount of CPU used at any moment. Results over the month of November are presented in Figure 5.7.

As expected, the average utilization rate is not 100% since the traffic at night is very low: the utilization rate is only around 70% - 80% during the week days. It is even lower during the week-end: although the traffic is lower, we could expect the datacenter to be fully used during the day.

However, since scheduled offers are reserved for the full week and cannot be cancelled for the week-end, it turns out to be more economic to use the already-paid resources rather than the private datacenter ones. This effect is especially important during Sundays, when the traffic is at its lowest level (see Figure 5.5c). This affects the global utilization rate that drops at 56%. We conclude that taking into consideration available commercial offers, especially regarding periodic traffic, can modify a lot the actual benefit a network operator can expect from building its own facilities.

5.4 Perspective and conclusion

In this contribution we proposed a model to assist the NFVO in the process of selecting cloud provider offers, in order for it to buy enough resources to embed all the required VNFCs at the best possible price in due time. Based on NFVO's workload predictions, our model allows to plan in advance long-term reservations, which come with reduced hourly prices. We applied this technique to a network operator use-case using a real dataset. We showed that the model keeps doing well when the amount of VNFCs, On Demand offers and length of the prediction increase. The long-term scheduled reservation offers however induce exponential complexity. We mitigated this shortcoming by taking advantage of the periodicity of the traffic, showing that selecting only one well-chosen scheduled offer already fairly reduces the overall placement cost. Finally, we stressed the fact that taking into account existing commercial offers is really important for a network operator when it comes to decide whether or not to build its own private cloud. Although the OPEX is reduced, the CAPEX may not be as worthy as expected since the projected utilization rate might be lower than anticipated.

This contribution has been published in the 34th International Conference on Information Networking (ICOIN 2020) [111].

For future work, we plan to develop a heuristic to handle the exponential complexity induced by reserved offers. We would also like to propose an algorithm dedicated to actually assign a specific VM to each VNFC, minimizing the migrations between VMs, which is the next and last step the NFVO has to perform to fully embed a network service.

CHAPTER 6

CONCLUSION

The global increase of traffic and the incoming of 5G are pushing traditional networks to their limits. New 5G service requirements such as ultra low latency, high number of connections or massive data transfers require new solutions to operate the network. To address those challenges new paradigms emerged. The most successful today are SDN, NFV, and the slicing. Those concepts are based on an abstraction and an isolation of the resources in order to better serve every specific use case while relying on a single shared physical infrastructure. While this strategy is very promising it also generates new challenges, especially in terms of management. Indeed, most of the implementations rely on a logically centralized control plane. To fully benefit from this type of architecture solutions must be developed to limit its drawbacks, especially regarding the difficulties of communication between the distributed data plane and the centralized control plane, and to maximize its benefits, such as the optimal management of the resources through dedicated algorithms.

In this thesis we organized our work to follow a main thread: the establishment of a network service using SDN/NFV capabilities. We focused on the resolution of different challenges that this operation induces. We tackled both architectural issues and algorithmic optimization.

6.1 Summary of the contributions of the thesis

The contributions of this thesis can be separated in three parts:

— **Establishment of a connection with guaranteed QoS using an SDN framework:** in this first contribution we focused on the base of a network service: the connectivity. Our objective was to establish a connection between two points of the network with a guaranteed QoS. Such capability allows the network operator to enforce SLA engagements, but it also implements the mandatory isolation between users in the context of slicing. Therefore, strict QoS management is critical. The difficulty here lied in the centralization of the control plane: QoS monitoring was reactive, allowing QoS violations, and generative of massive amount of control traffic. To solve this issue we maintained the QoS policy design in the control plane, to fully benefit from centralized resource management, but we relocated the policy enforcement into the dataplane, to avoid control traffic and potential QoS violations. Results showed a very effective enforcement of the QoS policy while no dedicated monitoring traffic was exchanged between control and data planes. Once ensured that the QoS of the virtual links between the VNFs was secured, we started to develop an algorithm to place full chains of VNFs.

— **Development of an algorithm to optimally use network resources:** in this second contribution we tackled the Virtual Network Function Graph Placement Problem. This

contribution can be decomposed into two aspects. First, we developed an algorithm that focuses on maximizing the number of Network Services that we could embed at runtime. Since those Network Services may have very strict QoS needs, especially in terms of latency, we couldn't rely only on central datacenters, but we also had to consider smaller edge facilities. We evaluated our algorithm against multiple topologies, various resource distribution patterns and different Network Service requirements. We demonstrated that it performs equal or better than a bandwidth-focused one. The best performances were observed for topologies with concentrated resources trying to host Network Services with latency constraints, which represents the likely scenario in practice. Second, we focused on a situation mostly ignored in current literature: the multi-tenant scenario, where VIMs may not disclose all their topology details to the NFVO, hence impacting the quality of the Virtual Network Function Graph Placement Problem resolution. We proposed an abstraction method to let the VIMs expose their resources to the NFVO while maintaining a certain confidentiality level. The difficulty here was to design an abstraction method that took into account node resources, while traditional methods only consider link resources. We demonstrated the performance of our strategy by comparing the holistic algorithm to the heuristic. We then proposed an adaptation of this method to allow its usage to the mono-tenant scenario, in order to reduce computation time. We showed that the heuristic was much less sensitive to the increase of the network size than its holistic counterpart. While the VNF chain was virtually placed into the network by the NFVO one problem remained: we had to buy the resources necessary to actually embed it.

— **Development of an algorithm to find optimal commercial offers to buy necessary resources:** in this last contribution we get interested in the selection of public clouds (VIMs) commercial offers by the NFVO to embed the Network Services into the network. Based on traffic forecasts, the NFVO may decide to buy in advance resources from public cloud in order to cover its estimated needs. Such anticipation may allow it to benefit from advantageous prices. However, the multiplicity of offers makes the choice of an optimal solution difficult. This choice is the objective of the algorithm we designed in this contribution. While many researches has been carried out to optimize the costs and/or profits of the different actors in and around the MANO framework, to the best of our knowledge this was the first attempt to optimize the VIM offer selection problem. Our results demonstrated that using our optimization strategy could be greatly beneficial in terms of costs compared to more straightforward or less planned options. In addition, we provided an insight of the opportunity for an operator to build its own cloud facility. We highlighted that, in a context of competition among resource providers, such facility may be unexpectedly underused.

We tackled in this thesis several key aspects required for the implementation of a Network Service in an SDN-NFV framework, from resource reservation to the NFV chain placement to the final embedding. The summary of our contributions projected into the ETSI MANO framework is represented in Figure 6.1.

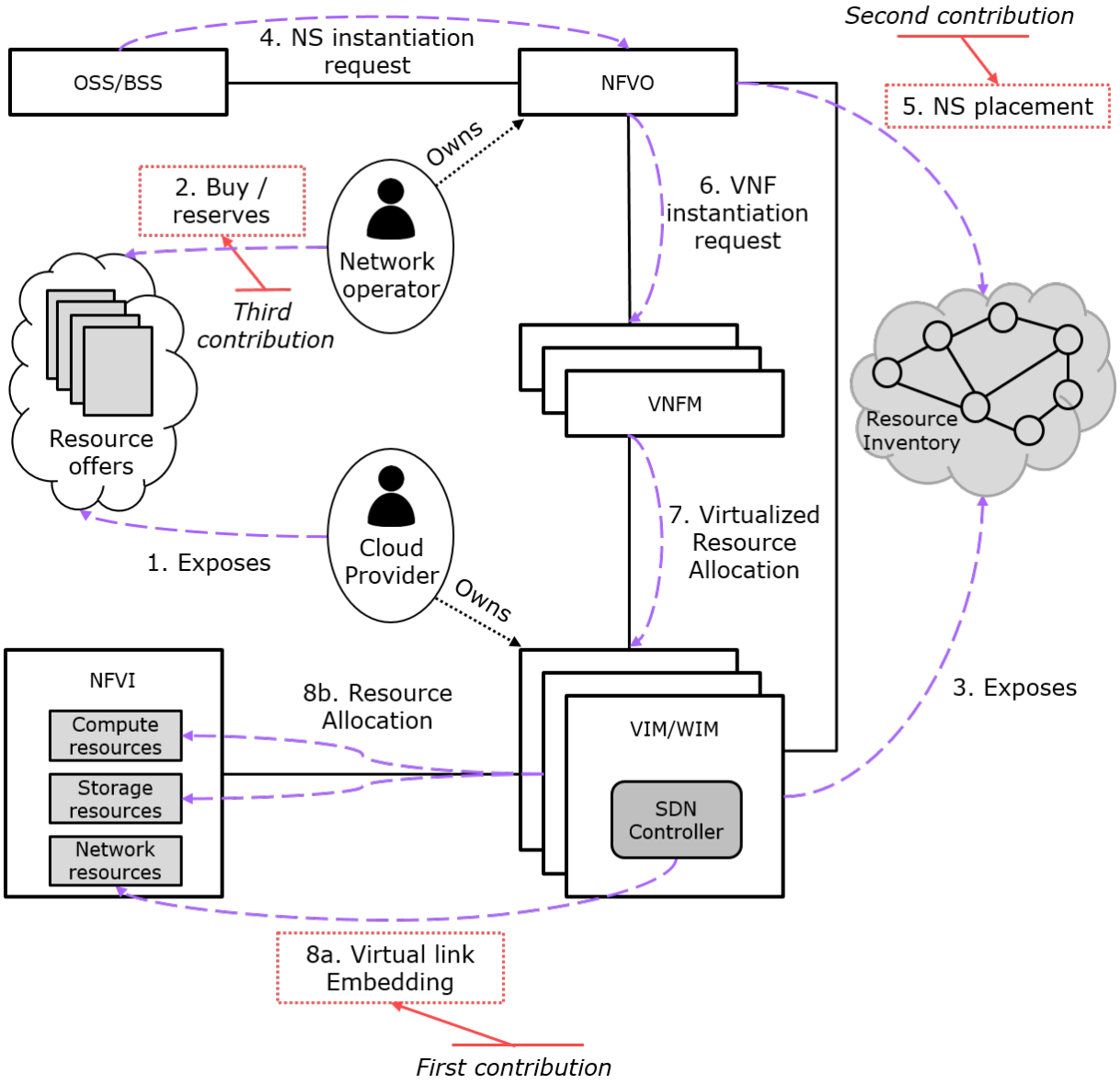


Figure 6.1 – Summary of the contributions

6.2 Perspectives

Regarding the different contributions of this thesis some perspectives have been detailed in their respective conclusions. In this section we provide some higher level research perspectives.

— **Node resource isolation:** while our first contribution focused on edge resource isolation it is important to emphasize that node resources must be isolated as well. This isolation is necessary to prevent Network Services from influencing each other when operating. It must be implemented at two levels: firstly, Virtual Deployment Units (two VMs for example) must be isolated so that independent VNFs do not interfere. Secondly, in case of shared VNFs, the load balancing process must be designed to fairly serve all services, according to their respective contracts. Regarding VDU isolation, the process might be very complex, in particular when it comes to memory sharing. Indeed, concurrent access requests to a common physical disk may induce performance drops. Those drops the result of multiple factors, that may not be known in advance, thus making them difficult to predict and manage through traditional analytical tools. To solve this issue, researchers propose to use machine learning, in order to make accurate predictions of application interactions when sharing a common memory [44].

— **The densification of edge resources:** as we evoked in Section 4 network operators will have to rely not only on vast central datacenters to get their resources, but also on edge facilities, smaller but necessary to address some use cases and relieve the core. In Section 5 we detailed how a network operator may buy resources from public cloud to instantiate their VNFs. However, this solution may only apply to services with loose delay constraints, as public cloud actors do not operate edge datacenters, and their solutions do not guarantee strict QoS in terms, for example, of latency. As a consequence, the network operator may have to build its own edge facilities. As it represents a slow and costly process, it would be very interesting to evaluate the optimal location and size of those edge points of presence. One possibility to estimate optimal location would be to modify the VNFGPP, allowing the algorithm to generate new points of presence in order to embed the VNFs, instead of being constrained by the exiting ones.

