Intent-based networking for 5G mobile networks

Fred Kwasi Mawufemor Aklamanu

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAS013

Thèse de doctorat

Intent Based Networking for 5G Mobile Networks

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Réseaux, Information et Communication

Thèse présentée et soutenue à Nozay, le le 5 Novembre 2020, par

FRED KWASI MAWUFEMOR AKLAMANU

Composition du Jury :

Djamal Zeghlache Professeur, Télécom SudParis (RS2M), France	Président
Walter Cerroni Professeur, University of Bologna (Unité de recherche), Italie	Rapporteur
Thi Mai Trang Nguyen Maître de conférences, Sorbonne Université (LIP6), France	Rapporteur
Djamal Zeghlache Professeur, Télécom SudParis (RS2M), France	Examinateur
Véronique Vèque Professeur, Université Paris Sud (RESTEL), France	Examinateur
Hacène Fouchal Professeur, Université de Reims (Unité de recherche), France	Examinateur
Éric Renault Professeur, ESIEE (LIGM), France	Directeur de thèse
Sabine Randriamasy Ingénieur de Recherche, Senior, Nokia Bell Labs (ENSA), France	Co-directeur de thèse

Acknowledgements

First of all, I am grateful to God for His guidance and protection throughout my PhD journey. This will not have been possible without His favor.

I thank my parents and siblings for their relentless prayers, encouragement and also being my support system throughout my fulfilling academic period.

My heart felt gratitude goes to Nokia Bell Labs, France for the opportunity to achieve such an academic dream within one of the prestigious research laboratory. I take the opportunity to sincerely show my appreciation to my research department, ENSA and the CNTP team for their support and guidance in my research adventure.

To my research supervisors, Eric Renault and Sabine Randriamasy, I'm grateful for your relentless supervision and especially patience. This milestone will not have been possible without you.

The last but definitely not the least, my girlfriend, your words of encouragement and moral support is why I am able to accomplish another academic milestone.

To my readers, thank you for taking the time to read my thesis.

Abstract

Mobile networks currently provide an imperative approach to network service provisioning and service life-cycle management. With the rapid technology disruptions, there is a wave which brings on board millions of users, huge data bulks and more complex network infrastructures which an imperative management approach will not scale up with the expected increase in demand. Software Defined Networking (SDN) and Network Function Virtualization (NFV) pave the way for programmability, flexibility and scalability of mobile networks. Both technologies offer significant advantages to the Network Operators (NOs) in terms of network management and service provisioning, which widens their market to 3rd party providers such as Virtual Network Operators (VNOs) and Over-The-Top (OTT) Application Providers. However these technologies still rely on imperative approaches to manage and provision network services.

A declarative approach to network and service management is essential to handle the growth of networks seamlessly, which an Intent-based networking (IBN) approach provides. IBN consists in organizing and abstracting sets of complex network management and configuration instructions so as to expose them to network tenants in the form of a simple and unambiguous service request called Intent. Intents involve the expression of **WHAT** while the network handles the **HOW**. This thesis proposes an Intent-based networking framework for vertical markets with the aim to speed up and simplify the task of network service provisioning and management. The thesis focuses on provisioning 5G network slices using a declarative approach, Intents. The framework aids both operators and network tenants to express their Intents in a high-level language which is close to human language, based on 4th generation language approach and language transformation (source-to-source) tools. The Intent-Based Networking framework is responsible for the end-to-end deployment of 5G network application slices by staging through different service execution phases including, service configuration, resource allocation, identifying optimal service placement strategy and service lifecycle monitoring without human intervention after Intent expression

Titre : Réseaux de 5ème génération basés sur des intentions

Mots clés : Réseau 5G, Virtualisation, Software Defined Networks (SDN), Intention

Résumé : Les réseaux mobiles utilisent actuellement une approche impérative pour la fourniture de services réseaux et la gestion du cycles de vie des services. Les sauts de technologie qui accompagnent la 5G vont attirer des millions de nouveaux utilisateurs et d'énormes volumes de données. Plus Les infrastructure de réseaux atteindront une complexité telle qu'une gestion en mode impératif ne pourra pas suivre la hausse escomptée en demandes de services. Les technologies Software Defined Networking (SDN) et Network Function Virtualisation (NFV) tracent la route pour la programmabilité, la flexibilité et l'évolutivité des réseaux mobiles. Les deux technologies offrent un avantage significatif aux Opérateur de réseau (NOs) en terme de gestion de réseaux et de fourniture de services, et élargissent leur marché aux fournisseurs tiers tels des opérateurs de réseau virtuels (VNOs) et des fournisseurs d'application Over-The-Top (OTT). Cependant, ces technologies reposent toujours sur des approches impératives de gestion et de fourniture de services réseaux. Une approche déclarative pour la gestion des réseaux services est nécessaire pour gérer leur accroissement du réseau de manière transparente, ce qu'offre une approche de réseautage basé sur l'intention (IBN). L'IBN consiste à organiser et à abstraire des ensembles d'instructions complexes de gestion et de configuration de réseaux afin de les exposer aux locataires du réseau sous la forme d'une demande de service simple et sans ambiguïté appelé Intention. L'intention décrit **QUOI** on demande tandis que le réseau gère **COMMENT** y répondre. La présente thèse propose un cadre de traitement basé sur les Intentions pour le traitement des requêtes par les marchés verticaux. L'étude se concentre sur l'approvisionnement des tranches de réseau 5G dédiées à des applications. La structure aide à la fois les opérateurs et les locataires du réseau à exprimer leur intention dans un langage de 4eme génération proche du langage humain et en langage de transformation (source-à-source)

Title : Intent-Based Networking for 5G Mobile Networks

Keywords : 5G Networks, Virtualisation, Software Defined Networks (SDN), Intent

Abstract : Mobile networks currently provide an imperative approach to network service provisioning and service life-cycle management. With the rapid technology disruptions, there is a wave which brings on board millions of users, huge data bulks and more complex network infrastructures which an imperative management approach will not scale up with the expected increase in demand. Software Defined Networking (SDN) and Network Function Virtualization (NFV) pave the way for programmability, flexibility and scalability of mobile networks. Both technologies offer significant advantages to Network Operators (NOs) in terms of network management and service provisioning, which widens their market to 3rd party providers such as Virtual Network Operators (VNOs) and Over-The-Top (OTT) Application Providers. However these technologies still rely on imperative approaches to manage and provision network services.
A declarative approach to network and service management is essential to handle the growth of networks seamlessly, which an Intent-based networking (IBN) approach provides. IBN consists in organizing and abstracting sets of complex network management and configuration instructions so as to expose them to network tenants in the form of a simple and unambiguous service request called Intent. Intents involve the expression of **WHAT** while the network handles the **HOW**. This thesis proposes an Intent-based networking framework for vertical markets with the aim to speed up and simplify the task of network service provisioning and management. The thesis focuses on provisioning 5G network slices using a declarative approach, Intents. The framework aids both operators and network tenants to express their Intents in a high-level language which is close to human language, based on 4th generation language approach and language transformation (source-to-source) tools. The Intent-Based Networking framework is responsible for the end-to-end deployment of 5G network application slices by staging through different service execution phases including, service configuration, resource allocation, identifying optimal service placement strategy and service lifecycle monitoring without human intervention after Intent expression.

Institut Polytechnique de Paris
91120 Palaiseau, France

Resumé

Les réseaux mobiles utilisent actuellement une approche impérative pour la fourniture de services réseaux et la gestion du cycles de vie des services. Les sauts de technologie qui accompagnent la 5G vont attirer des millions de nouveaux utilisateurs et d'énormes volumes de données. Plus Les infrastructure de réseaux atteindront une complexité telle qu'une gestion en mode impératif ne pourra pas suivre la hausse escomptée en demandes de services. Les technologies Software Defined Networking (SDN) et Network Function Virtualisation (NFV) tracent la route pour la programmabilité, la flexibilité et l'évolutivité des réseaux mobiles. Les deux technologies offrent un avantage significatif aux Opérateur de réseau (NOs) en terme de gestion de réseaux et de fourniture de services, et élargissent leur marché aux fournisseurs tiers tels des opérateurs de réseau virtuels (VNOs) et des fournisseurs d'application Over-The-Top (OTT). Cependant, ces technologies reposent toujours sur des approches impératives de gestion et de fourniture de services réseaux. Une approche déclarative pour la gestion des réseaux services est nécessaire pour gérer leur accroissement du réseau de manière transparente, ce qu'offre une approche de réseautage basé sur l'intention (IBN). L'IBN consiste à organiser et à abstraire des ensembles d'instructions complexes de gestion et de configuration de réseaux afin de les exposer aux locataires du réseau sous la forme d'une demande de service simple et sans ambiguïté appelé Intention. L'intention décrit **QUOI** on demande tandis que le réseau gère **COMMENT** y répondre. La présente thèse propose un cadre de traitement basé sur les Intentions pour le traitement des requêtes par les marchés verticaux. L'étude se concentre sur l'approvisionnement des tranches de réseau 5G dédiées à des applications. La structure aide à la fois les opérateurs et les locataires du réseau à exprimer leur intention dans un langage de 4eme génération proche du langage humain et en langage de transformation (source-à-source)

Cette thèse est organisée comme suit:

Chapitre 2 - État de l'art

Ce chapitre commence par un examen approfondi des Software Defined Networks et de leurs cadres basés sur l'intention. Les différents modules des architectures SDN IBN Framework font également l'objet d'une étude. Les défis liés aux cadres SDN pour la gestion de réseau sont également abordés. Le chapitre se poursuit par une plongée dans le paradigme de virtualisation de la fonction réseau (NFV) pour la gestion et l'orchestration des services. De plus, les cadres IBN open source, la recherche dans le milieu universitaire et le point de vue de l'industrie sur l'IBN sont également étudiés. Le chapitre approfondit également nos travaux de recherche sur l'IBN.

Dans ce chapitre, nous avons échangé et mis en évidence différentes technologies de pointe pour conduire l'IBN. Nous avons commencé par une enquête sur les IDN qui a mis en évidence le besoin de réseaux de configuration, d'autogestion et d'auto-optimisation pour l'avenir. Cette enquête clarifie également les définitions relatives aux IDN. Le chapitre s'est poursuivi avec les tendances en matière de technologie de réseau en introduisant le concept de mise en réseau définie par logiciel en tant que catalyseur pour la programmabilité et la gestion du réseau. De plus, nous avons étudié les contrôleurs SDN, et leur rôle pour les futurs réseaux. Deux contrôleurs SDN open source, ONOS et ODL ont été étudiés avec leurs architectures. Malgré les avantages du SDN, nous avons réalisé que la configuration et le réglage des paramètres réseau nécessitent le personnel qualifié. De plus, les réseaux compatibles SDN nécessitent une reconfiguration fréquente, car la taille du réseau augmente. Nous avons examiné le NFV et la manière dont il rend les réseaux flexibles, agiles et évolutif pour piloter la programmabilité du réseau. L'architecture NFV MANO de l'ETSI a été étudiée également. Les différentes couches liées à l'architecture NFV MANO ont été mises en évidence et la flexibilité qu'il offre dans la création de services à travers la chaîne de fonctions de service. Si l'architecture MANO offre une architecture découplée, ce qui manque une telle architecture est un critère de placement intelligent pour les VNF. L'architecture MANO considère uniquement la disponibilité des ressources réseau pour le placement VNF dans les réseaux d'opérateurs.

Aucune autre stratégie de placement de réseau n'est envisagée pour les VNF. La stratégie de placement réseau sera essentielle dans les réseaux 5G pour pouvoir accueillir différents services de réseau et leurs SLA associés. Outre la prise en charge de différents services réseau, le coût d'exploitation des centres de données doit être pris en compte lors du placement du service réseau. Nous avons également étudié les solutions IBN existantes couvrant les implémentations open source, solutions industrielles et recherche académique. Nous avons commencé avec les modules de réseautage basé sur l'intention dans les contrôleurs ONOS et ODL SDN. Le fonctionnement du cycle de vie IBN pour aider à la programmabilité et la gestion du travail de manière agnostique ont également été étudiés. Un autre projet IBN de l'ONF, Boulder, qui vise à fournir la portabilité Intent à travers différents contrôleurs SDN pour éviter le verrouillage de l'intention du contrôleur SDN. Le projet Boulder a été intégré à la fois à l'exécution d'ONOS Intent et à la carte réseau ODL avec l'introduction d'une nouvelle couche, couche d'exécution Open Intent au-dessus de la couche Intent des deux ONOS et contrôleurs ODL SDN. Bien que le projet Boulder propose une couche à intégrer plusieurs moteurs d'intention, peu de travail a été fait sur ce projet et peu d'informations sont disponibles sur le projet. Les implémentations open source, spécialement ONOS et ODL, sont adaptés aux réseaux optiques et ne fournissent que la connectivité Intent.

Enfin, les produits industriels de Nokia, Huawei, Cisco et Apstra dans l'écosystème du réseautage basé sur l'intention a été étudié. Ces entreprises en sont aux premières étapes en vue de fournir des réseaux entièrement automatisés, car les produits IBN récents ne sont pas déployés pour prendre en charge les opérations réseau de bouts en bout. Ce qui manque dans l'état de l'art actuel, c'est une solution pour prendre en charge la fourniture de services pour les applications verticales. Dans les réseaux mobiles 5G, la prise en charge des applications verticales sera indispensable en raison des nombreux services qu'il apporte. Le processus de fourniture de ces services devraient être entièrement automatisés pour réduire les interventions humaines dans les réseaux.

Chapitre 3 - Structure IBN pour les tranches d'application réseau Over-The-Top

Ce chapitre présente les contributions à la mise en réseau basée sur l'intention pour le réseau mobile 5G. Le cadre présente plusieurs modules nécessaires pour une approche réseau basé sur l'intention pour la fourniture de services réseau et la gestion du cycle de vie. Nous tirons parti du Software Defined Networking (SDN) et la virtualisation des fonctions réseau (NFV) pour le déploiement de services. Le cadre proposé fournit une abstraction de réseau à partir du réseau les locataires de travail pour aider à exprimer les intentions de manière déclarative. Une preuve de concept a été développée pour la tranche d'application réseau 5G IoT Over-The-Top (OTT) à l'aide du cadre IBN proposé.

Nous proposons un cadre de mise en réseau basé sur l'intention pour gérer certains de ces défis et ensuite nous échangeons sur les différents modules présents dans le cadre de l'architecture IBN. De plus, nous évaluons notre cadre IBN avec un Proof-of-Concept (PoC) pour un déploiement de tranche d'application réseau IoT 5G.

Notre cadre de mise en réseau basé sur l'intention aide à combler le fossé technologique entre SDN et NFV en fournissant une plate-forme unifiée. Notre cadre s'étend au-delà du réseau connectivité pour fournir des services d'application de réseau OTT, dans notre cas une tranche de réseau 5G à travers la coordination avec l'interface IBAS qui se trouve au-dessus de la technologie SDN et NFV. Nous fournissons également une expression d'intention simplifiée qui résume les aspects techniques du réseau des locataires du réseau. Cette approche permet de réduire les contrôles de vérification constants entre les locataires du réseau et les propriétaires d'infrastructure pour déterminer si tous les exigences de service sont remplies. La cartographie de l'intention accélère le processus d'identification de l'intention et aussi fournit les ressources nécessaires sans l'intervention des infrastructures propriétaires ou exploitants. Notre expérience, PoC identifie certains avantages clés du cadre:

Processus automatisé à partir de la demande d'intention au déploiement du service avec un minimum d'interaction humaines. Une boucle de rétroaction continue sur les intentions et la surveillance des SLA pour aider à respecter le réseau sortie d'intention souhaitée par le locataire.

Une fonction de calendrier qui offre la flexibilité de planifier des demandes d'intention répétitives.

Nous démontrons que le réseautage basé sur l'intention va au-delà de la connectivité réseau en tirant parti de la technologie NFV pour créer un réseau automatisé, flexible et agile. Nous présentons un algorithme qui joue un rôle essentiel dans l'identification des infras-nœuds de structure dans le processus de déploiement d'intention et montre la stratégie pour minimiser la consommation d'énergie et de ressources dans les réseaux des opérateurs.

Chapitre 4 - Utilitaire et algorithme A-Star (UA*) pour la tranche de réseau Over-The-Top placement

Nous avons présenté un nouvel algorithme, UA* pour le chaînage des tranches de réseau et le placement sur des topologies aléatoires et datacenter. UA* est robuste pour prendre en charge différentes tailles de tranches de réseau et optimise également l'utilisation des ressources au sein d'une infrastructure ou d'un domaine d'opérateur. La fonction utilitaire robuste normalise et rend les différentes ressources et connectivité vecteurs comparables. UA* diminue le nombre de nœuds utilisés dans le déploiement de la tranche de réseau comme par rapport aux méthodes de pointe. L'avantage est une consommation d'énergie réduite dans le réseau infrastructure des opérateurs et centres de données. Nous avons évalué notre algorithme sur des graphes aléatoires et une topologie de centre de données qui n'est pas étudiée dans beaucoup d'écrits. Enfin, nous montrons également l'impact de différentes stratégies de placement des tranches de réseau sur les ressources réseau, qui n'a pas été abordé dans la littérature.

Chapitre 5 - Conclusion et perspective La thèse se termine par un résumé des contributions et présente quelques perspectives de travaux futurs.

Pour les futures opportunités de recherche ou l'orientation dans le cadre de l'IBN, il existe encore un certain nombre de domaines qui peuvent être améliorés. Une syntaxe de langage normalisée pour IBN est un élément important dans lequel la recherche peut être améliorée. En raison de la diversité des équipements réseau fournisseurs et leurs langages d'équipement propriétaires, un langage IBN plus unifié est nécessaire. Ceci est particulièrement important, car cela facilitera la tâche d'une intention abstraite expression. De plus, une approche de traitement du langage naturel peut ouvrir la voie à une meilleure expression d'intention. Les opérateurs de réseau ou les propriétaires d'infrastructure

peuvent utiliser les données de configuration du réseau à des fins d'apprentissage des langues. Dans le contexte du mobile 5G réseaux, l'ontologie pourrait être une autre approche pour unifier les différents domaines du réseau c'est-à-dire RAN, transports et réseaux centraux. Néanmoins, un Machine Learning pourrait être utilisé pour aider au dimensionnement des ressources pour les tranches de réseau. Ce sera un avantage supplémentaire pour l'IBN. La stratégie de déploiement des tranches d'application OTT peut être encore améliorée en considérant différentes classes de locataires du réseau en termes de hiérarchisation de leurs demandes de services. Cela contribuera à améliorer la qualité de service pour les locataires de réseau privilégiés sur les réseaux. De plus, la migration des tranches de réseau peut être intégrée à notre algorithme UA* pour déplacer certains VNF d'une tranche sur différents nœuds d'infrastructure pour un meilleur service et une meilleure rentabilité. Les nœuds d'infrastructure peuvent être classés plus en détail basé sur les fonctionnalités matérielles et mappé au type de tranche de réseau pour affiner davantage le sous-graphique pour le placement des tranches de réseau. Enfin, différents protocoles de communication devraient être étudiés pour faciliter les échanges d'informations entre les contrôleurs SDN et les orchestrateurs NFV qui sont au centre d'IBN.

Contents

List of Figures

List of Tables

Abbreviations

SLA	**S**ervice **L**evel **A**greement
ONF	**O**pen **N**etworking **F**oundation
SDN	**S**oftware **D**efined **N**etworks
NFV	**N**etwork **F**unction **V**irtualisation
OPeX	**O**perational **E**xpenditure
CAPEX	**C**apital **E**xpenditure
SBI	**S**outh **B**ound **I**nterface
CRAN	**C**loud **R**adio **A**ccess **N**etwork
MVNO	**M**obile **V**irtual **N**etwork **O**perator
IoT	**I**nternet **o**f **T**hings
M2M	**M**achine **2**to **M**achine
eMBB	**e**nhanced **M**obile **B**road **B**and
IBN	**I**ntent **B**ased **N**etworking
TXL	**T**uring **X**extender **L**anguage
UA*	**U**tility **A***-Star
OTT	**O**ver **T**he **T**op
ACL	**A**ction **C**ontrol **L**ist
IP	**I**nternet **P**rotocol
GPA	**G**raph **P**olicy **A**bstraction
GBP	**G**roup **B**ased **P**olicy
API	**A**pplication **P**rogrammable **I**nterface
ONOS	**O**open **N**etwork **O**perating **S**ystem
ODL	**O**open **D**ay **L**ight
HSS	**H**ome **S**ubscriber **S**erver
MME	**M**obility **M**anagement **E**ntity

P-GW	**P**acket **G**ate **W**ay
S-GW	**P**acket **S**erving **W**ay
PNF	**P**hysical **N**etwork **F**unction
SFC	**S**erve **F**unction **C**haining
MANO	**MAN**agment **O**rchestration
ESTI	**E**uropean **T**elecommunication **S**tandard **I**nstitute
VIM	**V**NF **I**nfrastructure **M**anager
VNFM	**V**NF **M**anager
NFVO	**V**NF **O**rchestrator
IDN	**I**ntent **D**riven **N**etwork
IETF	**I**nternet **E**ngineering **T**ask **F**orce
ENI	**E**xperimental **N**etwork **I**ntelligence
YANG	**Y**et **A**nother **N**ext **G**eneration
ONF	**O**pen **N**etworking **F**oundation
CLI	**C**ommand **L**ine **I**nterface
GUI	**G**raphical **U**ser **I**nterface
SB	**S**outh **B**ound
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
NIC	**N**etwork **I**nternet **C**omposition
QoS	**Q**uality **o**f **S**ervice
WORA	**W**rite **O**nce **R**un **A**nywhere
VPN	**V**irtual **P**rivate **N**etwork
FCAPS	**F**ault **C**onfiguration **A**ccounting **P**erformance **S**ecurity
DSL	**D**omain **S**pecific **L**anguage
NEMO	**NE**twork **MO**deling
MDA	**M**odel **D**riven **A**rchitecture
DNA	**D**igital **N**etwork **A**rchitecture
AOS	**A**pstra **O**perating **S**ystem
SDK	**S**oftware **D**evelopment **K**it
PoC	**P**roof **o**f **C**oncept
IBAS	**I**ntent **B**ased **A**pplication **S**licing
COASP	**C**loud **O**ver-The-Top **A**pplication **S**licing **P**latform
3GL	**3**rd **G**enerational **L**anguage

BNF	**B**ackus **N**aur **F**orm
CPU	**C**entral **P**rocessing **U**nit
CPU	**C**entral **P**rocessing **U**nit
FEU	**F**ront **E**nd **U**nit
EC	**E**dge **C**loud
CC	**C**entral **C**loud
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
VM	**V**irtual **M**achine
EPC	**E**volve **P**acket **C**ore
ILP	**I**nteger **L**inear **P**rogramming
MILP	**M**ixed **I**nteger **L**inear **P**rogramming
IDN	**I**ntent **D**riven **N**etwork

List of Abbreviations

3GL 3rd Generational Language.

ACL Action Control List.

ANI Autonomic Networking Intelligence.

AOS Apstra Operating System.

API Application Programmable Interface.

BNF Backus Naur Form.

CAP Capital Expenditure.

CCC Central Cloud.

CLI Command Line Interface.

COA Cloud Over-The-Top Application Slicing Platform.

CPU Central Processing Unit.

CRA Cloud Radio Access Network.

DNA Digital Network Architecture.

DSL Domain Specific Language.

ECC Edge Cloud.

eMB enhanced Mobile Broad Band.

ENI Experimental Network Intelligence.

EPC Evolve Packet Core.

EST European Telecommunication Standard Institute.

FCA Fault Configuration Accounting Performance Security.

FEU Front End Unit.

GBP Group Based Policy.

GPA Graph Policy Abstraction.

GUI Graphical User Interface.

HSS Home Subscriber Server.

HTT Hyper Text Transfer Protocol.

IBA Intent Based Application Slicing.

IBN Intent Based Networking.

IDN Intent Driven Network.

IET Internet Engineering Task Force.

ILP Integer Linear Programming.

IoT Internet of Things.

IPP Internet Protocol.

M2M Machine 2to Machine.

MAN MANagment Orchestration.

MDA Model Driven Architecture.

