Gestion de la confiance et de la responsabilité dans des environnements multi-acteurs, dynamiques et avec plusieurs niveaux de délégation de responsabilités

Yacine Anser

HESAM UNIVERSITÉ

le cnam

École Doctorale Sciences Des Métier De l'Ingenieur

Centre d'Étude et de Recherche en Informatiques et Communications

THÈSE DE DOCTORAT

présentée par : **Yacine ANSER**

soutenue le : **18 Mars 2024**

pour obtenir le grade de : **Docteur d'HESAM Université**

préparée au : **Conservatoire national des arts et métiers**

Discipline : **Informatique**

Spécialité : **Informatique**

Trust and Liability Management in Multi-Actor, Dynamic Environments with Hierarchical Delegation

THÈSE dirigée par : Mme BOUZEFRANE Samia
Professeur des Universités, Cnam
Et co-encadrée par :
Mme GABER Chrystel
Docteure - Recherche et Développement, Orange Innovation Caen
M. YACOUB Meziane
Maître de Conférences, Cnam

Jury

M. Lyes KHOUKHI	Professeur des universités, ENSICAEN	Président
M^me^ Samia BOUZEFRANE	Professeur des Universités, Cnam Paris	Direction de la thèse
M^me^ Thi-Mai-Trang NGUYEN	Professeure des universités, Université Sorbonne Paris Nord	Rapporteur
M. Abderrezak RACHEDI	Professeur des universités, Université Gustave Eiffel (UGE)	Rapporteur
M. Fabrice MOURLIN	Maître de conférences, Paris 12	Examinateur
M^me^ Chrystel GABER	Docteure, Orange Innovation Caen	Co-encadrant de la thèse
M. Meziane YACOUB	Maître de Conférences, Cnam	Co-encadrant de la thèse

THÈSE

Acknowledgements

I would like to express my deepest gratitude to my thesis committee, starting with my rapporteurs, Mme. Mrs. Thi-Mai-Trang NGUYEN, Professor at Sorbonne Paris Nord, and Mr. Abderrezak RACHED, Professor at Gustave Eiffel, for the honor of evaluating the work presented in this thesis. I am also deeply grateful to Mr. Lyes KHOUKHI, Professor at ENSICAEN, for presiding over my thesis defense. My sincere thanks to Mr. Fabrice MOURLIN, lecturer at Paris 12, for his role as an examiner.

I owe a tremendous debt of gratitude to my thesis advisors, Mrs. Samia BOUZEFRANE, Professor at Cnam Paris, Mrs. Chrystel GABER, researcher at Orange Innovation Caen, and Mr. Meziane YACOUB, Lecturer at Cnam, for their expertise, patience, and support.

This thesis was conducted within the INPISRE-5Gplus project framework. I extend my gratitude to all who contributed to a rich and engaging research atmosphere, particularly Mr. Gürkan Gür, Senior Lecturer at ZHAW School of Engineering, and Mr. Onur Kalinagac, who worked as a research assistant at ZHAW School of Engineering during the INPIRE-5Gplus project. Their insights and contributions significantly enhanced my work.

I would like to express my sincere gratitude to all those who dedicated their time to review my thesis and actively participated in enhancing its quality.

Lastly, I must convey my profound gratitude to my family for their unwavering support and continuous encouragement throughout my academic journey.

Abstract

The rapid growth of the Internet of Things (IoT) has given rise to multitude of services that are revolutionizing various industries. These services, including Smart Home Automation, Industrial IoT, and Autonomous Vehicles, are primarily driven by the integration of key technologies such as Cloud computing, Edge Computing and supported by architecture like microservices architecture. The combination of these technologies and architecture leads to the formation of what is referred to as the Cloud-Edge-IoT continuum, a dynamic environment characterized by collaborative efforts across multiple domains, active participation of various stakeholders, and multiple levels of responsibilities delegation. In this complex environment, the responsibilities of each actor involved can become blurred and it can be challenging to determine the specific roles and obligations of each party, this can lead to confusion, overlapping responsibilities, and potential gaps in fulfilling obligations. Additionally, legal and financial responsibility should be shared proportionally among all involved parties in the event of cyber attacks or service quality failures. It is crucial to implement efficient accountability and liability management mechanisms.

The objective of this thesis is to tackle the challenge of accountability and liability management through two aspects: the identification of responsibilities and the analysis of their effectiveness in the operational phase. To study these two aspects, we propose the following contributions: a descriptor which reflects the commitments taken by stakeholders throughout the supply chain and metrics which measure liability and trust with regards to those commitments. While several models exist in the continuum, none of them includes the notions of responsibility, accountability, and liability. To address this gap, we propose TRAILS (sTakeholder Responsibility, AccountabIlity and Liability deScriptor), a modular and generic descriptor. It tracks a component or service throughout its lifecycle, enabling all supply chain participants to outline their commitments. TRAILS is paired with an ontology, which is used to evaluate its associated profile in terms of a referencing policy or to examine the ontology's

content. Based on the TRAILS descriptor and using machine learning network methods, we introduce three categories of metrics to assess liability and trust. The first category, Commitment Trust Scores, aims to evaluate the level of confidence in a specific instance of a service, a service based on the observations collected on each of its instances as well as providers based on the observations collected from all the services they provide, based on the commitments outlined in the models. These scores enable the categorization of confidence levels regarding whether the behavior aligns with the expected commitments. The second category, Financial Exposure, focuses on quantifying the potential financial loss for a service provider as a whole, considering the current composition of services. This metric provides insights into the monetary risks associated with the existing service arrangements. Finally, the third category, Commitment Trends, involves tracking patterns and trends in breach rates of Service Level Agreements (SLAs) and the confidence level of instance commitments. By analyzing these trends, it becomes possible to predict future breaches and identify potential areas of concern.

We implemented our contributions in a framework called LASM (Liability-Aware Security Manager). LASM is a modular tool designed to help service providers in incorporating liability considerations when building, running, and managing architectures that involve multiple sub-components (such as microservices, infrastructure, or hybrid solutions) provided by various service providers. An ontology is employed within the LASM in order to reason about the responsibility models.

Keywords: Cloud-Edge-IoT Continuum, Responsibility Management, Liability Management, Accountability Management, Trust, 5G, NFV.

Résumé

La croissance rapide du nombre d'objets connectés (Internet des Objets, ou IoT) a donné naissance à une multitude de services révolutionnant diverses industries. Ces services, tels que l'automatisation des maisons intelligentes, l'IoT industriel et les véhicules autonomes, sont principalement encouragés par l'intégration de technologies clés telles que le cloud computing, le edge computing et soutenus par des architectures telles que l'architecture microservices. La combinaison de ces technologies et architectures conduit à la formation de ce que l'on appelle le continuum Cloud-Edge-IoT, un environnement dynamique caractérisé par des efforts collaboratifs dans plusieurs domaines, la participation active de divers intervenants et plusieurs niveaux de délégation de responsabilité. Dans cet environnement complexe, les responsabilités de chaque acteur impliqué peuvent devenir floues ; il peut être difficile de déterminer les rôles spécifiques et les obligations de chaque partie, ce qui peut entraîner confusion, chevauchement des responsabilités et des lacunes potentielles dans l'accomplissement des obligations. De plus, la responsabilité juridique et financière devrait être partagée de manière proportionnelle entre toutes les parties impliquées en cas d'attaques cybernétiques ou de défaillances de qualité de service. Il est crucial de mettre en place des mécanismes efficaces de responsabilité et de gestion des responsabilités.

Cette thèse répond à deux dimensions de la gestion de la responsabilité, à savoir l'identification des responsabilités et l'analyse de leur effectivité. La première contribution de la thèse est un descripteur reflétant les engagements pris par les intervenants tout au long de la chaîne d'approvisionnement. La seconde contribution consiste en des métriques mesurant la responsabilité et la confiance à l'égard de ces engagements. Bien qu'il existe plusieurs modèles dans le continuum, aucun d'eux n'englobe les notions de responsabilité, d'obligation de rendre des comptes, de responsabilité juridique. Pour combler cette lacune, nous proposons TRAILS (sTakeholder Responsibility, AccountabIlity and Liability deScriptor), un descripteur modulaire et générique. Il suit un composant ou un service tout au long de

son cycle de vie, permettant à tous les participants de la chaîne d'approvisionnement de définir leurs engagements. TRAILS est associé à une ontologie, qui est utilisée pour évaluer le profil associé sur la base d'une politique de référencement ou pour interroger l'ontologie. En utilisant le descripteur TRAILS ainsi que des méthodes d'apprentissage automatique, nous introduisons trois catégories de métriques pour évaluer la responsabilité et la confiance. La première catégorie, les Scores de Confiance des Engagements, vise à évaluer le niveau de confiance dans une instance spécifique d'un service, un service spécifique ainsi que sur les fournisseurs, et cela basé sur les observations recueillies auprès de tous les services qu'ils fournissent, en fonction des engagements décrits dans les modèles. Ces scores permettent de catégoriser les niveaux de confiance quant à savoir si le comportement est conforme aux engagements attendus. La deuxième catégorie, l'Exposition Financière, se concentre sur la quantification de la perte financière potentielle pour un fournisseur de services dans son ensemble, en tenant compte de la composition actuelle des services. Cette métrique fournit des informations sur les risques monétaires associés aux arrangements de services existants. Enfin, la troisième catégorie, les Tendances des Engagements, implique le suivi des modèles et des tendances en matière de taux de violation des accords de niveau de service (Service Level Agreement, SLA) et du niveau de confiance des engagements d'instance. En analysant ces tendances, il devient possible de prédire les violations futures et d'identifier les domaines potentiels de préoccupation.

Nous avons implémenté nos contributions dans un outil appelé LASM (Liability-Aware Security Manager). LASM est un outil modulaire conçu pour aider les fournisseurs de services à intégrer les considérations de responsabilité lors de la construction, de l'exécution et de la gestion d'architectures impliquant de multiples sous-composants (tels que des microservices, une infrastructure ou des solutions hybrides) fournis par divers fournisseurs de services. Une ontologie est utilisée au sein du LASM afin de raisonner sur les modèles de responsabilité.

Mots-clés : Cloud-Edge-IoT Continuum, Confiance, Gestion de la responsabilité, Gestion de l'imputabilité, Gestion des pénalités.

Contents

Acknowledgements **i**

Abstract **iii**

Résumé **v**

List of Tables **xiii**

List of Figures **xv**

Chapters

1 Introduction **13**

1.1 Introduction 14

1.2 Context 14

1.3 The objectives of the thesis 17

1.4 Motivating example: Smart IoT Campus Service 20

1.5 The manuscript's structure 22

1.6 Publications 23

1.6.1 International conference with proceedings and selection committee 23

1.6.2 International workshop with proceedings and selection committee 23

1.6.3 Technical Reports 24

1.6.4 Protected Software (Agency for the Protection of Programs) 24

1.6.5 Journal Paper under review for a major revision 24

2 Background **25**

2.1 Introduction . . . 26
2.2 Liability and Accountability in computing . . . 26
2.2.1 Definition . . . 26
2.2.2 Rationale . . . 27
2.2.3 Related concepts . . . 32
2.3 Describing and Orchestrating Cloud Services - TOSCA . . . 34
2.3.1 TOSCA - General Concept . . . 34
2.3.2 TOSCA - Conceptual Layers . . . 35
2.3.3 TOSCA Entities & the concept of substitution mapping . . . 36
2.3.4 Packaging . . . 38
2.3.5 TOSCA Simple Profile for Network Functions Virtualization (NFV) . . . 38
2.4 Ontology . . . 38
2.4.1 Fundamental Concepts . . . 39
2.4.2 Simple Protocol and RDF Query Language (SPARQL) . . . 40
2.4.3 Semantic Web Rule Language (SWRL) . . . 41
2.5 Machine Learning and Neural Network Algorithms . . . 41
2.5.1 K-means . . . 41
2.5.2 Artificial Neuron & Artificial Neural Network (ANN) . . . 41
2.5.3 Multi-Layer Perceptron (MLP) . . . 42
2.5.4 Self-Organizing Map (SOM) . . . 46
2.5.5 Dataset Transformation . . . 50
2.5.6 Assessment Software . . . 51
2.6 Conclusion . . . 52

3 State-of-the-Art 53

3.1 Introduction . . . 54
3.2 Coexisting Profiles in the Cloud-Edge-IoT Continuum . . . 54
3.2.1 Introduction . . . 54
3.2.2 Inspire-5Gplus Manifest . . . 54
3.2.3 Service-Level Agreement (SLA) . . . 57
3.2.4 Existing Profiles on the Internet of Things (IoT) Ecosystem . . . 68

3.2.5 Existing profiles in the Network Function Virtualization (NFV) ecosystem . . . 73
3.2.6 Conclusion . . . 79
3.3 Liability and Trust Metrics . . . 81
3.3.1 Introduction . . . 81
3.3.2 Trust Computation within the Cloud-Edge-IoT Continuum . . . 81
3.3.3 Liability metrics within the Cloud-Edge-IoT Continuum . . . 83
3.3.4 Monitoring and Detecting SLA Breaches . . . 84
3.3.5 Financial Exposure To Risk Metric . . . 85
3.3.6 Conclusion . . . 86

4 Contribution 1: TRAILS, Extending TOSCA NFV Profiles for Liability Management in the Cloud-Edge-IoT Continuum 89
4.1 Introduction . . . 90
4.2 Energy-aware Service-Level Agreements in 5G NFV architecture . . . 90
4.2.1 VNFD Energy Extension . . . 91
4.2.2 Energy-aware Service Level Agreement template for Network communications . 94
4.2.3 Exploring Use Cases: Practical Scenarios . . . 95
4.2.4 Conclusion . . . 97
4.3 TRAILS: Extending TOSCA NFV profiles for liability management in the Cloud-to-IoT continuum . . . 98
4.3.1 Introduction . . . 98
4.3.2 The TRAILS Model . . . 98
4.3.3 Illustrative Example . . . 101
4.3.4 Evaluation . . . 104
4.3.5 Discussion, Conclusion and Perspective . . . 113

5 Contribution 2: The Liability and Trust Metrics 115
5.0.1 Introduction . . . 116
5.0.2 LASM Analysis Service (LAS) Architecture . . . 116
5.0.3 Instance Trust Score (ITS) . . . 119
5.0.4 MicroService Trust Score (MTS) and Service Provider Trust Score (SPTS) . . 119

5.0.5 Temporal Evolution on the Self-Organized Map of the ITS and the SLA Violation Risk . . . 120
5.0.6 Financial Exposure to Penalty Risk (FEPR) . . . 121
5.0.7 Evaluation and Result . . . 121
5.0.7.1 Use case n°1 - PacketFabric SLA . . . 122
5.0.7.2 Results . . . 124
5.0.7.2.1 Instance Trust Score . . . 124
5.0.7.2.2 Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS) . . . 126
5.0.7.2.3 Trend Variations of Instance Trust Score and SLA Violation Rate . . . 127
5.0.7.2.4 Financial Exposure to Penality Risk . . . 130
5.0.7.3 Use case n°2 - Edgex SLA: . . . 130
5.0.7.4 Results . . . 133
5.0.7.4.1 Instance Trust Score . . . 133
5.0.7.4.2 Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS) . . . 136
5.0.7.4.3 Trend Variations of Instance Trust Score and SLA Violation Rate . . . 136
5.0.7.4.4 Financial Exposure to Performance Risk . . . 139
5.0.8 Conclusion, Discussion and Future Work . . . 140

6 Conclusion **143**
6.1 Synthesis of the Context and Problem Statement . . . 144
6.2 Contributions . . . 145
6.3 Perspectives . . . 146

Bibliography **149**

Appendices

A TRAILS Grammar **171**

B ACM MobiCom2023 Poster **219**

List of Tables

1 Conformité aux exigences du manifeste INSPIRE-5Gplus, version avec TRAILS 9

3.1 Inspire-5Gplus manifest characteristics 55
3.2 Mandatory and Recommended manifest attributes of the SUIT manifest. 69
3.3 General characteristics of VNFD 75
3.4 General Characteristics of NSD 78
3.5 Compliance with Inspire-5Gplus manifest requirements 1 80

4.1 NFVI reference attribute 93
4.2 Capacity level attribute 94
4.3 The Energy attributes 94
4.4 Energy-aware SLA metrics 95
4.5 Results of measurement tests 96
4.6 Compliance with Inspire-5Gplus manifest requirements 2 105

5.1 Time Windows and Corresponding ITS (Comply: as expected, ↓ : less than expected, ↑ : higher than expected) 123
5.2 Classification Results 125
5.3 MTS and SPTS (L: Low, M: Medium, H: High) 127
5.4 SOM - Evaluation 127
5.5 SLA penalties for Edgex: Availability, Latency and Error Rate 132
5.6 Time Windows and Corresponding ITS (Comply : as expected, ↓ : less than expected, ↑ : higher than expected) 133
5.7 Classification Results. 134
5.8 Microservice Trust Score and Service Provider Trust Score (L: Low, M: Medium, H: High) 136

5.9 SOM: Evaluation 137

List of Figures

1 Structure d'un nœud TRAILS 9

1.1 The Cloud-Edge-IoT Continuum 15
1.2 Liability & Accountability Management Functional Blocks [27] 18
1.3 LASM Architecture — Highlighting Contributions n°1 and n°2 and their outputs . . . 20
1.4 Smart IoT Campus Service — Liability and Accountability challenges 22

2.1 Responsibility between the Responsible Entity and Assigning Entity 27
2.2 Relation of TOSCA concepts [17] 35
2.3 Simplified UML class diagram of the TOSCA entities 37
2.4 Composition of services in TOSCA 37
2.5 MLP Learning Process 43
2.6 Hexagonal grid for 4x5 map 46

3.1 Inspire-5Gplus manifest lifecycle 56
3.2 ETSI SLA Model [93] 59
3.3 WSLA Meta-Model [98] 60
3.4 WS-Agreement Meta-Model [98] 62
3.5 SLA* Meta-Model [100] 63
3.6 CSLA Meta-Model [95] 64
3.7 MUD architecture [4] 71
3.8 MUD & IoT lifecycle [123], The arrows in the diagram indicate the flow and relationship between different stages of both the IoT lifecycle and the MUD lifecycle 73
3.9 VNF Package Lifecycle [5] 77

4.1 VNFD Extension Methodology 93

4.2 Extension of the TOSCA NFV metamodel 99
4.3 High-level structure of a TRAILS node 100
4.4 TRAILS MEC Streaming Service Directory Tree 104
4.5 TRAILS's topologic view 111
4.6 TRAILS's responsibility view 112
4.7 Time evolution to perform a request to the regard to the size of the ontology 112

5.1 Overview of LASM Analysis Service (LAS) 117
5.2 Confusion matrix 125
5.3 One-vs-Rest ROC curves: High Trust vs (Medium Trust & Low Trust) 126
5.4 ITS-TV Map with data insight. 128
5.5 SVR-TV Map with data insight. 129
5.6 Test environment for Use Case 2, VM indicates in which virtual machine the services are deployed 131
5.7 Confusion matrix 134
5.8 One-vs-Rest ROC curves: High Trust vs (Medium Trust & Low Trust) 135
5.9 ITS-TV Map with data insight. 138
5.10 SVR-TV Map with data insight. 139

B.1 ACM MobiCom2023 Poster Presenting the Liability-Aware Analysis Service 220

Résumé de Thèse en Français.