— **From RAN to Cloud RAN (CRAN):** the research presented in this work, and especially the VNF placement, was primarily directed to core networks. However, RAN also faces challenges raised by new use cases. The efficiency of an SDN-NFV based sliced core network would be strongly undermined if the access remained monolithic. Although core and access share a lot of similarities, they also display fundamental differences. The virtualization and the centralization into the cloud of the RAN functions to turn the RAN into a CRAN provides specific benefits, such as softer handover, better management of cell interference or load balancing between cells sharing a same area [53]. On the other hand, CRAN presents its own challenges: the radio resource is difficult to isolate, data volumes exchanged between the lower layers may be very large, and delays imposed by attachment protocols are very tight. All those challenges and benefits advocate for a specific management of the RAN. CRAN algorithms must be specifically designed to take advantage of large cloud computing resources to perform massive parallelism, in order to complete critical radio treatment operations in due time [129]. From a function

placement perspective, centralizing all functionalities in the cloud may be impossible due to fronthaul performance constraints, both in terms of latency and bandwidth. To solve this issue a flexible CRAN may be envisioned, leaving some functionalities close to the antenna [31]. The choice of the split may be partly done via placement algorithm based on network capacities, as the one presented in Chapter 4.

— **Network Service scaling:** as detailed in Section 2.3.5, scaling is one of the advantage of NFV, being part of its flexibility. However, implementing this functionality is not straightforward. First of all, scaling must be fully automated in order to keep up with the dynamicity expected from the NFV concept. Dedicated workflows are already proposed to tackle this challenge [3], but it also implies to develop efficient sensors, and elements to gather this monitoring information and turn it into scaling decision. As many proposal suggests to add artificial intelligence into network management [59], scaling process could fall under under machine learning attributions.

PUBLICATIONS OF THE THESIS

Journals:

1. Quang Pham Tran Anh, Jean-Michel Sanner, **Cédric Morin** and Yassine Hadjadj-Aoul, « Virtual network function-forwarding graph embedding: A genetic algorithm approach », *in: International Journal of Communication Systems* (2019), e4098.

International conferences:

1. **Cédric Morin**, Géraldine Texier, and Cao-Thanh Phan, « On demand QoS with a SDN traffic engineering management (STEM) module », *in: 2017 13th International Conference on Network and Service Management(CNSM)*, IEEE, 2017, pp. 1-6.
2. Quang Pham Tran Anh, Jean-Michel Sanner, **Cédric Morin** and Yassine Hadjadj-Aoul, « Multi-objective multi-constrained QoS Routing in large scale networks: A genetic algorithm approach », *in: 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, IEEE, 2018, pp.55-60.
3. **Cédric Morin**, Géraldine Texier, Christelle Caillouet, Gilles Desmangles and Cao-Thanh Phan, « VNF placement algorithms to address the mono- and multi-tenant issues in edge and core networks », *in: 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019)*, IEEE, 2019, pp. 1-6.
4. **Cédric Morin**, Géraldine Texier, Christelle Caillouet, Gilles Desmangles and Cao-Thanh Phan, « Optimization of Network Services Embedding Costs over Public and Private Clouds », *in: The 34th International Conference on Information Networking (ICOIN 2020)*, IEEE, 2020, pp. 1-6.

CHAPTER 7

ANNEX

7.1 Inter-VNFs delay bounds

The acceptable delay between two VNFs, denoted $D^{v,w}$, depends on the delay factor and the network diameter. Its bounds are determined using the following formulas:

$$scaleDelay = \frac{delayFactor * networkDiameter}{2}$$

$$0 \leq D^{v,w} \leq scaleDelay \tag{7.1}$$

All values between those two bounds are equally probable. Table 7.1 shows the delay bounds, as well as the average and maximum hops allowed to connect two VNFs in Cost and Edge51 topologies, according to different delay factors. Note that the minimum number of hops is always zero.

Table 7.1 – Consecutive VNFs delay bounds details for Cost and Edge51

Topology	Cost			Edge51		
Delay factor	Delay bounds	Average hops	Max hops	Delay bounds	Average hops	Max hops
50	0 - 200	1	2	0 - 150	0.75	1.5
100	0 - 400	2	4	0 - 300	1.5	3
150	0 - 600	3	6	0 - 450	2.25	4.5
200	0 - 800	4	8	0 - 600	3	6
300	0 - 1200	6	12	0 - 900	4.5	9
400	0 - 1600	8	16	0 - 1200	6	12
800	0 - 3200	16	32	0 - 2400	12	24

7.2 SDN - Control distribution

SDN offers many advantages and opportunities for network management. Early controllers, such as the first versions of ODL [104], Ryu [1], Beacon [52] or Floodlight [77], were implemented as single pieces of software running on a single physical machine. This was not a problem back then, as those controllers were primarily used for demonstration purposes over very small networks without any industrial-level quality requested. However, centralization also brings significant drawbacks that make such controllers unsuitable for carrier-grade deployments:

— responsiveness: as opposed to traditional distributed systems, a centralized one may suffer from delays between edge switches and central controller

— scalability: one controller can only process a limited amount of events. Moreover, applying complex routing algorithms over a large topology may be computationally intractable. In addition, the control plane may not be able to provide enough bandwidth in the vicinity of the controller as all control traffic of the network will end up in this bottleneck.

— reliability: a unique controller represents a Single Point Of Failure (SPOF). If the controller fails, either accidentally or as a result of an attack, the network falls.

Those drawbacks have been studied in various surveys [22] [97] [20] [117]. Proposed solutions consist in augmenting the number of controllers, leading to distributed architectures.

7.2.1 Definitions and terminology

Before detailing the main solutions proposed to distribute the control plane, we first have to define the terminology associated with distribution, as all authors do not use exactly the same.

When a controller is installed on a single machine we call it *physically centralized*. On the contrary, when several instances working as part of a global architecture are installed on several machines we call it *physically distributed*. In the rest of this section all controllers have the ability to be physically distributed.

As we explained in the previous section, one of the defining aspect of SDN is to present a single, unified interface to northbound applications. This property is straightforward when the controller is physically centralized, as it is often pictured, but it has to be maintained when the controller is distributed. Such distributed but seemingly unique controller is called *logically centralized*. However, in [20] and [18] authors define logical centralization as the fact that each controller has the same view of the network and the same responsibilities, it is then mostly a load balancing process. As opposed, they define logical distribution as a situation were each controller has its own view of the network, and its own switches to manage. We believe that this distinction introduces confusion: logical centralization in SDN should only refer - as it is the case in the vast majority of publications - to the appearance of unicity toward the northbound interface, not toward inner controllers in the topology. All the controller architectures that are presented in the remaining of this section are logically centralized, according to our definition.

We however distinguish two types of distributed architectures: flat and hierarchical. In flat architectures all controllers have the same level of responsibility, and they communicate with

1. Ryu github, `https://osrg.github.io/ryu/`

each other through east/west interfaces. In practice, they are either implemented using traditional network protocols such as BGP or PCEP [76], database synchronization protocols, or custom protocols [117]. Authors in [18] propose a dedicated protocol for controller East/West interfaces called Communication Interface for Distributed Control plane (CIDC). Their system can be used on three different modes depending on the situation: notification about events only, advertisement and requests to implement full services, and both. In hierarchical architectures, controllers are layered: the lower layer is responsible for managing the infrastructure layer, and upper layers manage lower layers via controllers' north/south interfaces.

7.2.2 Distribution challenges

Distributed SDN, such as any distributed system, faces the challenge formalized by the CAP theorem [24], which states that controllers cannot archive simultaneously the three following goals:

- consistency (all of them have the same information)
- availability (address requests in a timely manner)
- partition tolerance (ability to recover from temporary partition, and maintain functionality in the meantime)

This means that solutions have to focus on two of those aspects, usually consistency and availability. Indeed, consistency is mandatory to avoid errors in the network, and availability is necessary to meet SLA engagements and avoid dropping packets (when the controller does not respond to switche instruction requests). Consistency gathers special attention, as the CAP theorem ignores one important aspect that must be taken into account in large topologies: latency. SDN architectures have to work around this problem, that may cause two controllers to be inconsistent because the synchronization takes time. To deal with this issue two types of consistency are used:

- strong consistency: controllers wait to be fully consistent before taking a decision regarding any event
- weak / eventual consistency: controllers take decisions based on potentially stale information, but implement mechanisms to converge toward consistency.

In the SDN context, consistency almost always refers to consistency between controller views. However, it is also essential to keep a consistency between the controller view and the actual state of the switches. Ravana [86] proposes to slightly modify the OpenFlow protocol to solve this issue. By implementing acknowledgement messages, allowing buffering, and adding an ID field into OpenFlow messages to identify the controller they originated from, they manage to ensure that switch events and controller commands are processed exactly once. Moreover, a shared log system ensures that events are processed in the same order by controller replicas. Regarding the dilemma between weak and strong consistency, authors in [122] proposed the Simple Coordination Layer (SCL). They point out that strong consistency is difficult to achieve, prone to failures and has a negative impact on availability. However, they note that it is not always useful to reach such level of consistency: indeed, the network is not affected by it, as long as it does not induce errors in the rules. The idea is then to run two levels of consistency in parallel. Weak consistency is used for liveness policies, and strong one is reserved for se-

curity policies. Their application, SCL, runs on top of a classical controller and turns it into a replicated one: each switch is controlled by multiple instances, and a quorum must be reached to perform an action. The objective here is to improve the resiliency, not the scalability. In their example, liveness policies include traffic engineering or shortest path routing, while security policies include isolation or waypointing (ensure that a flow goes through the specified list of middleboxes).

According to [20], flat architectures are more adapted to datacenters, as controllers and switches are very close to each other and can reach strong consistency. On the opposite, a hierarchical architecture is more adapted to wider networks, with numerous switches scattered across a large geographical area that requires abstraction and several levels of management relying on weak consistency. This last architecture is more complex, and in studied solutions it does not exceed two hierarchical levels. We detail some of the most important contributions for both flat and hierarchical architectures.

7.2.3 Flat architectures

B4 [82] was published in 2013 and describes the SDN Openflow architecture implemented by Google to manage the interconnection traffic between their data centers. The main objective of this architecture is to provide fine engineering traffic functions to optimize inter-datacenter flow distribution along the time and to reduce link over-provisioning between them. Parts of the flows are managed with traditional routing distributed functions, but large data transfers are managed using the traffic engineering module hosted in a server on the north of the OpenFlow controllers. They claimed very impressive results with a strong reduction of link over-provisioning. Switches are designed using commodity hardware and controllers are organized in clusters at the edge of each datacenters with a mechanism of election to select a new master controller in case of controller failure. Although implementation and technical aspects are not deeply detailed, this first contribution demonstrated back in 2013 the feasability and the advantages of SDN in an actual, production network. Part of the implementation is based on ONIX, presented below.

In [90] authors present ONIX, a distributed controller whose objectives are:

— generality (wide range of applications accepted),
— scalability, reliability (must handle both its own failures and network equipment failures),
— simplicity (building applications must be easy)
— control plane performance (it should not impede the application, *e.g.,* by introducing too much latency)

To achieve these goals, ONIX is made of two components: the ONIX controller itself, with eventually multiple instances, and on top of them the control logic, that defines the desired behaviour of the network and sends requests to ONIX instances in order to implement this behaviour. The major task of ONIX controllers is to collect, synchronize, and present to the control logic a Network Information Base (NIB) that contains all physical entities (switches) and network entities (tunnels). If the network grows, a single controller instance could run out of memory because of the NIB size, or out of CPU because of the amount of events to process. To avoid this, the control logic partitions the network, and configures each instance of ONIX to be responsible

for one partition, and in particular to maintain the NIB related to this partition updated. To solve scalability issues in terms of both memory shortage and amount of traffic requested for NIB synchronization, each controller presents its own part of the network as a single switch. This is called the aggregation. Lastly, ONIX presents two databases: one strongly consistent, the other weakly consistent. It uses one or the other, depending on the application needs. The problem is that ONIX relies on the application itself to solve potential problems raising from inconsistencies, which partially breaks the logical centralization. ONIX proposes various mechanisms to face failures. In case of controller failure, other controllers are responsible for detecting the problem and sharing the responsibility of the switches. To prevent connectivity failures between controllers multiple paths are established.