MIL Mixed Integer Linear Programming.

MME Mobility Management Entity.

MVN Mobile Virtual Network Operator.

NEM NEtwork MOdeling.

NFV Network Function Virtualisation.

NIC Network Internet Composition.

ODL Oopen Day Light.

ONF Open Networking Foundation.

ONO Oopen Network Operating System.

OPe Operational Expenditure.

OTT Over The Top.

PNF Physical Network Function.

PoC Proof of Concept.

QoS Quality of Service.

SBB South Bound.

SBI South Bound Interface.

SDK Software Development Kit.

SDN Software Defined Networks.

SFC Serve Function Chaining.

SLA Service Level Agreement.

SNM Simple Network Management Protocol.

TXL Turing Xextender Language.

UA* Utility A*-Star.

VIM VNF Infrastructure Manager.

VMM Virtual Machine.

VNF VNF Manager.

VPN Virtual Private Network.

WOR Write Once Run Anywhere.

YAN Yet Another Next Generation.

Chapter 1

Introduction

Mobile Networks have experienced a rapid evolution due to the growing demand of network users and furthermore the new horizon of different network services as a result of innovative technologies. The next generation mobile network 5G, opens opportunities for different vertical market players to be tenants on network operators' infrastructure. The concept of network slicing is a key enabler for 5G mobile networks involving allocation of dedicated isolated physical or virtual resources as an end-to-end service for network infrastructure tenants and vertical market players. Network Operators will open up part of their infrastructure to vertical market tenants for example autonomous vehicle players, energy and agricultural sectors. Tenants will like to enter requests that specify a desired tenant application with a given quality level in terms that are agnostic to both the infrastructure technology and network operation expertise. Likewise, network services may also be requested at a higher level by operation engineers. The infrastructure will have to process these requests so as to translate them in a set of adequate network operation instructions. This involves, checking the request validity, mapping it against available network services, checking its feasibility in terms of timing and resources and lastly compiling it in a set of system commands. All these processing steps will require a lot of effort from network operators or infrastructure owners to manage the various services and their life cycle. To sustain the expected large amount of incoming service requests, management of the networks will require some programmability and automation to reduce human intervention. Network programmability, renders networks flexible, agile for better configuration and management. This will play an important role in mobile networks like 5G.

Software Defined Networking (SDN)[3] is a network programmable approach, providing a centralized control of infrastructure network equipment for configuration, monitoring and management. SDN allows network administrators to remotely control several network equipment from a central point because of the separation of the control plane from the data plane in network equipment. Additionally, Network Function Virtualisation (NFV) brings flexibility and agility to networks. NFV aids network operators, infrastructure owners and vertical markets to scale their services faster and efficiently. NFV contributes to the reduction of the infrastructure Operational Expenditure (Opex). NFV isolates network functions from network equipment hence provides the possibility to scale many network functions on the same piece of network equipment. These technologies will be game changers in mobile networks as a result of the numerous advantages they offer over traditional networks. SDN as a technology provides a physical separation of the control and data plane of networking equipment thus providing a centralized control for configuration and management. This approach accommodates the co-existence of multi-vendor network equipment through standardized and open South Bound Interfaces (SBIs) [4]. NFV offers the opportunity to place different virtual network functions on the same piece of hardware or in a distributed manner instead of dedicating a piece of hardware to a specific network function. Today, different virtual network functions can be deployed for example in a Cloud Radio Access Network (CRAN) architecture [5]. Properly leveraging on both SDN and NFV will help network operators, Mobile Virtual Network Operators (MVNO) and other infrastructure tenants to reduce both their Capital Expenditure (CAPEX), Operational Expenditure (OPEX) and maintain a scalable, flexible infrastructure to meet growing network traffic demands. The infrastructure may thus timely and reliably deliver a variety of network services such as video streaming, online gaming and measurement collection from connected sensors for example in healthcare or agriculture (IoT) domains.

1.1 Background and Challenges

Though SDN and NFV addresses some of the concerns about network service management, some issues remain unresolved by these innovative technology paradigms. Despite the emergence of these technologies, mobile networks continue to rely heavily on human know-how for network service provisioning and network management. Network

engineers continue to intervene on network service life cycle management. Network operators or infrastructure owners spend enormous time in network service planning, verification and deployment. These human efforts are sometimes prone to errors and contribute to degradation in the quality of network services and cost penalties for infrastructure owners and network operators. Network administrators are required to master configuration syntax for different vendor equipment on the network. Furthermore, different network policy rules are required to be defined for known and perhaps unknown network events. This raises lots of concerns because these policies turn to be static and will require reconfiguration depending on network activities and evolution.

What lacks in today's networks is a fully automated network service provisioning and management system. Today's networks require lots of human effort for provisioning different network services. These tasks require highly skilled network administrators, enormous amounts of time for equipment configuration and an execution process. The process is also highly erroneous as prior and post evaluation of the network services and management is not fully automated. This is a major setback for existing networks and so would it be for future mobile networks. Mobile network operators or infrastructure owners who want to outsource part of their infrastructure will need to provide a fully automated system. They will have to deliver different kinds of 5G network services for different network services. Network services are required to meet several Service Level Agreements (SLAs). An estimated 30 billion connected devices are expected on 5G networks across different network slices like Internet of Things (IoT), Machine-to-Machine (M2M) communication, autonomous cars, extreme mobile broadband (eMBB) and critical machine communication [6].

One way to address the above mentioned problems related to human intervention is to automate the processing of incoming requests, from the analysis of their expression to the generation of system commands. The application specific and network technology agnostic expressions of service request is left to the tenant while the network expertise and know-how to deliver the service is left to the infrastructure. The infrastructure would open an interface to a framework that automatically and reliably receives, processes and serves tenant requests. This approach is known as Intent-Based Networking, where an Intent is a network-agnostic request to deliver the connectivity and resources needed to deploy a given application. The intent-based networking approach is paving

way for a new networking business landscapes and to address the problem of network service management. End-user communication devices such as smartphones, tablets and private PCs have become widespread as they started interacting with their users via pleasant applications and commands written in common languages. As a consequence, the application business has been blooming at the expense of the network infrastructure operations. We believe that one way to couple applications and network equipment growth is to offer a user-friendly way to use and manage the network infrastructure and network services. Moreover, 5G network management involves an increased part of NFV and Software Defined Network Control (SDN-C). Exposing network operations and services in this context will require powerful abstraction mechanisms and Intent-Based Networking (IBN) [7] will make this possible. IBN consists in organizing and wrapping sets of complex network management and configuration instructions so as to expose them to network users in the form of a simple and unambiguous service request called an Intent. An Intent expresses what is requested, for example, "Connect Host A and Host B" while leaving the *how to achieve* it to the network management. The term network users here covers network operation engineers wanting simplicity and robustness as well as their customers, tenants or applications, wanting simple and reliable access to network connectivity services. Cloud Radio Access Networks (CRAN) are being designed to reduce Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) for network operators. A CRAN splits the base station functions into distributed remote radio heads working as antennas and a pool of centralized virtual baseband units running in data centers [2]. It involves multiple networks, technologies, efficient NFV and SDN-C. This further amplifies the need for a reliable Intent interface sitting on top of coordinated multi-technology network controllers and able to expose wireless network services while hiding their complexity. Such a tool does not exist yet and motivates our research. IBN state-of-art restricts to fixed connectivity for network applications and low level policy configurations.

The research on Intent-Based Networking is novel with limited research work and with an incomplete research background. Our research work is exploratory and first academic results are now prevailing in this domain. Our research work on IBN proposes a unified framework for network service provisioning adapting a declarative approach. The essence of a declarative approach is to automate and accelerate service provisioning and facilitate network management. Since network operators and infrastructure owners will offer part

of their networks to vertical tenants, a simplified and non-technical approach is required to help network tenants easily place their Intents.

1.2 Research Contribution

The objective of this thesis is to address the aforementioned challenges of network infrastructure and network service life cycle management. Our research primarily focuses on **automating network service provisioning and lifecycle management**, **Intent language expressions** and **network service placement in infrastructure network**. The fore-mentioned challenges have been addressed by the following contributions.

Software Defined Networking (SDN) and Network Function Virtualisation (NFV) have been used to help addressing the challenges of IBN. Our IBN research work has been conducted in the context of SDN and NFV. We extend the IBN scope into three directions:

- We extend the usual IBN scope into three directions
  - From fixed to mobile networks
  - From SDN controlled to SDN and NFV supported service delivery
  - From low-level applications to tenant-level vertical application
- To address the problem of a non-existent declarative method for IBN network service provisioning, a conceptual framework for Intent-Based Networking for 5G Mobile Networks has been proposed which aims at exposing network services and facilitate the task of service provisioning and life cycle management for network application slices. This framework has been implemented and a Proof-of-Concept carried out for an IoT network application slice use-case.
- In this contribution, we tackle the problem of Intent language expression as there is no formal Intent language. An extensive study has been done on fourth generation languages with an emphasis on translation languages, a subset of transformation languages. A conceptual language has been proposed for Intent expression for networks. A study of the Turing eXtender Language (TXL) for language translation from a close-to-human language to a low-level network equipment language

has been carried out. We also discuss some of the challenges related to these translation languages.

- We address the problem of network function placement and network resource optimisation in network operator networks or infrastructure network. We have defined a method for network slice placement and chaining algorithm, Utility and A-Star (UA*) for network application slices. We have then implemented this method and evaluated its theoretical performances on large topologies. UA* integrates within the proposed IBN Framework to enable an automated and resource aware placement of network application slices.
- An extension to the proposed IBN Framework has been made for a fine-grain network application slice and microservice energy monitoring to aid network operators and infrastructure owners for billing and identify potential energy saving opportunities within their infrastructure.

1.3 Outline of Dissertation

This dissertation is organized as follows:

Chapter 2 – State of the Art
This chapter starts with an extensive review on Software Defined Networks and their Intent-based Frameworks. The different modules of SDN IBN Framework architectures are also discussed. Challenges related to SDN Frameworks for network management are addressed as well. The chapter continues with a dive into the Network Function Virtualisation (NFV) paradigm for service management and orchestration. Additionally, open source IBN frameworks, research in academia and industry perspective on IBN are discussed. The chapter also positions our research work on IBN.

Chapter 3 – IBN Framework for Over-The-Top Network Application Slices
This chapter introduces contributions to Intent-Based Networking for 5G Mobile Networks. The framework introduces several modules necessary for an Intent-based networking approach for network service provisioning and life cycle management. We leverage on Software Defined Networking (SDN) and Network Function Virtualisation (NFV) for service deployment. The proposed framework provides network abstraction from network tenants to help express Intents in a declarative manner. A Proof-of-Concept has

been developed for 5G IoT Over-The-Top (OTT) Network Application Slice using the proposed IBN framework.

Chapter 4 – Utility and A-Star (UA*) Algorithm for OTT Network Slice Placement

We present an algorithm for network-slice application placement within network infrastructure. The algorithm aims to enhance the network service provisioning using a strategic placement policy for network slices. Furthermore, UA* helps to reduce the operational expenditure (OPEX) by minimising the number of servers used to host network slices.

Chapter 5 – Conclusion and Future Research

The dissertation is concluded with a summary of the contributions and some perspectives for future research are discussed.

Chapter 2

State of the Art

The chapter begins with a recent survey on emerging applications for Intent-Driven Networks (IDN) and research on IDN. The survey reviews existing technologies, clarifies definitions and summarises features for IDN. Basic IDN architecture and key technologies of IDN are discussed as well. We then take a dive into different enablers for Intent-Based networking for network service provisioning and network management. We present the subject of Intent-based Networking (IBN) and also highlight some industrial players within the ecosystem of IBN. Next, we look at how an Intent-based networking approach is vital to handle the growth of network users and also automate the process of service delivery. Network users or tenants here refer to Over-The-Top (OTT) market players or vertical markets who will have a share of a network operators' infrastructure to place different network Intents. In the absence of programmable networks, network administrators will continue to manually configure network equipment and react to changing network states.

We also discuss IBN in academia, standardisation, industry and open source community. In conclusion, we discuss some of the challenges related to these technologies to achieve a fully automated service and network management system.

2.1 Networking Technology Trends and IDN Survey

In a recent work [8], authors provide a comprehensive survey on Intent-Driven Networks (IDN). This survey is one of the first that gives a general overview of the scope of

Intent-based networking. The survey looks into existing technologies like SDN and NFV, clarifies definitions and summarises features of IDN. The survey highlights our research work on IBN for 5G networks. In addition, diversity gains and challenges are analyzed briefly. In this survey, existing technologies, including SDN and NFV, are discussed with their primary advantages being low-cost and efficient operation of network construction. Despite the advantages of these technologies some technical challenges are exposed by the survey as following:

- **High complexity and security issues**: High flexibility software often implies high complexity of the program. In SDN architecture, the system needs to maintain multiple logical networks simultaneously while applications share resources and their functions do not impact one another. The SDN controller of an SDN network may have security problems, such as overload, single point of failure, and vulnerability to network attacks.

- **Separation of service and network**: The future networks will not only support traditional communication services but also provide compatibility for a large number of emerging industries. The trend of services is gradually shifting from core applications based on existing technologies to new technologies and applications in the vertical industry. However, existing networks use a single network function to support multiple application services, which makes it difficult to provide a customized network. The separation of application service and network cannot realize the timely network delivery, thereby deteriorating quality of service and quality of experience [9].

- **Manual and error-prone network configuration**: Networking equipment have numerous parameter settings which are configured manually, including conflict and consistency analysis. However, configuration failures are still inevitable and the expense of configuring network precisely are high. Therefore, to achieve self-managed networks and automatic configuration are still quite challenging.

These challenges related to existing technologies drive the need for a more robust, intelligent and self-aware technology, the need for an Intent-Driven Network. The survey goes on to define IDN as an intelligent network, which can automatically convert, verify, deploy, configure, and optimize itself to achieve target network state according to the

intent of the operators, and can automatically solve abnormal events to ensure the network reliability. IDN provides a full-life cycle management for network elements under the condition of collecting network events.

2.2 IBN in standardisation

IBN has found its voice in the standardisation bodies like the Internet Engineering Task Force (IEFT) and European Telecommunication Standards Institute (ETSI). The Network Management Research Group (NMRG) under the IETF drives standardisation work on IBN. NMRG defines terminology for IBN. They have also defined some taxonomy related to IBN [10]. NMRG has drawn an IBN tool suite from both academia and industry [1] and this includes our research work on IBN for network slice deployment. The IBN tool suite consists of identifying IBN lifecycle processes. Figure 2.1, shows some identified processes and work-in-progress. IETF also works on an automated network management and autonomic networking integrated model and approach (ANIMA). The aim of ANIMA is to improve network management efficiency by using a YANG data modeling language to model configurations [8]. Another working group, Industry Specification Group (ISG) under ETSI have also defined a system called Experimental Network Intelligence (ENI). ENI is a policy driven module that understands network configuration and takes actions according to network changes. Aside ENI, IETF is also researching zero touch network and service management.

The IEFT also defines an internet draft [11] that describes a prototype implementation of Socket Intents for the Berkeley Software Distribution (BSD) Socket API as an illustrative example on how Socket Intents can be implemented. It described the experiments made with the prototype and lessons learned from trying to extend the BSD Socket API. Socket Intents allow an application to express what it knows, assumes, expects or wants to prioritize regarding its own network communication. This information can then be used by the Operating System(OS) to perform destination selection, path selection and transport protocol stack instance selection. Socket Intents prototype for the BSD Socket API is a first attempt to automate transport option selection within the OS. It is primarily targeted at path and destination address selection and tries to be as close as possible to the semantics of the BSD Socket API. The prototype mostly excludes the problem of transport protocol stack instance selection.

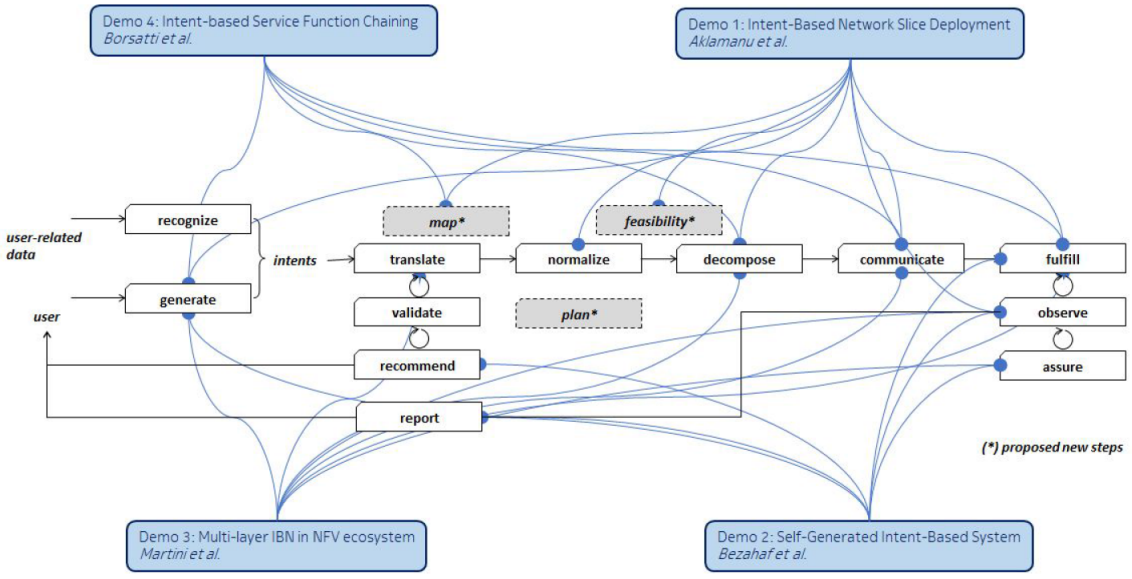


FIGURE 2.1: IBN tools suite (source: [1])

Traffic Management (TM) Forum introduces OPEN APIs with focus on service catalog, service ordering and service inventory. TMF provides a Service Catalog Management API REST specification which allows the management of the entire lifecycle of the Service Catalog elements and the consultation of service catalog elements during several processes such as ordering process. This includes all the processes required to design, build, deploy, maintain and ultimately retire the catalog element [12]. TMF provides specification on service inventory, a REST API allowing intents to provide a consistent mechanism to query and manipulate a service inventory [13]. Services in the service inventory can be retrieved using filtering criteria. The REST API for Service Order Management provides a standardised mechanism for placing a service order with all of the necessary order parameters. It allows users to create, update and retrieve Service Orders and manages related notifications. [14].

2.3 Policy-Based Network Management

Majority of today's networks and services are driven and managed with policies. These are imperative instructions which network operators or infrastructure owners use to instruct the network on what to do in different situations. Networks can also be driven by business policies. Business policies have to do with applications respecting different Service Level Agreements (SLAs). Often, policy design requires highly skilled experts

to come up with different Access Control Lists (ACLs) to react to network changes. As networks grow, these ACLs need to be re-evaluated by experts and re-designed to meet network growth.

Raymer et al. [15] propose a policy information model, DEN-ng, providing different set of views for different constituencies in a policy consortium. The views have their own set of grammar and terminologies. The aim of the informational model is to encircle the entire network and its operational environment.

An N-State policy framework is proposed in [16] for the aggregation of network events into few manageable states. Authors in this work map current and desired network events to these aggregated network states. Their solution which is based on finite state machines helps to frame PBNM policies.

Also in [17], a set of toolkits based on the Ponder language [18] for specification, deployment and management of policies is proposed. The toolkit comprises a policy compiler, used to generate the implementation code for heterogeneous management and security platforms, a hyperbolic tree viewer for efficient manipulation of the domain structure and effective navigation across the domains.

Works in [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] and [29] all provide abstraction policies for network management. However, their policy expressions are tied to low-level specific IP addresses, MAC addresses and switch ports. These frameworks are more tailored towards the infrastructure owners who have better knowledge of the infrastructure than network tenants.

A Graph Policy Abstraction (GPA) approach is introduced in [30] for network configuration using a high-level policy language. The work also tackles detection and resolution of network policy conflicts leveraging on a graph structure. GPA models and composes service chaining policies as well.

IBN has found grounds in academia, with lots of interest in automation frameworks and northbound Intent interfaces. In [31], network requirements are expressed as network policies using an intent-based modeling abstraction. Authors in [32]

mercian2017indira, propose a service description framework where Intent semantics are constructed in a graph structure. This is done through natural language processing for creation of required network services by interaction with different nodes within a graph

structure. IBN has been extended in [33] to the security domain to provide security services in a multi-layer IP network, Ethernet and optical networks. Other research works like [34], [35], and [36] have focused on intent translation, from natural language expressions into network configurations for network service deployment.

2.4 Group-Based Policy for OpenStack

Group-Based Policy (GBP) is a framework in OpenStack to manage network infrastructure through a declarative policy abstraction driven by the open community. GBP framework promotes network automation and security. GBP provides an intent-driven model that presents simplified application-oriented interfaces to network users. The concept of GBP is based on the following key principles;

- **Group**: This has to do with entities within the infrastructure with the same characteristics, a set of endpoints. They also have the same policies.
- **Reusable policy rule sets**: This provides connectivity relationship between different Groups on how they communicate. The connectivity here involves layer 2 and 3 functions, switching and routing respectively. The rule sets are reusable for new created groups.
- **Policy layering**: Allows for different role policies in the network. Layering allows policies to be applied to applications and likewise infrastructure.
- **Network services**: This feature allows connecting multiple services from layer 4 to layer 7 in a sequence known as service chaining. Services in this layer includes firewalls and load balancers.

GBP works in a single administrative domain meaning they do not extend to different network domains. For example a GBP cannot be globally applied from RAN to core network in a mobile network.

2.5 Software Defined Networking

SDN is the new architectural paradigm for networks which involves the physical separation of the responsibilities of the control plane from the data plane[1]. The goal is to provide a centralised control plane to help the management of physical or virtual devices in the network compared to the traditional approach. With traditional approach, where network devices are distributed with both control and data planes on a single equipment.

SDN provides lots of flexibility for network equipment configuration and as well as network management. Since all equipment in the network can be configured remotely from a centralised control, network administrators save enormous amount of time compared to traditional approaches. Additionally, monitoring of the network equipment is much easier for network operators or network administrators to react to network changes through reconfiguration. SDN provides a global view of the network infrastructure topology and a single point for network configuration. With SDN, network operators and administrators do not need to bother about vendor equipment lock-in as different vendor equipment can co-exist in the network and communicate with a single or multiple SDN controllers. This simplifies network design and planning for network operators and administrators. Figure 2.2 shows a logical view of the SDN architecture.

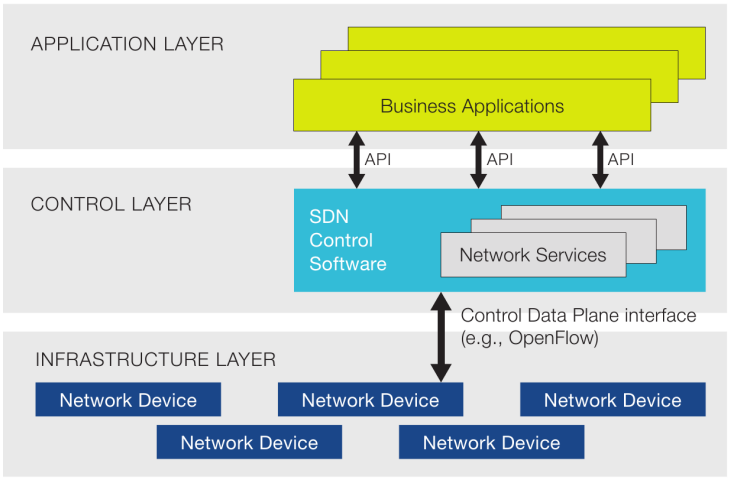


FIGURE 2.2: SDN Architecture

- **Application Layer:** This layer is exposed to network tenants to take advantage of SDN services through a NorthBound API. Network application developers can

equally take advantage of the exposed NorthBound APIs to programmatically control the networks.

- **Control Layer:** Co-ordinates the different request from the application and infrastructure layer. The control layer serves as an intermediary for providing abstract information of the infrastructure to the application. This abstraction helps not to expose the network to security vulnerabilities.
- **Infrastructure Layer:** This layer comprises of various network equipment ranging from network switches and routers. It receives instruction from the application through the control layer for equipment configuration and management.