Cette thèse aborde la problématique de la gestion de la responsabilité et de la confiance au sein du continuum Cloud-Edge-IoT.

L'essor de l'Internet des Objets (IoT) transforme radicalement les industries et la société, en améliorant la connectivité, la productivité et la prise de décision grâce aux données collectées. Néanmoins, gérer le volume massif de données générées représente un défi majeur. Les architectures cloud actuelles, orientées vers un traitement centralisé des données, peinent à répondre aux besoins spécifiques des applications IoT, surtout en termes de latence et de réactivité. En réponse, le continuum Cloud-Edge-IoT a été développé. Ce continuum intègre des technologies essentielles telles que les concepts de NFV (Network Function Virtualization), de 5G, de Edge Computing et de SDN (Software Defined Network) ainsi que les architectures microservices, créant ainsi une infrastructure robuste et efficace, optimisée pour la latence et la bande passante. Des projets comme l'initiative européenne Cloud, Edge & IoT Continuum [**?**] œuvrent pour l'intégration de ces technologies, stimulant l'innovation et établissant des standards d'interopérabilité.

Le continuum Cloud-Edge-IoT, de par sa nature dynamique, crée des défis en matière de gestion de la responsabilité. Cette architecture fusionne divers domaines où chacun opérant selon ses propres standards et protocoles et chaque domaine agit comme une entité légale distincte. La nature multi-acteurs du continuum, qui inclut des participants variés comme les fournisseurs de services, les fabricants de composants réseau ou les développeurs d'applications, rend la détermination des responsabilités complexe. Cette complexité est amplifiée par le chevauchement et l'évolution constante des rôles de chaque acteur. Aussi, les multiples niveaux de délégation ajoutent une couche supplémentaire de complexité. L'attribution de responsabilité en cas d'incidents ou de manquements peut s'avérer ardue, du fait de la réaffectation fréquente des obligations au sein de cette hiérarchie. La complexité inhérente aux divers aspects du continuum Cloud-Edge-IoT appelle à l'élaboration de nouveaux outils pour une gestion efficace de la responsabilité et de la confiance.

Nous avons identifié trois blocs fonctionnels (en anglais `Functional Block`, FB) essentiels pour la gestion de la responsabilité, qui comprennent : 1) la définition des responsabilités, 2) la surveillance et la collecte de preuves, et 3) l'analyse et l'attribution de responsabilité. Ces blocs fonctionnent en synergie pour assurer une gestion efficace de la responsabilité et de la confiance dans des environnements complexes tels que le Cloud-Edge-IoT. Cette thèse présente deux contributions qui se concentrent respectivement sur le premier et le troisième bloc fonctionnel. La première contribution de cette

thèse est la création d'un descripteur, TRAILS (`sTtakeholder Responsability, AccountabIlity, Liability deScriptor`), qui capture les engagements des différentes parties prenantes à travers la chaîne d'approvisionnement. TRAILS est un descripteur modulaire et générique. Il suit un composant ou un service durant tout son cycle de vie, offrant aux acteurs de la chaîne d'approvisionnement la possibilité de spécifier clairement leurs engagements. De plus, TRAILS est lié à une ontologie, qui sert à évaluer le profil associé en fonction d'une politique de référence.

La deuxième contribution concerne la collection de preuves à travers des métriques de responsabilité et de confiance. Dans un deuxième temps, nous définissons des métriques de responsabilité et de confiance, regroupées en trois catégories : les Scores de confiance, l'Exposition financière et le Suivi des tendances. Pour démontrer et évaluer ces contributions, nous avons développé l'outil LASM pour `Liability-Aware Security Manager`.

Nous avons réalisé deux revues de la littérature, l'une axée sur les modèles et descripteurs actuellement utilisés dans le continuum Cloud-Edge-IoT, et l'autre sur les métriques de responsabilité et de confiance. En ce qui concerne le premier état de l'art, notre objectif était d'examiner les profils existants, puis de comparer leurs caractéristiques aux exigences du manifeste INSPIRE 5G+ [1]. Le manifeste INSPIRE-5Gplus formalise les concepts de responsabilité, `d'accountability` (obligation de rendre des comptes) et de `liability` (la responsabilité légale d'une entité envers une autre pour des actions entraînant des conséquences juridiques et financières) dans un document structuré. Il permet aux acteurs de définir leurs engagements et les conditions associées de manière claire. Ce document modulaire facilite l'attribution de responsabilités et la clarification des preuves de conformité requises, s'appliquant à divers composants ou services. Les éléments du manifeste correspondent directement aux clauses contractuelles, incluant obligations, conditions d'utilisation, objectifs, récompenses et sanctions, avec un expert juridique garantissant leur cohérence.

Dans l'écosystème IoT, différents profils IoT ont été développés pour répondre à divers défis et exigences. Ces profils proposent des solutions standardisées pour des besoins variés tels que la communication fluide entre appareils, la sécurité robuste et la gestion efficace des appareils. Nous décrirons les principaux profils comme le MUD (Manufacturer Usage Definition) et le manifeste SUIT (Software Updates for Internet of Thing).

Manifeste SUIT : l'IETF (`Internet Engineering Task Force`) a créé une solution de mise à jour du firmware adaptée aux besoins uniques des appareils IoT [2]. Ce processus protège contre les

modifications non autorisées du firmware, préservant l'intégrité et la confidentialité des images du firmware. Le manifeste SUIT joue un rôle clé dans le processus de validation de la mise à jour, contenant des informations essentielles pour l'intégrité de l'image, son applicabilité, les considérations de stockage, etc.

Lightweight M2M (LwM2M) Data Model : LwM2M, développé par OMA SpecWorks [3], est un protocole pour la gestion à distance des appareils IoT et autres applications M2M. Il utilise le protocole CoAP pour encapsuler les données applicatives. Trois entités interagissent : les Clients LwM2M sur les appareils finaux, le Serveur Bootstrap LwM2M pour l'initialisation, et le Serveur LwM2M pour la maintenance des connexions. Le modèle de données LwM2M se compose de deux niveaux. Au premier niveau, on trouve des objets avec des attributs comme Nom, ID, Instances (simples ou multiples), et le statut obligatoire ou non. Au second niveau, les ressources de chaque objet sont définies avec des attributs comme ID, Nom, Type d'opérations (lecture, écriture, etc.), Instances, statut Obligatoire ou non.

Manufacturer Usage Definition (MUD) profile : pour contrer les risques de cybersécurité dans l'IoT, le MUD de l'IETF contrôle le comportement des appareils IoT pour une déploiement sécurisé [4]. Les fabricants définissent des profils de comportement pour les appareils, en utilisant des politiques ou listes de contrôle d'accès (`Access Control List`, ACL) pour réduire les surfaces d'attaque. Le MUD est adopté dans les milieux de recherche et de normalisation, notamment par le NIST (`National Institute of Standards and Technology`).

L'architecture NFV utilise des descripteurs pour représenter et gérer les fonctions et services réseau dans des environnements virtualisés. Ces descripteurs standardisés facilitent l'automatisa- tion et l'orchestration. Les principaux descripteurs dans l'architecture NFV sont le Descripteur de Fonction Réseau Virtuel (VNFD) et le Descripteur de Service Réseau (NSD).

Virtual Network Function Descriptor (VNFD) : un VNFD définit le déploiement et le comportement opérationnel d'une fonction réseau virtualisée (VNF) [5]. Il se structure en trois composants clés : la topologie, les aspects de déploiement, et les opérations de gestion du cycle de vie (LCM) du VNF.

Network Service Descripto (NSD) : un NSD [6] est une composition de composants réseau définis par des VNFD où les connexions sont des liens virtuels. Les liens virtuels sont décrits à l'aide des `Virtual Link Descriptor` (VLD) et la topologie est décrite avec le `VNF Forwarding Graph Descriptors`

(VNFFGD).

Les modèles et profils présentés visent divers objectifs comme renforcer la sécurité, déployer et gérer des composants ou services réseau. Notre objectif est de combler les lacunes identifiées. Les profils étudiés montrent un manque d'expression d'engagement, une absence de spécification claire de l'`accountability`, et ne traitent pas la `liability`. Seuls les descripteurs VNFD et NSD, utilisant TOSCA [7], satisfont au critère de modularité. Cependant, leur spécificité limite leur généralité sur le continuum Cloud-Edge-IoT.

Le deuxième volet de notre revue de littérature se focalise sur les modèles de confiance et de responsabilité, explorant les méthodes de mesure de ces deux aspects dans l'écosystème Cloud-Edge-IoT, en fonction des engagements établis dans les SLA (Service Level Agreement). De plus, nous portons une attention particulière à la détection des violations de SLA ainsi qu'à la gestion des risques financiers dans le cadre des SLAs. L'étude de Govindaraj *et al.* [8] classe les modèles de confiance dans le cloud en trois catégories : basés sur les recommandations, la réputation et les SLA (Service Level Agreement). Nous nous intéressons particulièrement aux modèles basés sur les SLA (car les engagements sont pris à l'aide de ce document).

Le modèle basé sur les SLA utilise ces derniers pour établir et mesurer la confiance entre fournisseurs et consommateurs. Les travaux comme ceux de Huang [9] soulignent que la surveillance de la QoS (Quality of Service) et la vérification des SLA sont essentielles pour la gestion de la confiance dans le cloud computing. Chandrasekar *et al.* [10] suggèrent une technique de surveillance de la QoS et une méthode de calcul de la confiance dynamique utilisant une approche basée sur l'état du système pour réduire les données réseau.

Concernant la responsabilité dans le cloud computing, des initiatives telles qu'A4Cloud (Accountability for Cloud) [11] ont développé des outils pour renforcer la gestion des données. Le projet TrustCloud [12] a souligné l'importance de la responsabilité dans le cloud, tandis que Cloudacc a visé la confiance dans les clouds fédérés. Pour la surveillance des SLA, des outils comme Sandpiper et SLA@SOI offrent des solutions de gestion et de détection des violations. Sur le plan des risques financiers, des études ont appliqué l'analyse des risques à l'économie des grilles, et le projet AssessGrid a exploré la négociation de contrats avec une prise en compte des risques. Cependant, la littérature manque de métriques pour évaluer la fiabilité des services cloud. Notre contribution cherche à pallier ce manque en proposant une évaluation complète des services et une nouvelle visualisation pour le

suivi des SLA, ainsi qu'une métrique pour les risques financiers, offrant une perspective évolutive dans le temps.

En ce qui concerne la responsabilité et la transparence des fournisseurs de services cloud, des projets comme A4Cloud [11] ont développé des outils et des modèles pour améliorer le contrôle et la transparence des données dans le cloud. Ko *et al.* [12] ont mis en avant l'urgence de la recherche sur la responsabilité dans le cloud avec leur outil TrustCloud. En ce qui concerne la détection des violations de SLA, plusieurs outils de surveillance orientés open-source sont disponibles, chacun offrant des fonctionnalités spécifiques. Wood *et al.* [13] introduisent Sandpiper, un outil qui automatise la surveillance, la détection des points chauds et la reconfiguration des VMs. Comuzzi *et al.* [14] proposent SLA@SOI, un outil incorporant une surveillance basée sur les SLA et la gestion des pénalités. Dans le contexte de la gestion des risques financiers, Antonopoulos et al. [15] discutent de l'application des techniques d'analyse des risques financiers à l'économie des grilles pour assurer la disponibilité, la capacité et la responsabilité en relation avec les applications financières. Ils construisent leur SLA de grille en considérant le prix relatif des ressources de différentes spécifications et le risque associé à l'incapacité d'un élément du portefeuille à fonctionner ou à compléter sa tâche dans un délai limité.

La revue de la littérature révèle une absence de métriques permettant d'évaluer un score de confiance pour une instance de service, une classe de services et le fournisseur de services. Le travail de Valer *et al.* [16] se concentre sur la sélection des services des parties prenantes dans un marché, mais reste spécifique au domaine de la 5G et n'offre pas la généralisation que nous réalisons avec notre contribution. Notre proposition vise à fournir une évaluation globale pour un service. Concernant la détection des violations de SLA, les outils présentés se concentrent principalement sur le cloud, utilisant une méthode commune de définition d'un seuil et de vérification si les observations le dépassent. En revanche, notre contribution vise à fournir une visualisation améliorée de l'évolution des SLAs. En utilisant une carte, nous définissons différentes zones pour faciliter une meilleure interprétation de la progression des SLA. En ce qui concerne la métrique d'exposition aux risques financiers, il semble y avoir un manque dans la littérature. Excepté le travail d'Antonopoulos *et al.* [15], aucune autre proposition pour une telle métrique n'est apparente. Ce qui distingue notre proposition est sa capacité à observer son évolution dans le temps.

Notre apport réside dans des métriques de confiance et responsabilité pour le Cloud-Edge-IoT, plus générales que les solutions existantes, avec une meilleure visualisation des SLA et une nouvelle

métrique de risque financier.

La première contribution de cette thèse est le modèle de responsabilité nommé TRAILS, étend le profil TOSCA NFV en intégrant une description de la responsabilité de la chaîne d'approvisionnement. Cette contribution, alignée sur le bloc FB.1, a été acceptée à la conférence Netsoft 2022. La deuxième contribution, liée au bloc FB.2, concerne l'établissement de métriques de responsabilité et de confiance, soumises et acceptées avec des révisions pour le journal IEEE TNSM (Transactions on Network and Service Management) numéro spécial `Networks, Systems and Services Operations and Management through Intelligence`.

Nous avons également mis en œuvre l'outil LASM, qui comprend plusieurs modules aidant dans la gestion des services et des profils TRAILS. Les modules LASM Referencing Service (LRS), LASM Visualized Service (LVS), et LASM Creation Service (LCS) appuie la première contribution, tandis que le LASM Analysis Service (LAS) appuie la seconde. Des présentations sur ces outil ont été réalisées au Salon De La Recherche 2022 d'Orange et à la conférence ACM MobiCom 2023.

L'état de l'art met en évidence l'absence d'un modèle pour décrire la responsabilité, `l'accountability` et la `liability` pour un service dans le Cloud-Edge-IoT continuum. Nous introduisons TRAILS pour pallier ce manque et améliorer la gestion de la responsabilité dans le continuum. TRAILS étend les profils TOSCA NFV [17] pour unifier les profils existants dans l'écosystème Cloud-IoT-Edge. Il permet de suivre un composant ou un service tout au long de son cycle de vie, permettant à tous les participants de la chaîne d'approvisionnement de définir leurs engagements. TRAILS prend la forme d'une archive qui suit le format CSAR (Cloud Service ARchive), largement adopté par de nombreux fournisseurs de services cloud. Ce modèle s'aligne sur FB.1 et sert de composant fondamental pour une gestion de la responsabilité.