While in ONIX all controllers are aware of being part of a distributed architecture, authors in [150] propose another approach. Their application, Hyperflow, can run on top of classical SDN controllers, here NOX, and turn them into a distributed controller without further modification. Each controller believes that it is alone and that it manages the whole network, while it actually controls only a local network. To achieve this, Hyperflow application follows a three step process. First, it discovers other controllers by subscription / advertisement to all Hyperflow applications through a dedicated channel. Second, it captures any incoming local event, such as topology modifications. When those events are considered to have a global impact they are advertised to the other controllers, which believe they were addressed directly to them by the network. This is done by WheelFS, a publish/subscribe mechanism that handles network partitioning. Third, when the local controller emits an OpenFlow order that targets a switch outside of its local network, Hyperflow captures it and redirects it to the correct controller. Tests show that Hyperflow has some difficulties in handling more than 1000 events per second, and that controller synchronization induces delays that may be problematic for the system responsiveness.

The ODL controller was first designed as a standalone one [104] used to perform early demonstrations on SDN benefits. Later releases provided new features to allow distribution [146]. The ODL cluster is based on a distributed datastore, synchronized using the RAFT protocol which guarantees strong consistency. This datastore allows each controller to propose the same network services, and to have the same view of the topology. The controllers in a cluster use heart-beat messages to check if their peers are still up. To load-balance the workload, namely the management of the switches, ODL relies on the master-slave system built in OpenFlow: each switch has one unique master controller that can read and write its state, and several slaves who are read-only. If the master-switch connection fails, or if the master wants to delegate the switch control for load balancing reasons, one of the slave can take over.

The other main open source controller, ONOS [19], was designed with distribution in mind. Similarly to ODL, in ONOS clusters each controller instance is responsible for a set of switches (master). Some other controllers may also be connected to these switches, as slaves. if the master fails, the slaves elect a new master for each one of the switches. Additionally, the cluster can adapt to the workload by scaling in/out, switching off/on instances. One of the main differences with ODL is the management of the shared database: ONOS has multiple stores, linked to the different services it offers. Depending on the nature of the store the consistency is

either strong (realized by Atomix [2], based on RAFT [119]), or weak (based on gossip protocol: every few second a controller picks up randomly another and compares their views, eventually leading to an update). For example, the switch mastership information is strongly consistent.

ODL, ONOS, and many other controllers based on OpenFlow, rely on the build-in master/slave mechanism described above to make connections between the control and infrastructure planes reliable and to archive switch migration for load balancing. The main advantage of this system is to be already functional and easy to implement, however the resulting migration is not disruption free. Authors in [47] propose Elasticon, a controller that focuses on dynamicity and switch migration. Its key features consist in monitoring the controller activity in terms of CPU usage, migrating switches, and eventually growing or shrinking the pool of controllers. When a controller is overloaded, Elasticon first determines which switches send the larger amount of messages. Those switches are the ones which are the more likely to cause the largest amount of processing. It then determines one or several controllers able to receive them. If none can be found a new one is created. Then, Elasticon proceeds to the migration. A disruption-free operation should guarantee the three following properties:

— liveness: at least one controller for each switch at all time and the current controller must end all pending transactions before migration,
— safety: exactly one controller to process every asynchronous message form the switch,
— serializability: the controller processes the events in the order in which they are sent by the switch.

OpenFlow master/slave system violates liveness. Authors propose instead a four step process, which adds additional features to OpenFlow. Elasticon switch migration mechanism is one of the few described in detail in the literature, as most proposed architectures tend to be static.

Dynamicity is also the topic of BalCon [30], an algorithm designed to create clusters of switches to be controlled by the same SDN controller. Their idea is to keep strongly connected switches together, as treating a flow over multiple controller domains increases the overhead. They develop a controller to run their experiments, whose actions can be divided into three steps: controller overload detection, clustering evaluation and modification, switch migration. Through their algorithm they manage to reduce and balance the global charge. However, they do not consider augmenting/shrinking their pool of controllers.

Most of the work studied before tends to focus on datacenter network management, as it could be expected from flat architectures. DISCO [126], on the other side, tackles explicitly wide networks. Like Hyperflow, DISCO is an application installed on traditional controllers, here Floodlight. In intra-domain, each controller is in charge of its own part of the network, using the abilities of the traditional SDN controller DISCO is running on. Regarding inter-domain, DISCO implements four agents in charge of discovering new domains, sending periodic updates of abstracted network status, advertising the connection of a new host to the network and checking the status of inter-domain connections. To communicate together, controllers use a lightweight channel based on Advanced Message Queuing Protocol (AMQP). The reduced bandwidth requested by those communications allows DISCO to operate over wide networks. However, WAN management is mostly addressed by hierarchical architectures.

2. Atomix framework, `https://atomix.io/`

7.2.4 Hierarchical architectures

KANDOO [78] is a hierarchical architecture designed to run over a datacenter. The main concern of the authors is that a lot of local applications report information to the controller, creating delays, overhead, and exhausting controller's capacities. To solve this issue, they introduce a layer of local controllers between the infrastructure and the controller, now referred to as root controller. Those local controllers are able to manage local applications, such as the detection of "elephant flows". The root controller can subscribe to specific notifications that are considered useful for the global network management. Authors present impressive results in terms of amount of data sent to the root controller (compared to a standalone scenario) in a datacenter environment, where many events are local.

In [71], authors introduce ORION. Their architecture is similar to KANDOO's one, except that the layer two controller, called here domain controller, can communicate with other domain controllers in order to address inter-domain scenarios and wide networks. This solution is referred to by the authors as hybrid, as it combines a hierarchical organization between the domain controller and its area controllers, and a flat one between domain controllers. Area controllers are responsible for collecting, abstracting, and transmitting to the domain controller the network topologies they manage. These topologies are aggregated and shared between domain controllers. They are also responsible for managing local events, such as a connection request between two hosts within their network. If a connection requires connecting two hosts from the same domain but different areas, the area controller reports to the domain one, which performs a first path computation based on the abstracted topologies, and delegates the exact implementation to concerned area controllers. If the connection involves several domains the domain controller delegates parts of the path establishment to corresponding peers. This architecture offers both the precision and the abstraction needed to manage large scale networks.

The idea of a controller responsible of the management of a sub-domain in the network and interacting with a master controller is also the base of the Distributed-SDN (D-SDN) architecture proposed in [132]. Although the paper is a general overview of D-SDN, it focuses on the security issues for the control delegation between the main controller and a secondary controller. Authors addressed the security concerns regarding the communications between the two layers of control, and studied the fault tolerance.

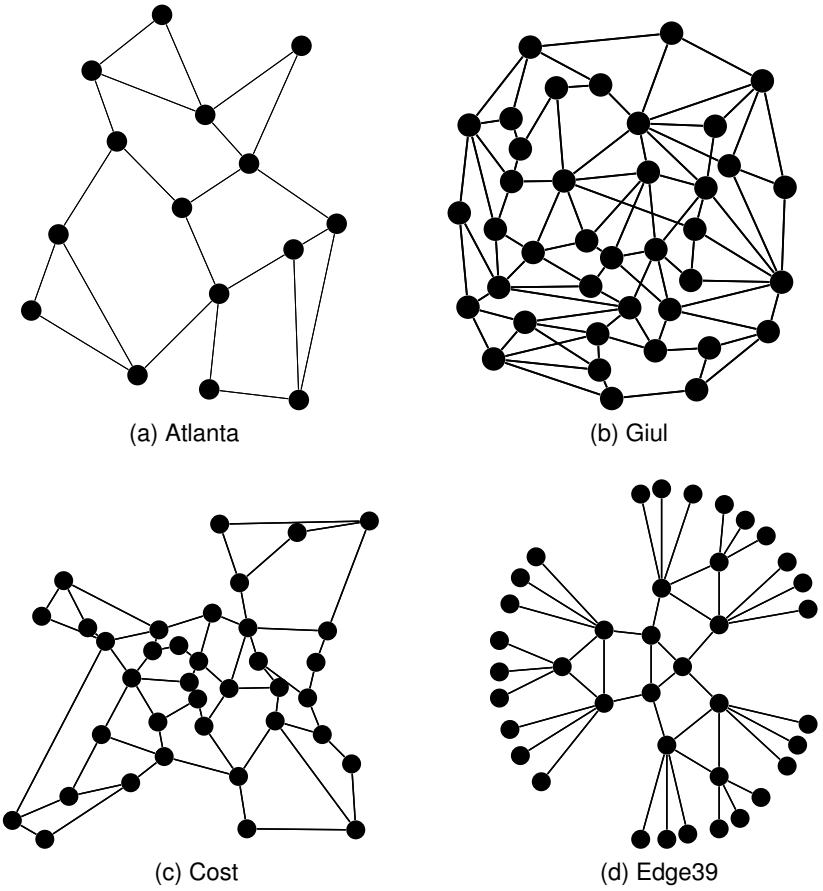
The propositions exposed in Sections 7.2.3 and 7.2.4 detail multi-controller architectures, and some of them further explicit the dynamic adaptation to the workload. However when it comes to geographically extended networks another concern must be solved: the physical location of the different controllers.

7.2.5 Controller placement

In order to be scalable, distributed systems have to allow their pool of controllers to scale in/out depending on the workload. For controllers managing networks deployed over small areas, new controllers placement does not really matter, as they will be all close to their switches anyway. For controllers over WANs, however, delays and extra bandwidth consumption induced by a poorly chosen placement may have a strong impact on the performances.

One of the first paper on this issue was published by Heller *et al.* in [79]. In this paper authors focus solely on latency in WANs. They try to minimize either the average or the worst latency. They show that, for any number of controllers, the latency obtained with K controllers is equal to the latency obtained with one controller divided by K. Consequently, the more controllers are added to the network, the less important is the improvement. Authors end up concluding that, for most of the cases, one well placed controller is actually enough. However, their study does not take into account resiliency, controller workload or bandwidth consumption, which advocate for a better distribution of the control.

Based on this work many other publications have been dedicated to this problem in the past years. Some of them are summarized in [155].



(a) Atlanta (b) Giul

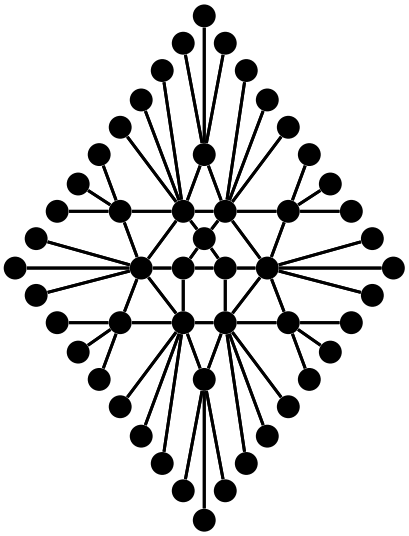
(c) Cost (d) Edge39

Figure 7.1 – Topologies (1/2)

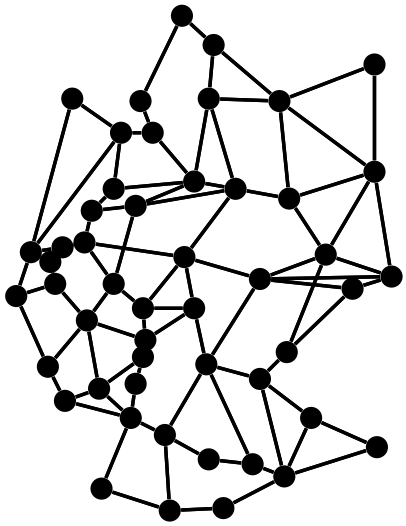
7.3 Topologies

Figures 7.1 and 7.2 represent the different topologies used in the evaluation of the algorithm used to solve the VNFGPP in the mono-tenant scenario, as detailed in Section 4.2.3.1.1. They are all taken from the SDNLib database [3], except for Edge39 and Edge51, that we designed ourselves, following classical designs of edge topologies.