Network operators and administrators can programmatically configure networks rather than adapting traditional methods for equipment configuration and network management. Furthermore, leveraging on SDN as a technology, network behavior can be changed in real-time and provisioning of new network services can be done in few hours rather than days to meet network service growth.

Moreover, network administrators can write network policies and get them deployed on network devices instead of waiting for vendor equipment specific features which involves extra cost and time.

SDN technology provides network abstraction by providing support for Application Programmable Interfaces (APIs) making it possible to implement network services. Some of these services include routing, access control, bandwidth management, traffic engineering, quality of service, processor and storage optimization, energy usage, and policy management [3]. Finally, SDN architecture extends to both wired and wireless technologies.

2.5.1 SDN Controllers

An SDN controller serves as the intelligence for an SDN enabled network. Network equipment are configured and policies are setup with the SDN controller and deployed onto network devices. New network devices can easily be on-boarded into the network as configurations are centralised by the SDN controller rendering networks scalable and flexible. There are quite a number of open source SDN controllers which provide different

features and modules. Some of these SDN controllers provide an Intent Based Network module which is detailed in section 2.7.1. Open Network Operating System (ONOS) [37] and Open Daylight (ODL) [38] are more matured open source SDN controllers which have an Intent Based Network module integrated.

2.6 Network Function Virtualisation

NFV is another technology driver to make mobile networks more agile for automation and facilitate management. The concept NFV is to decouple network functions from dedicated hardware rendering them as software based network functions. These network functions are also known as Virtual Network Function (VNF). The virtualisation of network components as functions in a mobile network can for example include, Home Subscriber Server (HSS) [39], Mobility Management Entity [39], Packet and Serving Gateway (P-GW and S-GW) [39] provides the opportunity to scale VNFs easily compared to dedicated physical network functions (PNF). NFV will immensely support the growth of network services and the diverse variety of new network services through the concept of network slicing [40] which is envisioned for 5G mobile networks. Different NFVs can be composed together, known as Service Function Chaining (SFC) [41] to provide different network services. NFV also helps in cost reduction, both CAPEX and OPEX in datacenters or network operator networks. Figure 2.3, provides a high-level of NFV Management and Orchestration (MANO) [2] reference architecture proposed by European Telecommunication Standard Institute (ESTI).

The NFV MANO architecture consists in three function layers:

- VNF Infrastructure Manager (VIM): The layer is responsible for the management of both physical and virtual resources in a NFV Infrastructure (NFVI) domain. These resources controlled by VIM include compute or storage.
- VNF Manager (VNFM): VNFM controls the lifecycle management of VNFs of the same type or different types. VNFM are restricted to domains thus multiple domains will have different VNFMs.

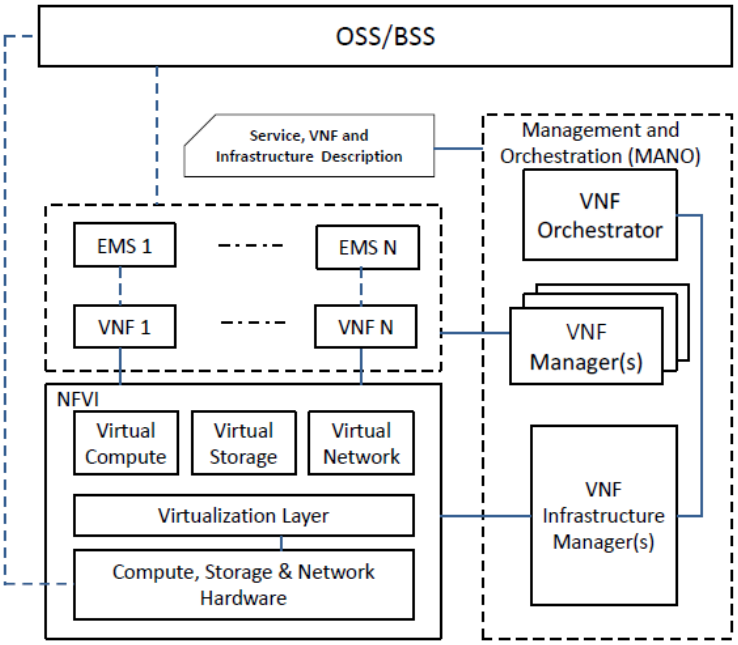


FIGURE 2.3: NFV MANO architecture (source: [2])

- NFV Orchestrator (NFVO): It handles end-to-end service creation. This means, multiple NFVs can be combined to provide a network service. NFVO is also responsible for resource and service orchestration.

2.7 Intent-Based Networking

Intent Based Networking (IBN), is a novel approach to service management and automation by simplification of low-level network configuration through generalized high-level policies known as Intents.

Intents are expressed in a declarative form which serve to separate network users, network administrators and vertical market players from network specificities. This renders network Intents in human-readable format to make them agnostic to the underlying network infrastructure. Intent requests do not provide any technology or service specific implementation such as routes for packet flow, the amount of computational resources required for service provisioning. Intents are network agnostics and provide a common basis for network service requests to diverse network technologies. The concept of Intents take advantage of translation and mapping from human-readable languages to

network equipment configuration language. Intents make use of a closed-loop for service persistence irrespective of the network state and network changes.

Mappings serve as an intermediary between network users and networks, permitting network-independent Intent requests to be properly interpreted by individual network equipment in their own system-specific languages. Intent implementations make use of continuous-loop comparison among existing and new Intent requests, mappings, and controlled-resource sets and states. This is to correctly enact and maintain service Intents, even as such Intents, mappings and controlled resource sets and states may evolve. Intent systems may usefully be represented and even implemented as discrete Intent[3].

In [42], IBM researchers propose a novel intent-based modeling abstraction for specifying the network as a policy governed service and present an efficient network virtualization architecture, Distributed Overlay Virtual Ethernet network (DOVE), realizing the proposed abstraction for data centers. The authors show in that work that it is both necessary and possible for a management abstraction at the virtual level to be conceptually different than that of a physical level. This is done through intent-based network management abstraction, offering a basis for a paradigm shift in the way connectivity services are specified and consumed. Their IBN modeling abstraction captures the network functionality as a blueprint that can be verified and approved prior to the deployment, and deployed prior to instantiating any of the endpoints.

In the same work [42], authors propose a network virtualization architecture created to carry out abstract network specifications down to the level of actual traffic delivery, policing, and control. The architecture is called Distributed Overlay Virtual Ethernet network (DOVE) and it builds upon two major principles—edge-terminated overlays and centralized control plane. Combining the intent-based network functionality abstraction with the DOVE architecture to create a comprehensive network virtualization solution whereby the physical and the virtual infrastructure layers are loosely coupled and operationally independent.

Authors in [43] provide a comprehensive survey on Intent-Driven Networks (IDN) where their work provides perspective from standardisation, open source communities and industry standpoint of view. In relation to standardisation, Internet Engineering Task

Force (IETF) and European Telecommunications Standards Institute (ETSI) have different working groups leading research in IDN. ETSI's ISG working group have come up with an Experimental Network Intelligence (ENI), a policy-based model-driven system. ENI takes care of network configuration and actions to related to network changes. Furthermore, IETF has been working an automated management and autonomic networking integrated model and approach (ANIMA), to help in efficient network management. ANIMA is base on Yet Another Next Generation (YANG) data modeling language for modeling network configurations. For open source communities, this is detailed in section 2.7.1.

In [44], authors propose a Software Resolved Networks (SRNs), a new architecture for IPv6 enterprise networks. They apply the fundamental principles of Software Defined Networks, i.e., the ability to control the operation of the network through software, but in a different manner that also involves the end hosts. Authors leverage their work on SRv6 to enforce and control network paths according to the network policies. Those paths are computed by a centralized controller that interacts with the end hosts through the DNS protocol. The authors work touches on a path awareness selection using an Intent approach.

Self-driving networks are facilitated using an Intent-based approach to express high level network policies and artificial intelligence (AI) in [45]. The authors propose an intent-refinement process that uses machine learning and feedback from the operator to translate the operator's utterances into network configurations.Their refinement process uses a sequence-to-sequence learning model to extract intents from natural language and the feedback from the operator to improve learning. The key insight of our process is an intermediate representation that resembles natural language that is suitable to collect feedback from the operator but is structured enough to facilitate precise translations. Their prototype interacts with a network operator using natural language and translates the operator input to the intermediate representation before translating to SDN rules.

2.7.1 Intent-Based Networking Frameworks

This section is dedicated to open source IBN Frameworks. These are frameworks which are integrated into some open source SDN controllers allowing network administrators or network operators to configure and manage network services through Intents. Examples

of open source SDN controllers include NOX [46], FloodLight [47], OpenDayLight (ODL) [38], Ryu [48], Trema [49] and Open Networking Operating System (ONOS) [37]. ONOS and ODL SDN controllers are more matured controllers and have their own Intent-Based frameworks. Though the focus is on IBN Frameworks, a general overview of ONOS and ODL SDN controllers' architectures are also discussed in subsequent sections and how they interact with the intent-based networking module.

2.7.2 Open Network Operating System (ONOS) Controller

ONOS is an SDN operating system developed by Open Networking Foundation (ONF) which is tailored for service provider networks. ONOS provides high availability, performance and scalability to Service Provider Networks. The SDN controller provides modular structure separating various subsystems into independent modules [50]. The architecture of ONOS is composed of a three (3) tier structure namely; NorthBound, Distributed Core and SouthBound. Figure 2.4 shows the three (3) tier architecture of ONOS controller. The layers are made of different modules responsible for performing various functions. One of such module is the Intent module which is of interest to this work and is detailed in section 2.9

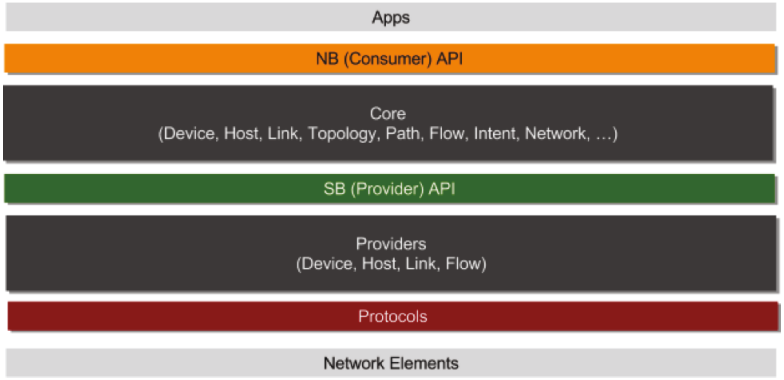


FIGURE 2.4: ONOS Tier Architecture

2.7.3 ONOS NorthBound Interface

This layer is the upper layer of the three (3) tier architecture presented in section 2.7.2, responsible for reception and transmission of information and network requests respectively to the distributed core layer. SDN applications reside on the NorthBound (NB) and monitor network activities through a graphical user interface (GUI) or command

line interface (CLI). Examples of such SDN applications include Topology Monitoring Applications, Device and Network Statistics Application. Applications leverage on the northbound application programmable interfaces (APIs) exposed by the subsystems or modules in the distributed core to synchronise the network state and present them to a network user in real-time.

2.7.4 Distributed Core

It is composed of a modular subsystem component exposing both NB and SouthBound (SB) APIs to applications and infrastructure elements residing in the NB and SB respectively. The subsystems are independent modules but however rely on other subsystems for information pertaining to the network state. Network elements are abstracted into generic models such that network elements like switches, hosts, and routers are not bound to specific protocols making them agnostic [51].

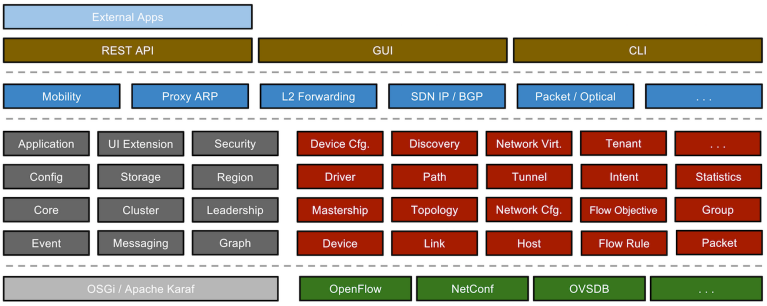


FIGURE 2.5: ONOS Distributed Core

Some of the subsystems include Device, Link, Path and Intent as indicated in the red rectangular shapes in 2.5 above. The core also leverages on Apache Karaf [52] for configuration and local or remote access to the ONOS instance through a command line.

2.7.5 ONOS SouthBound Interface

This is the lowest tier of ONOS architecture and communicates with network infrastructure with specific device protocols. SB APIs exposed by the distributed core communicate with the south bound interface to register and subscribe to the subsystem services to synchronise with network events. Available to infrastructure devices are SB plug-ins including OpenFlow [53], NETCONFIG [54] , OVSDB [54] and Simple Network

Management Protocol (SNMP) [55] for network equipment that support these protocols. Other southbound protocols can be developed to support other equipment which do not have their protocols readily available in ONOS.

2.8 Generic ONOS subsystem model

The subsystems available in ONOS distributed core also follow a three (3) layer structure with each layer residing in one of the main tier of ONOS architecture [51]. The layers are made of the Application, Manager and Provider components which reside in the NBIF, distributed core and SBIF respectively.

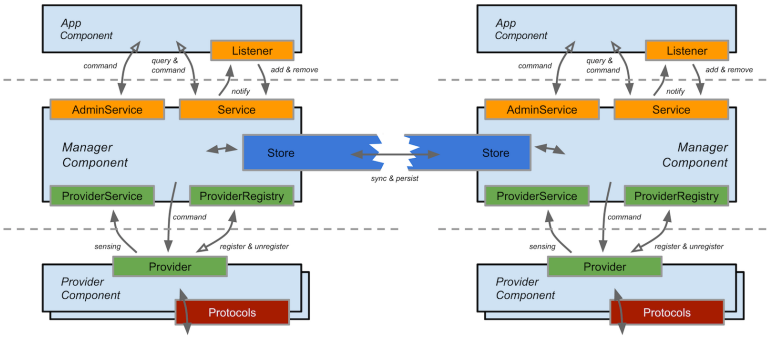


FIGURE 2.6: ONOS subsystem structure

2.8.1 Application Component

Applications reside in this part of the component layer consuming network resources through NB APIs (AdminService and Service) exposed by ONOS core. The applications register with the AdminService API of the subsystem and subscribe to services of these subsystems, a concept of publish/subscribe [51]. The application or user is updated about network events or any change relating to the subscribed service. The application can equally send request for network or service updates through the NB APIs.

2.8.2 Manager Component

The manager serves as the 'middle man' between the application and provider layers. It exposes two (2) north bound APIs (AdminService and Service) and two (2) south bound APIs (ProviderRegistry and ProviderService). Northbound APIs permit applications to

subscribe and receive network event changes by means of the events triggers within the subsystems. As part of the manager component, a store, responsible for tracking and persisting network state changes and synchronisation with other instances of ONOS within a clustered environment, that is multiple instances of ONOS [51].

2.8.3 Provider Component

Network elements communicate with the provider with device specific protocols like OpenFlow, NETCONFIG and SNMP in the south bound [51]. The provider components register these network devices through a southbound API (ProviderRegistry) exposed by the Manager component. Registered devices can subscribe to services of the subsystem through its manager API (ProviderService) in order to be synchronized with event changes on the network.

2.9 ONOS Intent Framework Subsystem

The intent framework is one of the subsystems allowing network applications or network administrators to express network control desires, intents [56]. The subsystem follows the generic subsystem model [51]. The intent framework has additional components, compilers and installers for intents. Figure 2.7 shows the architectural view of the ONOS subsystem and modules.

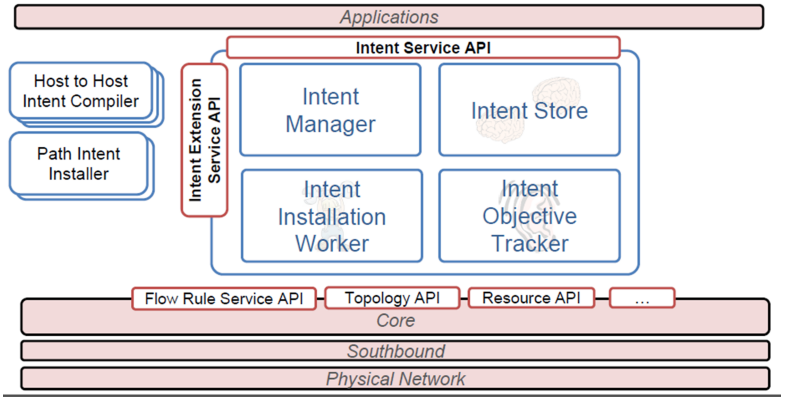


FIGURE 2.7: ONOS Intent Framework

The core accepts intents from the NBIF and translates them into device instructions. Intent requests are transformed via a compilation process into installable (non-decomposable

or primitive) intents. Installers are then responsible for actionable operations on the network [56]. The Intent framework is composed of default intents, compilers and installers based on defined use cases. Components of the intent framework designed by the ONF are extensible and provide room for enhancement of intents or development of new intents depending on the network context or use case.

2.9.1 ONOS Intent Model

The Intent model aligns with the basic English language grammar structure, subject, verb, and predicate. In the context of IBN, this evaluates to subject, criteria (conditions) and actions [56]. Though IBN model accepts constraints, this is optional. We can relate subject to network resources including links, devices and hosts affected by the intents. The verb is equivalent to actions or instructions to be performed based on a set of defined criteria. Criteria, answers the question of WHEN ie. a set of defined policies targeted for a network. Constraints or property explains the level of accessibility to network resources e.g. bandwidth, type of link and optical frequency for optical networks.

2.9.2 ONOS Intent State Machine

Before an intent is executed into actionable instructions or into low-level device specific instructions, intents are compiled and then followed by an installation phase. The intent state machine see figure 2.8 shows the flow of the intent compilation and installation phase. The intent state machine is composed of four (4) transitional states (compiling, installing, recompiling and withdrawing) and five (5) parking states (install request, installed, failed, withdraw request and withdrawn) [56]. Upon receipt of the intent on the NBIF, depending on the type of intent request, the associated compiler is invoked. For example, to create an end-to-end connection or host-to-host intent, a host-to-host compiler is invoked. After successful compilation, an installation phase follows. On the other hand if the compilation fails due to changes in the network resources, an intent lands in a failed parking state. A successful installation phase transits the intent into an installed state otherwise the intent moves into a recompilation phase. Upon a successful recompilation phase, the intent shows as an installed state otherwise a failed state. An installed intent can be removed through a withdrawal request which moves to a transitional state, ‘withdrawing state’ then finally into a withdrawn state.

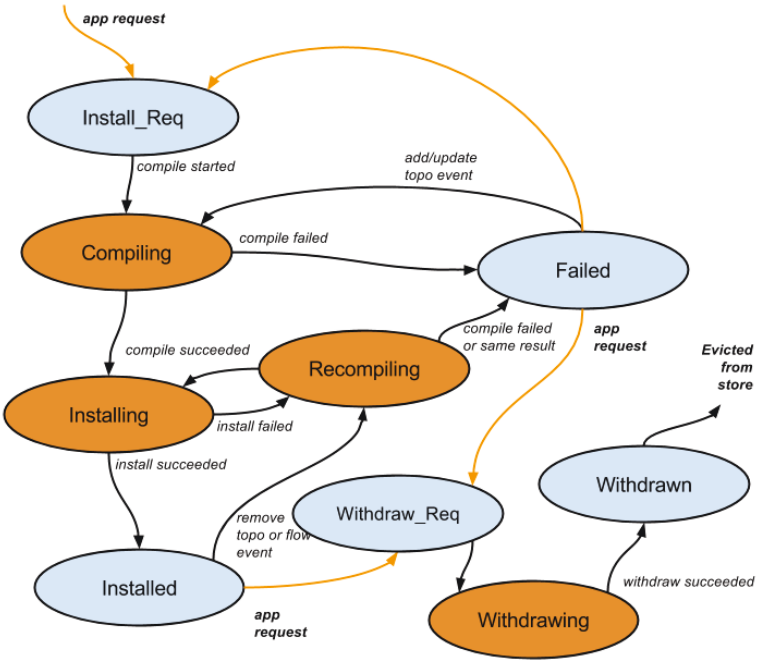


FIGURE 2.8: Intent State Machine

2.10 Network Intent Composition (NIC) for Open Day-Light

The NIC is one of the modules in the core layer of the ODL SDN controller. The aim of the intent model in the NIC architecture of ODL is not different from that of ONOS in terms of functionalities. This involves description of network services desire in a high level abstraction format and translation into network specific instructions. Though both Intent models in ODL and ONOS have the same goal, however these two SDN controllers do not conform to the same architecture.

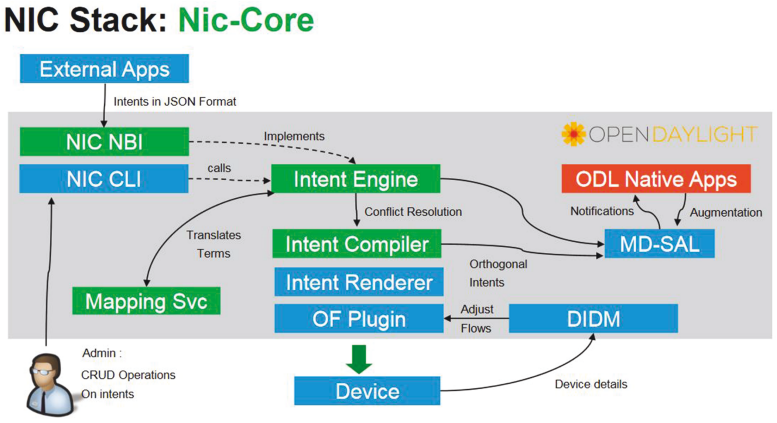


FIGURE 2.9: NIC Core Stack

Figure 2.10 above shows an overview of the internal machinery of NIC. The core components of NIC are detailed as following;

- **Network Intent Composition NBI (NIC NBI)**: This interface receives intents from network applications in JSON format expressing what the application aims to achieve without providing network specific details [57]. Intents could be sent by applications or through the CLI.
- **Intent Engine**: Responsible for implementing and monitoring Intents. It sends Intents to the compiler for translation into low level instructions. Additionally, it also forwards Intent terms to the mapping service.
- **Intent Compiler**: Translates high level instructions into low level i.e. network instructions and also responsible for Intent conflict resolution. Figure 2.10 explains how conflicts are resolved within the NIC.

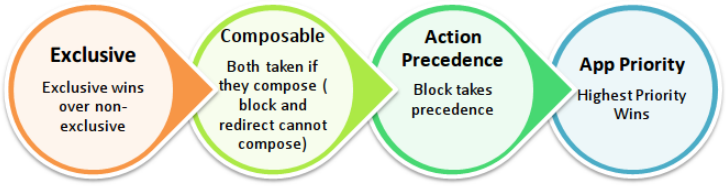


FIGURE 2.10: NIC Intent Conflict Resolution

The first step involves checking the state of the action i.e. allow or block, the exclusive action. For example, block command takes precedence over the non-exclusive commands. The Intent compiler checks whether the actions are composable that is fuse-able into a single instruction, eg. a block and monitor instruction. This is followed by the action precedence step where a block action has a high precedence over other actions, for example, an allow command. The last step of the conflict procedure is execution of application based on priority. For example, a quality of service (QoS) application has a higher priority over a security application. This is not a default behavior in all cases but depends on the type of services and Intent rule definition [58].

- **Mapping Service**: This part of the intent model maps the high level request into the low level device specific instructions that is, the sources and destinations

(IP addresses, devices), action (allow, block, monitor, redirect, etc) and conditions (could be time of day or other type of constraints depending on the service).

2.11 Boulder Intent Project

Boulder is an Intent-Based NBI project focused on SDN controllers driven by the Open Source SDN Community. The project aims at providing semantics and an information model for applications and provides a level of intent portability across the diverse SDN controllers available on the market[59]. Boulder project provides a new layer of abstraction between the NorthBound API of SDN controllers and applications. Figure 2.11 illustrates the boulder architecture with the introduction of an Open Intent Runtime layer.