En suivant le concept de séparation des préoccupations de Dijkstra, nous avons élaboré la structure de données TRAILS, présentée dans la Figure 4.3. L'en-tête (`header`) fournit un aperçu global du composant ou du service, identifiant son type, son modèle, ainsi que l'entité qui porte la responsabilité globale, le `LeadAuthor`. La validation (`Validation`) documente la date, l'acteur, la portée et les résultats de la validation du composant/service. La propriété `Authors` liste tous les acteurs intervenus lors de la création du composant/service. `Commitment` détaille les engagements d'un acteur, à l'aide du SLA, et les caractéristiques du service. `Usage condition` établit les prérequis nécessaires pour que les engagements pris sur le composant/service soit valable, incluant les dépendances matérielles

et logicielles, l'intégration des sous-services, ou encore le comportement réseau attendu du service. `Commitments` et `Usage conditions` détaillent les facettes complémentaires de la responsabilité liée au composant/service.

La `liability` est assurée par le fait que les propriétés sont signées par leur auteur ou la partie responsable à l'aide d'une paire de clés publique/privée gérée via une Infrastructure à Clé Publique (PKI). Nous distinguons les auteurs qui prennent la responsabilité d'une propriété spécifique et les `LeadAuthors` qui intègrent plusieurs composants et propriétés fournis par d'autres acteurs. Ainsi, les auteurs ne signent que les propriétés auxquelles ils s'engagent, tandis que les LeadAuthors signent toutes les propriétés dans le cadre de l'intégration qu'ils ont réalisée. Pour ce faire, nous séparons les engagements et les propriétés dans des fichiers que les auteurs peuvent signer individuellement. Nous les regroupons ensuite dans une archive CSAR qui est signée par le LeadAuthor concerné. La séparation est également démontrée par le fait que le modèle TRAILS peut être utilisé pour étudier un composant de réseau sous l'angle de sa topologie ou de ses chaînes de responsabilité. La vue de la topologie est un graphe orienté où chaque composant est un nœud et chaque lien décrit une connexion entre deux nœuds. La vue de la responsabilité est un graphe orienté avec une racine (le LeadAuthor final qui propose le service modélisé). Chaque sommet représente un couple d'un Auteur et d'une Réclamation. Chaque arête orientée représente une responsabilité d'un acteur envers un autre. Les `Commitments` sont représentés par une arête d'un fournisseur vers son client, tandis que les `Usage condition` sont représentés par une arête d'un client vers son fournisseur.

Le Tableau 4.6 fournit une comparaison entre TRAILS et les profils étudiés dans l'état de l'art en tenant compte des critères décrits dans le manifeste INSPIRE-5Gplus. TRAILS remplit les critères de généralité car il peut être utilisé pour les dispositifs IoT, les VNF et les NS, et exploite des profils couramment utilisés pertinents pour chaque domaine tels que SUIT, les profils MUD et leurs extensions. TRAILS trace les responsabilités de chaque acteur impliqué dans la chaîne d'approvisionnement. Plusieurs parties prenantes impliquées dans la création d'un service peuvent définir leurs responsabilités indépendamment les unes des autres. Les fournisseurs de la chaîne d'approvisionnement peuvent définir des responsabilités pour eux-mêmes et leurs utilisateurs. Si les utilisateurs acceptent d'utiliser le service décrit par TRAILS, ils peuvent définir leurs propres responsabilités et l'intégrer comme un nouveau service. Dans ce cas, un TRAILS peut être généré. Ainsi, TRAILS remplit simultanément les critères de responsabilité et de modularité. Il convient de

noter que TRAILS offre également la traçabilité des services. La `liability` est exprimée dans TRAILS par les SLA. La signature des engagements, ainsi que les conditions d'utilisation (`usage condition`), contribuent à atteindre les critères de responsabilité. Aussi, TRAILS assure `l'accountability` en incluant les SLI (Service Level Indicator), qui fournit la preuve que les résultats ont été atteints ou non.

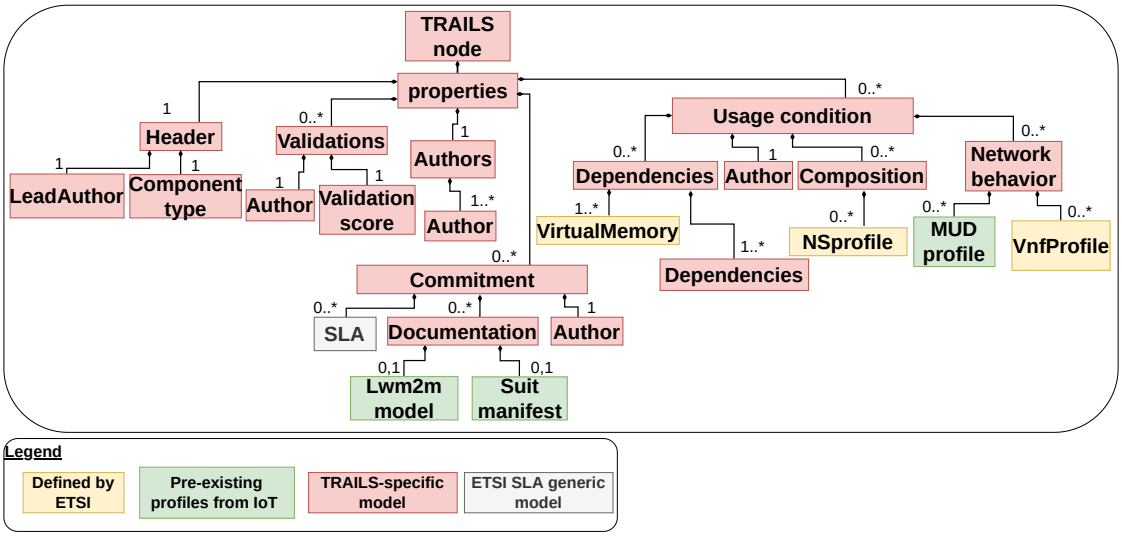


Figure 1: Structure d'un nœud TRAILS

Features	VNFD	NSD	MUD profile & extension	SUIT manifest	LwM2M model	TRAILS
Responsibility	□	□	□	□	-	■
Accountability	-	-	-	-	-	■
Liability	-	-	-	-	-	■
Modularity	■	■	-	-	-	■
Genericity	-	-	-	-	-	■

■: la propriété est prise en charge
□ : la propriété est partiellement prise en charge
- : la propriété n'est pas prise en charge

Table 1: Conformité aux exigences du manifeste INSPIRE-5Gplus, version avec TRAILS

La validité de la sémantique TRAILS a été démontrée en l'utilisant pour modéliser des composants et services réseau existants. Nous avons modélisé différents services : un moniteur IoT de pression artérielle de SmartMeter, un service de gestion IoT d'Amazon Web Service (AWS), un service de diffusion de contenu d'IBM et un service de réseau virtuel d'Equinix. Nous avons utilisé les manuels d'utilisation pour remplir les en-têtes TRAILS, lister les normes de validation, et définir les conditions

d'utilisation, notamment les fichiers MUD et les normes ISO. Les SLAs ont été intégrés dans les engagements (section `Commitment`) de TRAILS. Pour chaque service, TRAILS inclut au moins un validateur et un fournisseur de service, avec des signatures numériques pour chaque engagement. Pour finir, nous avons généré des archives CSAR signées par les `LeadAuthors` correspondants.

TRAILS et LASM offrent une assistance précieuse à l'administrateur du service dans la gestion des responsabilités à trois niveaux distincts. Initialement, lors du référencement d'un composant réseau pour la création de services, l'administrateur peut évaluer la conformité des solutions de sous-traitance aux politiques de cybersécurité. Ensuite, pour l'orchestration, il est possible de sélectionner des composants spécifiques qui garantissent la conformité des services aux exigences contractuelles. Enfin, dans le cadre de l'analyse des causes fondamentales, TRAILS et LASM contribuent à identifier la cause probable des problèmes et à évaluer les responsabilités associées, offrant ainsi un fondement solide pour les négociations juridiques sans recourir à des pénalités automatiques.

Après le descripteur TRAILS, nous étudions les métriques de confiance et de responsabilité créées par le LAS de la LASM, illustrées par l'exemple d'une architecture de microservices dans le continuum cloud-edge-IoT.

L'outil LAS prend en entré des ensembles de données étiquetés fournis par des experts en gestion des risques, ainsi que les SLA des fournisseurs de services (qui se trouvent dans TRAILS) pour générer trois catégories de métriques : Score de Confiance en l'Engagement, Exposition Financière et Tendances de l'Engagement. Le LAS calcule trois types de Scores de Confiance en l'Engagement, à savoir le Score de Confiance en l'Instance de Microservice (ITS), le Score de Confiance en le Microservice (MTS) et le Score de Confiance en le Fournisseur de Services (SPTS). Pour ce faire, il utilise le MLP (`Multi-Layer Percptron`) et la méthode k-means. Le LAS calcule l'Exposition Financière au Risque de Pénalité (FEPR) inspirée de la métrique d'exposition financière calculée dans le domaine des investissements. Enfin, deux types de Tendances de l'Engagement sont générés. À l'aide de SOM (`Self-Organizing Map`), le LAS suit les changements de l'ITS et du risque de violation des SLA au fil du temps. Cela génère deux autres résultats, à savoir la Variation de la Tendance du Score de Confiance de l'Instance (ITS-TV) et la Variation de la Tendance du Risque de Violation des SLA (SVR-TV).

Le LAS inclut un module de préparation de données et utilise un MLP pour classifier la fiabilité des données. On évalue les performances avec des métriques clés et en cas de dérive, le modèle est ré-entraîné plus rapidement. On emploie l'algorithme k-means pour définir des prototypes de service,

et des cartes SOM pour détecter des risques, déclenchant des alertes si nécessaire. La FEPR mesure le risque financier lié aux violations des SLA, calculé en fonction du taux de violation des SLA et des pénalités associées.

Pour illuster notre contribution, nous présentons l'application pratique du LAS à travers un cas d'utilisation en utilisant Edgex, un cadre logiciel pour la gestion des IoT. L'évaluation du LAS avec EdgeX a démontré son efficacité pour analyser les métriques de confiance et de responsabilité dans les microservices. Le modèle MLP a précisément classifié les niveaux de confiance des instances, reflétant leur adhérence aux SLA. Les scores MTS et SPTS ont varié en fonction de la conformité aux SLA, illustrant la capacité d'adaptation du LAS. Les cartes SOM ont offert une visualisation en temps réel des tendances et des risques de violation des SLA, soulignant des zones critiques pour une intervention proactive. La métrique FEPR a établi un lien entre la conformité aux SLA et les implications financières, soulignant l'importance de la gestion des SLA pour minimiser les risques financiers et maintenir la qualité des services.

Pour conclure, la thèse met en lumière les complexités de la gestion des responsabilités dans l'environnement Cloud-Edge-IoT et propose des solutions innovantes à travers TRAILS et le cadre LAS. TRAILS clarifie les engagements dans la chaîne d'approvisionnement, tandis que LAS évalue la fiabilité et les risques financiers des services. Les améliorations futures envisagées visent à renforcer la gestion des responsabilités et à adapter le LAS à des environnements de services plus complexes, utilisant des techniques d'apprentissage automatique avancées pour une analyse plus précise et dynamique.

Chapter 1

Introduction

Contents

1.1 Introduction **14**
1.2 Context **14**
1.3 The objectives of the thesis **17**
1.4 Motivating example: Smart IoT Campus Service **20**
1.5 The manuscript's structure **22**
1.6 Publications **23**
1.6.1 International conference with proceedings and selection committee 23
1.6.2 International workshop with proceedings and selection committee 23
1.6.3 Technical Reports 24
1.6.4 Protected Software (Agency for the Protection of Programs) 24
1.6.5 Journal Paper under review for a major revision 24

1.1 Introduction

This chapter introduces the context of our work, focusing on managing liability and trust in a dynamic, multi-actor, multi-domain environment with multiple layers of delegation. We provide a detailed overview of this environment and why it poses a challenge. We demonstrate the challenges through a concrete example. Lastly, we present the focus of our thesis and our main contributions, which revolve around defining supply chain responsibilities through a responsibility model and defining liability and trust metrics.

1.2 Context

The general context of the thesis revolves around managing liability and trust within the Cloud-Edge-IoT continuum.

The exponential expansion of the Internet of Things (IoT) revolutionizes industries and societies. It enables seamless connectivity, transforming business operations and human interaction with technology. It also enhances efficiency, productivity, and data-driven decision-making. Statista[1] predicts that by the year 2030, over 50 billion IoT devices worldwide will be connected to the Internet, generating an enormous volume of data. Managing and harnessing the potential of these massive data influx poses significant challenges. To ensure seamless service provision and meet criteria such as ubiquity, reliability, high performance, efficiency, and scalability, a robust architecture is crucial. Traditional cloud architectures face challenges meeting the performance demands of IoT applications that rely on real-time data processing and immediate responsiveness, like smart health or smart transport. Additionally, the rapid increase in IoT devices and data exacerbates network congestion and scalability issues within these architectures. Moreover, the distance between traditional cloud setups and IoT devices leads to higher communication overhead and reduced efficiency. This data transfer between IoT devices and remote clouds adds latency, especially impacting latency-sensitive applications. This issue has been extensively studied in both research and industry circles, leading to a common solution termed the Cloud-Edge-IoT continuum. For example, in [18], Lingen *et al.* propose a model-driven approach merging technologies such as cloud and edge computing to address this challenge. Also, the International Electrotechnical Commission (IEC) acknowledges the significance of the Cloud-Edge-IoT

[1] https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/

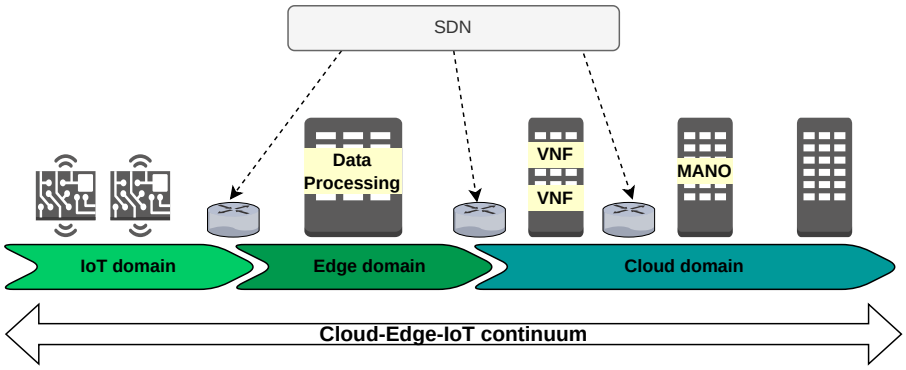


Figure 1.1: The Cloud-Edge-IoT Continuum

continuum in providing a robust and efficient infrastructure for seamless integration, scalable data processing, and real-time analytics across IoT devices, edge resources, and cloud platforms. Several groups have also been formed to help achieve this convergence, for example, A European collaborative initiative named The European Cloud, Edge & IoT Continuum [19] emerges to drive the integration of cloud, edge, and IoT technologies, foster innovation, and establish interoperability standards, ultimately accelerating the development and deployment of advanced applications and services across various industries within Europe. Additionally, Orange and INRIA (Institut national de recherche en sciences et technologies du numérique) have launched a joint laboratory focusing on the "Cloud to IoT" continuum to enhance research in network virtualization, cloud computing, and software infrastructures [20]. This partnership aims to drive innovation, develop advanced solutions, and establish seamless integration between cloud computing and IoT technologies. Europe showcases its importance within the Cloud-Edge-IoT continuum through its proactive investments in research and development pertaining to this domain. A notable example is Europe's provision of financial assistance for research projects specifically aimed at enhancing the security aspects of the continuum [21].

The Figure 1.1 provides an overview of the Cloud-Edge-IoT continuum architecture and the concepts it leverages. The following offers insights into diverse technologies, including Edge computing, 5G, NFV (Network Function Virtualization), MANO (MANagement and Orchestration), and SDN (Software-Defined Networking), all supported by Microservices architectures.

Edge computing. ETSI (European Telecommunications Standards Institute) defines the edge computing as a system which provides an IT service environment and cloud-computing capabilities at the edge of an access network which contains one or more types of access technology, and in proximity to its

users [22]. This technology plays a crucial role in the continuum by reducing latency, optimizing network bandwidth, enhancing scalability, and improving the overall performance of IoT systems. It enables efficient and real-time data processing at the edge, complementing the capabilities of cloud infrastructure and facilitating the seamless integration of IoT devices into the continuum architecture.

5G Technology. The advent of 5G, the fifth-generation wireless network, marks a significant leap forward with its promise of faster connectivity, reduced latency, expanded bandwidth, and enhanced scalability. It plays a pivotal role in the Cloud-Edge-IoT continuum by enabling efficient data transfer, supporting the extensive connectivity demands, and accommodating the processing needs of IoT devices within this integrated architecture [23].

NFV. This framework provides a solution for virtualizing network functions, facilitating adaptable resource allocation and scalability across the continuum. This approach supports the dynamic provisioning and management of network resources, including Virtual Network Functions (VNFs), to enhance their efficiency in response to the unique requirements of IoT applications.

SDN. This concept separates the control plane from the data plane in network infrastructure, providing centralized management and control. It enables efficient traffic routing, dynamic resource allocation, and network programmability, enhancing the flexibility and adaptability of the continuum.