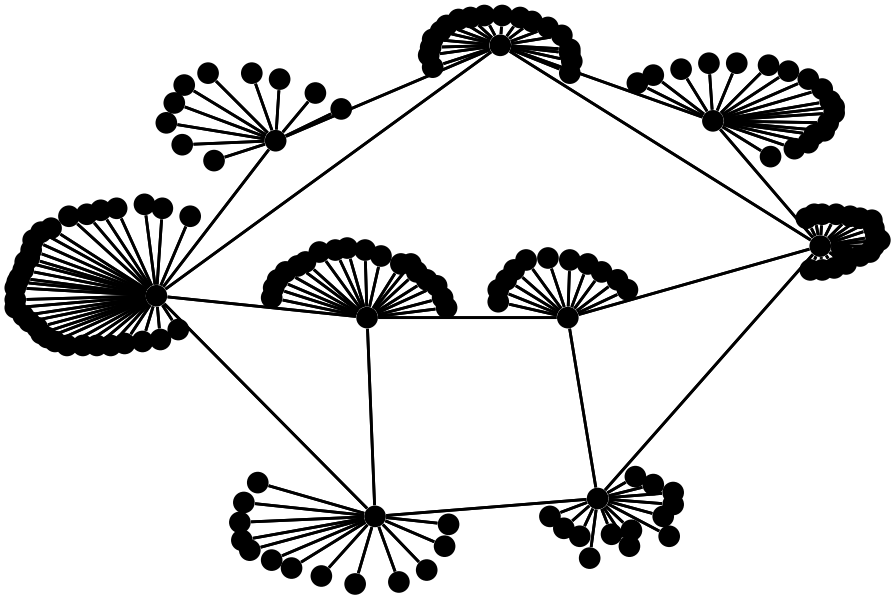
3. http://sndlib.zib.de



(a) Edge51



(b) Germany

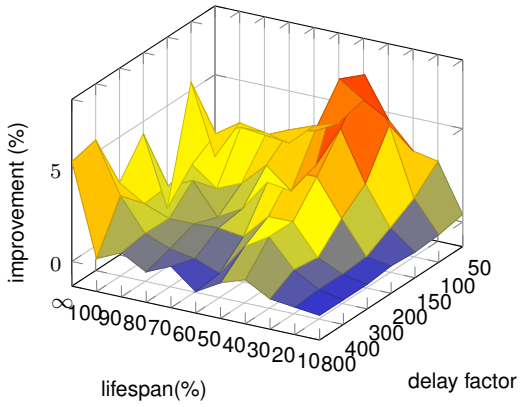


(c) Brain

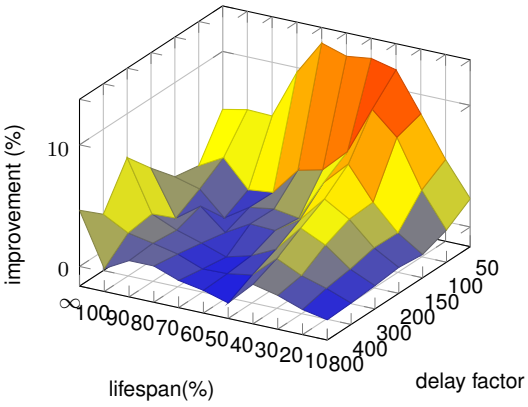
Figure 7.2 – Topologies (2/2)

7.4 Mono-tenant performances - complete results

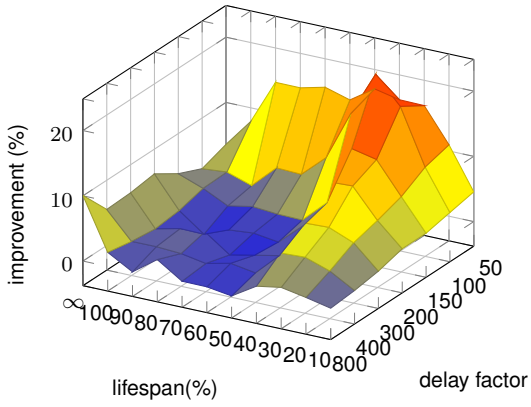
This Section presents the extensive results obtained through the experimental procedure detailed Section 4.2.3.6. The Figures represent the performance of our algorithm, in terms of number of embedded Network Services, compared to a baseline algorithm. The evolution of this performance is tested against two NS-related parameters: the lifespan and the strength of the latency constraints, represented by the delay factor (the smallest the factor, the strongest the constraint). Across the Figure two other parameters are evaluated: resource concentration and topology architecture type.