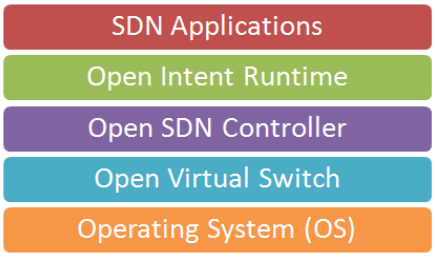


FIGURE 2.11: Intent Boulder architecture

A new independent layer, Open Intent Runtime is introduced on top of the NorthBound Interface of the SDN architectural tier. The conceptual idea is borrowed from Sun Microsystems' slogan for JAVA, Write Once and Run Anywhere (WORA). The aim is to provide cross-platform intent adaptivity for different environments (SDN controllers) without rewriting applications for specific SDN controllers. This helps to achieve portability and expression of intents are not bound to an SDN controller's intent framework semantics. The complexity of the network is encapsulated from the network application developer and network administrators and other users of the infrastructure. Section 2.11.1 introduces how boulder is integrated with the ONOS SDN controller.

2.11.1 Boulder to ONOS Mapping

The boulder layer Open Runtime is integrated with ONOS to provide Intent portability. Though ONOS provides an Intent framework in its distributed core, intents are controller

bound that is cannot be deployed on other families of controllers. Figure 2.11.1 below depicts the boulder integration with ONOS SDN controller.

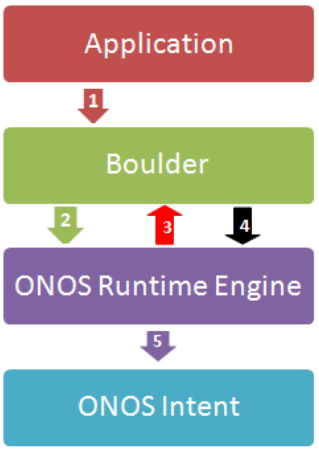


FIGURE 2.12: Boulder ONOS mapping architecture

At the top of the four (4) stack architecture is the SDN application layer where requests are made for network resources and eventually translated to device specific instruction on infrastructure devices. The application requests are expressed in high level request, that is, no network configuration details are provided. An example of a high level intent request would be to create a secure Virtual Private Network (VPN) tunnel between two (2) sites of an organization in remote areas. Configuration details such as type of authentication and encryption are not mentioned in the request. The application or user only makes a request for a VPN service to be provisioned. The Boulder (Open Runtime) layer receives the application request and forwards the information to the ONOS Runtime Engine. The runtime environment compiles and sends a feedback to the boulder indicating a successful compilation of the request. The boulder layer authorizes the ONOS Runtime to submit intent request to the ONOS Intent Framework for provisioning of the required service. The Intent request goes through a compilation phase, and then after an installation procedure where Intents are translated into batch operations containing flow rules. These flow rules are not device specific but follow the model object representation (match-action) pair. Depending on the type of infrastructure device protocol, these instructions are installed on the protocol specific devices. An example of infrastructure device would be OpenFlow enabled switches, in this case OpenFlow rules are installed on the devices [56]. It is the responsibility of the OpenFlow manager to

coordinate with the OpenFlow rule provider, a south bound API which generates OpenFlow messages from the flow rules. These flow rules are then installed in the device to realise an Intent.

2.12 IBN solutions in industrial products

Network vendors are constantly seeking cutting-edge technologies to improve their products for their customers. Vendors are already adapting Intent-driven solutions in their products. This is to make network management easier and less time consuming. Network vendors like Nokia, Cisco, Huawei and Apstra are all involved with IBN. The subsections discuss some of these IBN solutions.

2.12.1 Nokia Altiplano

Altiplano [60] is a Nokia product that provides a unified access management solution for fixed-access services to efficiently manage the transition from traditional networks to an Intent-based networking one. This unified access management platform also helps to reduce network complexity and decrease OPEX. The aim of such a solution is to support the growth demands on networks. Altiplano, puts a premium on network programmability and automation to fulfill high demands of services and drive operational efficiency. Nokia Altiplano, helps to reduce the time to provision and market network services. This product has led to an open software framework to transition from traditional access networks to a more open telco-grade one. Through unification of different access networks, Nokia Altiplano is agnostic to different technologies and enhances Fault, Configuration, Accounting, Performance and Security (FCAPS) processes. Nokia's solution is hardware vendor agnostic to eliminate vendor lock-in problems. This is possible because Nokia leverages on an open source framework that is interoperable with other SDN controllers and automation systems. Nokia Altiplano supports both physical and virtual networks bridging the gap between legacy and software defined-networks. Nokia Altiplano follows the six (6) different stages to help operators provision fixed-access services. Figure 2.13 describes the different stages involved;

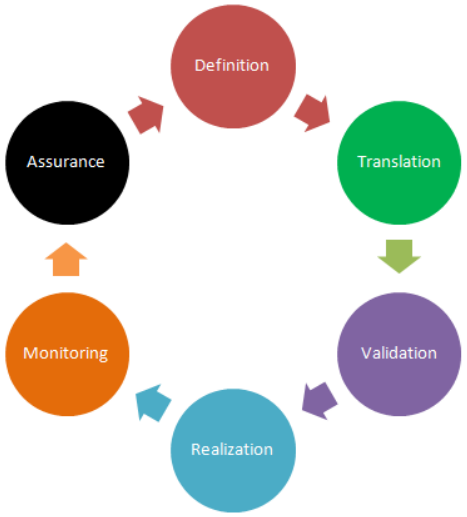


FIGURE 2.13: Nokia Altiplano Intent lifecycle

- **Definition**: The first stage begins with providing high level goals, that is an intent.
- **Translation**: The Intent is translated to into network configuration policies based on the type of devices in the infrastructure.
- **Validation**: This stages involves the correctness of the syntax of intents and the network configuration policies.
- **Realisation**: It involves the deployment and execution of the network configuration policies.
- **Monitoring**: The state of the network is monitored and reported when Intents are violated.
- **Assurance**: This step ensures the integrity of Intents through automated actions to maintain the state of the Intent.

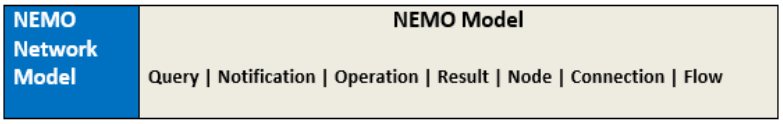
2.12.2 Network Modeling (NEMO)

NeMo is a Huawei Technologies driven project, which is a domain specific language (DSL) designed for abstraction of network models and provides a NBI for applications and network operators to express Intents. NeMo is driven by YANG [61] to describe the model and the Model Driven Architecture (MDA)[62]. NeMo is made up of an engine which acts as network middleware, which translates high level service intents

to network instructions based on MDA. This allows applications/users to leverage on intent-based policies to create virtual networks with controlled policy flows. NeMo follows the generic approach of intent networking by providing a prescriptive approach rather than a descriptive one, eliminating network details pertaining to the infrastructure. The northbound API of NeMo establishes a connection between the application and the controller. NeMo provides a set of commands to express intent on a network categorized into basic network and controller communication commands [62]. Basic network commands include node, link, flow and policy. Communication commands are connect, disconnect, transact, commit, notification and query. These commands are exchanged between the applications and SDN controller via a REST API.

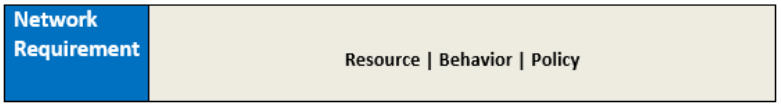
2.12.3 NeMo Methodology

NeMo presents an approach which abstracts resources using a Domain Specific Language (DSL) and executes user and application intents into network instructions. Figures 2.14, 2.15 and 2.16 explain the process from network abstraction to deployment of virtual networks through intents.



The NeMo Model provides a set of commands that can be executed on the network.

FIGURE 2.14: NeMo Network Model



Network requirement includes the network state i.e. behavior, available resources and binding network policies.

FIGURE 2.15: NeMo Network Requirement

An example of how the NeMo DSL is used create a virtual network.

FIGURE 2.16: NeMo Intent Expression

The process involves network abstraction into network primitives then expressed as service level abstraction (NeMo DSL), compilation (NEMO Engine) and then deployment.

2.12.4 NeMo Engine Framework

The framework shows different levels of compilation and rendering of the intents into device configurations. Since NeMo aims at creation of virtual networks for tenants on operator or infrastructure owners' networks, this creates tenant space isolation [62]. Multiple occupants of the infrastructure only have a view of their networks not others, even though on the same physical infrastructure. Figure 2.17 shows the different layers of the framework.

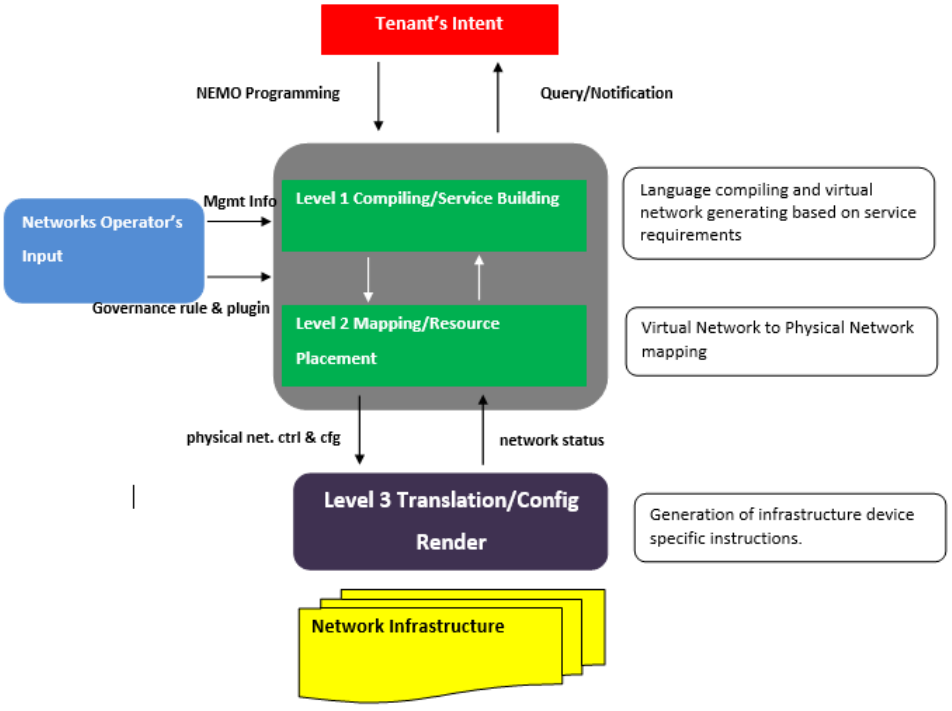


FIGURE 2.17: NEMO Engine Framework

From the Tenant's request, a compilation phase begins and pre-provisioning of network resources based on availability follows. A network service request is sent to level 2 (Mapping Resource Placement) which checks for availability of physical resources on the network (network status). If the needed resources are available, level 2 pushes the request i.e. network configurations down to the third level (Translation/Configuration Render), a layer just above the physical network. Level 3 is responsible for provisioning of the virtual services to tenants in isolated virtual spaces then, the service is provisioned on the physical resources.

2.12.5 Cisco Intent-Based Networking

Cisco's Digital Network Architecture (DNA), [63], stems to bridge the gap between business and IT through Intent-based networking. Cisco's DNA aids data center operators to express Intents and translate into device-level configurations. This is not possible in traditional networks as individual network configuration or policies have to be derived by individual for each network equipment. Cisco DNA provides a policy-based abstraction enabling network operators to express their Intents and then validate the desire outcome of these Intents. Cisco DNA, equally provides a continuous monitoring of Intents through network data collection for analysis and network optimisation. Cisco's view of a complete IBN system should be built around three (3) pillars;

- **Translation**: Helps network operators to express their expected outcome for a service that is "WHAT" without having knowledge of "HOW" it would be achieved.
- **Activation**: This is an automated process to install interpreted Intents into network policies for deployment.
- **Assurance**: Involves continuous monitoring and verification of Intents.

2.12.6 Apstra Operating System (AOS)

The AOS is a product of Asptra, is a vendor agnostic intent-based networking operating system designed for data centers [64]. AOS adapts a closed-loop approach to ensure Intent persistence. AOS can be extended to support different network environment via Software Development Kits (SDK) and RESTful APIs. AOS follows different stages to network and configure networks in a data center using an intent-based networking approach. The stages are numerated and explained as following;

- **Design phase**: Design involves the use of vendor-agnostic design templates through a GUI or RESTful APIs to express an intent.
- **Build phase**: The build phase analyses the Intent in the design phase then dimensions resources and device configuration for the next phase of the cycle.
- **Deploy phase**: Execution of the the different device configurations are carried out in this phase with their corresponding resources allocated.

- **Continuous Validations**: AOS performs an automated continuous monitoring and validation of Intents. In the case of violations, AOS raises alarms relating the specific problem.
- **Automatic validation**: Operations regarding installation of new equipment in the data center and configuration changes are validated automatically in a closed-loop function.

2.13 Conclusion and Remarks

In this chapter, we discussed and highlighted different state of the art technologies to drive IBN. We started by a survey on IDN which highlighted the need for self-configuring, self-managing and self-optimising networks for the future. This survey also clarified definitions related to IDN. The chapter continued with network technology trends by introducing the concept of software defined networking as an enabler for network programmability and network management. Furthermore, we studied SDN controllers and their role for future networks. Two open source SDN controllers, ONOS and ODL have been discussed and their architectures as well. Despite the advantages of SDN, we realised that configuration and setting of network parameters require highly skilled personnels. Additionally, SDN aware networks need frequent re-configuration as network size grows.

We looked into into NFV and the manner in which it renders networks flexible, agile and scalable to drive network programmability. ETSI's NFV MANO architecture has been discussed as well. The different layers related to NFV MANO architecture has been highlighted and the flexibility it offers in service creation through service function chaining. Though MANO architecture provides a decoupled architecture, what is lacking in such an architecture is an intelligent placement criteria for VNFs. MANO architecture only considers network resource availability for VNF placement in operator networks. No other network placement strategies are considered for VNFs. Network placement strategy will be key in 5G networks to be able to accommodate different network services and their associated SLAs. Apart from accommodating different network services, cost of operating data centers need to be considered during network service placement.

We have also investigated existing IBN solutions covering open source implementations, industry solutions and academic research. We started with the Intent-based networking modules in ONOS and ODL SDN controllers. IBN lifecycle operation to help achieve network programmability and management in an agnostic manner have been also discussed. Another IBN project by ONF, Boulder, which aims to provide Intent portability across different SDN controllers to avoid SDN controller intent lock-in. The Boulder project has been integrated with both ONOS Intent run-time and ODL NIC with the introduction of a new layer, Open Intent runtime layer above the Intent layer of both ONOS and ODL SDN controllers. Though Boulder project proposes a layer to integrate multiple Intent engines, not much work has been done on this project and not much information is available about the project. The open source implementation, specially ONOS and ODL are tailored towards optical networks and only provide connectivity Intents.

Finally, industry products from Nokia, Huawei, Cisco and Apstra in the ecosystem of Intent-based networking have been discussed. These companies are in the early stages of providing fully automated networks as recent IBN products are not deployed to support end-to-end network operation.

What is missing in today's state of the art is a solution to support service provisioning for vertical applications. In 5G mobile networks, support for vertical applications will be essential because of the numerous services it brings on-board. The process of providing such services should be fully automated to reduce human intervention in networks.

In chapter 3, we propose an Intent-Based Networking framework to handle some of these challenges then discuss the different modules present in the IBN framework architecture. Additionally, we evaluate our IBN framework with a Proof-of-Concept (PoC) for a 5G IoT network application slice deployment.

Chapter 3

Intent-Based Application Service Framework (IBAS)

We discuss our proposed Intent framework tailored towards network operators and OTT or vertical players assumed to be tenants on operator infrastructure. The framework enables both operators and their tenants to request for network services in non-technical terms. Furthermore, tenants do not need to have knowledge about the underlying infrastructure before requesting for services. These tenants may come from different business sectors not necessarily telecommunications such as health, energy and agriculture domains requesting for services using an Intent-based approach.

The chapter details the different components that make up the Intent framework and their interaction to help deliver network services. A Proof-of-Concept (PoC) and a use case deployment for a 5G IoT OTT network application slice are equally discussed later in this chapter. The framework has been extended to include energy monitoring for network slice components also through an IBN approach. Finally, we show some evaluation results for the 5G IoT OTT application use-case and conclude.

3.1 Limitations of existing solutions

Related work provides some innovative solutions for Intent-based networking. However, to address the problem of Intent-based vertical application deployment on 5G networks, these solutions present some limitations. The main limitations are categorized as follows;

- **Technology specific intent expressions**: Both ONOS and ODL SDN controllers provide Intent frameworks which help to express Intents through a Command Line Interface (CLI). The expression of Intents are composed using technical details, switch or host identity are provided when expressing the Intents. This assumes that OTT or vertical tenants should have knowledge of the operator infrastructure which raises two challenges:
  - The tenant would need to use information that is beyond its skills.
  - Exposing details on their infrastructure may pose a security threat to operators. As such, the ONOS and ODL Intents are better suited for operation engineers needing to gain time and avoid human errors with abstractions of connection establishment among switches.

  Likewise, Huawei's NeMo project provides a technical Intent expression solution where infrastructure knowledge should be known before an Intent can be placed. In NeMo's Intent expression, the IP addresses of the network devices need to be provided for an Intent expression. The solutions discussed above are not viable for OTT or vertical markets who want to leverage network operators' infrastructure for their service demands. Additionally, this means network administrators need to be in constant relation with OTT or vertical tenants to transform their Intents into technical specifications and followed by service provisioning.

- **Intents focused on connectivity and SDN controllers**: Most of the existing work on Intent-based networking applies to SDN and network connectivity only. Existing solutions provide Intents which help establishing network connectivity between end-users and network equipment or equipment only. OpenFlow protocol is pre-dominantly used in related works to establish connectivity on network devices. This is because it is a non-proprietary protocol, it is easy to program network equipment with OpenFlow and finally, it is supported by the open source communities. The Intents investigated in our research are aimed to go beyond network connectivity to deliver application-level services to OTT and vertical markets. IBN functionalities for SDN controllers like ONOS and ODL which do provide network connectivity are tailored towards only infrastructure owners who already have knowledge of the infrastructure.

- **Limited Intent conflict resolution**: In related work, only ONOS and ODL controllers have a mechanism for resolving potential intent conflicts. Two intents may have certain criteria such as bandwidth constraints on network links that can potentially cause a conflict for example a potential service degradation or resource allocation problem. An Intent engine needs to intervene to resolve such Intent conflicts. Though such mechanisms are found both in ONOS and ODL, these are not matured functionalities to fully handle intent conflicts. The Intent engine is not always able to resolve conflicts and network administrators need to solve the problem manually. The IBN feature we aim at is expected to receive thousands of service requests possibly for the same type of service and must be able to deal with intent conflicts.

3.2 Proposed OTT Intent-Based Networking Framework

In this section, we present our Intent-Based networking framework which tackles the limitations discussed in section 3.1, the IBN Framework provides network tenants and network operators with a simplified service interface to express Intents. Intents are a combination of the key fields exposed to network tenants upon infrastructure capabilities and values for these key fields are provided by network tenants. In this document, network tenants, operators or infrastructure owners using the IBN Framework (IBNF) will be referred to as Intent Users. The framework speeds up Intent request placement and service provisioning. The placement and chaining of network function is done by a Utility and A-Star (UA*) algorithm detailed in Chapter 4. The network exposes a list of different types of vertical applications to network tenants. The network infrastructure complexity is hidden from the network tenants as Intents are expressed in a human readable format that is, a declarative manner. A characteristic of the IBN framework that distinguishes it from a user interface is that it provides a system feedback for network service feasibility to guarantee its reliability. To ensure Intents expectations are met, network resources need to be readily available to execute Intents. Our notion of Intent goes beyond network connectivity and SDN controllers. Our Intents cover end-to-end network requests such as application slice setup for tenants offering services such as enhanced Mobile Broad Band (eMBB) and Internet of Things (IoT) services to third

(3rd) party clients. To address Intent conflict resolution, this work defines a first level feature that prevents conflicts in time and resources by

- providing a calendaring feature that plans intent execution in time.
- before scheduling an intent, it checks resources availability for the execution period of the request and reserves them before scheduling an intent.

The modules of the OTT IBN platform are found on top of a Cloud-Over-The-Top Application Platform (COASP) shown in Figure 3.1. The OTT IBN framework is managed by an Infrastructure owner or network operator and provides it as platform for vertical markets. COASP is part of the network infrastructure which interfaces with the IBNF. The IBNF sends it the sets of deployment, connectivity and control instructions inferred from the translated tenant Intent and subsequent analytics and computations. Measurements are gathered from the infrastructure related to control, feasibility and reliability. The functions of the various modules of the IBN framework are detailed in the following sub-sections.

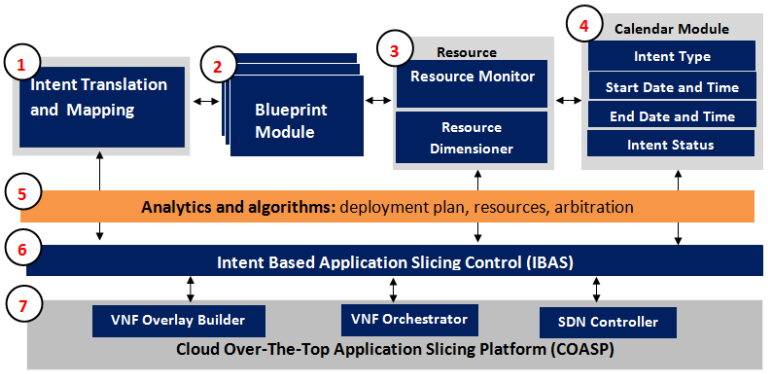


FIGURE 3.1: OTT IBN Framework

3.2.1 General Assumptions and Approach

This section provides the general working assumption of the OTT IBN framework and how the different modules of the framework interact. Interaction between the different modules are based on RESTful API calls. RESTful APIs are exposed by each module and consumed by other modules. Network tenants are exposed to the framework through a graphical interface (GUI) where they input their request without being exposed to infrastructure details. Services provided by the infrastructure are stored in a database

(DB) as pre-defined service templates. Service template composition is made-up of different independent network component known as a micro-service. These templates are parameterised based on network tenant inputs. The subsequent sections provide information relating to the modules that make up the OTT IBNF.

3.2.2 Intent Translation and Mapping

This module serves as the interface between network tenants and the network infrastructure. The module receives Intent Request (IR) issued by tenants or even by the network operator in a human readable language. We propose a semantic model inspired by fourth generation programming languages (4GL), based on a database language, Structured Query Language (SQL). A parser filters the key fields in the intent request and maps them to the corresponding service template in the blueprint database. The service template is then parameterised based on the intent request inputs. The translation and mapping method is detailed in section 3.3.3. This enables a human friendly expression with the aim to avoid network specificities. 4GLs are discussed in section 3.3.1.1. Such an approach permits both vertical players that is network tenants and infrastructure owners or network operators to express Intents in an abstracted manner. An example of an Intent expression using our semantic model is as follows;

> **intent action** [action-type] **intent-name** [application-type] **number-of-devices** [number] **location** [name-of-place] **start-end-date** [date] **duration** [time] **QoE** [string]

The Intent key fields shown in the expression are explained as follows;

- **Intent action**: This is an action word, that is a verb just as in the English language. Examples of action words include **add**, **delete**, **update** and **list**. The intent actions follow the database principle of CRUD i.e. **C**reate, **R**ead, **U**pdate and **D**elete.
- **Intent-name**: This field refers to the type of application provided by the network. The type of application can fall under the category IoT, Mobile Brand Band and ultra-low latency applications.