MANO. It ensures efficient utilization of computing, networking, and storage resources, optimizing their allocation based on the requirements of the applications. It enables automated deployment and scaling of services, ensuring that the continuum can handle varying workloads and adapt to changing demands in real time.

Microservices architectures. This approach involves breaking down applications into smaller, independent services, creating a more granular structure. Each service is responsible for specific business functions and capabilities, ensuring independence from other services. This makes microservices highly adaptable for deployment in the continuum, offering reusability and requiring minimal centralized management and orchestration. They promote independence in service development and maintenance.

The complex nature of the continuum leads to various challenges, as addressed in [24, 25]. These challenges include privacy and security concerns, interoperability issues, efficient data management, and the selection of suitable communication protocols. In the context of this CIFRE thesis proposed by Orange Innovation Caen and in collaboration with the Networks and IoT Systems (Réseaux et

Objets Connectés, ROC) team that belongs to the computer science and communications department (Cedric), at CNAM Paris and as part of the European inspire-5Gplus project, we will explore how liability and trust can be managed in this environment. The difficulties arise from the dynamic essence of the architecture, characterized by its capacity for flexibility and adaptation to change. This adaptability enables the architecture to modify its structure or behavior to meet varying requirements or conditions. This continual evolution makes it challenging to establish a fixed accountability, as responsibilities might constantly shift. Additionally, the multi-domain aspect complicates matters due to the diverse nature of the involved domains (edge computing, cloud infrastructure, and IoT) each operating with distinct protocols, standards, and architectures. Integrating these domains involves overcoming interoperability issues, addressing differences in data formats, communication protocols, and security measures. Governance and ownership become challenging too, as each domain operates as a distinct legal entity with its own policies, regulations, and control mechanisms, making unified governance and decision-making difficult. Furthermore, the multi-actor nature of the Cloud-Edge-IoT continuum involves a wide range of entities: integrators, service providers, infrastructure providers, device manufacturers, and application developers. Each entity commits to specific responsibilities to deliver services, but in this complex environment, these responsibilities can become blurred. In a multi-tenant architecture, allocating responsibilities among stakeholders becomes particularly challenging, as highlighted in [26]. Finally, the continuum includes various tiers of delegation. Delegation refers to the assignment of duties from one entity to another, within an established hierarchical framework of authority. The continuum typically features multiple layers of delegation, which can obscure the delineation of responsibilities. The distribution of duties across different levels complicates the identification of ultimate accountability in the event of a breach.

1.3 The objectives of the thesis

Liability and accountability management require three Functional Block (FB) [27]. These FBs and their interconnections are visually depicted in Figure 1.2.

FB.1 comprises critical components essential for establishing service governance, ensuring compliance with regulations and contractual obligations, and defining liability relationships among involved actors. For instance, the Responsibility Assignment Matrix (RAM) clarifies project roles, aiding liability management and conflict resolution by specifying task ownership. Additionally, the Software

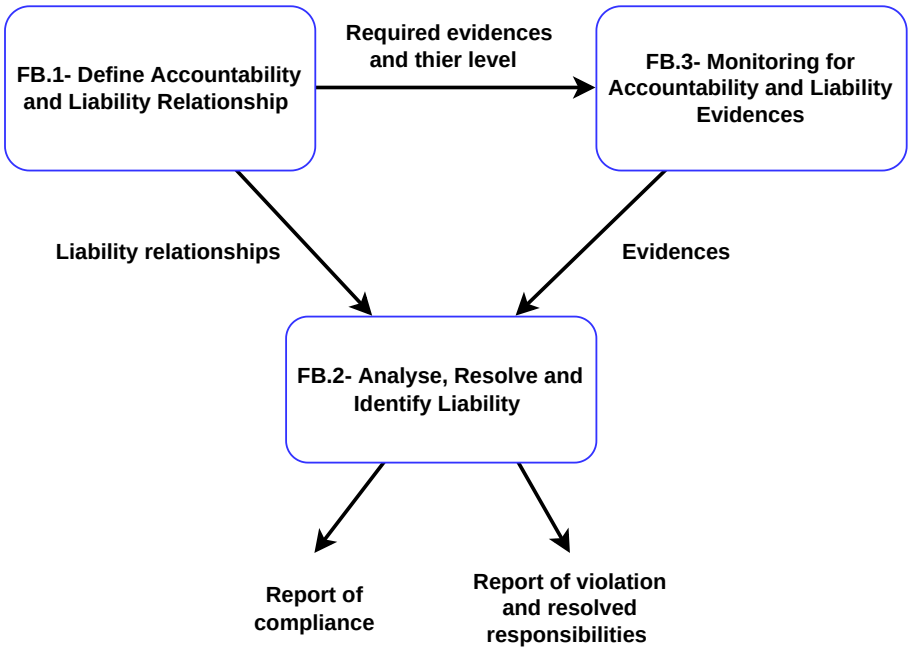


Figure 1.2: Liability & Accountability Management Functional Blocks [27]

Bill of Materials (SBOM) [28] catalogues software components, boosting transparency in the supply chain and outlining contributors' roles and liabilities. Finally, the Service Level Agreement (SLA) sets contractual terms, responsibilities, liability limits, and evidence requirements in case of service disruptions or breaches within the provider-user relationship. The second FB in the liability and accountability management process is centered around the crucial task of monitoring for accountability evidence. Once the governance framework, regulatory compliance measures, and contractual obligations have been established in the FB-1, it is essential to demonstrate compliance and to identify and trace events or incidents. For that we have mechanisms such as forensics, logging, and auditing that form essential components in liability management. Forensic analysis investigates incidents and breaches, identifying root causes and responsible parties. Logging records critical data, tracing events for an objective timeline. Auditing examines this data to assess any violations of agreements. Also, remote attestation that validates a system's security posture, providing verifiable proof of compliance. Finally, Root Cause Analysis that identifies underlying reasons for incidents, aiding in liability assessment and prevention. FB.3's primary objective involves analyzing the evidence of events/incidents collected by FB.2. Leveraging the liability relationships identified by FB.1, FB.3 evaluates compliance or potential violations and assigns responsibilities accordingly. Furthermore,

FB.3 generates reports for administrators or jurists, supporting forensic investigations and facilitating dispute resolution. Additionally, the outputs from FB.3 can be utilized by billing systems to calculate penalties or expected remediation, whether from the customer to the Service Provider or from the Service Provider to its subcontractors. In this thesis, we focus on FB.1 and FB.3, with the following contributions:

- The definition of accountability and liability relationship through TRAILS (sTakeholder Responsibility, AccountabIlity, Liability deScriptor) A modular and generic descriptor and its associated ontology which incorporate notions related to responsibility, accountability, and liability of the supply chain.

- Accountability and liability evidence through three categories of metrics to assess liability and trust, namely the Commitment Trust Score, Financial Exposure, and Commitment Trends.

- The framework LASM, which stands for Liability-Aware Security Manager, serves to demonstrate and evaluate our contribution.

The LASM is represented in Figure 1.3, it aids administrators in making management decisions to fulfill service commitments [29]. The tool comprises several modules: the first, named LASM Visualized Service (LVS), focuses on presenting services and associated data. The second module, LASM Referencing Service (LRS), catalogs available network components and their TRAILS profiles. It incorporates an ontology to offer tools for evaluating a new component's TRAILS in alignment with a referencing policy or searching for a profile with specific features. The fourth module, LASM Analysis Service (LAS), assesses various metrics related to trust, responsibility, or the reputation of components and authors. Finally, the LASM Creation Service (LCS) which helps the administrators creating TRAILS profile. The LASM Monitoring Service (LMS) has been replaced by GRALAF presented in chapter 2.

As shown by the Figure 1.3, the LRS, the LVS and the LCS were implemented for the first contribution and the LAS for the second contribution. A presentation showcasing the first contribution was delivered at the Orange Salon De La Recherche 2022, while another presentation featuring the second contribution was presented at the ACM MobiCom 2023 conference. A poster describing the demonstration is available in the appendix B

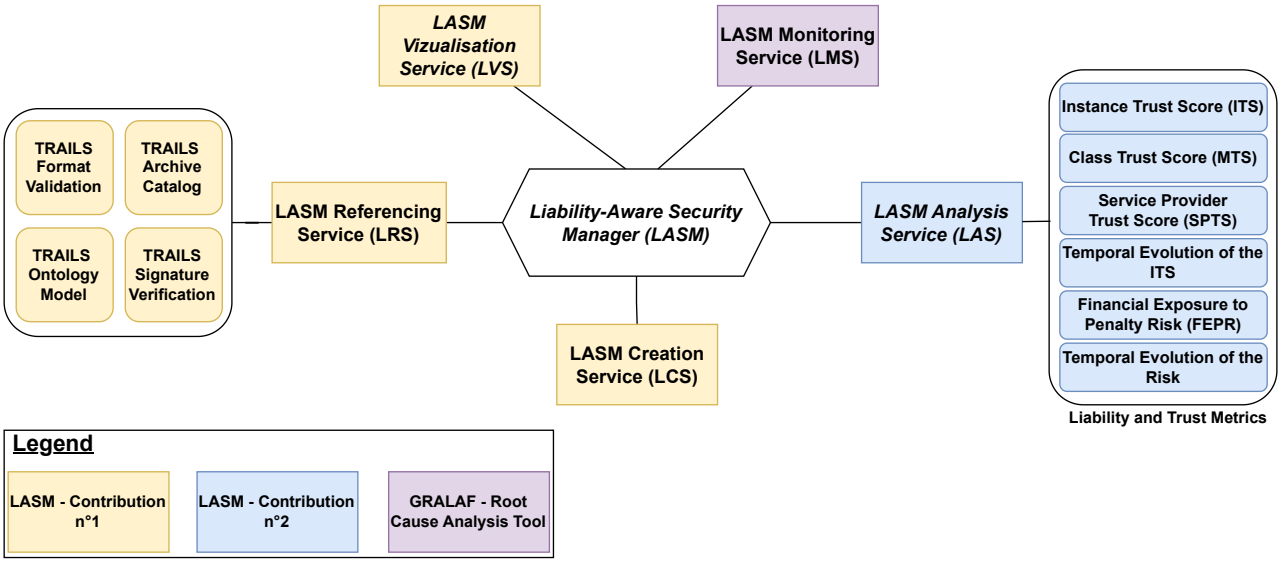


Figure 1.3: LASM Architecture — Highlighting Contributions n°1 and n°2 and their outputs

1.4 Motivating example: Smart IoT Campus Service

In the following, we present a use case to demonstrate the challenges related to Liability and Accountability in the Cloud-Edge-IoT Continuum and how mechanisms for liability and accountability management can assist in addressing these challenges. We consider a Service Provider (SP) that deploys a service across a wide infrastructure, spanning from the Cloud to an IoT campus, managed by a Slice Provider (SLP). The SLP, in turn, subcontracts the management and monitoring of SP's IoT campus to the Subcontractor (SC). Under normal conditions, the SLP directs packets collected from SP's IoT devices to SP's Cloud Delivery Network (CDN) application. The SLP operates SP's slice with a basic assurance level, ensuring low packet loss and optimized energy consumption. However, if any anomaly is detected in the IoT devices, the contract between SP and SLP mandates SP to implement a high level of assurance video streaming service. This service must provide proof of transit by specific nodes, a high level of video streaming solution availability, and guaranteed end-to-end isolation of the video streaming feed to control and confirm any potential threat. Figure 1.4 provides a visual representation of the described scenario. This use case involves three actors: the SP (service integrator), the SLP, and the SC, and three levels of delegation. In the event of a failure to meet the quality of service, it is essential not to automatically hold the SP solely responsible. For example, if the service fails to offer video streaming during an anomaly, it is essential to identify the responsible party

for this and hold them accountable for their actions. Legal and financial liability should be distributed proportionally among all parties involved in the delivery of the service. Also, in order to achieve efficient operations and regulatory compliance while maintaining cybersecurity standards, the SP must define and implement a governance structure. This structure will serve as a framework for managing the service effectively. Additionally, the SP must verify that any subcontracted solutions align with their established governance policies. To demonstrate compliance, the SP will require justification showing adherence to relevant regulations and contractual obligations. Ultimately, mechanisms are required to aid in establishing the service's governance, proving compliance with regulations and contractual obligations, and defining liability relationships among the involved actors. To identify the responsible party in the event of contractual obligation violations, the SP must diligently collect relevant evidence related to the situation at hand. This evidence serves as the cornerstone of the process, as it provides the necessary factual basis to ascertain what is at fault and to what extent. For example, if the service fails to offer video streaming, the SP can utilize detailed logs and the Proof of Transit protocol to securely verify whether, within a given path, all packets traverse all the nodes that they are supposed to visit to offer this service. Armed with this evidence, the SP can identify the node responsible for the failure of the service. With a well-defined delineation of liability among the various actors and access to clear and reliable evidence, the SP can effectively evaluate the compliance or potential violation and hold the responsible parties answerable for their actions or lack of compliance. With the help of liability and accountability analysis mechanisms, he can produce reports for administrators or jurists, aiding in forensic investigations and streamlining dispute resolution. This reinforces trust in the SP's ability to enforce contractual agreements but also ensures fairness and transparency in addressing any contractual breaches. In conclusion, the outlined use case within the Cloud-Edge-IoT Continuum underscores the critical need for robust liability mechanisms that clearly define the responsibilities of each party involved in the supply chain. These mechanisms should not only delineate accountability but also provide tangible evidence, such as liability and trust metrics, to support claims and resolutions. The establishment of such mechanisms is imperative to manage the complex interplay of services and delegations across the continuum effectively. It ensures that in instances of service failure or contractual breaches, the responsible entity can be accurately identified and held accountable.

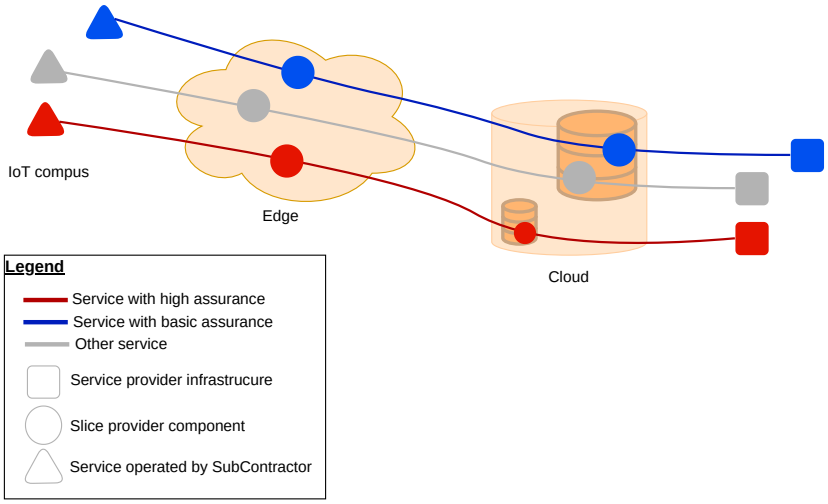


Figure 1.4: Smart IoT Campus Service — Liability and Accountability challenges

1.5 The manuscript's structure

This thesis is structured into three main parts as follows:

- The first part formalizes the context and challenges addressed by the contributions of the thesis. It includes the previously presented Chapter 1. introduction, Chapter 2. enhances the understanding of liability and accountability within information systems by offering deeper context, while also providing an overview of the interconnected concepts and technologies pertinent to the thesis. Chapter 3. provides an overview of the state-of-the-art on existing documents and profiles within the cloud-Edge-IoT, as well as metrics of liability and trust. Chapter 3. also positions our approach with regard to the state of the art.

- The second part presents the scientific contributions of this thesis. In Chapter 4, we present the responsibility model TRAILS and introduce the liability and trust metrics. Within these chapters, we showcase our research prototypes and experiments.

- The third part concludes this thesis by summarizing the contributions made and outlining prospects for future research work in Chapter 5.

1.6 Publications

The various works presented in this document have been the subject of various publications listed below.

1.6.1 International conference with proceedings and selection committee

- **Y. Anser**, C. Gaber, J. -P. Wary, S. N. M. García and S. Bouzefrane, "TRAILS: Extending TOSCA NFV profiles for liability management in the Cloud-to-IoT continuum," 2022 IEEE 8th International Conference on Network Softwarization (NetSoft), Milan, Italy, 2022, pp. 321-329, doi: 10.1109/NetSoft54395.2022.9844027.

- Onur Kalinagac, Wissem Soussi, **Y. Anser**, Chrystel Gaber, Gürkan Gür, "Root Cause and Liability Analysis in the Microservices Architecture for Edge IoT Services", 2023 IEEE International Conference on Communications (ICC): Next-Generation Networking and Internet Symposium.

- **Y. Anser**, Chrystel Gaber, Jean-Philippe Wary, Samia Bouzefrane, Meziane Yacoub, Onur Kalinagac, Gurkan Gur, Romain Cajeat. 2023. Demonstrating Liability and Trust Metrics for Multi-Actor and Dynamic Edge and Cloud Microservices. In Proceedings of the 28th Annual International Conference on Mobile Computing And Networking (MobiCom '23). Demonstration paper.