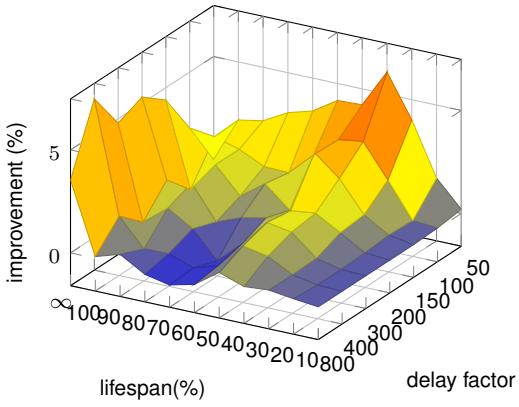
(a) Edge39Low



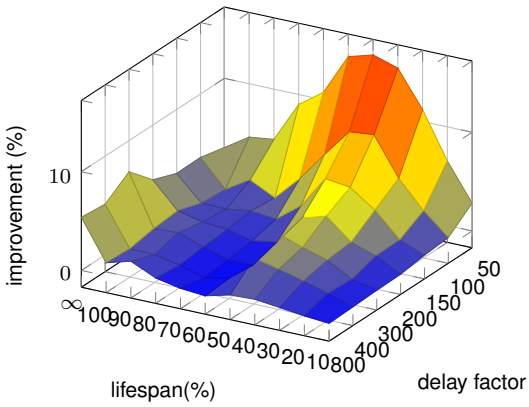
(b) Edge39Medium



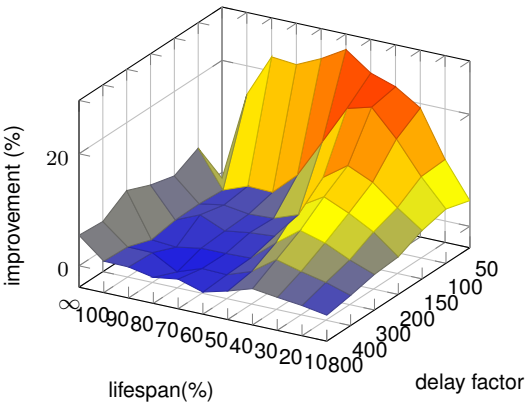
(c) Edge39High



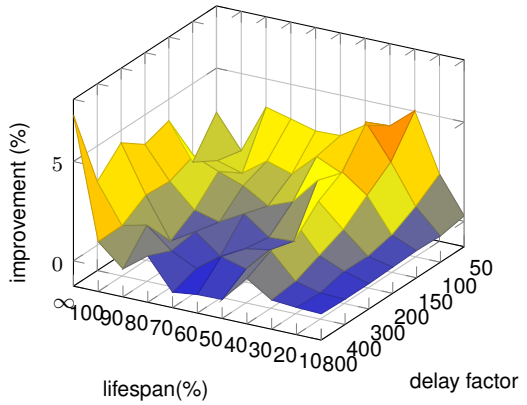
(d) Germany50Low



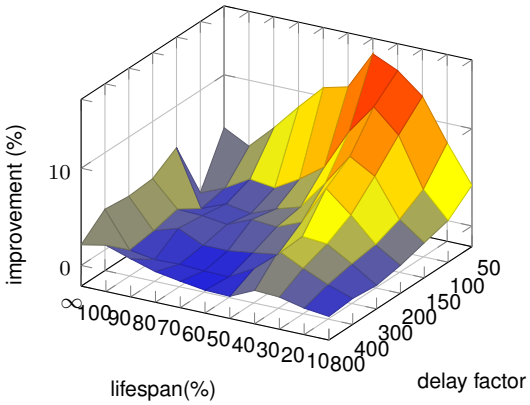
(e) Germany50Medium



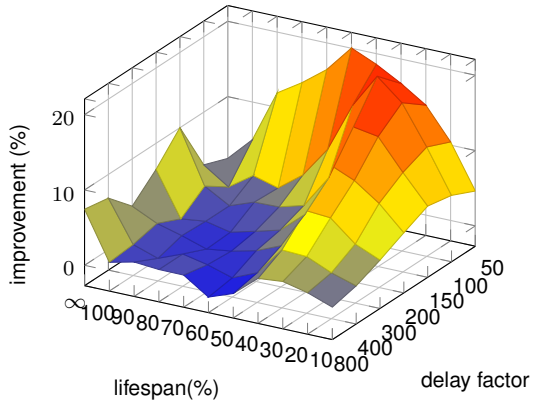
(f) Germany50High



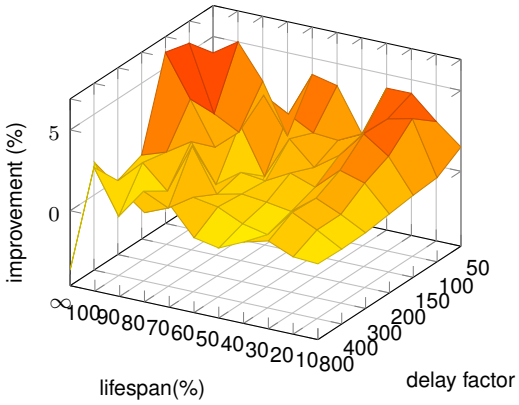
(a) Giul39Low



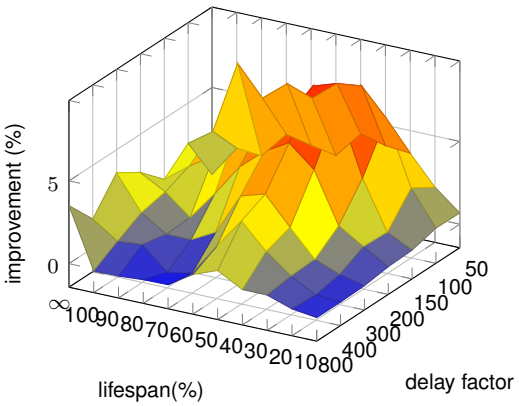
(b) Giul39Medium



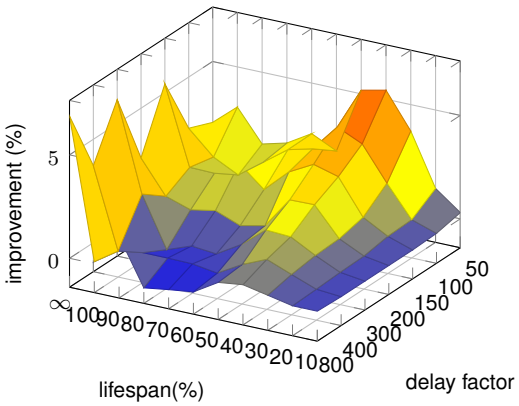
(c) Giul39High



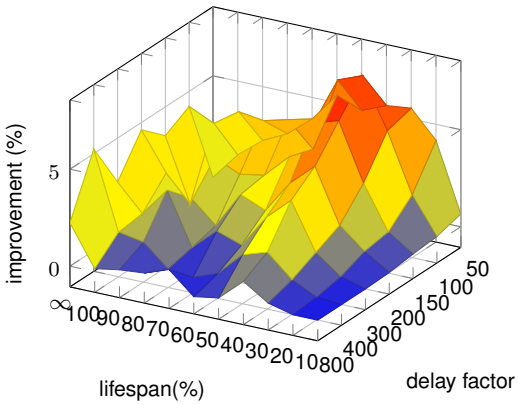
(d) Atlanta



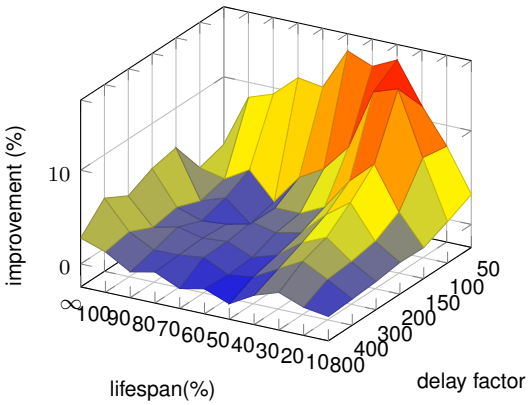
(e) Brain



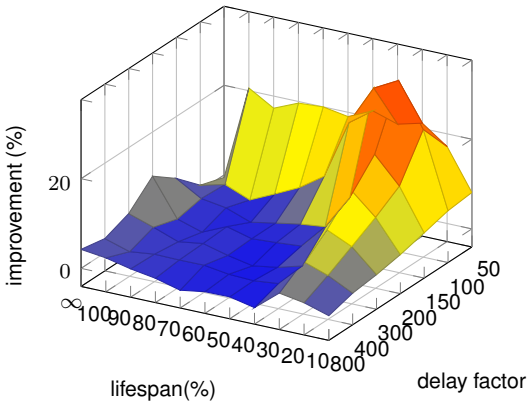
(f) Cost



(a) Edge51Low



(b) Edge51Medium



(c) Edge51High

BIBLIOGRAPHY

[1] « 3GPP TR 28.801 V15.0.0 - Study on management and orchestration of network slicing for next generation network », Dec. 2017.

[2] V. Abhishek, I. A. Kash, and P. Key, « Fixed and market pricing for cloud services », *in*: 2012 Proceedings IEEE INFOCOM Workshops, Mar. 2012, pp. 157–162, DOI: 10.1109/INFCOMW.2012.6193479.

[3] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez, and J. Folgueira, « Automated Network Service Scaling in NFV: Concepts, Mechanisms and Scaling Workflow », *in*: *IEEE Communications Magazine* 56.*7* (July 2018), pp. 162–169, ISSN: 1558-1896, DOI: 10.1109/MCOM.2018.1701336.

[4] B. Addis, D. Belabed, M. Bouet, and S. Secci, « Virtual network functions placement and routing optimization », *in*: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Oct. 2015, pp. 171–177, DOI: 10.1109/CloudNet.2015.7335301.

[5] B. Addis, G. Carello, and M. Gao, « On a virtual network functions placement and routing problem: Some properties and a comparison of two formulations », *in*: *Networks* 75.*2* (2020), pp. 158–182.

[6] B. Addis, M. Gao, and G. Carello, « On the complexity of a Virtual Network Function Placement and Routing problem », *in*: *Electronic Notes in Discrete Mathematics* 69 (Aug. 2018), pp. 197–204, DOI: 10.1016/j.endm.2018.07.026.

[7] A. V. Akella and K. Xiong, « Quality of Service (QoS)-Guaranteed Network Resource Allocation via Software Defined Networking (SDN) », *in*: IEEE, Aug. 2014, pp. 7–13, ISBN: 978-1-4799-5079-9 978-1-4799-5078-2, DOI: 10.1109/DASC.2014.11.

[8] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, P. Bertin, and A. Kerbellec, « On evaluating different trends for virtualized and SDN-ready mobile network », *in*: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), Prague, Czech Republic: IEEE, Sept. 2017, pp. 1–6, ISBN: 978-1-5090-4026-1, DOI: 10.1109/CloudNet.2017.8071534.

[9] O. Alhussein, P. T. Do, J. Li, Q. Ye, W. Shi, W. Zhuang, X. Shen, X. Li, and J. Rao, « Joint VNF Placement and Multicast Traffic Routing in 5G Core Networks », *in*: GLOBECOM 2018 - 2018 IEEE Global Communications Conference, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018, pp. 1–6, ISBN: 978-1-5386-4727-1, DOI: 10.1109/GLOCOM.2018.8648029.

[10] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, and E. Gourdin, « Virtual Function Placement for Service Chaining with Partial Orders and Anti-Affinity Rules », *in*: *Networks* 71.*2* (Aug. 8, 2017), pp. 97–106.

[11] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, « What Will 5G Be? », *in*: *IEEE Journal on Selected Areas in Communications* 32.*6* (June 2014), pp. 1065–1082, ISSN: 0733-8716, DOI: 10.1109/JSAC.2014.2328098.

[12] P. T. Anh Quang, J. Sanner, C. Morin, and Y. Hadjadj-Aoul, « Multi-objective multi-constrained QoS Routing in large-scale networks: A genetic algorithm approach », *in*: 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), Oct. 2018, pp. 55–60, ISBN: 978-1-5386-9493-0, DOI: 10.1109/SaCoNeT.2018.8585634.

[13] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, « A view of cloud computing », *in*: *Communications of the ACM* 53.*4* (Apr. 2010), p. 50, ISSN: 00010782, DOI: 10.1145/1721654.1721672.

[14] O. Arouk, T. Turletti, and N. Nikaein, « Multi-objective placement of virtual network function chains in 5G », *in*: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet), Prague, Czech Republic: IEEE, Sept. 2017, pp. 1–6, ISBN: 978-1-5090-4026-1, DOI: 10.1109/CloudNet.2017.8071537.

[15] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, « PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks », *in*: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Nov. 2013, pp. 1–7, DOI: 10.1109/SDN4FNS.2013.6702548.

[16] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, « A multi-source dataset of urban life in the city of Milan and the Province of Trentino », *in*: *Scientific Data* 2.*1* (Dec. 2015), p. 150055, ISSN: 2052-4463, DOI: 10.1038/sdata.2015.55.

[17] F. Ben Jemaa, G. Pujolle, and M. Pariente, « QoS-Aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture », *in*: GLOBECOM 2016 - 2016 IEEE Global Communications Conference, Washington, DC, USA: IEEE, Dec. 2016, pp. 1–7, ISBN: 978-1-5090-1328-9, DOI: 10.1109/GLOCOM.2016.7842188.

[18] F. Benamrane, M. Ben mamoun, and R. Benaini, « An East-West interface for distributed SDN control plane: Implementation and evaluation », *in*: *Computers & Electrical Engineering* 57 (Jan. 2017), pp. 162–175, ISSN: 00457906, DOI: 10.1016/j.compeleceng.2016.09.012.

[19] P. Berde, W. Snow, G. Parulkar, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, and P. Radoslavov, « ONOS: towards an open, distributed SDN OS », *in*: ACM Press, 2014, pp. 1–6, ISBN: 978-1-4503-2989-7, DOI: 10.1145/2620728.2620744.

[20] O. Blial, M. Ben Mamoun, and R. Benaini, « An Overview on SDN Architectures with Multiple Controllers », *in*: *Journal of Computer Networks and Communications* 2016 (2016), pp. 1–8, ISSN: 2090-7141, 2090-715X, DOI: 10.1155/2016/9396525.

[21] P. Bosshart, G. Varghese, D. Walker, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, and A. Vahdat, « P4: programming protocol-independent packet processors », *in*: *ACM SIGCOMM Computer Communication Review* 44.*3* (July 28, 2014), pp. 87–95, ISSN: 01464833, DOI: 10.1145/2656877.2656890.

[22] M. Boucadair and C. Jacquenet, « Software-Defined Networking: A Perspective from within a Service Provider Environment », RFC7149, RFC Editor, Mar. 2014, DOI: 10.17487/rfc7149.

[23] M. Bouet, J. Leguay, T. Combe, and V. Conan, « Cost-based placement of vDPI functions in NFV infrastructures », *in*: *International Journal of Network Management* 25.*6* (2015), pp. 490–506, DOI: 10.1002/nem.1920.

[24] E. Brewer, « CAP twelve years later: How the "rules" have changed », *in*: *Computer* 45.*2* (Feb. 2012), pp. 23–29, ISSN: 0018-9162, 1558-0814, DOI: 10.1109/MC.2012.37.

[25] I. Bueno, J. I. Aznar, E. Escalona, J. Ferrer, and J. A. Garcia-Espin, « An opennaas based sdn framework for dynamic qos control », *in*: Future Networks and Services (SDN4FNS), 2013 IEEE SDN for, IEEE, 2013, pp. 1–7, ISBN: 978-1-4799-2781-4, DOI: 10.1109/SDN4FNS.2013.6702533.

[26] B. Carpenter and S. Brim, « Middleboxes: Taxonomy and Issues », tech. rep., RFC3234, RFC Editor, Feb. 2002, DOI: 10.17487/rfc3234.

[27] F. Carpio, S. Dhahri, and A. Jukan, « VNF placement with replication for Loac balancing in NFV networks », *in*: 2017 IEEE International Conference on Communications (ICC), May 2017, pp. 1–6, DOI: 10.1109/ICC.2017.7996515.

[28] R. Casellas, R. Martínez, R. Muñoz, R. Vilalta, L. Liu, T. Tsuritani, and I. Morita, « Control and Management of Flexi-grid Optical Networks With an Integrated Stateful Path Computation Element and OpenFlow Controller », *in*: *J. Opt. Commun. Netw.* 5.*10* (Oct. 2013), Invited, A57–A65, DOI: 10.1364/JOCN.5.000A57.

[29] R. Casellas, R. Muñoz, R. Martínez, R. Vilalta, L. Liu, T. Tsuritani, I. Morita, V. López, O. González de Dios, and J. P. Fernández-Palacios, « SDN Orchestration of OpenFlow and GMPLS Flexi-Grid Networks With a Stateful Hierarchical PCE [Invited] », *in*: *Journal of Optical Communications and Networking* 7.*1* (Jan. 2015), A106, ISSN: 1943-0620, 1943-0639, DOI: 10.1364/JOCN.7.00A106.

[30] M. Cello, Y. Xu, A. Walid, G. Wilfong, H. J. Chao, and M. Marchese, « BalCon: A Distributed Elastic SDN Control via Efficient Switch Migration », *in*: 2017 IEEE International Conference on Cloud Engineering (IC2E), Apr. 2017, pp. 40–50, DOI: 10.1109/IC2E.2017.33.

[31] C.-Y. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, « FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN », *in*: ICC 2017 - 2017 IEEE International Conference on Communications, Paris, France: IEEE, pp. 1–7, ISBN: 978-1-4673-8999-0, DOI: 10.1109/ICC.2017.7996632.

[32] B. Chatras, U. S. Tsang Kwong, and N. Bihannic, « NFV enabling network slicing for 5G », *in*: 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Mar. 2017, pp. 219–225, DOI: 10.1109/ICIN.2017.7899415.

[33] Z. Chen, S. Zhang, C. Wang, Z. Qian, M. Xiao, J. Wu, and I. Jawhar, « A Novel Algorithm for NFV Chain Placement in Edge Computing Environments », *in*: GLOBECOM 2018 - 2018 IEEE Global Communications Conference, Abu Dhabi, United Arab Emirates: IEEE, Dec. 2018, pp. 1–6, ISBN: 978-1-5386-4727-1, DOI: 10.1109/GLOCOM.2018.8647371.

[34] M. Chowdhury, M. R. Rahman, and R. Boutaba, « ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping », *in*: *IEEE/ACM Transactions on Networking* 20.*1* (Feb. 2012), pp. 206–219, ISSN: 1063-6692, 1558-2566, DOI: 10.1109/TNET.2011.2159308.

[35] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, « Virtual Network Embedding with Coordinated Node and Link Mapping », *in*: IEEE INFOCOM 2009 - The 28th Conference on Computer Communications, Rio De Janeiro, Brazil: IEEE, Apr. 2009, pp. 783–791, ISBN: 978-1-4244-3512-8, DOI: 10.1109/INFCOM.2009.5061987.