- **Number-of-devices**: This key field expects the number of data exchanging end devices for an Intent request. Data exchanging end devices refer to either IoT sensors or User Equipment (UE). An estimation of the required network resources can be calculated based on the number of expected number of end devices for the Intent. The network resource estimation is done based on transmission rate and type of the data exchanging end devices which are grouped into different class profiles.

- **Location**: This key field refers to a group of one or more geographical locations where a service is supposed to be deployed. For example, this can be the name of a region or city or a stadium where an event is to take place.

- **Start-end date**: The service can be scheduled for deployment at a certain start and end date. Network resource reservation is done based on the inputs.

- **Duration**: Determines how long a service should last in terms of hours, minutes and seconds.

- **Quality of Experience (QoE)**: This is an SLA parameter provided in an Intent expression to be respected by the service. The metric helps the network select from the different classes of QoE i.e. low, medium and high with their associated cost.

Fourth generation languages are discussed further in section 3.3.1.1. In this manner, network tenants do not need to have knowledge of the network infrastructure. Additionally, a human friendly expression permits network tenants to express intents understanding what exactly they need. The intent framework handles the translation and mapping of the high-level intent expression into network language syntax. This is done through storing pre-defined services in the blueprint module, figure 3.1. A parser filters the key fields in the intent request and maps them to the corresponding service template in the blueprint database. The service template is then parameterised based on the intent request inputs. Network tenants have the option to schedule their Intents or request an immediate application deployment. An example of Intent request can be expressed as following; "I want an **IoT slice** deployed in **Paris** for a duration of **2 hours** to support **1,000,000 sensor devices** and a QoE set to **low**. The interface receives the request,

filters the key words and maps them to the appropriate network service blueprint, calculates required network resources based Intent User input, then provisions the IoT network slice.

3.2.3 Blueprint Module

The blueprint module is a repository for network service templates. The blueprint module is responsible for mapping Intent requests to service templates. Each Intent request is associated with a network service template. The service templates are defined by infrastructure owners or network operators. Service templates comprise of a chain of VNFs i.e. the relationship between the different VNFs. The "blueprints" are defined based on the network capabilities and stored in a template repository. The blueprints are reusable by other network tenants but configurations vary based on a network tenants request. Custom service templates can be composed based on network tenants' demand.

3.2.4 Resource and Intent Monitoring Module

After Intents have been requested by network tenants, resources need to be provisioned for Intents. Resource reservation is based on the type of Intent, that is to say the service to be deployed. Resources refer to both computational and link resources. Computational resources involve CPU and memory while link resources have to do with bandwidth and latency requirements. The Intent framework ensures these resources can be provided for the Intent requests from network tenants. The module also continuously monitors allocated and unallocated resources to estimate resource dimensioning for incoming Intent requests. In the case where allocated resources duplicate for example due to an increase in network traffic or network dynamicity, the resource and monitoring module needs to re-dimension resources to respect the initial SLA of the Intent.

3.2.5 Intent Calendaring Module

Intent calendaring is a feature of the Intent framework which helps to schedule different Intents for future deployments. Network tenants might not need the Intents deployed immediately and to this cause the Intent calendar queues the different Intent request. Occasional MVNO, MNO or 3rd Party tenants may require Intents scheduled and rolled

out during specific date and time of the day due to cost effective reasons, nature of supported end devices or for other reasons. As an example, an IoT Intent could be scheduled for specific hours of the day. The sensor devices spend more time in sleep mode and only transmit data periodically or within specific intervals.

3.2.6 Intent History Module

All Intent records are logged for troubleshooting purposes and furthermore, for analytics. The information stored here could be of great value to infrastructure owners or network operators to predict network tenant behavior patterns and potentially for some data analytics. Infrastructure owners can scale their network resources based on behavioural pattern of network resource consumption in relation to Intents. The thesis does not detail data analytics portion of the Intent history module as this is a futurist feature of the framework.

3.2.7 Intent Based Application Slicing Control Module (IBAS)

IBAS serves as the central hub or heart of the framework providing an interface with the different control entities in the underlying network infrastructure as shown in Figure 3.1. IBAS coordinates with the other modules in the framework to see to the successful deployment of Intent requests. Basically, IBAS interacts with network infrastructure entities like an SDN controller, an overlay network builder, and a VNF orchestrator. These are technologies which permits a scalable, flexible and agile network as seen in chapter 2. Since there is no standardised interface between an SDN controller and a VNF orchestrator, IBAS serves to bridge this gap through the northbound interface exposed by an SDN controller and a RESTful interface available on VNF orchestrators to aid coordinating between the two technologies. IBAS will intermittently send RESTful queries to a VNF orchestrator which is responsible for orchestrating VNFs to know the state of these elements before calling upon an SDN controller to provide connectivity. IBAS also enables Intent-based networking to expand towards both SDN and NFV worlds as compared to existing work which is limited connectivity. Furthermore, IBAS is responsible to maintain the state of Intents in coordination with other modules to satisfy network tenant SLAs. The overlay builder is responsible for creation of overlay

networks for the different networks tenants and their Intents. The aim is to isolate different network tenant Intents for security and privacy reasons.

3.2.8 Intent Deployment Message Flow

Figure 3.2 displays the message signalling between the different sub-modules in the IBN framework for Intent deployment. In this scenario, we assume the network has the necessary resource to provision an Intent. Network tenants send their Intent through the OTT IBN Framework GUI, a request message containing: Intent action, Intent name, number of end devices, location for deployment, start and end date, duration and quality of service as described in 3.2.2. End devices here refer to the sources and sinks of application traffic, for instance IoT sensors, video senders and receivers. The request message is then sent to the blueprint module to search for the appropriate service template. An acknowledgement message is sent for the affirmative to the request interface signifying existence of such a service template. Capacity requirements are then sent to the resource and Intent monitoring module to estimate the needed infrastructure resources and check for their availability. A message is then sent to the tenant indicating feasibility of the Intent and a deployment message is sent to the IBAS control module in the absence of Intent calendaring. On the other hand, if network tenant's Intent includes calendaring, the Intent is scheduled for deployment at the specified time. IBAS control communicates with the different control managers through RESTful API calls to deploy the network application service.

3.2.9 Analytics and algorithms

This feature mainly concerns data collection to aid the framework perform data-driven actions. These actions relate to historical Intent resource consumption patterns and network performance predictions. Information gathered through this module will help to take timely decisions for network slices both at deployment and execution time.

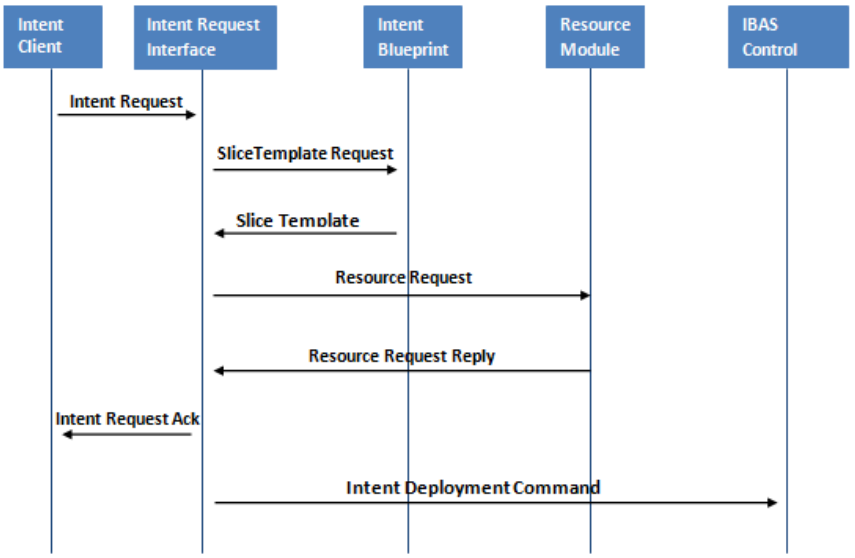


FIGURE 3.2: Intent Message Flow

3.3 Programming Language Generations

Programming languages are classified into several generation ranging from first through fifth. The first (1GL) and second (2GL) generation languages are low-level languages, machine and assembly language [65] respectively and do not provide abstraction thus they are not discussed in the scope of the thesis. We discuss generation languages that provide high-level abstraction that are closer to human languages, third, fourth and fifth generation languages.

3.3.1 Third Generation Programming Languages

This is a class of programming language that provides high-level abstraction with the aim of increasing productivity for programmers. Third generation programming languages also known as 3GL take a less procedural approach [66] compared 1G and 2G languages. 3GL is the specification of programming tasks in a step-by-step algorithmic manner. More can be accomplished with 3GL in an equivalent number of statements as compared to 1GL and 2GL. 3GLs are high-level languages, standardised and independent of hardware. 3GLs are composed of tokens [67] and are further broken down into commands, types and functions just like in the English language, where we have verbs, nouns and adjectives.

3.3.1.1 Fourth Generation Languages

Fourth generation languages, 4GL for short, are groupings of programming languages that attempt to get closer than third generational languages (3GLs) to human language, form of thinking and conceptualization [68]. These languages are closer to human than typical high-level programming languages. Since Intents aim to eliminate any network technical expressions, 4GL seem as an idea fit for Intent expression language. The main domains and families of 4GL are database queries, report generators, data manipulation, analysis and reporting, screen painters and generators, GUI creators, mathematical optimization, web development and general purpose languages.

3.3.1.2 Fifth Generation Languages

Fifth generational languages, 5GL, aim to provide a more natural language for programming leveraging on artificial intelligence. 5GL also involves logic programming and automatic problem solving [69]. In logic programming a program consists of facts about a certain subject, stated as a collection of sentences which express information that can be used to solve problems or to answer questions. The focus of the 5th generation programming language is on constraint programming. Examples of 5th generation programming languages are Mercury, Prolog, and OPS5. These are purely declarative programming languages and supports higher-order programming [70]. These languages turn to take constraints and solve programming problems than have programs written by programmers.

3.3.2 Source-to-source language translation tools

Source-to-source language translation is a method of transforming one high-level language to another language without rewriting the entire language to the target language [71]. These are tools also used for software analysis. An example of such language translation can be transforming Java [72] source code to a Python [73] code. Source-to-source language translator services as the intermediary where the source language syntax and semantics are defined and transformed to the equivalent in the target language. There are several of these tools available as open and proprietary languages. This thesis exploits open source languages as source codes are ready available. Some of

the open source source-to-source translation tools are Turing Extender Language (TXL) [74], Spoon [75], C Intermediate Language (CIL) [76], Stratego/XT [77] and Coccinelle [78]. The following sub-sections discuss these source-to-source languages.

3.3.2.1 Coccinelle

Coccinelle is a program matching and transformation engine which provides a Semantic Patch Language (SmPL) for specifying desired matches and transformations in C code [79]. Coccinelle was initially targeted towards performing collateral evolutions in Linux. Such evolutions comprise the changes that are needed in client code in response to evolutions in library APIs, and may include modifications such as renaming a function, adding a function argument whose value is somehow context-dependent, and reorganizing a data structure. Beyond collateral evolutions, Coccinelle is successfully used for finding and fixing bugs in system code [80]. Coccinelle leverages on a declarative approach based on syntax of patches. It uses an abstracted approach for high level search. With Coccinelle, a small semantic patch is capable of modifying hundreds of files.

3.3.2.2 SPOON

Spoon is an open-source library to analyze, rewrite, transform, transpile Java source code. It parses source files to build a well-designed Abstract Syntax Tree (AST) with powerful analysis and transformation API [75]. It fully supports modern Java versions up to Java 11. Spoon is an official Inria open-source project, and member of the OW2 open-source consortium. Spoon frameworks reifies programs in a meta-model, that allows direct modification of the program structure at compile-time (CT). The meta-model is often seen as an AST which helps end-users extract useful information. Spoon uses a visitor pattern which is applied to the meta-model or AST. Each node in the AST implements an *accept* method so that it can be visited by a processing visitor object allowing modification to the AST. *Processing visitor*, a user-defined processor is responsible for program processing. The processing visitor applies processors to the program until no processing condition is remaining to be applied.

3.3.2.3 C Intermediate Language (CIL)

CIL is a high-level representation of C programs and also source-to-source transformations of C programs.[76]. CIL works at lower levels than ASTs by breaking of complicated constructs in the C programming language. CIL goes beyond just a high-level than typical intermediate languages rendering it easy to analyse and manipulate C programs to have a representation that resembles the original source program. CIL has fewer constructs compared to the C programming language. It breaks down complicated constructs of C into simpler ones, and thus it works at a lower level than abstract-syntax trees [76]. CIL is also more high-level than typical intermediate languages (e.g assembly language) designed for compilation. The representation makes it easy to analyze and manipulate C programs, and emit them in a form that resembles the original source.

3.3.2.4 Stratego/XT

Stratego/XT is a language and toolset for program transformation. Stratego is a small and efficient domain-specific language for program transformation. It is based on the paradigm of programmable rewriting strategies, which makes it extremely well-suited for traversing and transforming all kinds of tree structures [77]. Its library and supporting tools are geared towards processing computer programs in the form of concrete or abstract syntax trees, as produced by text parsers. It supports the development of transformation components at a high level of abstraction [81]. The XT toolset offers a collection of extensible, reusable transformation tools, such as powerful parser and pretty-printer generators and grammar engineering tools. Stratego/XT supports the development of program transformation infrastructure, compilers, program generators, and a wide range of meta-programming tasks. Reusability is a key leading principle in the design and implementation of Stratego/XT. Large-grained components can be reused in Stratego/XT thus a strong focus on transformation components. Language transformation can be started immediately without development of a parser. Apart from Stratego/XT being a Domain Specific Language (DSL), it integrates a DSL for aspects such as syntax definition, pretty-printing, term schemas, and unit testing. Transformation is implemented using stratego language with systems built with Stratego/XT. Tree-like structures are known as terms in Stratego. Stratego programs are represented

in terms, terms are essentially trees. The abstract syntax of a programming language is described by means of a signature.

3.3.2.5 Turing Extender Language (TXL)

TXL is designed to support software analysis and source-to-source transformation [68]. TXL grammar is expressed in a Backus Naur Form (BNF) [82], context-free unambiguous notation. TXL is pre-built with grammar support for several programming languages including Java, Python, Ada [83], C# [84] JavaScript [85] and Yacc [86]. A set of transformation rules are then required for pattern matching and replacement pairs into a target language. TXL generates a parser that is able to resolve ambiguities through left-recursion and a strategy to apply the code transformation rules. Figure 3.3 shows the stages involved in TXL language transformation.

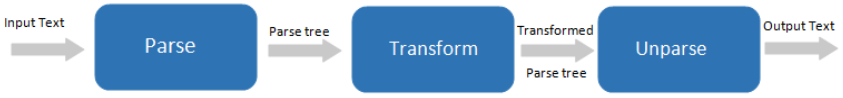


FIGURE 3.3: TXL Transformation Stages

- **Parse**: This stage involves extraction of lexical token from the input text or language into a parse tree which is hidden from the user.
- **Transform**: Transformation involves identifying patterns from the parse tree and matching them based on some defined replacement rules.
- **Unparse**: The last stage involves retrieval of the transformed match patterns from the transformed parse tree and outputting it in the format of the target language.

3.3.3 Application to the IBAS Framework

As described in the previous sections:

- 4GL provides a human readable representation format for instructions with high-level keywords and minimal syntax constraints.
- TXL associates a token to a keyword of a language L1 and defines its translation in another language L2.

TXL is not language-specific: it parses keywords in L1 expressions, maps to them to L2 keywords and renders the expression in L2 format. Additionally, TXL is flexible as mappings can be defined by any entity, in particular by the IBAS Intent translation and mapping module.

The IBAS Intent format is a sequence of so-called keyfields with their values as follows: intent action [action-type] intent-name [application-type] number-of-devices [number] location [name-of-place] start-end-date [date] duration [time] QoE [string]

Given example keyfield values, the sequencing of IBAS Intent key values renders a human readable 4GL format-like expression as follows: "DEPLOY IoT-slice 100000 Paris-Saclay-91 Tue, 10 Nov 2020 08:12:31 GMT 02:00:00 Quality-silver"

IBAS Intents are not expected to be translated to some other language, because they may end up in expressions declined in several possible languages, depending on infrastructure technology. For example, SDN Controller instructions may use an ONOS or Openday-light specific language, while instructions to orchestrator my use a K8S or Swarm specific language. Besides, keyfields may map to actions as different as commands, configuration and database lookup.

Our Intent translation and mapping method leverages on the transformation languages concept as follows:

- it assimilates intent keyfields to keywords
- it defines how intent keyfields are mapped to instructions, templates and configurations rather than keywords in another language

The keyfield mapping in the IBAS Intent and translation module is defined as follows:

intent action : maps to a command of value [action-type] send to the IBAS Control. This command can be viewed as an Intent as its expression, such as "add", "delete", "update", "list", is technology agnostic. The IBAS Control further *translates* it in expressions that are specific to the implementation of SDN Controller or Orchestrator it interfaces with.

intent-name [application-type] : maps to a lookup instruction in the Blueprint Module with the search key equal to [application-type]

number-of-devices [number] , QoE [string] , duration [time]: are sent to the dimensioning module that computes the VNF resources budget and QoS constraints

duration [time] , start-end-date [date]: are sent to the Calendaring module

location [name-of-place]: is sent to the IBAS analytics and algorithms, labeled (5) in figure 3.1 OTT IBN Framework, that maps [name-of-place] to an area in the infrastructure where the slice will be deployed.

3.3.4 Conclusion

We favour 4GL for Intent expression by adapting a database query expression, Structured Query Language (SQL) for our Intent semantic model. Based on our 4GL semantic expression for Intents, TXL seems to suit our approach for source-to-source translations i.e. translating from an SQL like expression to network equipment instructions or configurations. This is due to the fact that TXL is not language specific and already has in-built lexical tokens or pre-defined non terminal types which are also extendable.

3.4 Proof of Concept: OTT 5G IoT Application Slice

In our PoC, we illustrate how our proposed Intent framework is used to deploy a 5G IoT network application slice. The PoC shows how a 5G IoT application slice can be requested by a network tenant through a human readable manner and provide full automation of the service delivery process. It also demonstrates how a network tenant can request and have an Intent provision without knowledge of the network infrastructure

details. The PoC also demonstrates the rapidity of an OTT slice setup and the time gained compared to the traditional approaches.

The PoC is performed on a hybrid virtualised Cloud Radio Access Network environment (CRAN). This comprises a combination of virtual machines and docker containers [87], a lightweight version of virtual machines. In the experiment, we used three Nokia airframe servers which we considered as cloud platforms mapped as Front-End Unit (FEU), Edge Cloud (EC) and a Central Cloud (CC) respectively. These airframe servers were physically connected with a switch. The cloud platforms are cross domain, thus not limited only to OTT network application slice deployment. The term microservice represents VNFs which perform unitary functions. Table 3.1 shows technical specification of our Nokia Airframes which served as the experimental platforms.

	Front End Unit	Edge Cloud	Central Cloud
OS	Ubuntu 16.04	Ubuntu 16.04	Ubuntu 16.04
RAM (GB)	128	128	16
CPU Cores	24	24	8

TABLE 3.1: Cloud Platforms

On CC, we installed a swarm manager [87], a VNF orchestrator is responsible for the deployment of 5G microservice components as docker containers. We also leverage on swarm manager as an overlay builder to create an overlay network for the Intent deployment. The overlay network could have been created with other tools. We only took advantage of what already existed in the orchestrator and swarm manager. The aim of the overlay network is to provide isolation between different network services. On FEU and EC platforms, Open Virtual Switches (OVS) [88] are installed and managed by ONOS SDN controller. OVS serves to interconnect with the 5G mircoservices on the overlay. Swarm manager deploys ONOS as a docker container on CC responsible for network connectivity and chaining of VNFs across the different cloud platforms. Additionally, on CC is a distributed key-value store, etcd[89] responsible for storing cluster information and state information of the 5G docker containers i.e. CPU and memory as a pool of resource across FEU and EC.

The communication between IBAS modules is based on standard HTTP web protocol which facilitates the exchange of information through RESTful APIs exposed by ONOS and swarm manager. The choice of container based approach is to render the network flexible and agile as this is an advantage of NFV as discussed in SoA, chapter 2.

3.4.1 Virtual Network Setup Phase

In this phase of the PoC, we consider the a virtual configuration of the overlay network which was created earlier. Calico [90] and Swarm clients [87] are deployed as docker containers on FEU and EC as part of the virutal network setup phase. Calico is responsible for layer 3 functions which involves routing. Swarm clients serve as an agent which receives deployment commands and provides information related to docker containers to swarm manager respectively. These micro-services are interconnected by the installed OVS on the respective cloud platforms for layer 2 connectivity while calico forwards packets across the different cloud platforms. We assume within a network infrastructure these different phases of network setup will already exist. In our PoC, we show how the infrastructure will be setup as we do not have a pre-exsiting one. Figure 3.4, shows the state of the infrastructure after the virtual network setup.

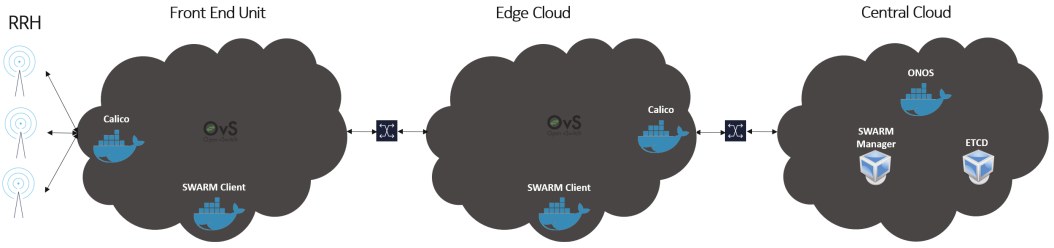


FIGURE 3.4: PoC Virtual Network Setup

3.4.2 5G Platform Setup Phase

The second phase is termed as 5G platform setup involving the deployment of 5G components. In a mobile network, we assume that there are already components installed in the RAN, Transport and Core domain. We emulate the same concept in our PoC to deploy components which serve as enabler functions for OTT network application slices. On FEU, three 5G auxiliary functions are deployed. We employ the word auxiliary because these functions might differ from network operator to the other. These functions run connectivity and user plane operations associated to the slice specific functions. Examples of auxiliary functions include virtual Base Band Units [91], Resource Managers [91] and Frequency Domain Scheduler. These functions facilitate the deployment of 5G network slices. Four other auxiliary functions are deployed on EC, including resource management functions, centralising FEU actions and edge network specific functions. Examples include a shared data layer, baseband service chaining and a local resource

manager. Figure 3.5 shows the state of the network with newly deployed components as part of the 5G platform setup.

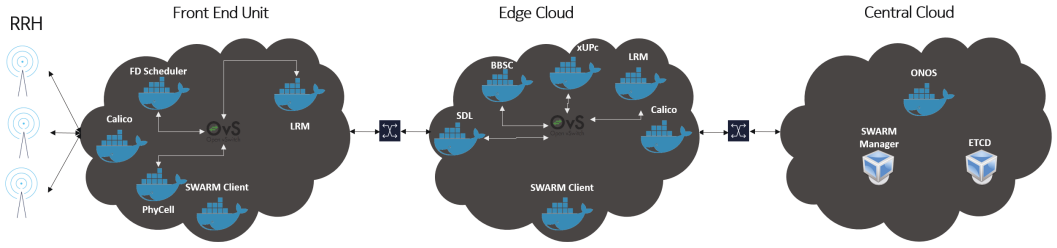


FIGURE 3.5: 5G Platform Setup

ONOS then installs OpenFlow rules on the OVS on EC and CC to enable traffic flow between the different auxiliary functions and other VNFs deployed on the CRAN platform.

3.4.3 5G Application Slicing Phase

VNFs which compose of the 5G application slice are deployed in this phase. This deployment is done based on the network tenant choice of application slice. The Intent request interface receives the request and associated application slice options. An Example of options include the number of sensor nodes or video clients to support for IoT or enhance mobile broadband application slice respectively and duration for slice deployment. In our case, we focus on a 5G IoT network application slice. These information are passed to the blueprint module for mapping with the appropriate template, IoT. Next phase involves dimensioning of resources, CPU and memory for the expected number of end clients, IoT sensors. The dimensioning is done based on pre-assumed resource profiling for the IoT sensors and the rate of packet transmission. Once the mapping is completed, the information is sent to IBAS which communicates with the orchestrator for the deployment process.