1.6.2 International workshop with proceedings and selection committee

- **Y. Anser**, J. -L. Grimault, S. Bouzefrane and C. Gaber, "Energy-Aware Service Level Agreements in 5G NFV architecture," EMSICC 2021 workshop, Co-located with 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy, 2021, pp. 377-382, doi: 10.1109/FiCloud49777.2021.00061.

- Chrystel Gaber, Ghada Arfaoui, Yannick Carlinet, Nancy Perrot, Laurent Valleyre, Marc Lacoste, Jean-Philippe Wary, **Y. Anser**, Rafal Artych, Aleksandra Podlasek, Edgardo Montesdeoca, Vinh Hoa La, Vincent Lefebvre, and Gürkan Gür. 2022. The Owner, the Provider and the Subcontractors: How to Handle Accountability and Liability Management for 5G End to End Service. In Proceedings of the 17th International Conference on Availability, Reliability and

Security (ARES '22). Association for Computing Machinery, New York, NY, USA, Article 68, 1–7. https://doi.org/10.1145/3538969.3544465

- C. Gaber and **Y. Anser**, "Modeling the accountability and liability aspects of a 5G multi-domain on-demand security services: an unexpected journey," 2022 1st International Conference on 6G Networking (6GNet), Paris, France, 2022, pp. 1-4

1.6.3 Technical Reports

- Inspire-5Gplus consortium, D4.4: Liability management in a 5G environment, 2019 INPIRE5Gplus project.

1.6.4 Protected Software (Agency for the Protection of Programs)

- **Y. Anser**, Chrystel Gaber, "LASM Referencing Service".
- **Y. Anser**, Chrystel Gaber,"LASM Analysis Service".

1.6.5 Journal Paper under review for a major revision

- **Y. Anser**, Chrystel Gaber, Jean-Philippe Wary, Samia Bouzefrane, Meziane Yacoub, Onur Kalinagac, Gurkan Gur. "Liability and Trust Analysis Framework for Multi-Actor Dynamic Microservices" in IEEE Transactions on Network and Service Management. Special Issue on Networks, Systems and Services Operations and Management through Intelligence.

Chapter 2

Background

Contents

2.1 Introduction **26**
2.2 Liability and Accountability in computing **26**
2.2.1 Definition 26
2.2.2 Rationale 27
2.2.3 Related concepts 32
2.3 Describing and Orchestrating Cloud Services - TOSCA **34**
2.3.1 TOSCA - General Concept 34
2.3.2 TOSCA - Conceptual Layers 35
2.3.3 TOSCA Entities & the concept of substitution mapping 36
2.3.4 Packaging 38
2.3.5 TOSCA Simple Profile for Network Functions Virtualization (NFV) 38
2.4 Ontology **38**
2.4.1 Fundamental Concepts 39
2.4.2 Simple Protocol and RDF Query Language (SPARQL) 40
2.4.3 Semantic Web Rule Language (SWRL) 41
2.5 Machine Learning and Neural Network Algorithms **41**
2.5.1 K-means 41
2.5.2 Artificial Neuron & Artificial Neural Network (ANN) 41
2.5.3 Multi-Layer Perceptron (MLP) 42
2.5.4 Self-Organizing Map (SOM) 46
2.5.5 Dataset Transformation 50
2.5.6 Assessment Software 51
2.6 Conclusion **52**

2.1 Introduction

The emergence of the Cloud-Edge-IoT continuum represents a logical progression in computing history. This model owes its existence to the integration of technologies like IoT, Edge, and the introduction of virtualization and microservices architectures. This combination has led to the creation of a dynamic, multi-domain, multi-actor environment with several levels of delegation. Managing liability and trust becomes a critical concern in such an architecture. In the following, we will first introduce the concept of liability in computer systems. Then, we will explore the technical concepts necessary for understanding our contributions such as the model TOSCA (Topology and Orchestration Specification for Cloud Applications) and the ontology used for the contribution n°1, and several machine learning and neural network methodologies like MLP (Multi-Layer Perceptron) and k-means.

2.2 Liability and Accountability in computing

In this subsection, we lay the foundations of liability in information systems, starting with a definition and exploring how this concept is perceived and utilized in the realm of computer science. We'll conclude by delving into related notions that are examined within this thesis.

2.2.1 Definition

Liability includes a wide range of interconnected ideas and vocabulary, making it a complex concept. To begin understanding this concept, we initially focus on two primary components: responsibility and accountability. Additionally, subsection 2.2.3 examines in more detail other related concepts. **Responsibility**, as described in the NIST (National Institute of Standards and Technology) [30], involves performing specific functions within a given context and being accountable for tasks or duties. Additionally, the mention of an organization's ability to delegate these responsibilities highlights the idea that within a structured environment, tasks and duties can be assigned or transferred to others while still holding someone ultimately accountable for their completion. In this thesis, we view responsibility as bidirectional (Figure 2.1). Responsibility, at its core, involves two facets, incorporating a usage condition. This condition outlines that not only are individuals given specific tasks within a defined context, but the organization assigning these responsibilities also holds the obligation to ensure and facilitate their successful completion. This dual aspect means that while individuals are

accountable for task fulfillment, the organization is responsible for overseeing and supporting the effective execution of these delegated duties.

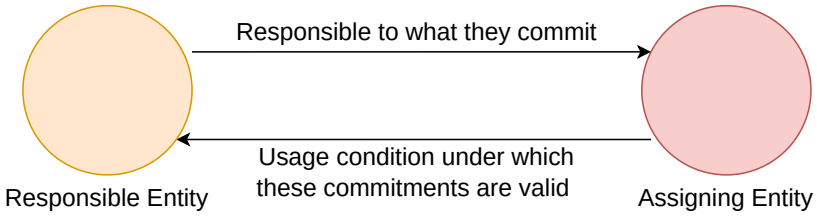


Figure 2.1: Responsibility between the Responsible Entity and Assigning Entity

The concept of **accountability** emerges as a direct outcome of this definition. Accountability is the property of being able to trace activities on a system to individuals who may then be held responsible for their actions [31]. Also, **liability** extends the concept of accountability further to the area of laws. Liability is a feature of political systems in which a body of laws is in place that permits individuals to recover the damages done to them by other actors, systems, or organizations. Due process is a related feature of law-governed societies and is a process in which laws are known and understood and there is an ability to appeal to higher authorities to ensure that the laws are applied correctly.

2.2.2 Rationale

The necessity of liability and accountability in computing is driven by the need to address the growing scale and complexity of online activities, such as social networking, remote work, and distance learning. In terms of security, preventive measures such as passwords, authentication protocols, firewalls, and access-control mechanisms alone are no longer sufficient to stop unauthorized parties from accessing confidential data, violating system policies, or engaging in actions they are not supposed to. Liability and accountability mechanisms are seen as a complement to preventive measures, aiming to hold individuals or entities responsible for their actions and ensuring consequences for policy violations. National Cybersecurity Agencies assume a leading role in addressing this matter. For example, in the United States, one of the objective of the national cybersecurity strategy [32] is to shift liability for insecure software products and services. Currently, the market lacks adequate consequences for entities releasing vulnerable software, leading to widespread neglect of secure development practices. The Biden administration plans to work with the private sector to establish legislation that holds software manufacturers accountable and sets higher standards of care. A safe harbor framework will be

created to protect companies adhering to secure software practices. Additionally, efforts will be made to encourage vulnerability disclosure, promote Software Bill Of Materials (SBOMs), and mitigate risks associated with unsupported software. Numerous research papers have highlighted the significance of incorporating mechanisms for liability and accountability. Fred B. Schneider [33] highlights the reality that attaining perfection in terms of security is a challenging task. Schneider suggests that liability provides a viable alternative to striving for perfection. Recognizing that perfection is beyond our reach, liability emerges as a practical approach. Rather than aiming for flawlessness, we can focus on establishing liability. By holding individuals or entities liable for their actions and decisions within computing systems, we introduce a means to address shortcomings and mitigate risks. While achieving perfection may be elusive, the attainability of liability offers a realistic and attainable goal. According to Lampson in [34], in the context of real-world security, deterrence plays a critical role by relying on the potential for punishment. The author illustrates this point by using the example of house burglary, highlighting that the security of a house is not solely reliant on a strong lock on the front door. Rather, it is primarily due to the risk of being apprehended and imprisoned, even though the likelihood of this occurring may be small. This significant risk serves to discourage potential burglars, making the act economically unviable. However, in the realm of securing a computer connected to the internet, deterring attacks becomes challenging due to the difficulty in identifying the perpetrators. Lampson suggests a potential solution by advocating for communication exclusively with accountable parties—those who can be held responsible and subsequently face punishment for their actions. According to the findings in [35], it is emphasized that in today's context, having the capability to not only detect errors but also identify the accountable entities for failures is of utmost importance. The authors argue that accountability has now emerged as a primary security service, holding a position alongside other essential security measures. In a recent work, Moshe Y. Vardi highlighted in [36] the significance of liability and accountability in the field of computing being highlighted. Vardi draws attention to the existing gap in accountability and liability within the computing marketplace. Recognizing the absence of accountability and liability, Vardi's work calls for a renewed focus on implementing these measures within the computing industry. This includes holding technology providers, developers, and other stakeholders responsible for the products and services they offer.

In the field of software engineering, computer scientists and lawyers alike have emphasized the

importance of liability and accountability in relation to software quality [37, 38, 39]. These concepts have been recognized as having significant implications for the field. Both disciplines acknowledge that addressing liability concerns can play a crucial role in ensuring the development and maintenance of high-quality software systems. In response to that, a project named LISE (Liability Isues in Software Engineering) has emerged [40]. The main contribution of the LISE project is the development of a formal framework for defining liability in a precise and unambiguous manner within the field of computing. By creating this framework, the project aims to establish a standardized approach for determining and assigning liability in software engineering practices. This contribution has the potential to bring clarity and consistency to addressing liability concerns. Additionally, the LISE project aims to provide methods and tools for establishing liability in the case of incidents. This includes the development of techniques for analyzing log files and identifying the responsible parties involved. To demonstrate the practical application of the framework and tools, the project presents a comprehensive case study. Overall, the LISE project significantly contributes to enhancing accountability and liability practices in the computing field, fostering transparency and responsible behavior.

Extensive research has been conducted on accountability within the domain of distributed systems. In [41], the authors agree to say that accountability is a fundamental design objective for services in federated distributed systems. Firstly, accountability acts as a valuable tool for achieving practical security by holding individuals or entities accountable for their actions and decisions. By doing so, it acts as a safeguard that promotes trust and deters malicious behavior. Furthermore, accountability is regarded as a primary design goal for services operating within federated distributed systems. In such systems, multiple autonomous entities collaborate to provide a unified service. Accountability ensures transparency, traceability, and the ability to attribute actions to specific participants. This design goal recognizes the importance of establishing clear lines of responsibility, enabling effective governance, and facilitating the resolution of conflicts or disputes that may arise. Also, Yumerefendi *et al.* highlights in [42] the importance of accountability in designing reliable network systems. For the authors, conventional techniques are insufficient to protect against covert manipulation by adversaries. By incorporating accountability into the system, faults can be detected, isolated, and tolerated, while discouraging malicious behavior. The future challenge lies in developing widely applicable techniques to achieve specified levels of accountability and expose attackers.

Helen Nissenbaum [43] wrote an essay where she critically examines the erosion of accountability in

computerized societies, highlighting barriers to accountability. The author emphasizes the significance of fostering a culture of accountability, especially for a technology that is struggling with reliability standards, as it ensures that even in the situations where things go wrong, there is a guarantee of being accountable for. As per the author's viewpoint, accountability can serve as a potent mechanism for driving improved practices, leading to the development of more reliable and trustworthy systems. Additionally, it argues that fostering a culture of accountability should encompass not only life-critical systems with severe consequences but also extend to malfunctions that cause individual inconveniences, emphasizing the importance of clear accountability for assigning appropriate punishment and providing compensation to victims in case of failure. One of the barriers that the author highlights is the problem of many hands. It refers to the difficulty of assigning accountability in the situations where multiple individuals or entities are involved in the development and implementation of computer systems. This issue affects accountability in computerized societies by obscuring the connection between an outcome and the person or entity responsible for it. In computerized societies, software systems are often produced in various institutional settings, including software development companies, corporations, government agencies, contractors, and educational institutions. With so many different actors involved in the process, it becomes challenging to determine who should be held accountable for any negative outcomes that may arise. The problem of many hands is not unique to computing but is also prevalent in other domains such as big businesses, governments and the military. However, computing is particularly vulnerable to this issue due to the complex nature of software development and the involvement of multiple stakeholders. Also, the author claims that the prevalence of bugs in software and the perception that they are inevitable pose a significant barrier to accountability in computing. While bugs are widely recognized as causing system failures, the belief that they cannot be avoided except in cases of obvious negligence hinders assigning responsibility. These are contrasts with other areas of the technology where accountability is assigned for known risks. A more discerning approach to bugs, distinguishing between natural hazards and avoidable failures, would enable better accountability. If experts deny this possibility, it suggests that computing may not be ready for its current applications. The author concludes the essay by suggesting three possible strategies for restoring accountability. She suggests putting forward an explicit standard of care. It serves as a non-arbitrary means to establish accountability in the computing field. Proposed guidelines for safer and more reliable computer systems, including simpler design, modular approach, quality assurance, independent auditing, redundancy,

and comprehensive documentation, can form the basis of this standard. By embracing and enforcing this standard, the computing profession can differentiate between negligent practices and unavoidable failures, support engineers' commitment to safety, and assess the integrity of the field. Also, she points the need to apply strict liability and producer responsibility. In considering the relationship between liability and accountability, it has been suggested that liability should not be seen as a substitute for accountability, as acknowledging or denying liability does not address one's answerability. However, according to the author, establishing effective liability policies can help express societal expectations and clarify lines of accountability. To this end, it is proposed to explore the implementation of strict liability for computer system failures, particularly in the case of consumer products sold in mass markets. For the author, supporters of strict liability argue that it benefits society by placing the burden of risk where it belongs, protecting against potential harm, seeking compensation for victims, and reducing the costs of litigation.

Accountability has emerged as a significant concern within the domain of cloud computing. The Cloud Accountability project (or A4Cloud for short) is the main project in this topic [44]. According to A4Cloud, accountability is important in cloud computing because it ensures legal compliance, promotes ethical behavior, builds trust in cloud relationships. In this project, accountability has been examined from four perspectives: legal, ethical, socio-economic, and technical. They have analyzed accountability requirements in data protection laws and regulations, emphasized the importance of ethical considerations beyond compliance, studied the socio-economic impact and governance of accountability in cloud ecosystems, and developed the Cloud Accountability Reference Architecture as a comprehensive framework for security and trust.

This thesis is a part of the inspire-5Gplus European project [45], which aims to implement a comprehensive automated framework for managing the security of network and services. The primary objective of the project is to ensure the protection, trustworthiness, liability, and accountability of 5G network infrastructures across various domains. The inspire-5Gplus consortium has established six work packages (WPs), with WP4, led by Orange, focusing on Liability aware Trusted 5G Security. This work package aims to enhance existing security systems by prioritizing trust, accountability, and liability throughout the entire supply chain of 5G infrastructures and services. To achieve this goal, the project proposes novel mechanisms to enforce liability for parties involved in the event of security breaches or system failures. One significant mechanism proposed is the use of a liability manifest

that formalizes liability within the environment [46], building upon the previous work conducted by Dragoni et al. [47]. Additionally, they explore the design, building blocks, and challenges of a Liability-Aware Security Management (LASM) system for 5G [29].

2.2.3 Related concepts

In this subsection, we delve into the concepts associated with liability and accountability management such as trust, reliability, and transparency. We also outline how these concepts impact liability management.

Trust. Trust is a non-reciprocal ($T_{i,j} \neq T_{j,i}$) peer-based property where the trustor forms an opinion on how good the trustee is on providing a specific service. It is the subjective degree of belief a trustor has on a trustee to perform a concrete task in this specific system [48]. It depends on the context and corresponds to a real number of positive collaborations between trust and trustee. According to [48], the trust is the most important behavioral factor in managing relationships and in overcoming risks/uncertainty. It relies either on the formalization of agreements (contracts), mutual confidence established by fruitful exchanges and acquaintance. The project INSPIRE-5GPlus highlights the fact that the concept of trust is complementary to liability and accountability. Each covers a different aspect related to the accomplishment of a task and the management of the underlying risks [49]. Felici *et al.* [50] highlight the fact that accountability is instrumental in guiding trust decisions; however, accountability alone is not enough to fully establish trust. While accountability is essential, it does not unconditionally imply trust. A crucial factor influencing trust decisions is the evidence provided to stakeholders. Therefore, the presence of accountability plays a significant role in trust decisions, but it is the evidence supporting it that truly reinforces trust. According to [46]. trust forms the foundation for liability management, as parties are more willing to collaborate when they have confidence in each other. Liability, in turn, ensures accountability and reinforces trust by providing consequences for non-compliance or breaches. Together, they create a stable and secure environment, promoting responsible behavior and reducing uncertainties.