[36] S. Choy, B. Wong, G. Simon, and C. Rosenberg, « The brewing storm in cloud gaming: A measurement study on cloud to end-user latency », *in*: 2012 11th Annual Workshop on Network and Systems Support for Games (NetGames), Venice, Italy: IEEE, Nov. 2012, pp. 1–6, ISBN: 978-1-4673-4578-1 978-1-4673-4576-7 978-1-4673-4577-4, DOI: 10.1109/NetGames.2012.6404024.

[37] « Cisco Visual Networking Index: Forecast and Trends, 2017–2022 », Cisco, 2019.

[38] S. Civanlar, M. Parlakisik, A. M. Tekalp, B. Gorkemli, B. Kaytaz, and E. Onem, « A QoS-enabled OpenFlow environment for Scalable Video streaming », *in*: 2010 IEEE Globecom Workshops, Dec. 2010, pp. 351–356, DOI: 10.1109/GLOCOMW.2010.5700340.

[39] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, « The dynamic placement of virtual network functions », *in*: 2014 IEEE Network Operations and Management Symposium (NOMS), May 2014, pp. 1–9, DOI: 10.1109/NOMS.2014.6838412.

[40] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, « Near optimal placement of virtual network functions », *in*: 2015 IEEE Conference on Computer Communications (INFOCOM), Apr. 2015, pp. 1346–1354, DOI: 10.1109/INFOCOM.2015.7218511.

[41] D. Cooperson and C. Chappell, « Telefónica's UNICA architecture strategy for network virtualisation », 2017, p. 28.

[42] E. Crabbe, I. Minei, J. Medved, and R. Varga, « PCEP Extensions for Stateful PCE », Internet-Draft, Work in Progress, Version 21, Internet Engineering Task Force, Dec. 2017, 54 pp.

[43] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, « DevoFlow: Scaling Flow Management for High-Performance Networks », *in*: *SIGCOMM Comput. Commun. Rev.* 41.*4* (Aug. 2011), pp. 254–265, ISSN: 0146-4833, DOI: 10.1145/2043164.2018466.

[44] J.-E. Dartois, J. Boukhobza, A. Knefati, and O. Barais, « Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization », *in*: *IEEE transactions on cloud computing* 14 (2019), pp. 1–14, DOI: 10.1109/TCC.2019.2898192.

[45] K. Deb and J. Sundar, « Reference Point Based Multi-Objective Optimization Using Evolutionary Algorithms », *in*: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 635–642, ISBN: 1595931864, DOI: 10.1145/1143997.1144112.

[46] S. Denazis, E. Haleplidis, J. H. Salim, O. Koufopavlou, D. Meyer, and K. Pentikousis, « Software-Defined Networking (SDN): Layers and Architecture Terminology », *in*: (2015), RFC7426.

[47] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, « ElastiCon; an elastic distributed SDN controller », *in*: 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Oct. 2014, pp. 17–27, ISBN: 978-1-4503-2839-5.

[48] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, « Microservices: yesterday, today, and tomorrow », *in*: *Springer* (Apr. 20, 2017).

[49] C. A. Duncan, M. T. Goodrich, and S. G. Kobourov, « Balanced Aspect Ratio Trees and Their Use for Drawing Very Large Graphs », *in*: *Graph Drawing*, ed. by S. H. Whitesides, red. by G. Goos, J. Hartmanis, and J. van Leeuwen, vol. 1547, Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 111–124, ISBN: 978-3-540-65473-5 978-3-540-37623-1, DOI: 10.1007/3-540-37623-2_9.

[50] S. Dwarakanathan, L. Bass, and L. Zhu, « Cloud Application HA Using SDN to Ensure QoS », *in*: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD), New York City, NY, USA: IEEE, June 2015, pp. 1003–1007, ISBN: 978-1-4673-7287-9, DOI: 10.1109/CLOUD.2015.137.

[51] A. Dwaraki and T. Wolf, « Adaptive Service-Chain Routing for Virtual Network Functions in Software-Defined Networks », *in*: ACM Press, 2016, pp. 32–37, ISBN: 978-1-4503-4424-1, DOI: 10.1145/2940147.2940148.

[52] D. Erickson, « The Beacon Openflow Controller », *in*: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, Hong Kong, China: Association for Computing Machinery, 2013, pp. 13–18, ISBN: 9781450321785, DOI: 10.1145/2491185.2491189.

[53] « Ericsson - Telefonica. Cloud-RAN architecture for 5G. White paper », 2015.

[54] « ETSI GR NFV-EVE 0018 V0.0.6 - Evolution and Ecosystem; Report on Multi-tenancy in NFV », Work in progress, Apr. 2020.

[55] « ETSI GR NFV-EVE 010 V3.1.1 - Licensing Management; Report on License Management for NFV », *in*: (Dec. 2017).

[56] « ETSI GR NFV-EVE 012 V3.1.1 - NFV Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework », Dec. 2017.

[57] « ETSI GR NFV-IFA 022 V3.1.1 - NFV Management and Orchestration; Report on Management and Connectivity for Multi-Site Services », Apr. 2018.

[58] « ETSI GR NFV-IFA 028 V3.1.1 - NFV Management and Orchestration; Report on architecture options to support multiple administrative domains », Jan. 2018.

[59] « ETSI GS ENI 005 V1.1.1 - Experiential Networked Intelligence (ENI); System Architecture », Sept. 2019.

[60] « ETSI GS NFV-EVE 005 V1.1.1 - NFV Ecosystem; Report on SDN Usage in NFV Architectural Framework », Dec. 2015.

[61] « ETSI GS NFV-IFA 005 V2.1.1 - NFV Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification », Apr. 2016.

[62] « ETSI GS NFV-IFA 009 V1.1.1 - NFV Management and Orchestration; Report on Architectural Options », July 2016.

[63] « ETSI GS NFV-IFA 010 V2.2.1 - NFV Management and Orchestration; Functional requirements specification », Sept. 2016.

[64] « ETSI GS NFV-IFA 011 V3.2.1 - NFV Management and Orchestration; VNF Descriptor and Packaging Specification », Apr. 2019.

[65] « ETSI GS NFV-IFA 029 V3.3.1 - Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS" », Nov. 2019.

[66] ETSI GS NFV-MAN 001, « Network Functions Virtualisation (NFV); Management and Orchestration », *in*: (Dec. 2014), Version 1.1.1.

[67] Fang Hao and E. Zegura, « On scalable QoS routing: performance evaluation of topology aggregation », *in*: IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, Tel Aviv, Israel: IEEE, 2000, pp. 147–156, ISBN: 978-0-7803-5880-5, DOI: 10.1109/INFCOM.2000.832183.

[68] N. Feamster, J. Rexford, and E. Zegura, « The Road to SDN: An Intellectual History of Programmable Networks », *in*: *SIGCOMM Comput. Commun. Rev.* 44.*2* (Apr. 2014), pp. 87–98, ISSN: 0146-4833, DOI: 10.1145/2602204.2602219.

[69] « Final system design and Techno-Economic analysis ».

[70] L. C. Freeman, « A Set of Measures of Centrality Based on Betweenness », *in*: *Sociometry* 40.*1* (Mar. 1977), pp. 35–41, DOI: 10.2307/3033543.

[71] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, and B. Hao, « Orion: A Hybrid Hierarchical Control Plane of Software-Defined Networking for Large-Scale Networks », *in*: IEEE, Oct. 2014, pp. 569–576, ISBN: 978-1-4799-6204-4 978-1-4799-6203-7, DOI: 10.1109/ICNP.2014.91.

[72] J. Gil Herrera and J. F. Botero, « Resource Allocation in NFV: A Comprehensive Survey », *in*: *IEEE Transactions on Network and Service Management* 13.*3* (Sept. 2016), pp. 518–532, ISSN: 1932-4537, DOI: 10.1109/TNSM.2016.2598420.

[73] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand, and L. S. Leong, « Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) based Cloud infrastructure », *in*: 2014 2nd International Conference on Information and Communication Technology (ICoICT), May 2014, pp. 505–510, DOI: 10.1109/ICoICT.2014.6914113.

[74] L. Gurobi Optimization, « Gurobi Optimizer Reference Manual », 2018.

[75] S. Gutz, A. Story, C. Schlesinger, and N. Foster, « Splendid Isolation: A Slice Abstraction for Software-Defined Networks », *in*: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 79–84, ISBN: 9781450314770, DOI: 10.1145/2342441.2342458.

[76] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, and O. Koufopavlou, « Software-Defined Networking (SDN): Layers and Architecture Terminology », RFC7426, RFC Editor, Jan. 2015.

[77] V. B. Harkal and A. Deshmukh, « Software Defined Networking with Floodlight Controller », *in*: *International Journal of Computer Applications* 975 (2016), pp. 23–27.

[78] S. Hassas Yeganeh and Y. Ganjali, « Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications », *in*: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 19–24, ISBN: 9781450314770, DOI: 10.1145/2342441.2342446.

[79] B. Heller, R. Sherwood, and N. McKeown, « The Controller Placement Problem », *in*: *SIGCOMM Comput. Commun. Rev.* 42.*4* (Sept. 2012), pp. 473–478, ISSN: 0146-4833, DOI: 10.1145/2377677.2377767.

[80] N. Huin, B. Jaumard, and F. Giroire, « Optimization of network service chain provisioning », *in*: 2017 IEEE International Conference on Communications (ICC), IEEE, May 2017, pp. 1–7, ISBN: 978-1-4673-8999-0, DOI: 10.1109/ICC.2017.7997198.

[81] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard, « Energy-Efficient Service Function Chain Provisioning », *in*: *Journal of Optical Communications and Networking* 10.*3* (Mar. 1, 2018), p. 114, ISSN: 1943-0620, 1943-0639, DOI: 10.1364/JOCN.10.000114.

[82] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, « B4: Experience with a Globally-Deployed Software Defined Wan », *in*: *SIGCOMM Comput. Commun. Rev.* 43.*4* (Aug. 2013), pp. 3–14, ISSN: 0146-4833, DOI: 10.1145/2534169.2486019.

[83] M. Jiang, M. Condoluci, and T. Mahmoodi, « Network slicing in 5G: An auction-based model », *in*: 2017 IEEE International Conference on Communications (ICC), Paris, France: IEEE, May 2017, pp. 1–6, ISBN: 978-1-4673-8999-0, DOI: 10.1109/ICC.2017.7996490.

[84] Y. Jin, T. Okabe, and B. Sendho, « Adapting Weighted Aggregation for Multiobjective Evolution Strategies », *in*: *Evolutionary Multi-Criterion Optimization*, ed. by E. Zitzler, L. Thiele, K. Deb, C. A. Coello Coello, and D. Corne, red. by G. Goos, J. Hartmanis, and J. van Leeuwen, vol. 1993, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 96–110, ISBN: 978-3-540-41745-3 978-3-540-44719-1, DOI: 10.1007/3-540-44719-9_7.

[85] M. Karakus and A. Durresi, « Quality of Service (QoS) in Software Defined Networking (SDN): A survey », *in*: *Journal of Network and Computer Applications* 80 (2017), pp. 200–218, ISSN: 1084-8045, DOI: 10.1016/j.jnca.2016.12.019.

[86] N. Katta, H. Zhang, M. Freedman, and J. Rexford, « Ravana: Controller Fault-Tolerance in Software-Defined Networking », *in*: Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research, SOSR '15, Santa Clara, California: Association for Computing Machinery, 2015, ISBN: 9781450334518, DOI: 10.1145/2774993.2774996.

[87] C. Kilcioglu and C. Maglaras, « Revenue Maximization for Cloud Computing Services », *in*: *SIGMETRICS* (Nov. 2015).

[88] I. Y. Kim and O. de Weck, « Adaptive Weighted Sum Method for Multiobjective Optimization », *in*: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York: American Institute of Aeronautics and Astronautics, Aug. 30, 2004, ISBN: 978-1-62410-019-2, DOI: 10.2514/6.2004-4322.