3.4.4 OTT IBN Framework Operation

Network tenants are provided with a GUI to make intent requests. The OTT IBN framework is powered by web technologies, NodeJS [92] for IBN framework backend and AngularJS [93] for the GUI through which network tenants interact with the IBN framework. The communication between the GUI and OTT IBN framework are based

on HTTP RESTful API calls. A NoSQL database, mongoDB serves as a repository for network service templates. The service template has properties such as intent action, intent name, end devices, location, date, duration and associated VNFs (VNF chaining) and QoE. Below is an example of a pre-defined JSON structure of a service template. An Intent ID, a unique identity is associated with each Intent request.

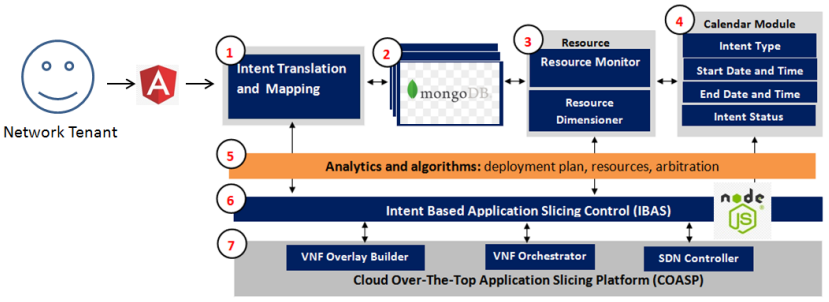


FIGURE 3.6: IBN Framework Implementation

Fig. 3.6 shows the technology mapping on the IBN Framework. Network tenants interact through a GUI implemented with a web front-end technology, AngularJS [93]. The AngularJS front-end GUI interacts with the Intent translation and mapping through a RESTful API interface. The Blueprint module is implemented as a mongoDB. All network service templates are stored in the database. IBAS is implemented with a server side technology, NodeJS which implements the functionalities of the other modules in the IBN framework and exposes these resources as Restful API interfaces.

IntentAction: String
IntentName: String,
EndDevices: Number,
Location: String,
Date: Date
Duration: Time,
QoE: String,
MicroserviceChain: [String]

The OTT IBN platform provides an Intent form where a network tenant declares "WHAT" they want to achieve on the network without knowledge of "HOW" to realise this goal, that is 5G IoT network application slice. The IBN framework then maps

IoT Intent to infrastructure system commands as part of a service deployment. This is achievable through parsing the key fields in the Intent request then progressively maps to the respective service template in the blueprint module. Figure 3.7 shows the Intent request form provided to a network tenant.

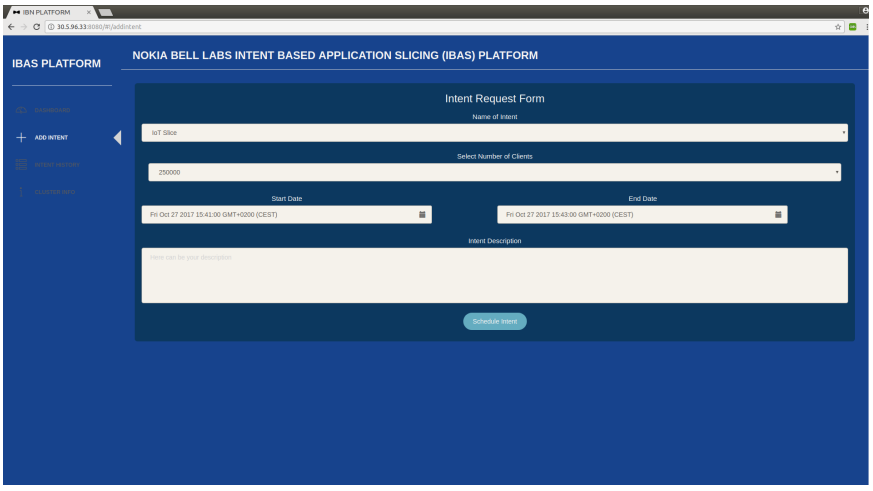


FIGURE 3.7: Intent Request Form

The OTT IBN platform provides a dashboard displaying both total CPU usage and memory consumption of an OTT application slice on the Virtual Network Infrastructure (VNI). The dashboard also displays real-time statistics of CPU and memory consumption of individual microservices of an OTT application slice. Figure 3.8, also shows the dashboard for resource monitoring.

FIGURE 3.8: Intent Dashboard

3.4.5 Validation of the OTT IBN Framework

The validation process involves the feasibility of Intent processing on the OTT IBN platform and the resulting OTT application slice deployment. The performance gain of the OTT IBN platform is measured by the duration for an application slice deployment. The validation scenario considers the 5G IoT application slice. An IoT traffic is simulated on the 5G IoT application slice to validate the slice deployment feasibility. The slice supports up to 600,000 sensors. These are simulated sensor nodes and are assumed to be stationary. We do not take into account the mobility of the nodes in the PoC as sensor traffic is simulated on a PC. A network tenant initiates a request for a 5G IoT application slice through an Intent request form on the OTT IBN platform. The tenant initially requests support for 100,000 sensors as part of the attributes for the slice. The Intent request is validated and sent to the Intent engine. The Intent engine checks for the Intent type, searches within the blueprint module for a Blueprint template match. The next step involves dimensioning of network resources for the 100,000 sensor nodes. Two CPU cores and 1 gigabyte of memory are allocated to the 5G IoT application slice out of the resource pool on the Nokia airframes shown in Table 3.1. The resources allocated are based on pre-assumed VNF profile number of IoT sensors and their packet transmission rate. IBAS control sends a message to swarm manager for 5G IoT micro-service deployment on EC after resource dimensioning, with a working assumption that the 5G connectivity is already set up. This is followed by installation of OpenFlow rules for 5G IoT by ONOS for connectivity with the existing 5G VNF components deployed in the 5G platform setup phase. The state of the VNI is shown in figure 3.9 after 5G IoT application slice deployment. Our validation separately measures the time taken for the three (3) different setup phases.

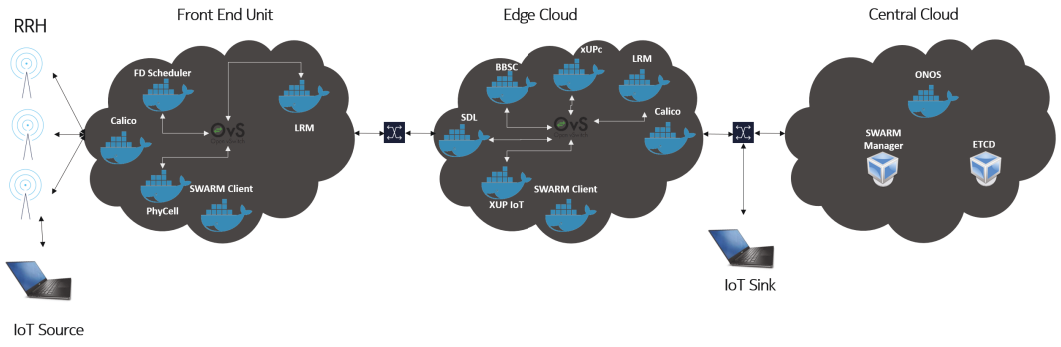


FIGURE 3.9: 5G IoT Network Slice Setup

The experiment is repeated 20 times to support different number of sensor nodes ranging from 100,000 through to 600,000. IoT traffic is then simulated on the 5G IoT network application slice. The traffic source application runs on a laptop connected to the FEU and the sink on another connected to CC. The IoT packet payload is less than 120 bytes and 46 bytes for the header. The IoT traffic is generated randomly respecting the payload criteria. Two types of IoT traffic profiles are simulated, normal and emergence traffic. Normal IoT traffic sends packets at a uniform low throughput while emergence traffic refers to a burst of packets amounting to a sudden rise in throughput. The experiment showed no significant changes for 5G IoT packet throughput and for 5G IoT resource thus we do not show the confidence for the overall repeated experiments. The results of the simulated traffic are detailed in sections 3.4.6 and 3.4.7 respectively.

3.4.6 Intent-based slice deployment verification

This section verifies the feasibility of the 5G IoT application slice deployment by verifying the presence of IoT traffic complying to expected evolution patterns. Figure 3.10 shows the average throughput of the IoT nodes for the traffic profiles simulated by an IoT application simulator on the 5G IoT network application slice. We observe a linear correlation between the number of sensor nodes and throughput, with a lower increase ratio of throughput to sensor number. The distribution is somehow linear because the IoT sensors are immobile and hence can transmit synchronously, thus causing no packet collisions.

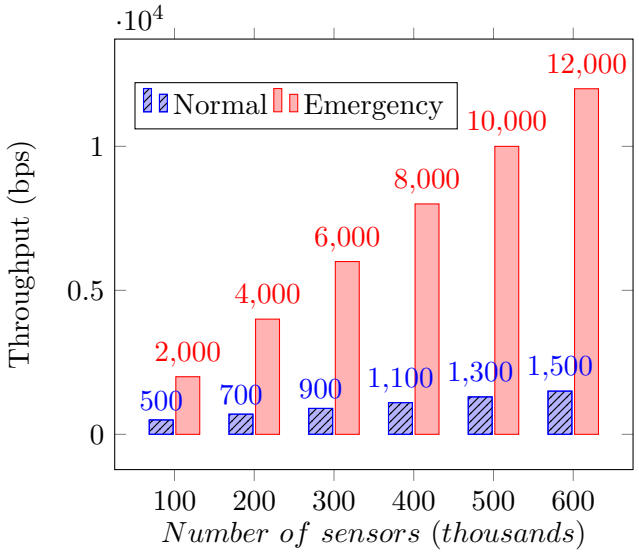


FIGURE 3.10: 5G IoT Packet Throughput.

3.4.7 Validation of the resource and Intent monitoring module

This section verifies the CPU and memory estimation done by the resources dimensioning module for the 5G IoT application slice. Figure 3.11 shows the impact of the number of IoT sensor packet transmission rate on the network resources in percentage, memory and CPU allocated for the slice. The 5G VNFs are dedicated to the 5G IoT application (slice isolation) so as to respect the concept of network slicing in 5G mobile networks. No alarming resource consumption is observed during the experiment. The ratio of the resource consumption to the number of sensors is more moderate for memory than for CPU. The resource consumption evolution patterns are used to help refine the expected Intent slice resources.

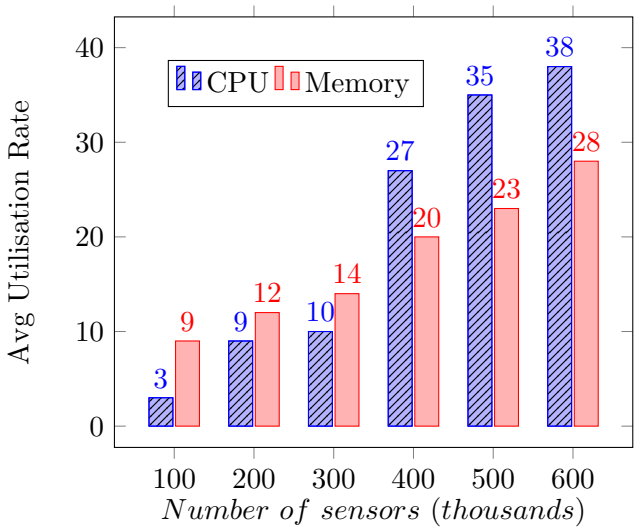


FIGURE 3.11: 5G IoT Resource Impact.

3.4.8 OTT IBAS performance evaluation

This section evaluates the impact of an Intent-based approach on the 5G IoT application slice setup duration. The duration, using the IBAS framework is provided for the three (3) setup phases described in Section 3.11. Table 3.2 displays the duration of virtual network setup phase, 5G VNF component setup and 5G IoT application slice deployment. The overall system setup elapses 86 seconds compared to an expected hour and half of a manual approach by a system expert.

	Number of VNFs	Time (secs)
Network Setup	5	13
5G Setup	7	63
IoT Setup	1	10
Total	13	86

TABLE 3.2: VNI Setup Duration.

3.5 Energy Assessment and Monitoring for Network Slices

This section discusses a new module included in the IBN framework for energy assessment and monitoring. The motivation behind energy monitoring is specifically for billing and possibly resource optimisation to reduce energy consumption in a data center or cloud infrastructure. The functions deploying the VNFs and executing the intent should integrate energy saving mechanisms. The subject on energy optimisation is beyond the scope of this thesis. Energy monitoring is a factor to be considered by network operators or infrastructure owners to determine network slice usage and their impact on energy consumption within their infrastructure.

In literature, GSMA-GST in liaison with 3GPP, the GSM Association defines Generic Slice Template (GST) attributes, (see "Generic Network Slice Template, Version 3.0, NG 116, May 2020"). Among the 37 currently defined, attribute number 7 "Energy efficiency" is defined as the "ratio between the performance indicator, in terms of data volume (DV), and the energy consumption (EC) when assessed during the same time frame". Example requirements are provided in bits/Joule in rural/dense/urban areas. The energy efficiency is evaluated when the application is running and may serve as a KPI. This attribute is FFS in 3GPP Rel17. First, GSMA does not specify ways to express energy awareness in Intents and neither do GSMA-related "Catalyst projects" who attempt to define questions to fill intent attribute values but focus on network capabilities in e.g. bandwidth and latency. The only hint provided in section 3.4.7 is a value in bit/Joule. However providing such a value requires a technical knowledge that should not be required from the client. Secondly, as mentioned in section 3.4.7 of the GSMA specification, "the energy efficiency is evaluated when the network is running". There is no pro-active mechanism to assess energy efficiency at the deployment stage. Lastly, the measurement granularities proposed in GSMA to assess energy efficiency are: weeks, months and years, which may be far beyond the time span of an application

duration and may miss applications deployed only a couple of times, in an ephemeral way.

IBAS introduces an economy (ECO) attribute and value settings for client Intents. The value of ECO entered by a client may take several forms

- A simple clicking on the **ECO** button may prompt an indication of eco-friendly slice deployment, send equal weights of a 3-tuple CPU, RAM, Storage (C, R, S) to the VNF-PF function, and return the number of servers saved with right to a random placement.
- Clicking ECO may also prompt a notification of automated optimal setting. For example, a string value like **ECO-PRIORITY** may send to the VNF-PF optimized weights of **C, R, S** computed upon optimal control or analytics.
- By default, the weight of C is at least doubled with respect to R and S clicking **ECO** may also prompt the possibility for the client to select policy-specific weights for **C, R, S** in case the client has sufficient skills or interest.

If the model is based on a GSMA-GST template, a new Network Slice Consumer specific attribute needs to be proposed to the GSMA, as the current GSMA "Energy efficiency" attribute is a KPI that conveys only bits/joule during run-time. The KPI value returned by this invention (named ECO-UA*) and representing the savings in energy may combine the saved number of servers and estimated power savings. The ECO attribute can also be conveyed with interfaces that are internal to IBAS. The Utility A-Star algorithm is detailed in chapter 4

The evaluation for the energy assessment and monitoring is performed on the existing 5G IoT application network slice use-case already presented in section 3.4. The solution validates a novel cloud-native energy monitoring solution, capable to work at micro-service level (VNFs). The proposed solution exploits lightweight software power meters, based on the PowerAPI tool [94], that can be deployed on different servers. Unlike current state-of-the-art technology, these probes do not require any external device to estimate the energy consumption of the different micro-services executing on the machine [95]. The estimation is part of a two-step process:

1. The probe performs raw metrics acquisition on a monitored server (e.g. hardware counters on central processing unit (CPU) and memory) and stores them in a database.

2. The backend accesses the raw information to compute per-micro-service (or per VM) real-time energy consumption estimation.

The software power meters do not require any complex and cumbersome calibration phase: after an offline configuration phase, the probes can be deployed on the server, ready to work. This is a fundamental aspect for an industrial utilization, where the servers are already deployed and cannot undergo a calibration phase which disrupts their operation. The overall energy-monitoring solution proposed and experimented in SooGREEN project discussed in [96] integrates both server-level monitoring (based on physical power meters) and micro-service level monitoring (based on software probes) into a complete cloud-native approach, see figure 3.12. While the physical power meter is statically installed, the software probes can be deployed as micro-services at system deployment or during execution, on demand. This allows the necessary flexibility to respond to different needs, such as an end-to-end always-on monitoring set-up, or the ability to schedule time-limited sessions, on selected data-centers or machines. The proposed energy monitoring solution is a first step towards efficient cloud networks. Several research topics are still open within the scope of network slice energy monitoring. On the energy monitoring side, end-to-end energy consumption estimation (for services or slices) is still an open challenge, mainly in the case of virtual network function (VNF) shared among different services or slices. To move toward minimized energy consumption, energy-aware orchestration algorithms still have to be defined.

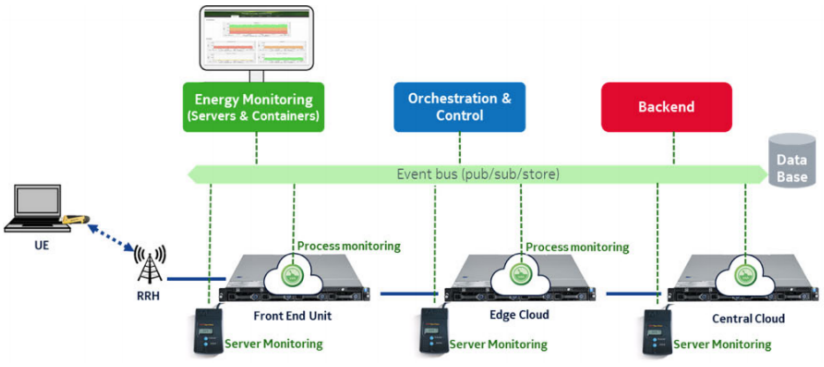


FIGURE 3.12: Cloud native energy monitoring

3.6 Conclusion

Our Intent-based networking Framework aids to bridge the technological gap between SDN and NFV by providing a unified platform. Our framework, extends beyond network connectivity to provide OTT network application services, in our case a 5G network slice through coordination with IBAS interface which lies on top of both SDN and NFV technology.

We equally provide a simplified Intent expression that abstracts network technicalities. This approach helps to reduce the constant verification checks between network tenants and infrastructure owners to determine whether all service requirements are met. Intent mapping accelerates the process of identifying the Intent and also providing the necessary resources without the intervention of infrastructure owners or operators.

Our experiment, PoC identities some key benefits of the framework:

- Automated process from Intent request to service deployment with minimal human interaction.
- A continuous feedback loop on Intents and SLA monitoring to help respect network tenant's desired Intent output.
- A calendaring feature which provides flexibility to schedule repetitive Intent request.

We demonstrate that Intent-based networking goes beyond network connectivity by leveraging on NFV technology to achieve an automated, flexible and agile network. In chapter 4, we present an algorithm which plays a vital role in identifying the infrastructure nodes in the process of Intent deployment and shows the strategy to minimise energy and resource consumption in operator networks.

Chapter 4

Utility and A-Star Algorithm

This chapter is dedicated to the placement and chaining of VNF functions in the scope of Intent deployment. Utility and A-Star algorithm will be referred to as UA* for the rest of the chapter. UA* aims to help infrastructure owners and network operators minimise the cost of operating their infrastructure during the decision process of service deployment. Furthermore, it helps to fully take advantage of the available resources in the infrastructure to reduce resource waste.

The chapter begins with an introduction and background to VNF placement and chaining for network slices. We present our model for UA* then provide simulation results and compare it's performance with the state-of the art approaches. We also discuss related works in VNF placement and chaining after which we conclude the chapter.

4.1 Introduction

The cost benefit of NFV is of huge interest to network operators and infrastructure owners. NFV provides Operational Cost (OPEX) gains, network scalability and flexible deployment strategies contrary to Physical Network Functions (PNF). The PNF requires tremendous time to scale and their management is cumbersome. In the next generation mobile network, 5G technology will open a new industry of services to vertical market players. These players are referred to as network tenants on operator and infrastructure owner networks. Vertical services will involve a hybrid of virtual and physical resources,

isolated and logically dedicated to tenants and referred to as a network slice. Network slicing will cover different network operator infrastructure domains namely RAN, transport network and evolve packet core (EPC) to deliver end-to-end services. To this end, 5G mobile networks will rely on NFV technology to offer an end-to-end network slice deployment. 5G network slices will be composed of many VNFs, as applications encapsulated either as Virtual Machines (VMs) or containers [87]. The VNFs will be chained together to deliver different types of network services, expected to support tenant applications. Emblematic examples of tenant slices include Massive IoT, eMBB and mission critical IoT. Application slices need to meet different SLAs and will share the same network infrastructure resources. Hence their deployment in the network operator or cloud environment must be resource efficient.

The VNF placement problem has been well investigated in literature. This is a well known NP-hard problem [97] and it has been frequently modeled as an Integer Linear Programming (ILP) or a Mixed Integer Linear Programming (MILP). Network slice placement differs from traditional VNF placement as an end-to-end application SLA needs to be guaranteed and the sequence of chains between VNFs can be complex as opposed to linear chains in traditional VNF placement solutions. Current work frequently considers the edge or link bandwidth between different VNFs whereas firstly latency is crucial for emblematic OTT applications and secondly, latency and bandwidth should be considered at the end-to-end level.

The present work, considers end-to-end network slice SLAs and also addresses the efficient resource allocation of network slice VNFs to maximise network operator cloud resources while minimising cost. We also look into non-linear network slice graphs which have not been addressed in the VNF placement and chaining literature. We adopt a classification and ranking approach to address VNF placement and chaining. The classification method reduces cost of network infrastructure graph traversal for network slice placement while ranking helps minimising the needed network resources. We filter and classify the different infrastructure nodes based on their resource capacity to host VNFs. The same is done on the network paths linking the nodes. A utility function then prioritises infrastructure nodes and links used to deploy and chain the VNFs.

UA* algorithm proposes a NFV placement and chaining model with the following optimization objectives:

(i) Minimising the number of active machines in the network infrastructure to reduce the energy consumed by servers and eventually cut down OPEX.

(ii) Providing a fast and scalable VNF placement and chaining solution for network slices.

The rest of the chapter is organized as follows; section 4.2, models the problem as a graph search. Section 4.3 presents the traditional A* algorithm and our Utility-based placement algorithm that extends traditional A* node classification set. Section 4.4 presents evaluation and performance results. Section 4.5, highlights related works in the scope of NFV placement and chaining. Finally, we conclude the chapter in Section 4.6.

4.2 System Model

To formalise the VNF placement and chaining problem, we identify network infrastructure nodes, nodes for short, that will accommodate VNFs and will need to meet computational resource constraints of the VNFs in the Intent blueprint template. Each VNF is assumed to be profiled with their respective resource requirement. Additionally, chaining involves defining connections between VNFs. The physical network links need to meet bandwidth and end-to-end latency constraints as per network slice demands.

4.2.1 Network Infrastructure Graph

The network infrastructure is modeled as a weighted undirected graph $NIG = (N, E)$ where N represents a set of nodes within a network operator infrastructure or a cloud environment and E is a set of physical links. Each node $n_i \in N$ is associated with a 3-tuple, $R_{ni} = (cpu, ram, storage)$, describing its resource availability and denoted by $Rn_i = (C_i, R_i, S_i)$. An edge $(i, j) \in E$ between 2 nodes n_i and $n_j \in N$ is characterised by a 2-tuple describing its performance in terms of bandwidth $B(i, j)$ and latency $L(i, j)$, denoted by $U(i, j) = (B_{ij}, L_{ij})$. These edges are assumed to be bidirectional.

4.2.2 Network Slice Graph

Network slice graph $NS = (F, C)$ describes all the VNFs involved in the execution of a tenant slice, together with their chaining. Each VNF, $F_v \in F$ is associated with a 3-tuple (cpu, ram, storage) describing its resource requirements and denoted by $RQ(F_v) = (Cv, Rv, Sv)$. NS is an oriented graph and C represents a set of edges $E_f(v, q)$ connecting the VNFs of set F. The presence of an edge $E_f(v, q)$ indicates that function F_v must be executed right before F_q. Depending on how they interact with one another, the VNFs within set F are chained in a linear or bifurcated fashion with connectivity constraints.