Risk. The risk is a measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of 1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence [11]. Risk management intends to mitigate risks and to identify operational trade-offs. An essential element of risk management is risk analysis, for which

an analytical definition is the following: risk analysis is an estimation of the occurrence of events, their possible consequences, their causes, and existing and/or planned countermeasures and mitigations [11]. The A4Cloud project takes an accountability-driven approach to risk mitigation [50]. It acknowledges that accountability measures are essential for mitigating and managing risks effectively.

The project A4Cloud defines accountability attributes [51]. According to A4Cloud, these attributes are fundamental concepts that support and are considered integral to accountability. These attributes include assurance, remediation, and transparency, all of which are derived directly from the definitions of accountability. These attributes exhibit interconnections based on various perspectives of analysis, such as societal, legal, and ethical viewpoints. For example, legal responsibilities imply obligations and may lead to sanctions, while social transparency involves both observability and verifiability. Overall, accountability attributes encompass key properties like transparency, conceptual elements such as remediation, consequences such as sanctions, and related objects including obligations and insurance. Below are definitions of attributes that we consider to be significant for the thesis.

Transparency. Transparency refers to the openness and visibility of information, processes, and actions in an organization or relationship. It involves sharing relevant information with stakeholders and being forthcoming about decisions, activities, and outcomes [52]. Transparency plays a vital role in liability and accountability management. By being transparent, organizations and individuals are more likely to take responsibility for their actions and decisions. This fosters a culture of accountability, making it easier to identify responsible parties in case of incidents or failures [50]. Also, transparent practices facilitate the early identification of potential risks and issues. When problems are detected promptly, appropriate measures can be taken to mitigate risks.

Assurance. Assurance is a positive declaration that creates confidence. It can be supported by evidence from an accountability system, which convinces third parties about the presence or absence of faults [51]. In accountability, assurance means providing evidence of compliance with governing rules and demonstrating trustworthiness. The Galway project [53] defines key elements of accountability as "internal oversight, assurance, reviews, and external verification." Accountable organizations must offer assurance to show they have proper governance, implemented suitable actions, and can explain and justify their decisions to relevant stakeholders.

Obligation. An obligation refers to a duty, commitment, or pledge that comes with potential repercussions if breached. These obligations are primarily categorized into three types: contractual (based

on formal agreements), regulatory (mandated by laws or rules), and normative (arising from societal norms). Depending on the context, user preferences can align with these categories, sometimes giving rise to legal obligations, and at other times, they may not carry any formal responsibility.

Sanction. Sanctions are the legal consequences of failing to comply with some requirement. The legal consequences arising from failing to uphold certain obligations result in diverse sanctions imposed by member states on accountable entities. These sanctions can range from court rulings to administrative measures. Sanctions have both a post hoc effect, placing a financial burden on the punished entity, and an preventative effect, promoting compliant behavior out of fear of punishment. Strong sanctions encourage investment in an accountability-based approach. Not only must there be robust penalties for improper actions, but they also motivate organizations to adopt an accountability-based approach by offering more leniency if they can demonstrate efforts to ensure proper implementation of actions.

Verifiability. Verifiability refers to a property of an object, process, or system that enables its behavior to be checked against specific requirements or a set of requirements. The degree of verifiability is directly influenced by the available evidence. It's worth noting that some argue that verifiability can intentionally be restricted in the contract specification. *Validation* is a closely associated concept that pertains to accountability. It enables users, operators, and third parties to verify after the fact whether a system has performed a data processing task as expected. Similarly, *verification* is a process that assesses whether a system adheres to relevant governing regulations.

2.3 Describing and Orchestrating Cloud Services - TOSCA

In order to define accountability and liability relationships among the Cloud-Edge-IoT continuum, we propose a responsibility model. This model is an extension of a well-know profile in the Cloud-Edge-IoT continuum named TOSCA (Topology and Orchestration Specification for Cloud Applications) NFV (Network Function Virtualization). This profile uses the TOSCA meta-model. As a result, we offer a contextual overview of TOSCA and the TOSCA NFV.

2.3.1 TOSCA - General Concept

The Topology and Orchestration Specification for Cloud Applications (TOSCA) is an OASIS (Organization for the Advancement of Structured Information Standards) standard language which was introduced in 2013. TOSCA defines a metamodel to describe the structure of composite cloud

applications and the corresponding management tasks in a standardized way. It is designed for automating, portability, and interoperability of complex cloud applications with multiple services [17]. TOSCA's objectives can be summarized into three areas: automated deployment and management of composite applications, ensuring portability of application descriptions, and promoting interoperability and reusability of application components. It aims to streamline complex application management, enable seamless portability, and facilitate effective communication and component reuse. To achieve this, TOSCA introduces two primary concepts: (1) Application topologies and (2) management plans. Application topologies serve as a structural representation of the application, detailing its components and their interconnections. Each node within the topology is associated with a set of operations for self-management. This not only describes the application's components and their relationships, but also explicitly declares its management capabilities. Management plans leverage these management capabilities to create higher-level tasks for application management. These plans can be executed fully automated, handling tasks such as deployment, configuration, and operation of the application. In Figure 2.2, we see an abstract TOSCA-based application description illustrating the relationship between the two main concepts: the application topology consists of nodes interconnected by relationships, and the management plans are initiated by external messages, utilizing the management operations of the nodes within the topology [54].

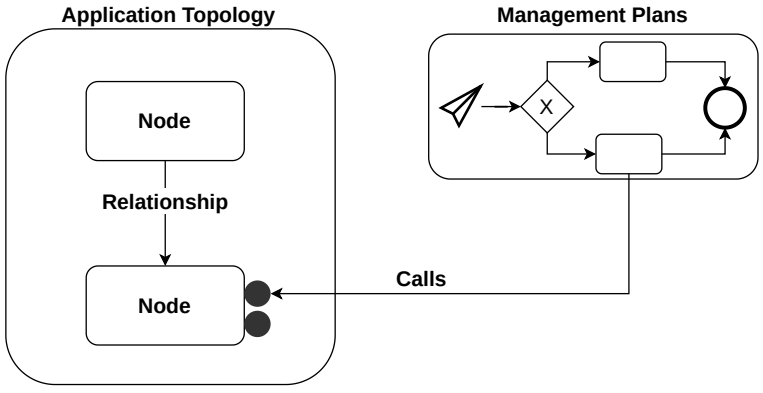


Figure 2.2: Relation of TOSCA concepts [17]

2.3.2 TOSCA - Conceptual Layers

To achieve a clear understanding of TOSCA, it is essential to differentiate between three conceptual layers. TOSCA establishes a metamodel and exchange format for (1) types and (2) templates, leading

to the third layer, the (3) instance layer. For example, the metamodel layer encompasses Node Templates and Relationship Templates, which are associated with reusable types—Node Type for Node Templates and Relationship Type for Relationship Templates, respectively. These types can be likened to abstract classes in Java, while the templates are analogous to concrete classes that extend these abstract classes. On the other hand, the instance layer represents actual instances of the components and relationships defined by the templates. [7]

2.3.3 TOSCA Entities & the concept of substitution mapping

To describe the topology and the management aspects of a cloud application or service, the TOSCA language introduces a set of special entities. The figure 2.3 is a simplified UML class diagram that gives an overview of these entities and their relations. The core entity is the Service Template; it encompasses all the necessary entities to define the structure, behavior, and orchestration of a cloud-based application or service. The primary entity is the Topology Templates. It defines the topology of the service and its components. it can be viewed as a directed graph. Its nodes, referred to as node templates, represent the application components, while its edges, called relationship templates, depict the connections between these components. Both node templates and relationship templates are associated with specific types - node types and relationship types, respectively. Node types define various characteristics of an application component, such as its requirements and capabilities used to indicate that one component relies on (requires) a feature offered by another component, or to specify that a component has particular demands concerning the hosting environment, such as the allocation of specific resources or the activation of a particular mode of operation. A node's lifecycle is governed by policies defined through policy types, using predefined data types. These policies automatically trigger actions based on specific events or conditions, ensuring efficient management and consistent behavior of the cloud application. On the other hand, relationship types describe properties of the connections among components. [7]

TOSCA defines a concept that allows the substitution of Node Templates within a Service Template's topology with the entire topology of another Service Template. By defining a substitutable Node Type attribute in a Service Template, it indicates the specific Node Types for which it can serve as a substitute. As indicated before, a Node Template represents an instance of a Node Type, the same mechanism applies to Service Templates by utilizing boundary definitions, enabling the expression

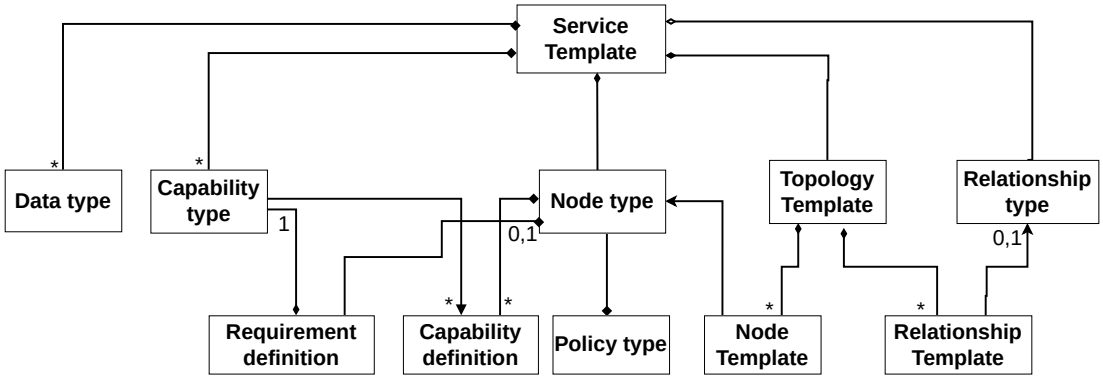


Figure 2.3: Simplified UML class diagram of the TOSCA entities

of properties, requirements, and capabilities within a Service Template's topology. When a Service Template substitutes a Node Template in another topology, these boundary definitions are analyzed to effectively handle and reconnect relationships and define properties. This feature facilitates abstract modelling of extensive and intricate topologies, allowing the use of substitutable Service Templates as a way to represent subsystems. For example, Figure 2.4 illustrates a scenario where Service Template 2 serves as a substitution for Node Template in Service Template 1.

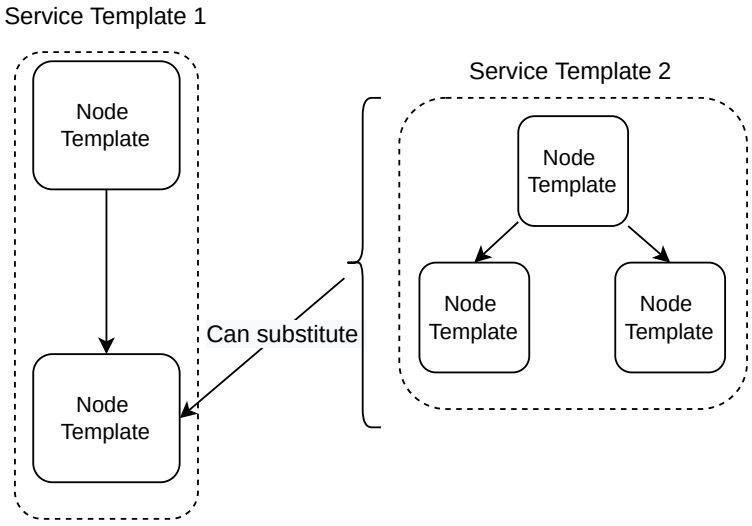


Figure 2.4: Composition of services in TOSCA

2.3.4 Packaging

The physical files associated with the Service Template, such as Implementation Artifacts, Deployment Artifacts, scripts, and XML schema files, are bundled together into what is known as CSAR (Cloud Service Archive). This standardized archive format enables applications to be fully self-contained, encompassing all necessary management functionalities within a single file, facilitating their installation. In essence, the CSAR can be considered a single, installable package for complex composite applications, along with their management capabilities. When deploying a TOSCA archive, it is deployed on a TOSCA runtime environment, which takes responsibility for installing the application package and processing the archive. TOSCA archives adhere to a standardized format, ensuring portability across different TOSCA runtime environments. As a result, they provide an exchange format for complex composite applications, along with their management functionalities [7].

2.3.5 TOSCA Simple Profile for Network Functions Virtualization (NFV)

A NFV profile was introduced which defines a specific data model for NFV using the TOSCA language. It includes several default Node Types that mostly align with the ETSI (European Telecommunications Standards Institute) definitions of components in the NFV domain [55]. This profile is designed to express all the necessary information to specify an individual VNF (Virtual Network Function) or a NS (Network Service) composed of multiple VNFs in a vendor-neutral manner. Consequently, VNFs or complete Network Services can be defined in a Service Template and packaged into a self-contained CSAR. This CSAR can then be provided to customers, who can import it into their TOSCA-compatible runtime.

2.4 Ontology

The proposed responsibility model has been associated with an ontology to facilitate reasoning and logical analysis. In this section, we're going to introduce the fundamental concepts of an ontology and the tools used for reasoning.

2.4.1 Fundamental Concepts

An ontology is an explicit and formal specification of the concepts, individuals and relationships that exist in some area of interest. It shares the following minimal set of components [56] :

- **Classes** represent the fundamental concepts of a specific domain. For instance, in a school ontology, we may have classes like ”Person,” ”Student,” and ”ProgramOfStudy”.
- **Properties** depict the relationships between different concepts in the domain. For example, the school ontology might define a property called ”participatesIn” to connect the concepts of ”Student” and ”Activity”.
- **Axioms** are statements that express fundamental truths or facts within the domain, and they are always considered true within that context.
- **Instances** represent individual entities that belong to specific classes and are linked together through properties. For instance, the statement ”Omar is a student” indicates that the individual entity ”Omar” belongs to the ”Student” class. Similarly, the statement ”Omar participates in the web class” establishes a link between the individual entity ”Omar” and another instance representing ”the web class” using the ”participatesIn” property.

Various standards such as RDF, RDFS, and OWL 2 are utilized for implementing ontologies, each with distinct expressiveness and inference mechanisms. These three standards are governed by the World Wide Web Consortium (W3C). We briefly summarize these standards in the following:

- Resource Description Framework (RDF): RDF [57] serves as a standard model for describing web resources and their relationships. It utilizes Internationalized Resource Identifiers (IRIs) to identify resources and describes these resources through properties and property values, forming RDF triples. The triples consist of subjects (IRIs or blank nodes), predicates (IRIs defining relationships), and objects (IRIs, literals, or blank nodes). RDF graphs comprise collections of these triples, and RDF vocabularies use IRIs within the graph.
- RDF Schema (RDFS): An extension of RDF, RDFS [57] enables the description of resource groups and their relationships. It introduces concepts like classes and instances, facilitating hierarchies through properties like `rdfs:subClassOf` and `rdfs:subPropertyOf`. Properties such

as `rdfs:range` and `rdfs:domain` define property values and resource classes, while `rdf:type` specifies resource instances.

- Web Ontology Language (OWL) 2: OWL 2 [58], a semantic web standard, extends OWL and provides formally defined semantics. It uses axioms as true statements to represent knowledge. Entities like classes, properties, and individuals describe domain objects. Expressions, involving entities, create intricate representations. Property expressions define relationships between resources through object properties (using `rdfs:subPropertyOf`, `owl:inverseOf`, and `owl:equivalentProperty`) and data properties (linking resources to literals). Class expressions describe complex classes using classes and property expressions, setting conditions for individuals to be instances of those classes.

Ontologies expressed using RDF, RDFS, and OWL 2 can be serialized using different syntaxes such as RDF/XML where the class `School:Student` can be represented as `<owl:Class rdf:about="http://School.fr/Student">`, and N-Triples where the same class can be represented as `<http://School.fr/Person> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Class>`. This format is harder to read for a human, but it is easier to parse for a computer. [59]

2.4.2 Simple Protocol and RDF Query Language (SPARQL)

SPARQL is a semantic query language for databases, enables the retrieval and manipulation of data stored in the RDF format. SPARQL has a wide array of features suitable for various use cases. Some notable examples of these features include: using RDF-defined terms like IRIs, language tags, and literals in its query syntax; offering different query forms for different purposes, such as SELECT for extracting values from a SPARQL endpoint, CONSTRUCT for generating RDF graphs based on query criteria, and ASK for testing the presence of specific data. Additionally, SPARQL provides a filtering technique with the FILTER keyword to restrict query results based on predefined aspects. Moreover, it supports aggregation through predefined aggregates like COUNT, SUM, MIN, MAX, and AVG, which enable counting occurrences, returning sums, finding minimum and maximum values, and calculating averages within aggregate groups. [60]

2.4.3 Semantic Web Rule Language (SWRL)

SWRL offers the ability to define rules using OWL entities, combining the simplicity of rule-languages with the power of automated reasoning. It presents a high-level abstract syntax for horn-like rules that align perfectly with OWL semantics. A SWRL rule comprises two parts: the antecedent and the consequent, both formed by positive conjunctions of atoms. As SWRL rules are expressed in terms of OWL concepts, the atoms within the rules can represent individuals, properties, or classes defined within the ontology. [61]

2.5 Machine Learning and Neural Network Algorithms

The second contribution incorporated machine learning and neural network methodologies. Within this section, we aim to present an overview detailing these specific approaches. This includes an exploration of the machine learning techniques employed and a comprehensive examination of the neural network methodologies utilized in our study.