[89] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, « Automated and Scalable QoS Control for Network Convergence. » *in*: *INM/WREN* 10.*1* (2010).

[90] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, et al., « Onix: A distributed control platform for large-scale production networks. » *in*: *OSDI*, vol. 10, 2010, pp. 1–6.

[91] T. Korkmaz and M. Krunz, « Source-oriented topology aggregation with multiple QoS parameters in hierarchical ATM networks », *in*: IWQoS'99 - Seventh International Workshop on Quality of Service, London, UK: IEEE, 1999, pp. 137–146, ISBN: 978-0-7803-5671-9, DOI: 10.1109/IWQOS.1999.766488.

[92] T. Korkmaz and M. Krunz, « Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks », *in*: *ACM Transactions on Modeling and Computer Simulation* 10.*4* (Oct. 2000), pp. 295–325, ISSN: 10493301, DOI: 10.1145/369534.369542.

[93] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers, « Providing bandwidth guarantees with OpenFlow », *in*: 2016 Symposium on Communications and Vehicular Technologies (SCVT), Nov. 2016, pp. 1–6, DOI: 10.1109/SCVT.2016.7797664.

[94] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, « A Multi-objective Heuristic for the Optimization of Virtual Network Function Chain Placement », *in*: 2017 29th International Teletraffic Congress (ITC 29), vol. 1, 2017, pp. 152–160.

[95] J. L. Le Roux, « Path computation element (PCE) communication protocol (PCEP) », RFC5440, RFC Editor, 2009.

[96] W. C. Lee, « Topology Aggregation for Hierarchical Routing in ATM Networks », *in*: *SIGCOMM Comput. Commun. Rev.* 25.*2* (Apr. 1995), pp. 82–92, ISSN: 0146-4833, DOI: 10.1145/210613.210625.

[97] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, « Logically Centralized? State Distribution Trade-Offs in Software Defined Networks », *in*: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 1–6, ISBN: 9781450314770, DOI: 10.1145/2342441.2342443.

[98] X. Li and C. Qian, « The virtual network function placement problem », *in*: 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Apr. 2015, pp. 69–70, DOI: 10.1109/INFCOMW.2015.7179347.

[99] P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.-C. Wang, and J. Bi, « Seamless interworking of SDN and IP », *in*: ACM SIGCOMM computer communication review, vol. 43, ACM, 2013, pp. 475–476.

[100] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, « Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions », *in*: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, Canada: IEEE, May 2015, pp. 98–106, ISBN: 978-1-4799-8241-7, DOI: 10.1109/INM.2015.7140281.

[101] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, and L. P. Gaspary, « A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining », *in*: *Computer Communications* 102 (Apr. 2017), pp. 67–77, ISSN: 01403664, DOI: 10.1016/j.comcom.2016.11.002.

[102] A. Mayoral López de Lerma, R. Vilalta, R. Muñoz, R. Casellas, R. Martínez, M. Svaluto Moreolo, J. M. Fàbrega, S. Yi Yan, A. Aguado, E. Hugues Salas, S. Peng, G. S. Zervas, R. Nejabati, D. Simeonidou, J. M. Gran Josa, V. Lopez, O. Gonzalez de dios, J. P. Fernández-Palacios, P. Kaczmarek, R. Szwedowski, T. Szyrkowiec, A. Autenrieth, N. Yoshikane, X. Cao, T. Tsuritani, I. Morita, M. Shiraiwa, N. Wada, M. Nishihara, T. Tanaka, T. Takahara, J. C. Rasmussen, Y. Yoshida, and K.-I. Kitayama, « First experimental demonstration of distributed cloud and heterogeneous network orchestration with a common Transport API for E2E services with QoS », *in*: OSA, 2016, Th1A.2, ISBN: 978-1-943580-07-1, DOI: 10.1364/OFC.2016.Th1A.2.

[103] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, « OpenFlow: enabling innovation in campus networks », *in*: *ACM SIGCOMM Computer Communication Review* 38.*2* (Mar. 31, 2008), p. 69, ISSN: 01464833, DOI: 10.1145/1355734.1355746.

[104] J. Medved, R. Varga, A. Tkacik, and K. Gray, « OpenDaylight: Towards a Model-Driven SDN Controller architecture », *in*: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, June 2014, pp. 1–6, DOI: 10.1109/WoWMoM.2014.6918985.

[105] S. Mehraghdam, M. Keller, and H. Karl, « Specifying and placing chains of virtual network functions », *in*: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), Oct. 2014, pp. 7–13, DOI: 10.1109/CloudNet.2014.6968961.

[106] A. Mendiola, J. Astorga, E. Jacob, and M. Higuero, « A survey on the contributions of Software-Defined Networking to Traffic Engineering », *in*: *IEEE Communications Surveys & Tutorials* (2016), pp. 1–1, ISSN: 1553-877X, DOI: 10.1109/COMST.2016.2633579.

[107] X. Meng, V. Pappas, and L. Zhang, « Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement », *in*: 2010 Proceedings IEEE INFOCOM, Mar. 2010, pp. 1–9, DOI: 10.1109/INFCOM.2010.5461930.

[108] R. Mijumbi, J. Serrat, J. Gorricho, S. Latre, M. Charalambides, and D. Lopez, « Management and orchestration challenges in network functions virtualization », *in*: *IEEE Communications Magazine* 54.*1* (Jan. 2016), pp. 98–105, ISSN: 1558-1896, DOI: 10.1109/MCOM.2016.7378433.

[109] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, « Network Function Virtualization: State-of-the-Art and Research Challenges », *in*: *IEEE Communications Surveys & Tutorials* 18.*1* (2015), pp. 236–262, ISSN: 1553-877X, DOI: 10.1109/COMST.2015.2477041.

[110] H. Moens and F. D. Turck, « VNF-P: A model for efficient placement of virtualized network functions », *in*: 10th International Conference on Network and Service Management (CNSM) and Workshop, Nov. 2014, pp. 418–423, DOI: 10.1109/CNSM.2014.7014205.

[111] C. Morin, G. Texier, C. Caillouet, G. Desmangles, and C. Phan, « Optimization of Network Services Embedding Costs over Public and Private Clouds », *in*: 2020 International Conference on Information Networking (ICOIN), Jan. 2020, pp. 360–365, DOI: 10.1109/ICOIN48656.2020.9016607.

[112] C. Morin, G. Texier, C. Caillouet, G. Desmangles, and C.-T. Phan, « VNF placement algorithms to address the mono- and multi-tenant issues in edge and core networks », *in*: 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019), IEEE, 2019, pp. 1–6.

[113] C. Morin, G. Texier, and C.-T. Phan, « On demand QoS with a SDN traffic engineering management (STEM) module », *in*: 2017 13th International Conference on Network and Service Management (CNSM), IEEE, 2017, pp. 1–6.

[114] « Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action », Oct. 2012.

[115] « NGMN 5G White Paper », Feb. 17, 2015, p. 125.

[116] « OF-CONFIG 1.2 OpenFlow Management and Configuration Protocol », 2014.

[117] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, « Distributed SDN controller system: A survey on design choice », *in*: *Computer Networks* 121 (July 2017), pp. 100–111, ISSN: 13891286, DOI: 10.1016/j.comnet.2017.04.038.

[118] ON.LAB, « Introducing ONOS - a SDN network operating system for Service Providers », 2014.

[119] D. Ongaro and J. Ousterhout, « In search of an understandable consensus algorithm », *in*: 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14), 2014, pp. 305–319.

[120] « OpenFlow Switch Specification v.1.3.2 », May 25, 2013.

[121] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, « Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges », *in*: *IEEE Communications Magazine* 55.*5* (May 2017), pp. 80–87, ISSN: 0163-6804, DOI: 10.1109/MCOM.2017.1600935.

[122] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, « SCL: Simplifying Distributed SDN Control Planes », *in*: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA: USENIX Association, Mar. 2017, pp. 329–345, ISBN: 978-1-931971-37-9.

[123] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, « The Design and Implementation of Open vSwitch », *in*: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), Oakland, CA: USENIX Association, May 2015, pp. 117–130, ISBN: 978-1-931971-218.

[124] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, « Traffic-Aware and Energy-Efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach », *in*: *IEEE Transactions on Services Computing* 13.*1* (Jan. 2020), pp. 172–185, ISSN: 2372-0204, DOI: 10.1109/TSC.2017.2671867.

[125] T. A. Q. Pham, J.-M. Sanner, C. Morin, and Y. Hadjadj-Aoul, « Virtual network function–forwarding graph embedding: A genetic algorithm approach », *in*: *International Journal of Communication Systems* (2019), e4098, DOI: 10.1002/dac.4098.

[126] K. Phemius, M. Bouet, and J. Leguay, « DISCO: Distributed multi-domain SDN controllers », *in*: 2014 IEEE Network Operations and Management Symposium (NOMS), May 2014, pp. 1–4, DOI: 10.1109/NOMS.2014.6838330.

[127] M. Pióro and M. Deepankar, *Routing, Flow, and Capacity Design in Communication and Computer Networks*, Elsevier, 2004, ISBN: 978-0-12-557189-0, DOI: 10.1016/B978-0-12-557189-0.X5000-8.

[128] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed, « Virtual network functions orchestration in wireless networks », *in*: IEEE, Nov. 2015, pp. 108–116, ISBN: 978-3-901882-77-7, DOI: 10.1109/CNSM.2015.7367346.

[129] V. Q. Rodriguez and F. Guillemin, « Towards the deployment of a fully centralized Cloud-RAN architecture », *in*: 2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain: IEEE, June 2017, pp. 1055–1060, ISBN: 978-1-5090-4372-9, DOI: 10.1109/IWCMC.2017.7986431.

[130] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, « Network service chaining with optimized network function embedding supporting service decompositions », *in*: *Computer Networks* 93 (Dec. 2015), pp. 492–505, ISSN: 13891286, DOI: 10.1016/j.comnet.2015.09.035.

[131] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, « Provably efficient algorithms for joint placement and allocation of virtual network functions », *in*: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, May 2017, pp. 1–9, DOI: 10.1109/INFOCOM.2017.8057036.

[132] M. A. S. Santos, B. A. A. Nunes, K. Obraczka, T. Turletti, B. T. de Oliveira, and C. B. Margi, « Decentralizing SDN's control plane », *in*: 39th Annual IEEE Conference on Local Computer Networks, Sept. 2014, pp. 402–405, DOI: 10.1109/LCN.2014.6925802.

[133] S. Scott-Hayward, S. Natarajan, and S. Sezer, « A Survey of Security in Software Defined Networks », *in*: *IEEE Communications Surveys & Tutorials* 18.*1* (2016), pp. 623–654, ISSN: 1553-877X, DOI: 10.1109/COMST.2015.2453114.

[134] J. M. Scott and I. G. Jones, « The ATM Forum's private network/network interface », *in*: *BT Technology Journal* 16.*2* (Apr. 1998), pp. 37–46, ISSN: 1573-1995, DOI: 10.1023/A:1009661714128.

[135] « SDN architecture », Open Networking Foundation, June 2014.

[136] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, « FlowQoS: QoS for the rest of us », *in*: ACM Press, 2014, pp. 207–208, ISBN: 978-1-4503-2989-7, DOI: 10.1145/2620728.2620766.

[137] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, « Design and Implementation of a Consolidated Middlebox Architecture », *in*: Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), San Jose, CA: USENIX, 2012, pp. 323–336, ISBN: 978-931971-92-8.

[138] « Setting the Scene for 5G: Opportunities & Challenges », ITU, 2018.

[139] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, « Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service », *in*: *SIGCOMM Comput. Commun. Rev.* 42.*4* (Aug. 2012), pp. 13–24, ISSN: 0146-4833, DOI: 10.1145/2377677.2377680.