End-to-end performance requirements are defined on the tenant slices, in terms of metrics such as bandwidth availability and data path latency between the first and last encountered VNFs F_s and F_t respectively. Such constraints are denoted by $C(s, t) = (BQ(s, t), LQ(s, t))$. Table 4.1 summarises the parameters for our VNF infrastructure and network slice graph system model.

Notation	Description
$NIG = (N, E)$	An infrastructure network graph with a set of nodes N and a set of links E
n_i	An infrastructure node $n_i \in N$
(C_i, R_i, S_i)	Node computational resource capacity
$B(i, j)$	Bandwidth of edge i and $j \in E$
$L(i, j)$	Edge latency of i and $j \in E$
$NS = (F, C)$	Network slice graph
F	Set of nodes within network slice graph $F \in NS$
C	Set of edges within network slice graph $C \in NS$
(C_v, R_v, S_v)	VNF node computational resource
$BQ(s, t)$	End-to-end bandwidth required in SLA
$LQ(s, t)$	End-to-end latency required in SLA

TABLE 4.1: Summary of UA* graph model notations.

4.2.3 VNF Placement Model

The first stage of network slicing is mapping VNFs of an Intent template to network infrastructure nodes. The goal is to minimise the number of nodes being used for network slices. Our **objective function** is expressed as follows:

$$Cost(slice) = min \sum_{i} \mathbb{1}_{n_i} \qquad \forall_{n_i} \in [1, N] \tag{4.1}$$

where

$$\mathbb{1}_{(n_i)} = (1 - \prod_{v} (1 - 1_{VNF_v \in n_i})) \tag{4.2}$$

$$(1_{VNF_v}) := \begin{cases} 1 & \text{if } VNF_v \text{ is active} \\ 0 & \text{if } VNF_v \text{ is inactive} \end{cases} \tag{4.3}$$

Though literature does not account for storage, our work considers it as a factor for network slice placement.

A network slice graph is expressed as a composition of different VNFs and their corresponding resource requirements in equations (4.4) and (4.5):

$$NS = \{F_1 .... F_v\} \qquad \forall_{F_v} \in F \tag{4.4}$$

where VNFs belonging to a slice have different computational resource requirements:

$$RQ(F_v) = (C_v, R_v, S_v) \qquad \forall_{Fv} \in NS \tag{4.5}$$

Network slice placement is subject to some constraints, computational resource requirement that need to be respected by infrastructure nodes. This is denoted by;

$$\sum_{F_v \in F} RQ(F_v) \leq \sum_{n_i \in N} RA_{n_i} \tag{4.6}$$

We determine the state of a node in the infrastructure network as *active* or *inactive*. Additionally, the amount of residual resources on *active* nodes and required network resources for incoming Intent network slice requests.

We represent the infrastructure topology as a balanced binary search tree (BST) to facilitate the search process for the VNF slice placement. The leaf nodes represent children

nodes and parent nodes are decision points in the tree. The branches (edges) within the BST serve as a connection path between nodes. The edges between infrastructure nodes are weighted with bandwidth and latency capacity information. We classify nodes and edges based on their residual capacity with our data-driven utility function U, that helps normalizing the computational and connectivity capacity parameters. UA* minimises the number of iterations on the infrastructure network topology graph to speed up the network slice placement by working on sub-trees and sub-nodes. It borrows from the Dijkstra [98] and greedy best-first search [99] to classify nodes into different sets. Whereas traditional A* uses two classification sets named **OPEN** and **CLOSED**, UA* augments the sets to four (4). We limit our set classification to four (4) to reduce the time-complexity for searching and classifying the nodes on the entire infrastructure graph NS.

* **OPEN**: List of unvisited nodes within a network infrastructure graph, NIG.
* **CLOSED**: List of visited nodes in NIG which have residual resources less than the smallest required amount over the set of VNFs within an network slice graph, NS.
* **CANDIDATE**: List of nodes that can host at least one VNF in the network slice request graph.
* **CRITICAL**: List of visited nodes with residual computational resource plus some security margin being equal or greater than the resource capacity requirement of a given VNF. Such nodes are only used in the eventuality of unavailable nodes to host VNFs with a performance penalty.

The VNF placement method and Utility function ranks the slice VNFs and the servers with a utility function, resp. $U(Fv)$ and $U(n_i)$ that takes values in $[0, 1]$.

1. $U(Fv)$ is maximal for the VNF having the highest priority,
2. $U(n_i)$ is maximal for the servers n_i having the less available resources.

$U(Fv)$ can depend on various factors, specific to the slice request or infrastructure constraints. Upon this ranking, UA* attempts to place the highest priority VNF in the most loaded server, so as to saturate the servers before using a new one. $U(n_i)$ is data-driven and defined upon resources as follows;

- UA* first builds an ideal vector $RA^* = (C^*, R^*, S^*)$ where each component is *smallest* value observed over all the n_i.
- for each server n_i having resource availability $RA(i) = (C^i, R^i, S^i)$, the utility vector $UV(i)$ is equal to:

$$\left(\frac{C^*}{C^i}, \frac{R^*}{R^i}, \frac{S^*}{S^i}\right) \tag{4.7}$$

Components of $RA(i)$ are non null, otherwise the corresponding node n_i is in the **CLOSED** set. The utility $U(n_i)$ is equal to a weighted norm of $UV(i)$, by default the $L1$ norm, with weights typically depending on slice and infrastructure context. RA^* does not necessarily exist, but is built with existing values and avoids ranking distortions. Thus $U(n_i)$ normalises vector components of heterogeneous nature, value range and allows to reliable comparison.

The candidate set is sorted with respect to $U(n_i)$. The placement decision is done on the selected candidate set instead of the entire infrastructure network graph. During VNF placement, UA* collects all the graph information needed to compute the paths among nodes hosting the slice VNFs. This speeds up the chaining process for network slices described in Section 4.3.2 since the sequencing of VNFs is not considered at this stage. Furthermore, the number of iterations over the infrastructure network graph is minimised, since the slice VNF placement is done on the candidate set.

4.3 Proposed extended path finding algorithm UA*

4.3.1 Traditional A* algorithm

The traditional A* algorithm is known for finding the best path to a goal and was introduced in [100]. It optimises the Dijkstra [101] search, through a cost function expressed as:

$$f(n) = g(n) + h(n) \tag{4.8}$$

where $g(n)$ is the exact cost of the path from the starting point to any node n within a graph and $h(n)$ is a heuristic function that estimates the cost from node n to the destination. A* combines both algorithms for graph traversal and path selection by prioritising node n with the lowest $f(n)$. A* marks nodes into unvisited and visited classes also known as **OPEN** and **CLOSED** set respectively. A* can behave like a Dijkstra algorithm when $h(n) = 0$ and guarantees to find the shortest path to the goal. On the other hand, if $h(n)$ is exactly equal to the cost of moving from node n to the goal, then A* follows the best path. If $h(n)$ is far greater than $g(n)$, A* turns into a greedy best-first-search. A good heuristic enables A* to benefit from both Dijkstra and greedy best-first search. This renders A* fast as it does not exploit several paths during the graph traversal. This property of A* makes it interesting depending on what needs to be achieved that is finding the shortest path or ideal path or have a balance of both.

4.3.2 Network Slice chaining

Network slice chaining involves establishing connectivity between the successive VNFs of a network slice. VNF chains are usually required to respect connectivity SLA constraints that are expressed in equations (4.10) and (4.11). The sequencing of the VNFs also need to be respected. The outgoing edges and neighbors of nodes of hosting VNFs are stored during the VNF placement process to avoid multiple graph iterations and speed up the chaining process. Assuming two successive VNFs in the chain are placed respectively in nodes i and j, UA* needs to compute the optimal path between the two nodes. To this end, we introduce an extension of the A* path finding algorithm and use the model in equation (4.9). In our algorithm, $g(j)$ represents the cost of moving from node n_i to node n_j within the infrastructure network graph. The cost is a vector composed of bandwidth and latency which is normalised using the formula in (4.9).

$$g(j) = \left(\alpha \frac{B_{(i,j)}}{maxB_{(i,j)}} + \beta \frac{minL_{(i,j)}}{L_{(i,j)}}\right) \quad \forall (i,j) \in e^{+}(i) \tag{4.9}$$

where $e^{+}(i) \subset E$ is the set of outgoing edges from i and $\alpha + \beta = 1$ are weights attributed to bandwidth and latency respectively.

Our heuristics, $h(n)$ is the number of hops to the next VNF in the sequence of the network slice. This determines the path to the next node hosting a slice VNF. In the

case where multiple paths have the same hop count to the next VNF, $g(j)$ serves as the tie-breaker to select the best path. UA* selects the path with the minimum hop count to the subsequent VNF meeting the network slice requirement and not the path with the optimal bandwidth and latency. Some connectivity constraints are placed on bandwidth and latency that need to be respected during the VNF chaining process.

Bandwidth constraint:

$$\sum BQ_{(i,j)} \leq \sum B_{(i,j)} \qquad \forall (i,j) \in E \tag{4.10}$$

In equation (4.10) the total bandwidth demanded by a network slice should not exceed the physical available bandwidth. The SLA constraint on latency is expressed by (4.11).

Latency constraint:

$$\sum LQ_{(s,t)} \leq L_{(i,j)} \qquad \forall (i,j) \in E \tag{4.11}$$

4.4 Performance Evaluation Method

To evaluate our approach, we performed some simulations and compared the results with the Greedy graph search approach [102, 103] and the Tabu search algorithm [104] as it also models the network as a graph.

4.4.1 Simulation Setup and Scenarios

We evaluate our UA* algorithm by using a python library, networkX [105], for graph generation and analysis. We simulate our experiment with the same networkX library by adopting similar simulation parameters as in [106] to better compare with existing algorithms. Literature usually evaluates proposed algorithms or heuristics on either random graphs or the classical Fat-Tree data center topology. We evaluate our algorithm on both Erdős-Renyi [107] and Barabási Albert [108] random graphs and on a data center fat-tree topology. Table 4.2 provides the different parameters for the network infrastructure graph and network slice graph for the simulation. Computational resource capacities of the nodes are randomly generated with a lower and upper bound for the

3-tuples (cpu, ram, storage). Table 4.2 also provides information on the network slice demands. The size of the network slices are randomly generated and associated with computational resources and end-to-end connectivity requirements. In the first round of our simulations, we assume all nodes are eligible as candidates for network slice placement.

Parameters	Infrastructure Network	Network Slice Demand
Graph size (nodes)	$100 - 1000$	$5 - 50$
Node[CPU]	$50 - 100$	$1 - 20$
Node[RAM]	$128 - 256$	$5 - 50$
Node[storage]	$500 - 1000$ units	$100 - 500$ units
Connectivity	$0.5, 0.8, 1.0$	0.5
Edge[bandwidth]	$50 - 100$ units	$1 - 50$ units
Edge[latency]	$1 - 100$ units	$5 - 10$ units

TABLE 4.2: Infrastructure and slice parameter value range.

4.4.2 Performance Metrics

The performance of UA* is evaluated with respect to four (4) metrics defined below:

(i) **Execution Time**: The time taken to map network slices on the infrastructure network nodes.

(ii) **Iterations**: The number of times the infrastructure topology graph is traversed during the network slice placement.

(iii) **Network slice nodes distribution** : The number of nodes used to host a network slice.

(iv) **Acceptance ratio with decreasing resource**: Provides a ratio of the network slices successfully placed on nodes while network infrastructure resources are being reduced gradually.

4.4.3 Scenarios

Different simulation scenarios were considered to evaluate the performance of UA*. The deployment is simulated on random topologies with value ranges as in Table II, of a set

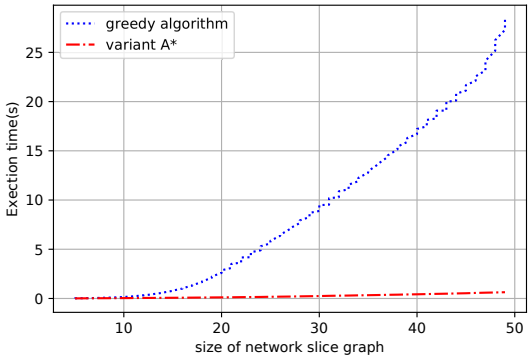


FIGURE 4.1: Erdős-Renyi graph.

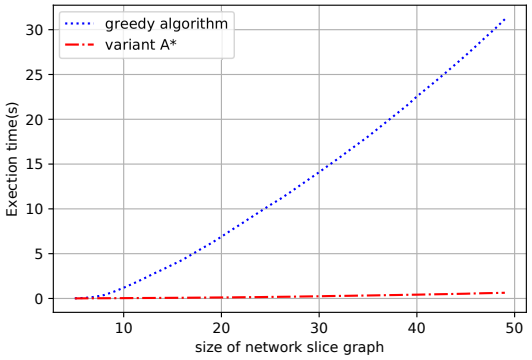


FIGURE 4.2: Barabási Albert graph.

S of 100 slices. The slice size is random, ranges from 5 to 50 with a granularity of 5. Greedy and UA* algorithms are compared against metrics (i), (ii) and (iii) on random graphs. A fat-tree DC topology is compared with UA*, greedy and Tabu algorithms against metric (iv), when server resources were progressively decreased.

4.4.4 Evaluation results on random graphs

This section discusses the results of UA* with respect to the metrics enumerated in Sec. 4.4.2. 10 different simulation runs are performed on the random graphs.

(1) Execution time: Figures 4.1 and 4.2 compare the execution time of UA* with greedy algorithm on Erdős-Renyi and Barabási Albert graphs respectively. The results show that UA* (variant A*) on the legend of the graph outperforms the greedy algorithm. First, the greedy algorithm takes longer to place all network slices because it does not have a global view of the network topology. Additionally, VNFs are placed on nodes one at time without foreknowledge of candidate nodes and where to place VNFs. This contributes to a longer execution time to successfully place network slices in the infrastructure.

(2) Iterations: Table 4.3 provides the average number of iterations for 100 different network slice requests over the infrastructure network graph. UA* performs fewer iterations over the infrastructure graph because sub trees are extracted and sorted with respect to the least available resource. This speeds up the process for placement of network slices whereas the greedy algorithm iterates over the entire infrastructure graph for each request.

Algorithm	iterations	avg. time(s)
greedy	15	650
UA*	5	215

TABLE 4.3: NIS graph iteration.

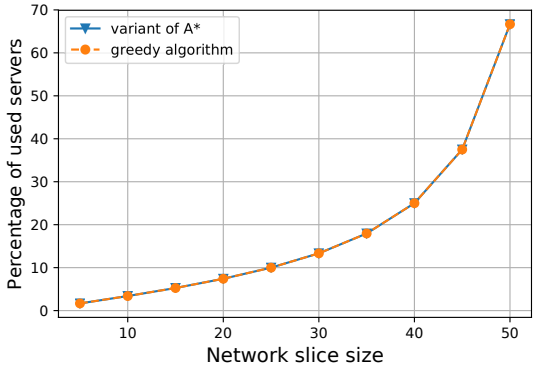


FIGURE 4.3: Policy = 1.

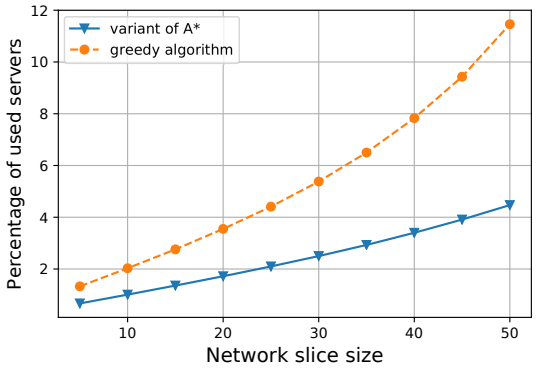


FIGURE 4.4: Policy = 3.

(3) Network slice distribution: Figures 4.3, 4.4, 4.5 and 4.6 describe the average percentage of nodes needed to host various VNFs of network slices. A placement policy, $p-policy$ in terms of number of VNFs allowed on a node is taken into account. $p-policy$ is varied in value set $\{1, 3, 5, 7\}$ and then measure the impact on the VNF distribution on infrastructure nodes. This policy is set up to avoid placing all VNFs making up a network slice on a single node and to eliminate the risk of having a single point of failure. When $p-policy = 1$, the same number of servers are used for the placement of network slices in both algorithms. Though the same percentage of servers are used, the execution time for network slice placement varies significantly as shown in Fig. 4.1 and 4.2. In Fig. 4.4, each node is limited to $p-policy = 3$ hence a significant reduction in the percentage of nodes in use. A percentage difference of 7.5% is observed between UA* and greedy algorithm which can help contribute to reduction of energy consumption. Figures 4.5 and 4.6, show results for $p-policy = 5$ and $p-policy = 7$. UA* not only continues to outperform greedy algorithm in the percentage of involved nodes but also in execution time.

Table 4.4 provide network slice node distribution for a VNF policy per node set to $p = 3$ and $p = 5$ respectively. The placement is performed for 50 different network slices of varying size and on a varying infrastructure size.

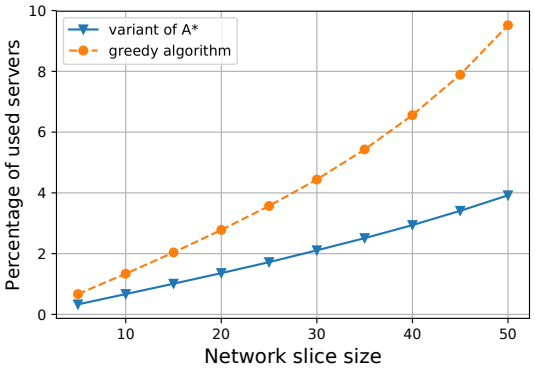


FIGURE 4.5: Policy = 5.

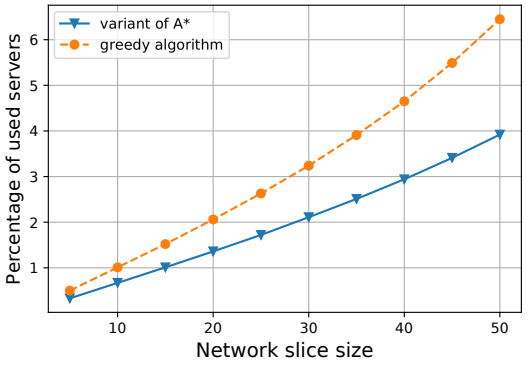


FIGURE 4.6: Policy = 7.

	100	**250**	**500**	**750**	**1000**
Greedy ($p = 3$)	X	X	424	625	800
UA* ($p = 3$)	X	X	350	400	750
Greedy ($p = 5$)	X	X	465	670	923
UA* ($p = 5$)	X	X	380	550	772

TABLE 4.4: Nb of servers used w.r.t infra size and policy $p - policy$.

Results show that an infrastructure network of size 100 and 250, cannot support placement for 50 network slices of varying size, $p - policy = 3$ or $p - policy = 5$. On the other hand, UA* outperforms greedy algorithm in terms of number of nodes. These results give forehand information about the kind of VNF policy per infrastructure to be adapted for a corresponding infrastructure topology size.

4.4.5 Evaluation results on Datacenter Topology

Fat-tree topology: Whereas random topologies also cover DC topologies, DC topologies is used to evaluate UA* against metric "acceptance ratio with decreasing resource". We used the same topology parameter settings as in [104]. Though the authors consider multiple domains, that is different Service Providers (SPs) in their work, the different SPs is mimicked as different clusters within the infrastructure network topology in our simulation setup. A comparison is done between UA* algorithm, tabu search and greedy algorithms. The same number of network slice requests (400) is used in our simulation, as done in [104].

UA* outperforms both greedy and Tabu algorithms due to the fact that it does a pre-node selection before the actual network slice mapping. Though the difference between

Decrease ratio	**0/10**	**2/10**	**4/10**	**6/10**	**8/10**	**10/10**
Greedy	100%	89.75%	60.5%	27.75%	2.75%	0%
Tabu	100%	89.75%	61.75%	28.32%	3.25%	0%
UA*	100%	90.0%	65.75%	30.75%	4.25%	0%

TABLE 4.5: Slice placement acceptance ratio with resources decreased by $N/10$ percent.

UA* and Tabu algorithm is marginal, UA* has a better execution time and fewer iterations over the infrastructure graph.

4.5 Related Work

There have been lots of work on VNF placement and chaining, considered as an NP-hard problem. The problem has been often modeled as an Integer Linear Programming (ILP) or Mixed (MILP) problem. The goal is to map a set of flows and function chains on a graph of many server nodes while optimising resources or respecting their limitations. As this is to be applied to near-instant network slice deployment, ILP-based methods get soon challenged by scalability and time-complexity. These challenges are overcome by adapting a heuristic based approach. [109] looks at VNF placement and chaining for a RAN in order to assist network operators to adapt a custom resource allocation for virtual radio access nodes. They formalise the problem as an ILP and also solve it with a VNF placement heuristic. Their approach identities potential infrastructure nodes that can host the VNFs and prioritise the nodes with the most available resources for Service Function Chains (SFCs) placement. [110] equally proposes an ILP model to ensure that allocated resources are not over or under provisioned for SFCs. A binary search heuristic is proposed to guide the ILP model to find a near-optimal solution for large networks. A MILP approach has been adapted in [111] for aggregated resource provisioning for network slices. Authors map SFCs to slice resource demands within a mobile network operator infrastructure. The demands in CPU, storage and transmission resources of the slices are represented as aggregated requirements rather than individual VNF requirements within slice resource demand. [112] also proposes MILP for VNF placement. A collection of heuristics, a deterministic rounding-based virtual network embedding algorithm (D-WiNE) and a randomized rounding-based virtual network embedding algorithm (R-ViNE) are proposed for the virtual network (VN) and

edge mapping on the infrastructure nodes and edges. D-WiNE maps VN requests sequentially on infrastrucutre nodes using binary values to indicate presence and absence of a virtual network on the infrastructure network. R-ViNE adapts a similiar approach as D-WiNE. (R-ViNE) normalises flow values obtained in D-WiNE within a range of 0 and 1 to determine infrastructure nodes for VN embedding.

As for heuristics based approaches, [103] proposes greedy algorithms for VNF placement and chaining and evaluate the algorithms for different network sizes, dimensions and different VNF chain lengths. [106] addresses the problem as an Eigen decomposition that handles joint VNF placement and traffic distribution. The placement and chaining is done in one shot and their strategy is compared to [103]. In [113], VNF placement is formalised as a decision tree problem to help scaling with the problem size. To this end, they derive a new decision tree algorithm based on the Monte Carlo Tree Search (MCTS). To speed up and improve already existing algorithms for VNF placement and chaining,[114] reduces the problem size by extracting sub-graphs through the preprocessing of a problem instance. These sub graphs serve as inputs to existing algorithms for efficient resource allocation for VNFs. [104] studied Multi-domain VNF mapping algorithms where applications are distributed in different ISP domains. Authors propose a modified version of depth first and breadth first search algorithm for faster VNF mapping in a multi-domain environment. [115] introduces a distributed and hierarchical algorithm for component placement. Their components cover all the networking and traffic processing functions needed to run an application. Heuristic methods usually ignore end-to-end application latency and tend to spread VNFs on separate infrastructure nodes, thus increasing the communication burden among VNFs.

4.6 Conclusion

We presented a novel algorithm, UA* for network slice chaining and placement on random and data center topologies. UA* is robust to support varying network slice sizes and also optimises resource utilisation within an infrastructure or operator domain. A robust utility function normalises and renders the different resource and connectivity vectors comparable. UA* decreases number of nodes used in the network slice deployment as compared to state-of-art methods. The benefit is a reduced consumption of energy in network operator infrastructures and data centers. We evaluated our algorithm on both

random graphs and a datacenter topology which is not seen much in literature. Last, we also show the impact of different network slice placement policies on network resources, which has not been addressed much in literature.