2.5.1 K-means

The k-means algorithm [62] is a clustering technique utilizing k codebook vectors, each mirroring the dataset's dimensionality to represent clusters. These vectors, symbolized as $m_j \in R^d$, iteratively segment the dataset into k clusters. Initially, the algorithm randomly initializes k cluster centroids, often as dataset points. It operates through two main steps: the assignment step, where data points are matched to the nearest centroid using Euclidean distance, and the centroid update step, recalculating centroids as the mean of assigned data points for each cluster. These steps iterate until convergence, when centroids stabilize and data points persist within their respective clusters.

2.5.2 Artificial Neuron & Artificial Neural Network (ANN)

An artificial neuron, or perceptron, is a fundamental computational unit in neural networks, it takes multiple input values, each associated with a weight that signifies its importance and a bias. The neuron computes a weighted sum of the inputs, introducing non-linearity through an activation function. This function transforms the sum into the neuron's final output, making decisions or predictions based on the inputs, their weights, and the activation function, while also considering the

bias to allow for shifts and fine-tuning of the neuron's response. Activation functions influence how neural networks learn and detect patterns within data. The popular ReLU activation [63], replacing negatives with zeros, accelerates training and mitigates gradient vanishing. Sigmoid functions like logistic and tanh suit binary classification [64], while softmax [65] excels in multi-class classification by converting outputs into probabilities.

An Artificial Neural Network (ANN) is a mathematical computing paradigm inspired by biological neural systems, with origins dating back to McCulloch and Pitts in 1943 [66]. These networks consist of interconnected artificial neurons and can be categorized as feed-forward or recurrent. In feed-forward networks like the MultiLayer Perceptron (MLP), signals flow in one direction, while recurrent networks involve feedback loops where neuron outputs become inputs. MLP considered in this thesis belongs to the feed-forward network.

2.5.3 Multi-Layer Perceptron (MLP)

An MLP is a type of ANN commonly used in supervised learning. It consists of multiple layers (input layer, one or more hidden layer and output layer) of interconnected artificial neurons and is designed to perform complex tasks by learning from data through training with labeled examples (Figure 2.5).

MLP Learning Process As shown by the Figure 2.5 the learning process of an MLP involves iteratively adjusting the network's weights and biases to minimize the loss function. This process includes a forward pass to make predictions, a backward pass (backpropagation) [67] to calculate gradients, and the use of optimization algorithms like Gradient Descent [68], Adam [69], and RMSprop [70] to update parameters. During training, the network learns to capture complex patterns in data. Common examples of loss functions used in training include Mean Squared Error (MSE) [71] for regression tasks and Cross-Entropy [72] for classification tasks. These loss functions quantify the disparity between the model's predictions and the actual target values, guiding the network toward better performance and enhanced pattern recognition.

Hyper Parameter In an MLP, hyperparameters are essential configuration settings that guide the network's structure and learning process. These parameters are not learned from the data but must be set prior to training. They include decisions about the number of hidden layers and neurons in each layer, the choice of activation functions for neurons, the learning rate that governs weight

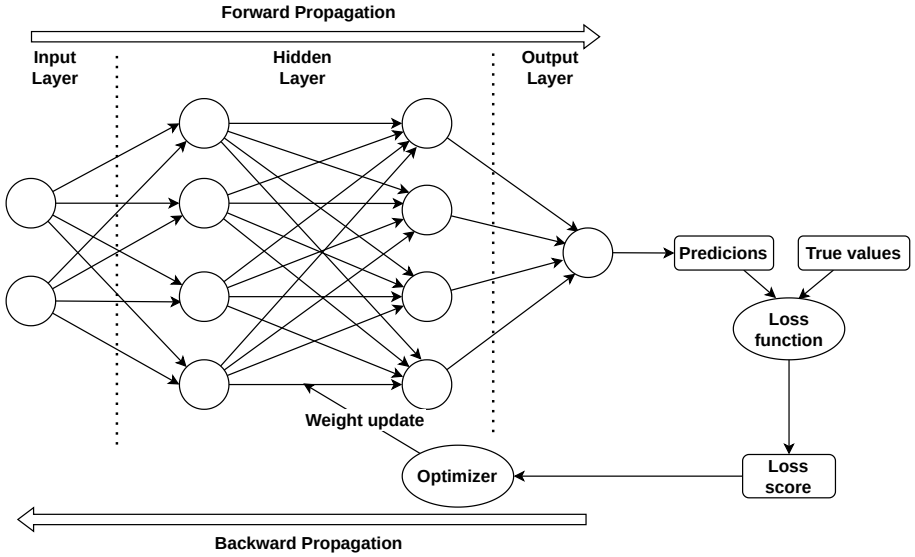


Figure 2.5: MLP Learning Process

updates, the batch size for data processing which defines the number of data points used in each forward and backward pass during training, the number of training epochs (how many times the entire training dataset is processed during training), the initialization of weights and biases, the optimization algorithm, the selection of an appropriate loss function, and choices related to early stopping and dropout rates [73] are all critical hyperparameters. Tuning these hyperparameters is a crucial step in configuring the MLP to perform optimally on a specific task and dataset.

`GridSearchCV` [74] and `RandomizedSearchCV` [75] are two popular hyperparameter optimization techniques. `GridSearchCV`, which stands for Grid Search Cross-Validation, offers a systematic and exhaustive approach to hyperparameter tuning. It explores all possible combinations of hyperparameter values within predefined ranges. This comprehensive search enables it to find the best-performing set of hyperparameters. However, this exhaustive search can be computationally expensive, especially when there are numerous hyperparameters to optimize. `RandomizedSearchCV`, on the other hand, takes a more efficient approach. Instead of considering all possible combinations, it randomly samples a specific number of hyperparameter sets from predefined distributions. This randomness reduces computational cost while still allowing it to find good hyperparameter values. It's particularly useful when you have a large search space, and you want to quickly identify promising hyperparameters without exploring every possible combination. `HalvingGridSearchCV` and `HalvingRandomSearchCV`

[76] are extensions of their counterparts, designed to further improve efficiency in hyperparameter optimizations. The two extensions employ iterative strategies to progressively narrow down the search space. `HalvingGridSearchCV` initiates with a grid search on a subset of hyperparameter combinations, evaluating performance and discarding suboptimal options. This iterative process refines the search space towards optimal hyperparameters. `HalvingRandomSearchCV` follows a similar iterative approach, starting with random sampling and retaining the best performing combinations. It efficiently identifies optimal hyperparameters, especially in computationally expensive search spaces.

Model Evaluation Model evaluation means assessing how well the trained neural network perform on new, unseen data. Cross-Validation (CV) [77] has become a popular technique for model evaluation because it offers several advantages and reduces the need for a separate validation set, which drastically reduces the number of samples which can be used for learning the model. It is essential for hyperparameter tuning. By evaluating the model's performance across different hyperparameter configurations, it helps select the best set of hyperparameters. In this thesis, We combine Grid Search and CV to find the best combination of hyperparameters. Also, as we are dealing with time-serie data, we used Time-Serie Split CV, a variation of CV where the data are splitting into sequential folds in order to preserve temporal order, ensuring the validation sets come after training sets, mimicking real-world scenarios better than random splits used in the ordinary CV.

The confusion matrix is often used during cross-validation to evaluate the model's performance in each fold. It allows you to calculate metrics like precision, recall for each fold and then average them to assess the model's overall performance. Cross-validation with a confusion matrix provides a more detailed understanding of how the model is making errors and where it excels. It is particularly valuable when you want to diagnose the specific strengths and weaknesses of your model in a classification task. Given the dataset's imbalance, we focus in this thesis on the following metrics:

1. Precision measures how many of the predicted positive instances were correctly classified for the specific class. It defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

   TP is the number of True Positive and FP the number of False Positive.

2. Sensitivity (or recall) measures how many of the actual positive instances were correctly classified

for the specific class. It is defined as follows:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.2}$$

FN is the number of False Negatives.

3. F1-Score which can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. It is defined as follows:

$$F1 - score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \tag{2.3}$$

In the case of multi-class classification, the F1 score can be computed as an average of the F1 scores for each class.

4. Specificity measures the ability of a model to correctly identify true negatives out of all actual negative instances. It is defined as follow:

$$Specificity = \frac{TN}{TN + FP} \tag{2.4}$$

TN is the number of True Negative. In multi-class classification, specificity for each class is calculated using the one against all approach.

5. G-mean is a measure that aims to balance and optimize accuracies across all classes. For binary classification, it's the squared root of the product of the sensitivity and specificity. For multi-class problems, it's a higher root of the product of sensitivity for each class.

Another method to evaluate the model is the Receiver Operating Characteristic (ROC) curve [78]. It's a graphical representation that illustrates a classifier's performance across different discrimination thresholds. It plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) for varying threshold values. This curve helps visualize the classifier's ability to distinguish between classes, showcasing how changes in the classification threshold impact its performance. A steeper ROC curve closer to the top-left corner indicates superior performance, while an area under the curve (AUC) closer to 1 suggests better overall model discrimination. The ROC Curve in multi-class classification supports two averaging strategies: one-vs-one computes pairwise ROC Curve, while one-vs-rest computes ROC Curve for each class against all others. Both use predicted labels in an array from 0 to a number of classes. In this thesis, we used the one-vs-rest strategies.

2.5.4 Self-Organizing Map (SOM)

Self-Organizing Map (SOM) [79] is an ANN that combines the properties of vector quantization and vector projection. Vector quantization is a data compression technique that represents data points using a set of code vectors for efficient storage, and vector projection is a mathematical operation used to find the component of a vector in the direction of another vector. SOM is an unsupervised learning algorithm, and like k-means, it consists of a set of M codebook vectors m_j represented as $M \in R^{MxD}$. In the upcoming sections, we'll explore the core attributes of SOM, focusing on its similarities and differences compared to the k-means algorithm. A key difference lies in the introduction of an output space that establishes relationships between prototype vectors.

In SOM, the prototype vectors have a specific order. They are positioned on a discrete output space lattice of dimension L, with each codebook vector having additional, unrelated coordinates. The unit within the output space is represented as $\theta_j \in N^{MxL}$, with individual coordinates denoted as θ_j^k, where k designates the specific coordinate. For this thesis, we consider a two-dimensional map. The horizontal and vertical coordinates of these units are expressed as θ_j^u and θ_j^v, respectively. The Euclidean distance, used to calculate the distance between two coordinates, is formulated as follows:

$$d(\theta_i, \theta_j) = \sqrt{\sum_{k=1}^{L} (\theta_i^k - \theta_j^k)^2} \tag{2.5}$$

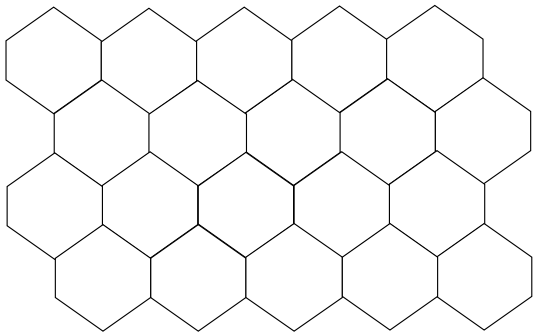


Figure 2.6: Hexagonal grid for 4x5 map

In this thesis, we utilize hexagonal maps for visualization. As depicted in Figure 2.6, here's a two-dimensional example (L=2). We have a horizontal axis of 5 units and a vertical axis of 4 units, resulting in M=4x5=20 units.

A key element in both the training phase and subsequent analysis of SOMs is the neighborhood

function, denoted as h_ϑ. This function is characterized by a monotonic decrease, meaning it consistently diminishes, and is mathematically described as $h_\vartheta : R^+ \longrightarrow R^+$. Its role is to quantify the proximity between two units on the map by accepting the distance between them in the output space as input. Specifically, for units θ_i and θ_j, the further apart they are, the smaller the value produced by the kernel of the neighborhood function, indicating a lower degree of closeness or influence between these units. There are numerous neighborhood functions, the one that is most commonly used is the Gaussian kernel h^G_ϑ, which is defined as follows :

$$h^G_\vartheta(\theta_i, \theta_j) = exp(-\frac{d(\theta_i, \theta_j)}{2\vartheta}) \tag{2.6}$$

The parameter ϑ controls the width of the neighborhood function. High values mean that it affects both distant and nearby units significantly, while low values focus primarily on the immediate neighbors of the map unit. The function value for distant nodes is decreasing exponentially and is close to zero for $d > \vartheta$.

The Ricker wave, also known as the Mexican hat function, is another commonly used neighborhood function. The Mexican hat function applies a penalty to neighbors that are slightly more distant from the center. If the model aims to discourage close matches, the Mexican hat function is a suitable option. It is defined as follows:

$$h^H_\vartheta(\theta_i, \theta_j) = (1 - \frac{d(\theta_i, \theta_j)}{\vartheta})exp(-\frac{d(\theta_i, \theta_j)}{2\vartheta}) \tag{2.7}$$

The SOM training algorithm, like k-means, iteratively updates prototype vectors using data samples. It aims to achieve vector quantization and projection by gradually moving prototype vectors towards their final positions. While the initial setup of model vectors has an impact, it's less critical than in k-means. To enhance predictability and determinism, systematic codebook initialization is preferred over random. Many methods have been proposed [80], In this thesis, we apply PCA (Principal Component Analysis) on the dataset to initialize prototype vectors.

Following the initialization, SOM undergoes training for a specified number of epochs, denoted as T. The data samples are introduced to the codebook in a random order. When processing a sample, x_i, during the current epoch, denoted as t, the codebook undergoes updates as follows:

$$m_j(t+1) = m_j(t) + \alpha(t) * h_\vartheta(t)(\theta_j, \theta_{I(x_i)}) * [x_i - m_j(t)] \tag{2.8}$$

This update step has to be repeated for all the m_j for every x_i presented.

The learning rate, denoted as $\alpha(t)$, is a critical parameter in the training process that adjusts over time t, typically decreasing to ensure convergence of the algorithm. The Best Matching Unit (BMU), symbolized by $\theta_{I(x_i)}$, is a crucial concept in the mapping, representing the unit whose prototype vector is closest to the input sample x_i, effectively capturing the most similar feature representation within the map. This proximity is quantified by selecting the unit with the minimum distance to x_i, as formalized by the equation:

$$I(x_i) = \arg \min_{j \in 1,\ldots,L} |x_i - m_j| \tag{2.9}$$

Here, $I(x_i)$ identifies the index of the BMU.

$h_\vartheta(t)(\theta_j, \theta_{I(x_i)})$ is the neighborhood function at time t, dependent on the positions $\theta_j, \theta_{I(x_i)}$. It ensures the arrangement of prototype vectors in the feature space mirrors the topology of the output space.

Once the learning phase is completed, we proceed to visualization. In the scientific literature [81], various methods have been discussed. This thesis will focus on the following techniques: U-Matrix, Component Planes, Cluster Visualization, and the Codebook Map.

U-Matrix The U-Matrix is computed as the feature space distance between prototype vectors, specifically when the map units in the output space are adjacent.

$$\mu(\theta_j, \theta_k) = \|m_j - m_k\| \tag{2.10}$$

These values represent distances between nodes θ_j and θ_k. To aid visualization, the average for each node is computed using the following formula:

$$\bar{u}(\theta) = \frac{1}{|\xi_j|} \sum_{k \in \xi_j} u(\theta_j, \theta_k) \tag{2.11}$$

where ξ_j is the set of indices of units adjacent to j. The U-Matrix helps find spots where units are in between, outliers, and crowded areas where units are really close together.

Component Planes In Component Place visualization, every piece of information within the codebook is presented at once. A singular component plane displays the value of a chosen input space variable

from the codebook. The component plane representing variable i is denoted as $m(i)$, corresponding to the i^{th} row in matrix M.

Cluster Visualization The arrangement of codebook vectors into clusters indicates regions on the map that are densely packed and similar in the feature space. This clustering involves utilizing the prototype vectors as input for a clustering algorithm, revealing how the SOM codebook nodes group together.

After training, it's necessary to evaluate our MAP. To do so, several measures have been introduced to assess the effectiveness of specific SOMs. The survey [82] offers an overview of these diverse approaches. In this thesis, we employed the following metrics:

Quantization Error Quantization Error is the basic method for assessing the vector quantification properties of a map. It's define as follows:

$$E_j^Q = \sum_{i \in \xi_j} \|x_i - m_i\| \tag{2.12}$$

It denotes the total quantization error that is computed for each unit θ_j by adding all distances from the unit's prototype vector m_i to the data samples it represents. Lower is better.