[140] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, and S. Rosenberg, « MDP and Machine Learning-Based Cost-Optimization of Dynamic Resource Allocation for Network Function Virtualization », *in*: 2015 IEEE International Conference on Services Computing (SCC), New York City, NY, USA: IEEE, June 2015, pp. 65–73, ISBN: 978-1-4673-7281-7, DOI: 10.1109/SCC.2015.19.

[141] T. Shimojo, Y. Takano, A. Khan, S. Kaptchouang, M. Tamura, and S. Iwashina, « Future mobile core network for efficient service operation », *in*: 2015 IEEE Conference on Network Softwarization (NetSoft), London, United Kingdom: IEEE, Apr. 2015, pp. 1–6, ISBN: 978-1-4799-7899-1, DOI: 10.1109/NETSOFT.2015.7116190.

[142] A. Singh, M. Korupolu, and D. Mohapatra, « Server-storage virtualization: Integration and load balancing in data centers », *in*: 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA: IEEE, Nov. 2008, pp. 1–12, ISBN: 978-1-4244-2834-2, DOI: 10.1109/SC.2008.5222625.

[143] F. Slim, F. Guillemin, A. Gravey, and Y. Hadjadj-Aoul, « Towards a dynamic adaptive placement of virtual network functions under ONAP », *in*: 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin: IEEE, Nov. 2017, pp. 210–215, ISBN: 978-1-5386-3285-7, DOI: 10.1109/NFV-SDN.2017.8169880.

[144] J. Song and R. Guerin, « Pricing and bidding strategies for cloud computing spot instances », *in*: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA: IEEE, May 2017, pp. 647–653, ISBN: 978-1-5386-2784-6, DOI: 10.1109/INFCOMW.2017.8116453.

[145] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, « Energy Efficient Algorithm for VNF Placement and Chaining », *in*: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Madrid, Spain: IEEE, May 2017, pp. 579–588, ISBN: 978-1-5090-6611-7, DOI: 10.1109/CCGRID.2017.84.

[146] D. Suh, S. Jang, S. Han, S. Pack, T. Kim, and J. Kwak, « On performance of OpenDaylight clustering », *in*: 2016 IEEE NetSoft Conference and Workshops (NetSoft), June 2016, pp. 407–410, DOI: 10.1109/NETSOFT.2016.7502476.

[147] Q. Sun, P. Lu, W. Lu, and Z. Zhu, « Forecast-Assisted NFV Service Chain Deployment Based on Affiliation-Aware vNF Placement », *in*: 2016 IEEE Global Communications Conference (GLOBECOM), Dec. 2016, pp. 1–6, DOI: 10.1109/GLOCOM.2016.7841846.

[148] « The Future of Network Virtualization and SDN Controllers », SDX Central, 2016.

[149] S. Tomovic, N. Prasad, and I. Radusinovic, « SDN control framework for QoS provisioning », *in*: 2014 22nd Telecommunications Forum Telfor (TELFOR), Nov. 2014, pp. 111–114, DOI: 10.1109/TELFOR.2014.7034369.

[150] A. Tootoonchian and Y. Ganjali, « HyperFlow: A distributed control plane for OpenFlow », *in*: Proceedings of the 2010 internet network management conference on Research on enterprise networking, vol. 3, 2010.

[151] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, « Analysis of Topology Aggregation Techniques for QoS Routing », *in*: *ACM Comput. Surv.* 39.*3* (Sept. 2007), 7–es, ISSN: 0360-0300, DOI: 10.1145/1267070.1267071.

[152] P. Van Mieghem and F. A. Kuipers, « Concepts of exact QoS routing algorithms », *in*: *IEEE/ACM Transactions on Networking* 12.*5* (Oct. 2004), pp. 851–864, ISSN: 1558-2566, DOI: 10.1109/TNET.2004.836112.

[153] P. Van Mieghem, « Topology information condensation in hierarchical networks », *in*: *Computer Networks* 31.*20* (Sept. 1999), pp. 2115–2137, ISSN: 13891286, DOI: 10.1016/S1389-1286(99)00066-3.

[154] R. Wallner and R. Cannistra, « An SDN Approach: Quality of Service using Big Switch's Floodlight Open-source Controller », *in*: *Proceedings of the Asia-Pacific Advanced Network* 35.*0* (June 10, 2013), p. 14, ISSN: 2227-3026, DOI: 10.7125/APAN.35.2.

[155] G. Wang, Y. Zhao, J. Huang, and W. Wang, « The Controller Placement Problem in Software Defined Networking: A Survey », *in*: *IEEE Network* 31.*5* (2017), pp. 21–27, ISSN: 0890-8044, DOI: 10.1109/MNET.2017.1600182.

[156] A. P. Wierzbicki, « The use of reference objectives in multiobjective optimization – theoretical implications and practical experience », *in*: *International institute for applied systems analysis* (Aug. 1979), p. 36.

[157] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, et al., « Black-box and Gray-box Strategies for Virtual Machine Migration. » *in*: NSDI, vol. 7, 2007, pp. 17–17.

[158] Xin Li and Chen Qian, « A survey of network function placement », *in*: 2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC), Jan. 2016, pp. 948–953, DOI: 10.1109/CCNC.2016.7444915.

[159] C. Xu, B. Chen, and H. Qian, « Quality of Service Guaranteed Resource Management Dynamically in Software Defined Network », *in*: *Journal of Communications* (2015), ISSN: 23744367, DOI: 10.12720/jcm.10.11.843-850.

[160] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, « HiQoS: An SDN-based multipath QoS solution », *in*: *China Communications* 12.*5* (May 2015), pp. 123–133, ISSN: 1673-5447, DOI: 10.1109/CC.2015.7112035.

[161] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, « Scalable Flow-Based Networking with DIFANE », *in*: Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM '10, New Delhi, India: Association for Computing Machinery, 2010, pp. 351–362, ISBN: 9781450302012, DOI: 10.1145/1851182.1851224.

[162] M. Yu, Y. Yi, J. Rexford, and M. Chiang, « Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration », *in*: *SIGCOMM Comput. Commun. Rev.* 38.*2* (Mar. 2008), pp. 17–29, ISSN: 0146-4833, DOI: 10.1145/1355734.1355737.

ACRONYMS

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

ARP Address Resolution Protocol.

ATM Asynchronous Transfer Mode.

AWS Amazon Web Service.

BGP Border Gateway Protocol.

BSS Business Support System.

CAPEX Capital Expense.

CIR Committed Information Rate.

CIS Container Infrastructure Service.

CPU Central Processing Unit.

CRAN Cloud RAN.

D-SDN Distributed-SDN.

DSCP Differentiated Service Code Point.

EPC Evolved Packet Core.

ETSI European Telecommunications Standards Institute.

GBR Guaranteed Bit Rate.

ILP Integer Linear Programming.

IoT Internet of Things.

IP Internet Protocol.

KPI Key Performance Indicator.

LP Linear Programming.

M2M Machine to Machine.

MANO Management and Orchestration.

MLPOC Multiple Logical Points Of Contact.

MME Mobility Management Entity.

MPLS Multi-protocol Label Switching.

NF Network Function.

NFV Network Functions Virtualisation.

NFVI Network Functions Virtualisation Infrastructure.

NFVO Network Functions Virtualisation Orchestrator.

NGMN Next Generation Mobile Networks.

NIB Network Information Base.

NS Network Service.

NSD Network Service Descriptor.

ODL OpenDaylight.

OF OpenFlow.

ONF Open Networking Foundation.

ONOS Open Network Operating System.

OPEX Operational Expense.

OS Operating System.

OSS Operations Support System.

OVS Open vSwitch.

PCC Path Computation Client.

PCE Path Computation Element.

PCEP Path Computation Element Protocol.

PCP Priority Code Point.

PIR Peak Information Rate.

PNF Physical Network Function.

PNNI Private Network-to-Network Interface.

QoE Quality of Experience.

QoS Quality of Service.

RAM Random Access Memory.

RAN Radio Access Network.

REST Representational State Transfer.

RIB Routing Information Base.

SDN Software Defined Network.

SLA Service Level Agreement.

SLPOC Single Logical Point Of Contact.

SPOF Single Point Of Failure.

STEM SDN Traffic Engineering Management.

TC Traffic Control.

TCAM Ternary Content Addressable Memory.

TDF Telediffusion de France.

TED Traffic Engineering Database.

ToS Type of Service.

UE User Equipment.

VCD Virtual Compute Descriptor.

VDC Virtual Data Center.

VDU Virtual Deployment Unit.

VIM Virtualized Infrastructure Manager.

VL Virtual Link.

VLAN Virtual Local Area Network.

VM Virtual Machine.

vMME virtual Mobility Management Entity.

VNE Virtual Network Embedding.

VNF Virtual Network Function.

VNFC Virtual Network Function Component.

VNFD Virtual Network Function Descriptor.

VNFFG Virtual Network Function Forwarding Graph.

VNFFGD Virtual Network Function Forwarding Graph Descriptor.

VNFG Virtual Network Function Graph.

VNFGPP Virtual Network Function Graph Placement Problem.

VNFM Virtual Network Function Manager.

VSD Virtual Storage Descriptor.

WAN Wide Area Network.

WIM Wide Area Network (WAN) Infrastructure Manager.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Algorithmes pour la mise en œuvre de services réseau avec une qualité de service garantie dans un environnement SDN-NFV

Mot clés : SDN, NFV, MANO, ingénierie de trafic, QoS

Résumé : Le trafic réseau ne cesse de croître et de nouveaux besoins font aujourd'hui leur apparition avec de très fortes contraintes en termes de ressources et de qualité de service. Pour y faire face la 5G propose une centralisation du contrôle et une abstraction des ressources basées sur SDN et NFV ce qui permet une automatisation du déploiement des services, une optimisation de l'usage des ressources et une réduction des coûts. SDN centralise le plan de contrôle du réseau et gère les ressources des liens tandis que NFV virtualise les ressources des nœuds et transforme les fonctions réseau en VNFs. Dans cette thèse, nous analysons et proposons de résoudre certaines difficultés que pose la mise en place de services réseau dans cet écosystème. Nous abordons d'abord la mise en place d'un service de connectivité à qualité de service garantie. Afin d'éviter un afflux de messages vers le contrôleur et un délai dans l'application de la politique d'accès nous transférons son exécution dans le plan de données. Nous proposons ensuite de rajouter des fonctions réseau à nos services. Nous formulons un algorithme de placement visant à épargner les ressources là où elles sont rares pour placer un maximum de services. Nous doublons cet algorithme d'une technique d'abstraction de la topologie du réseau afin de permettre aux entités tierces d'exposer et vendre leurs ressources sans divulguer les détails de leurs infrastructures. Enfin, pour permette à l'opérateur réseau d'acheter les ressources nécessaires à l'établissement de ses services pour un coût minimal nous proposons un algorithme d'optimisation des réservations basé sur les offres des opérateurs de datacenters ainsi que sur des prévisions de trafic.

Title: Algorithms for implementing network services with a guaranteed quality of service in an SDN-NFV environment

Keywords: SDN, NFV, MANO, Traffic Engineering, QoS

Abstract: Network traffic keep growing and new needs are emerging today with strong requirements in terms of resources and quality of service. To face them 5G relies on control plane centralization and abstraction of resources based on SDN and NFV which allow network services deployment automation, resources utilization optimization and cost reductions. SDN centralizes the network control plane and manages link resources while NFV virtualizes node resources and turns network functions into VNFs. In this thesis we analyze some of the difficulties raised by the establishment of a network service in this context and we propose solutions to solve them. We first tackle the setup of a connectivity service with guaranteed quality of service. To avoid a massive flow of monitoring information to the controller and a delay in the enforcement of the policy we transfer the policy's execution to the data plane. Then our goal is to add network functions to our services. We design a placement algorithm that spare resources where they are scarce to save them for services that need them to place a maximum of services. We develop a topology abstraction technique to allow other tenants to expose and sell their resources without disclosing the details of their infrastructures. Finally, to help the network operator to buy the resources required to instantiate its network services at a minimal cost we propose an algorithm that optimizes resource reservations based on cloud operators' commercial offers and network operator's traffic predictions.