Chapter 5

Conclusion and Future Research

The aim of this dissertation was to investigate

- Intent-Based Networking for the management of 5G Mobile Networks and their services
- how Intent Based networking can facilitate network automation and minimise the time spent in service provisioning

This final chapter provides a review of the research contributions and discusses future research direction.

5.1 Contributions

As part of the contributions, the dissertation began with a survey on existing technologies paving the way for Intent Based networking. We have introduced the SDN technology and its prospects to drive IBN. We further discussed open-source SDN controllers with a focus on ONOS and ODL controllers which both provide an Intent framework. We gave an overview of the working mechanism of their Intent frameworks and also highlighted their limitations.

In the start-of-art, we also discussed industrial IBN solutions from Nokia, Huawei, Cisco and Apstra. All these companies present IBN solutions to address network automation

through IBN but in different contexts. Details related to their industrial products have been provided in section 2.12.

The state-of-art and related works have been complemented with the following two main contributions:

1. **Over-The-Top IBN Framework (Chapter 3)**: The first contribution is the proposal of an IBN framework for Over-The-Top network application slices. The goal of the framework is to automate the process of service provisioning for network tenants on operator networks and also for the network operators. The framework presents five modules to facilitate the network service automation process as discussed in Chapter 3. The Intent translation and mapping module which interfaces with network users enable them to express Intents in a non-technical and human-friendly manner. Network tenants do not need to have knowledge of the underlying infrastructure before expression. This interface conceals and abstracts Intents from network tenants who might not necessary have technical knowledge.
   The concept of blueprints is introduced in this framework to avoid network service composition at every Intent request. They also provide service abstraction and are parameterised based on the type of incoming Intent. This provides lots of flexibility for service blueprint re-use. Concerning resource management, thanks to IBAS which interfaces with an SDN controller to have information related to network links and an orchestrator provides information regarding node resources. This module is particularly important to help resource dimensioning for Intents. The calendar module provides flexibility for Intent deployment for a during a particular date and hour. Finally, we implemented a PoC for the deployment of a 5G network application slice adapting an Intent based approach.

2. **UA* Algorithm (Chapter 4)**: The proposed UA* algorithm helps network and infrastructure owners to strategically place network slices to reduce their OpEX. Infrastructure owners spend a lot of money to maintain their infrastructure and hence such an approach will be of much benefit in terms of cost cutting and infrastructure management. The goal of UA* is to minimise the number of infrastructure nodes used to host network slices and additionally provide a fast and efficient placement strategy. UA* proposes a utility function for infrastructure and VNF resource tuple, CPU, RAM and storage normalisation. A score is attributed to the

normalised infrastructure nodes and network slice VNFs resource tuples respectively. UA* also provides a pre-node selection and classifies infrastructure nodes into four (4) different groups, open, candidate, critical and closed sets. Candidate and critical sets are extensions proposed to the traditional A* pathing algorithm which have two sets, the open and the closed set. Furthermore, UA* extends the cost function of traditional A* by attributing weights to link resources, bandwidth and latency respectively. This aids to put importance on the SLA required by the network tenant and also serves to break ties in case of multiple path to a destination.

UA* introduces a VNF placement policy, the distribution of VNF per infrastructure node. The placement policy analyses the impact of the number of infrastructure nodes used to serve the network slices. The cost benefits in terms of the number of infrastructure nodes that can be potentially turned off are equally evaluated using the UA* algorithm.

5.2 Future Research

For future research opportunities or direction in the scope of IBN, there are still a number of areas that can be improved. A standardised language syntax for IBN is one important area where the research can be enhanced. Due to the diversity of network equipment vendors and their proprietary equipment languages, a more unified IBN language is needed. This is particularly important as it will facilitate the task of an abstracted Intent expression. Furthermore, a natural language processing approach can pave way for a better intent expression. Network operators or infrastructure owners can use historic network configuration data for language learning purposes. In the context of 5G mobile networks, ontology could be another approach to unify the different network domains i.e. RAN, transport and core networks. Nevertheless, a Machine Learning could be used to help resource dimensioning for network slices. This will be an additional benefit to the IBN.

The OTT application slice deployment strategy can be further improved by considering different classes of network tenants in terms of prioritising their service demands. This will help improve service quality for privilege network tenants on the networks. Additionally, network slice migration can be incorporated into our UA* algorithm to

move some VNFs of a slice on different infrastructure nodes for a better network service experience and cost benefit reasons. Infrastructure nodes can be further categorised base on hardware features and mapped to type of network slice to further narrow down sub-graph for network slice placement.

Finally, different communication protocols should be investigated to facilitate information exchange between SDN controllers and NFV orchestrators which sit at the center of IBN.

Chapter 6

List of Publications and Communications

6.1 Conference Proceedings

- Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, Abdelkrim Hebbar: "Intent-Based Real-Time 5G Cloud Service Provisioning", IEEE Global Communications Conference 9-13 December 2018, Abu Dhabi, UAE Gateway to a Connected World – December 2018.

- Fred Aklamanu, Sabine Randriamasy, Eric Renault: "Utility and A*-based Algorithm for network slice placement and chaining", IEEE Global Communications Conference 9-13 December 2019 Hawaii, US Revolutionizing Communications– December 2019.

- Fred Aklamanu, Sabine Randriamasy, Eric Renault: "Intent-Based 5G IoT Application Network Slice Deployment", 10th International conference on the Network of the Future - 0ctober 2019.

- M. Masoudi, F. Aklamanu et al., "Green Mobile Networks for 5G and Beyond," in IEEE Access, vol. 7, pp. 107270-107299, 2019. doi: 10.1109/ACCESS.2019.2932777

6.2 Demonstrations

- "Intent-Based 5G IoT Application Slice Energy Monitoring", Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, Abdelkrim Hebbar, Alberto Conte, Bilal Al Jammal, Warda Hamdaoui, IFIP Networking 2018 Conference - Student Posters/Demos session, May 16 2018.

- "Gestion de l'énergie basée sur les intentions dans les réseaux 5G", Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, Abdelkrim Hebbar, Alberto Conte, Bilal Al Jammal, Warda Hamdaoui, DigiHall Day Event, Paris-Saclay, France, May 22 2018.

- Demo: "Intent-Based 5G IoT Application Slice Energy Monitoring", Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, Abdelkrim Hebbar, Alberto Conte, Bilal Al Jammal, Warda Hamdaoui, MSPN 2018 Conference - Posters/Demos session, June 17 2018.

- Nokia 5G Smart Campus Event (Bell labs FX Days) - Intent-based real-time 5G cloud service provisioning - Fred Aklamanu, Sabine Randriamasy, Imran Latif - 10-12 October 2017.

6.3 Patent

- Method to allocate resources to at least one application, system and computer readable medium (NC103809), Sabine Randriamasy, Imran Latif, Fred Aklamanu

Bibliography

[1] NMRG. Report-nof-nmrg-demo-session, . URL https://datatracker.ietf.org/meeting/interim-2019-nmrg-10/materials/slides-interim-2019-nmrg-10-sessa-report-nof-nmrg-demo-session.

[2] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Steven Latré, Marinos Charalambides, and Diego Lopez. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, 2016.

[3] ONF. Open networking foundation. software-defined networking:the new norm for networks. technical report. 2012.

[4] ON.Lab. ONOS intent framework, May 2016. URL https://wiki.onosproject.org/display/ONOS/Intent+Framework.

[5] et.al. Imran Latif. Cloud ran architecture for smart cities. In *The 1st American University in The Emirates International Research Conference (AUEIRC)*. Springer, 2017.

[6] ITU-R, IMT Vision. Framework and overall objectives of the future development of imt for 2020 and beyond, 2015.

[7] SDxCentral. Intent: Don't tell me what to do! (tell me what you want, February 2015. URL https://www.sdxcentral.com/articles/contributed/network-intent-summit-perspective-david-lenrow/2015/02/.

[8] Lei Pang, Chungang Yang, Danyang Chen, Yanbo Song, and Mohsen Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020.

[9] Min Chen, Yixue Hao, Shiwen Mao, Di Wu, and Yuan Zhou. User intent-oriented video qoe with emotion detection networking. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[10] NMRG. Intent-based networking - concepts and definitions, . URL https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-00.

[11] IETF. A socket intents prototype for the bsd socket api. URL https://tools.ietf.org/html/draft-tiesel-taps-socketintents-bsdsockets-02.

[12] TMF. Service catalog api rest, . URL https://www.tmforum.org/resources/specification/tmf633-service-catalog-api-rest-specification-r18-5-0/.

[13] TMF. Service inventory api rest, . URL https://www.tmforum.org/resources/specification/tmf638-service-inventory-api-rest-specification-r18-5-0/.

[14] TMF. Service ordering management api rest, . URL https://www.tmforum.org/resources/specification/tmf641-service-ordering-api-rest-specification-r18-5-0/.

[15] David Raymer, John Strassner, Elyes Lehtihet, and Sven Van der Meer. End-to-end model driven policy based network management. In *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 4–pp. IEEE, 2006.

[16] Sitaram Kowtha and Xi Jiang. An n-state driven policy-based network management to control end-end network behaviors. In *Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 5–pp. IEEE, 2006.

[17] Nicodemos Damianou, Naranker Dulay, Emil Lupu, Morris Sloman, and Toshio Tonouchi. Tools for domain-based policy management of distributed systems. In *NOMS 2002. IEEE/IFIP Network Operations and Management Symposium.'Management Solutions for the New Communications World'(Cat. No. 02CH37327)*, pages 203–217. IEEE, 2002.

[18] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *International Workshop on Policies for Distributed Systems and Networks*, pages 18–38. Springer, 2001.

[19] Nate Foster, Rob Harrison, Michael J Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. Frenetic: A network programming language. *ACM Sigplan Notices*, 46(9):279–291, 2011.

[20] Christopher Monsanto, Joshua Reich, Nate Foster, Jennifer Rexford, and David Walker. Composing software defined networks. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 1–13, 2013.

[21] Andreas Voellmy, Junchang Wang, Y Richard Yang, Bryan Ford, and Paul Hudak. Maple: Simplifying sdn programming using algorithmic policies. *ACM SIGCOMM Computer Communication Review*, 43(4):87–98, 2013.

[22] Tim Nelson, Andrew D Ferguson, Michael JG Scheer, and Shriram Krishnamurthi. Tierless programming and reasoning for software-defined networks. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 519–531, 2014.

[23] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. Netkat: Semantic foundations for networks. *Acm sigplan notices*, 49(1):113–126, 2014.

[24] Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for netkat. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 343–355, 2015.

[25] Andrew D Ferguson, Arjun Guha, Chen Liang, Rodrigo Fonseca, and Shriram Krishnamurthi. Participatory networking: An api for application control of sdns. *ACM SIGCOMM computer communication review*, 43(4):327–338, 2013.

[26] Robert Soulé, Shrutarshi Basu, Parisa Jalili Marandi, Fernando Pedone, Robert Kleinberg, Emin Gun Sirer, and Nate Foster. Merlin: A language for provisioning network resources. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 213–226, 2014.

[27] Timothy L Hinrichs, Natasha S Gude, Martin Casado, John C Mitchell, and Scott Shenker. Practical declarative network management. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*, pages 1–10, 2009.

[28] Mohammad Banikazemi, David Olshefski, Anees Shaikh, John Tracey, and Guohui Wang. Meridian: an sdn platform for cloud network services. *IEEE Communications Magazine*, 51(2):120–127, 2013.

[29] Hyojoon Kim, Joshua Reich, Arpit Gupta, Muhammad Shahbaz, Nick Feamster, and Russ Clark. Kinetic: Verifiable dynamic network control. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 59–72, 2015.

[30] Chaithan Prakash, Jeongkeun Lee, Yoshio Turner, Joon-Myung Kang, Aditya Akella, Sujata Banerjee, Charles Clark, Yadi Ma, Puneet Sharma, and Ying Zhang. Pga: Using graphs to express and automatically reconcile network policies. *ACM SIGCOMM Computer Communication Review*, 45(4):29–42, 2015.

[31] Yoonseon Han, Jian Li, Doan Hoang, Jae-Hyoung Yoo, and James Won-Ki Hong. An intent-based network virtualization platform for sdn. In *Network and Service Management (CNSM), 2016 12th International Conference on*, pages 353–358. IEEE, 2016.

[32] Mariam Kiran, Eric Pouyoul, Anu Mercian, Brian Tierney, Chin Guok, and Inder Monga. Enabling intent to configure scientific networks for high performance demands. *Future Generation Computer Systems*, 79:205–214, 2018.

[33] Thomas Szyrkowiec, Michele Santuari, Mohit Chamania, Domenico Siracusa, Achim Autenrieth, Victor Lopez, Joo Cho, and Wolfgang Kellerer. Automatic intent-based secure service creation through a multilayer sdn network orchestration. *IEEE/OSA Journal of Optical Communications and Networking*, 10(4):289–297, 2018.

[34] Yoshiharu Tsuzaki and Yasuo Okabe. Reactive configuration updating for intent-based networking. In *2017 International Conference on Information Networking (ICOIN)*, pages 97–102. IEEE, 2017.

[35] Arthur S Jacobs, Ricardo J Pfitscher, Rafael H Ribeiro, Ronaldo A Ferreira, Lisandro Z Granville, and Sanjay G Rao. Deploying natural language intents with lumi. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, pages 82–84, 2019.

[36] Yehia Elkhatib, Geoff Coulson, and Gareth Tyson. Charting an intent driven network. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–5. IEEE, 2017.

[37] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6. ACM, 2014.

[38] *The OpenDaylight SDN Platform*. 2014. URL https://www.opendaylight.org/.

[39] ETSI. Etsi gr nfv 001. network functions virtualization (nfv); use cases. 2017. URL https://docbox.etsi.org/ISG/NFV/Open/Publicationspdf/Specs-Reps/NV20001v1.2.120-20GR20-20NFV20Use20Cases20revision.pdf.

[40] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K Marina. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100, 2017.

[41] Ahmed M Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A Carella, Stefan Covaci, and Thomas Magedanz. Service function chaining in next generation networks: State of the art and research challenges. *IEEE Communications Magazine*, 55(2):216–223, 2016.

[42] Rami Cohen, Katherine Barabash, Benny Rochwerger, Liran Schour, Daniel Crisan, Robert Birke, Cyriel Minkenberg, Mitchell Gusat, Renato Recio, and Vinit Jain. An intent-based approach for network virtualization. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 42–50. IEEE, 2013.

[43] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.2969208.

[44] David Lebrun, Mathieu Jadin, François Clad, Clarence Filsfils, and Olivier Bonaventure. Software resolved networks: Rethinking enterprise networks with ipv6 segment routing. In *Proceedings of the Symposium on SDN Research*, pages 1–14, 2018.

[45] Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, and Lisandro Zambenedetti Granville. Refining network intents for self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 15–21, 2018.

[46] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.

[47] *Floodlight OpenFlow Controller*. Accessed: 2013-04-28, 2014. URL http://www.projectfloodlight.org/floodlight.

[48] *Ryu SDN Controller*. Accessed: 2013-03-30, 2013. URL https://osrg.github.io/ryu/.

[49] *Trema SDN Controller*. Accessed: 2013-04-28, 2013. URL https://trema.github.io/trema/.

[50] ON.Lab. *ONOS Overview*. onosproject.org, California, 2016. URL https://docs.google.com/presentation/d/1Y4S82YZyskqnKAzW4kKm-6IIO04h_nnrRQyfrsCdt-I/edit?pref=2&pli=1#slide=id.p.

[51] B Varkonyi A Koshibe. *ONOS System Components*. onosproject.org, California, 2016. URL https://wiki.onosproject.org/display/ONOS/System+Components.

[52] Achim Nierbeck, Jamie Goodyear, Johan Edstrom, and Heath Kesler. *Apache Karaf Cookbook*. Packt Publishing Ltd, 2014.

[53] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[54] C Cascone A Campanella. *Southbound: Protocol, Providers, Drivers*. onosproject.org, California, 2016. URL https://wiki.onosproject.org/display/ONOS/Southbound%3A+Protocol%2C+Providers%2C+Drivers.

[55] Chris Hare. Simple network management protocol (snmp)., 2011.

[56] Jonathan Hart Ayaka Koshibe. *Intent Framework*. onosproject.org, California, 2016. URL https://wiki.onosproject.org/display/ONOS/Intent+Framework.

[57] Open DayLight Wikis. *Open DayLight High Level Features*. www.opendaylight.org, 2016. URL https://wiki.opendaylight.org/images/8/8b/Nic-arch-conversation.pdf.

[58] Ciena C Janz and HP R Amorim. *What is Intent Anyway*. www.youtube.com, 2015. URL https://www.youtube.com/watch?v=QvEK_CFIGik.

[59] Bithika. *Aspen + Boulder=Intent*. opennetworkingfoundation, California, 2013. URL https://www.opennetworking.org/?p=1866&option=com_wordpress&Itemid=316.

[60] Nokia. Unified access management. *Nokia White Paper*, 2017. URL https://onestore.nokia.com/asset/202237.

[61] M Bjorklund and L Berger. Yang tree diagrams. *draftietf-netmod-yang-tree-diagrams-06 (work in progress)*, 2018.

[62] Pedro A . Gutierrez T Zhou Huawei Helen Chen. *Apply Intent for Service Level Network Programming*. wiki.opendaylight.org, Californi, 2013. URL https://wiki.opendaylight.org/images/e/ef/Apply_Intent_for_Service_Level_Network_Programming.pdf.

[63] Cisco Systems. Cisco intent based networking solution. *Cisco Systems*, 2019. URL https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/digital-network-architecture/nb-06-ibn-sol-overview-cte-en.html.

[64] Apstra Incorporated. Apstra operating system. *Apstra White Paper*, 2017. URL https://cn.teldevice.co.jp/_cms/wp-content/uploads/2018/03/eb347cd151cbc3d313bf1c98deda2a77.pdf.

[65] Greg Morrisett, David Walker, Karl Crary, and Neal Glew. From system f to typed assembly language. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 21(3):527–568, 1999.

[66] Alan L Tharp. The impact of fourth generation programming languages. *ACM SIGCSE Bulletin*, 16(2):37–44, 1984.

[67] S. K. Misra and P. J. Jalics. Third-generation versus fourth-generation software development. *IEEE Software*, 5(4):8–14, 1988.

[68] James R Cordy. Txl source transformation in practice. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 590–591. IEEE, 2015.

[69] Steven Hardy. Towards more natural programming languages. *Artificial intelligence programming environments*, pages 79–114, 1987.

[70] Isabel Gouveia Lima and Philip C Treleaven. Programming languages for fifth generation computers. *Computer Physics Communications*, 38(2):221–231, 1985.

[71] Paul F Albrecht, Philip E Garrison, Susan L Graham, Robert H Hyerle, Patricia Ip, and Bernd Krieg-Brückner. Source-to-source translation: Ada to pascal and pascal to ada. *ACM SIGPLAN Notices*, 15(11):183–193, 1980.

[72] Ken Arnold, James Gosling, David Holmes, and David Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 2000.

[73] Guido Rossum. Python reference manual. 1995.

[74] James R Cordy. The txl source transformation language. *Science of Computer Programming*, 61(3):190–210, 2006.

[75] Renaud Pawlak, Carlos Noguera, and Nicolas Petitprez. Spoon: Program analysis and transformation in java. 2006.

[76] George C Necula, Scott McPeak, Shree P Rahul, and Westley Weimer. Cil: Intermediate language and tools for analysis and transformation of c programs. In *International Conference on Compiler Construction*, pages 213–228. Springer, 2002.

[77] Eelco Visser. Program transformation with stratego/xt. In *Domain-specific program generation*, pages 216–238. Springer, 2004.

[78] Henrik Stuart. Hunting bugs with coccinelle. *Master's Thesis*, 2008.

[79] Neil D Jones and René Rydhof Hansen. The semantics of "semantic patches" in coccinelle: Program transformation for the working programmer. In *Asian Symposium on Programming Languages and Systems*, pages 303–318. Springer, 2007.

[80] INRIA. *What is Coccinelle*. LIP6, 2019. URL http://coccinelle.lip6.fr/.

[81] Stratego/XT. *Stratego/XT*. LIP6. URL https://strategoxt.org/.

[82] Donald E Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7(12):735–736, 1964.

[83] Jean D Ichbiah, Bernd Krieg-Brueckner, Brian A Wichmann, John GP Barnes, Olivier Roubine, and Jean-Claude Heliard. Rationale for the design of the ada programming language. *ACM Sigplan notices*, 14(6b):1–261, 1979.

[84] Anders Hejlsberg, Scott Wiltamuth, and Peter Golde. *C# language specification*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[85] Simon Holm Jensen, Anders Møller, and Peter Thiemann. Type analysis for javascript. In *International Static Analysis Symposium*, pages 238–255. Springer, 2009.

[86] Stephen C Johnson et al. *Yacc: Yet another compiler-compiler*, volume 32. Bell Laboratories Murray Hill, NJ, 1975.

[87] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. volume 2014, page 2. Belltown Media, 2014.

[88] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open vswitch. In *NSDI*, pages 117–130, 2015.

[89] Core OS. A distributed, reliable key-value store for the most critical data of a distributed system. URL https://coreos.com/etcd/.

[90] Project Calico. Project calico. URL https://www.projectcalico.org/.

[91] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati. Network function virtualization in 5g. *IEEE Communications Magazine*, 54(4):84–91, 2016.

[92] Jim R Wilson and Jacquelyn Carter. *Node. js the right way: Practical, server-side javascript that scales*. Pragmatic Bookshelf, 2013.

[93] Peter Bacon Darwin and Pawel Kozlowski. *AngularJS web application development*. Packt Publ., 2013.

[94] Ali El-Amine, Hussein Al Haj Hassan, and Loutfi Nuaymi. Analysis of energy and cost savings in hybrid base stations power configurations. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–7. IEEE, 2018.

[95] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. A review of energy measurement approaches. *ACM SIGOPS Operating Systems Review*, 47(3):42–49, 2013.

[96] Meysam Masoudi, Mohammad Galal Khafagy, Alberto Conte, Ali El-Amine, Brian Françoise, Chayan Nadjahi, Fatma Ezzahra Salem, Wael Labidi, Altuğ Süral, Azeddine Gati, et al. Green mobile networks for 5g and beyond. *IEEE Access*, 7: 107270–107299, 2019.

[97] Andreas Fischer, Juan Felipe Botero, Michael Till Beck, Hermann De Meer, and Xavier Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, 2013.

[98] Yong Deng, Yuxin Chen, Yajuan Zhang, and Sankaran Mahadevan. Fuzzy dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*, 12(3):1231–1237, 2012.

[99] Tatsuya Imai and Akihiro Kishimoto. A novel technique for avoiding plateaus of greedy best-first search in satisficing planning. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[100] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[101] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[102] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[103] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Steven Davy. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9. IEEE, 2015.

[104] Jorge Martín-Pérez and Carlos J Bernardos. Multi-domain vnf mapping algorithms. In *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6. IEEE, 2018.

[105] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[106] Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. Vnf placement and chaining in distributed cloud. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 376–383. IEEE, 2016.

[107] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[108] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[109] Roberto Riggio, Abbas Bradai, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, and Toufik Ahmed. Virtual network functions orchestration in wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 108–116. IEEE, 2015.

[110] Marcelo Caggiani Luizelli, Leonardo Richter Bays, Luciana Salete Buriol, Marinho Pilla Barcellos, and Luciano Paschoal 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)*, pages 98–106. IEEE, 2015.

[111] Quang-Trung Luu, Michel Kieffer, Alexandre Mouradian, and Sylvaine Kerboeuf. Aggregated resource provisioning for network slices. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[112] Mosharaf Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON)*, 20(1):206–219, 2012.

[113] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. An efficient algorithm for virtual network function placement and chaining. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 647–652. IEEE, 2017.

[114] Andreas Blenk, Patrick Kalmbach, Johannes Zerwas, Michael Jarschel, Stefan Schmid, and Wolfgang Kellerer. Neurovine: A neural preprocessor for your virtual network embedding algorithm. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 405–413. IEEE, 2018.

[115] Maryam Barshan, Hendrik Moens, and Filip De Turck. Design and evaluation of a scalable hierarchical application component placement algorithm for cloud resource allocation. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 175–180. IEEE, 2014.