Topographic Error Topographic Error is used for assessing the quality of vector projection, disregarding the quantization. It is defined as the percentage of data samples for which the BMU is not adjacent to the second-BMU. It defined as follows:

$$E^T = \frac{1}{N} \sum_{j}^{N} t(x_j) \tag{2.13}$$

N represents the total number of data samples considered when calculating the Topographic Error and t is defined as follows:

$$t_{x_j} = \begin{cases} 0 & \text{If } f(x) \text{ and } g(x) \text{ are neighbors,} \\ 1 & \text{Otherwise} \end{cases} \tag{2.14}$$

Where f returns the BMU and g returns the second BMU. For this metric, the lower is better, 0 indicates that all BMU and second-BMU nodes are neighbors, 1 indicate that BMU and second-BMU nodes are never neighbors.

Distortion Distortion is the loss function the SOM aims to minimize. It's computed by summing the squared Euclidean distances between the samples and SOM prototypes. These distances are weighted

by the neighborhood function, which relies on the distances to the best-matching unit on the map. Lower values of this metric indicate better performance

Silhouette Score Silhouette Score is used to evaluate the quality of clustering in data analysis. It assesses how well-separated clusters are and ranges from -1 to 1. The silhouette score s for a single sample is then given as:

$$E^s = \frac{b - a}{max(a, b)} \tag{2.15}$$

a represents the average distance between a particular sample and all other samples within the same cluster, while b signifies the average distance between that sample and all points in the closest neighboring cluster. The silhouette score for a group of samples is obtained by averaging the silhouette scores for each individual sample. A score close to 1 indicates that the data point is well-clustered and distant from other clusters. A score close to -1 suggests that the data point may have been assigned to the wrong cluster.

Neighborhood Preservation Neighborhood Preservation evaluates if data points that are close on the map also tend to be close to each other in the input space. This metric is measured by the formula is described in [83]. The result ranges from 0 to 1. Higher values indicate better performance.

2.5.5 Dataset Transformation

The training and prediction phases of MLP classifier can be impacted by the challenge of dealing with imbalanced datasets, which occurs when there is a discrepancy in the number of samples across different classes.

In this thesis, we deal with multi-class labeling problem, to measure the degree of imbalance of the dataset we use the measure proposed in [84] as an alternative for the well known imbalance-ratio used with binary class.

Numerous methods documented in the literature were explored, they are categorized into three groups: over-sampling, under-sampling, and a combination of both methods. The following presents an overview of the methods experimented with during this thesis.

Synthetic Minority Oversampling Technique (SMOTE) presented in [85], this method addresses class imbalance by generating synthetic samples for the minority class. It works by creating synthetic examples of the minority class by linearly interpolating between existing instances

Adaptive synthetic sampling (ADASYN) presented in [86], this method is similar to SMOTE, but it generates different number of samples. It dynamically adjusts the creation of synthetic samples using a weighted distribution. It focuses more on challenging regions, making it potentially more effective when dealing with extremely imbalanced datasets.

NearMiss presented in [87], it's an under-sampling technique working by selecting a subset of the majority class data points that are closest to the minority class samples, effectively reducing the number of majority class instances.

Combination of over- and under-sampling Due to the fact that SMOTE can generate noisy sample, Batista *et al.* propose in [88] SMOTETomek, a method combining over and under-sampling, this method uses SMOTE and Tomek links. The latter identifies pairs of instances (one from the majority class and one from the minority class) that are nearest neighbors and removes the majority class instance. Another method has been proposed by the same authors [89]. In this one, they use Edited Nearest Neighbours (ENN) in order to under-sample the majority class by removing certain instances.

Scaling To improve the model performance and ensure fair treatment of features, we scale the features to a uniform range using the following formula:

$$x_{scaled} = x_{tmp} * (max - min) + min \tag{2.16}$$

where $min = -1$ and $max = 1$ and:

$$x_{tmp} = \frac{(x - min(X))}{(max(X) - min(X))} \tag{2.17}$$

x represents the sample to be scaled and X the vector sample.

2.5.6 Assessment Software

In order to evalute the trust and liability metrics that we propose, we use two software, namely GRALAF for Graph Based Liability Analysis and Edgex. We describe briefly in the following the two software.

GRALAF Developed by O.Kalinagac *et al.* [90], GRALAF is a tool that performs near-real time anomaly detection and Root Cause Analysis (RCA) in a microservice environment based on events monitored by Prometheus. GRALAF uses the NOTEARS algorithm [91] to build a Causal Bayesian Network (CBN) from a dataset created by injecting faults into services. This CBN aids in understanding

the causal relationships between service fault states and their metrics. Subsequently, it aims to identify significant changes in microservice performance metrics or SLA violations. In this thesis, we used it in order to monitor service metrics.

Edgex The Edgex is used for IoT device management and is an open source software framework that offers device and application interoperability at the IoT edge. Edgex service is divided into four services, specifically the core, supporting, system management and devices services. Each service is composed of one or several microservices. Each service or microservice is provided by a service/microservice provider. For the evaluation, we focused on the core service, namely the core-metadata microservice. It communicates with other microservices such as core-command, UI and device-mqtt.

2.6 Conclusion

In this chapter, we presented the concepts and underlying technologies relevant to our subject. We began by introducing the foundational concepts associated with the concept of liability and accountability computing. Furthermore, we explored the meta-model TOSCA and TOSCA NFV and the fundamental concept of an ontology. Subsequently, we delved into the machine learning and neural network algorithms used during this thesis.

Chapter 3

State-of-the-Art

Contents

3.1 Introduction **54**
3.2 Coexisting Profiles in the Cloud-Edge-IoT Continuum **54**
3.2.1 Introduction 54
3.2.2 Inspire-5Gplus Manifest 54
3.2.3 Service-Level Agreement (SLA) 57
3.2.4 Existing Profiles on the Internet of Things (IoT) Ecosystem 68
3.2.5 Existing profiles in the Network Function Virtualization (NFV) ecosystem . 73
3.2.6 Conclusion 79
3.3 Liability and Trust Metrics **81**
3.3.1 Introduction 81
3.3.2 Trust Computation within the Cloud-Edge-IoT Continuum 81
3.3.3 Liability metrics within the Cloud-Edge-IoT Continuum 83
3.3.4 Monitoring and Detecting SLA Breaches 84
3.3.5 Financial Exposure To Risk Metric 85
3.3.6 Conclusion 86

3.1 Introduction

In this thesis, we address the challenge of liability management by proposing solutions related to defining the responsibility and liability relationship and metrics for liability and trust (FB.1 and FB.3 in Figure 1.2 Chapter 1). This chapter thus presents two state-of-the-art reviews for these contributions: one focusing on existing profiles within the cloud-Edge-IoT continuum, and the other centered on liability and trust metrics.

3.2 Coexisting Profiles in the Cloud-Edge-IoT Continuum

3.2.1 Introduction

In the following sections, we will present the state of the art conducted for the first Contribution, the responsibility model TRAILS (sTakeholder Responsibility AccountabIlity Liability deScriptor). This review begins by outlining the liability management strategy suggested by Inspire-5Gplus for the 5G ecosystem, with a special focus on the Inspire-5Gplus manifest. Next, we will delve into service level agreements, detailing their structure and existing models. We explore the already existing profiles in the cloud-Edge-IoT continuum and conclude by highlighting the shortcomings of these various profiles.

3.2.2 Inspire-5Gplus Manifest

Inspire-5Gplus project objectives are to introduce approaches that enable liability end-to-end delivery of 5G services. To achieve this goal, the project proposes several mechanisms, among which liability manifests play a crucial role. The Inspire-5Gplus manifest is a structured document that aims to formalize *responsibility*, *liability* and *accountability*. The characteristics of a such document has been described in [92]. The characteristics of the Inspire-5Gplus manifest are summarized in Table 3.1. The primary feature is to enable supply chain stakeholders to clearly state their committed responsibilities and the specific conditions under which these responsibilities hold valid (usage conditions). Additionally, manifests provide the flexibility for users to assign themselves responsibilities by defining operation limitations. Also, the Inspire-5Gplus manifest offers a means to clarify the demonstration requirements expected from each stakeholder. It exhibits modularity, allowing the composition of multiple components, and it effectively captures the relationships between

stakeholders throughout the product's lifecycle. Another notable characteristic of the Inspire-5Gplus manifest is its generic nature, suitable to any type of component or service, whether it is an IoT or VNF, located in the Cloud or at the Edge. The manifest will reflect the clauses of a contract. The contract between two entities comprises the following clauses: obligations and usage conditions, measurable objectives, and rewards and penalties. Thus, there is a one-to-one correspondence between the manifest and the contract. A legal expert will be responsible for ensuring that all contract clauses are reflected in the manifest and vice versa. Otherwise, it means that the contract contains clauses that are not relevant to the service or that they are not consistent with each other. Finally, Inspire-5Gplus manifest requires supply chain stakeholders to sign their contribution to the manifest to materialize their commitment to their responsibilities.

Features	**Description**
Responsibility	Enable stakeholders to define responsibilities and associated conditions (usage conditions) clearly.
Accountability	Allow for a clear specification of what each stakeholder must demonstrate.
Liability	Enable the clear assignment of responsibilities and their acceptance, similar to a contract.
Modularity	Enable the composition of multiple components while accurately documenting the relationships among stakeholders.
Genericity	Allow the description of any component or service type, such as IoT, VNF, or Network Service.

Table 3.1: Inspire-5Gplus manifest characteristics

The lifecycle of the Inspire-5Gplus manifest has been designed to generalize the stages that a component goes through from the manufacturer to the end-user. We arrive at the manifest's lifecycle, as depicted in Figure 3.1. Initially, manifest will describe a class of component. The component is constructed by the *manufacturer* using building blocks supplied by software editors, hardware manufacturers, or Service Providers. Subsequently, the *manufacturer* presents a first version of the manifest, which is based on feature descriptions and preliminary usage conditions. Then comes the testing stage, where the *validator* assesses the component by conducting tests, evaluating risks, and ensuring compliance with relevant requirements. Drawing from their observations, the *validator* may incorporate additional properties or describe controls and requirements, referred to as usage condition.

These constraints are essential for the infrastructure operator to uphold regular functionality and prevent exploitation of known vulnerabilities. After that, the manifest is presented as a service offered to the *infrastructure operator*, operating as an annex to the contractual agreement that binds the involved parties. It defines the service, guarantee, and SLA that the infrastructure operator expects from the manufacturer. Furthermore, the infrastructure operator includes the component in its Catalog and may conduct further tests. It identifies operation limitations, similar to usage conditions, but tailored to meet specific infrastructure requirements, company policies, or local regulations, and these constraints are not accessible to other stakeholders. From there, the operator proposes an internal service to the deployment entity. This entity adds instantiation details of the component to the manifest. It utilizes the component to provide a service to a *vertical service provider* who provides specialized industry-specific solutions. Finally, the manifest is instantiated by the *vertical service provider*, it is employed to determine if and how to monitor and manage the component. Additionally, it serves as a foundation for defining the expected behavior for monitoring purposes.

The Inspire-5Gplus manifest is included within the first FB in the Liability & Accountability management functional block (Figure 1.2) referred to as "Defining accountability and liability relationships" since they hold crucial information for the identification of commitments and responsibilities.

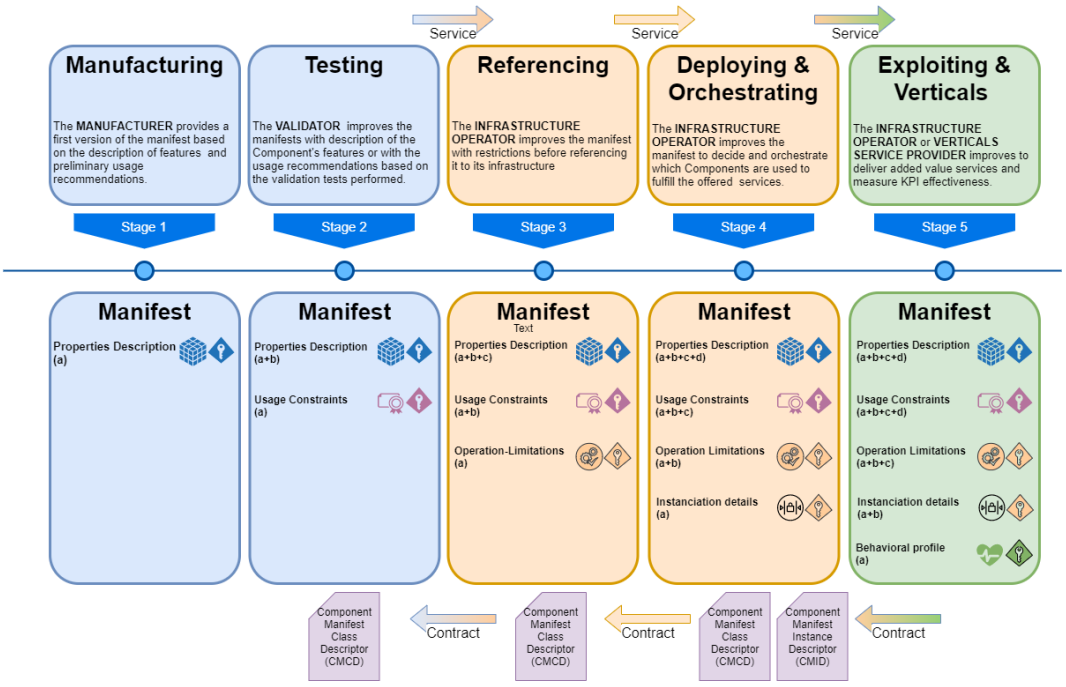


Figure 3.1: Inspire-5Gplus manifest lifecycle

3.2.3 Service-Level Agreement (SLA)

We have identified several definitions of SLAs by domain. In the telecommunications field, [93] is the key document focused on standardizing SLAs. ETSI defines the SLA as a contract that defines an agreement between two parties: the user and the service provider. It describes the terms and conditions for service delivery. On the user's side, it identifies the user's requirements, while on the provider's side, it outlines the provider's commitments to the client. In the web service, an SLA is defined as an agreement used to guarantee web service delivery. It defines the understanding and expectations from service provider and service consumer [94]. In the realm of cloud computing, an SLA is defined as a mutually agreed-upon contract between a service provider and a customer, wherein specified parameters outline the expected service standards the provider must ensure [95]. Based on the different study mentioned above, we can conclude a general structure of an SLA which are: the involved parties, the validity period of the agreement, the scope of services covered within the agreement, guarantees in terms of targets, penalties and the suspension or termination and sanctions. The most commonly used terms in SLAs, as encountered throughout this thesis, are defined in the following.

Quality of Service (QoS) QoS refers to a service's ability to meet various user requirements, such as availability, performance and reliability. The primary components of quality of service are provided through metrics characterized by a type, a unit, and a calculation function.

Quality of Experience (QoE) QoE is a quality metric that provides a holistic measure of the users' perception of the quality. [96]

Quality of Protection (QoP) QoP refers to the security measure put in place for the service. It includes data confidentiality, integrity, availability and access control [97].

Key Performance Indicator (KPI) KPI are measurable metrics that organizations use to evaluate and track their progress toward specific goals and objectives.

Service Level Objective (SLO) SLO is a specific, measurable target or goal that defines the level of performance or service quality a service provider commits to delivering to its customers or users through the SLA.

Service Level Indicator (SLI) SLI is an SLA clause that refers to a quantitative measurement or metric used to assess the performance or quality of a specific aspect of a service.

Several models have been proposed in order to formalize the structure of the SLA. Understanding commonly used models is crucial for identifying components within the proposed responsibility model. Below is an overview of the models explored throughout this thesis.

ETSI SLA Model ETSI introduces a comprehensive SLA model [93], illustrated in Figure 3.2, which comprises several crucial components. These components include the definition of `Parties` representing the contractual entities, categorized as Signatory parties (contractual parties) and Third Parties (optional trusted third parties). Additionally, the model comprises `Service Level Objectives (SLOs)`, reflecting user needs, it is grouped into four categories: performance SLOs (related to availability and response time), security SLOs (concerning authentication and encryption), data management SLOs (addressing mirroring and backup), and personal data protection SLOs. The `Service` element defines the service offers tailored to meet user demands specified in the SLO. Moreover, `Constraints` describe conditions imposed by either the provider or customer, spanning strategic constraints (affecting deployment priorities), financial constraints (involving payment and usage patterns), legal constraints (including licensing and compliance), and technical constraints (specifying prerequisites). `Use Condition` specifies limitations set by the provider within contracts, influencing end-to-end QoS in supply chains with multiple providers. The `Coverage` element addresses geographical characteristics using maps and tables to delineate service locations. `Guarantees` outline provider commitments for specific SLOs, with user-requested guarantees and compliance percentages determined through statistical models. Furthermore, the model covers `E2E Management Action` describing provider actions to achieve required SLOs, such as dynamic reconfiguration. It also considers `SLA Violations` with threshold values indicating breaches, triggered by factors like non-compliance with expected QoS. Additionally, `Penalty` defines penalty policies in case providers fail to meet client requirements, including penalties based on the guaranteed service availability ratio. Lastly, `SLA Cost` details the monetary commitment needed to access specific SLA levels aligned with desired user strategies, such as environmental sustainability, best effort, or dedicated QoS. This holistic SLA model provides a structured framework for defining and managing service agreements, ensuring clarity and alignment between service providers and users across diverse industries and scenarios.

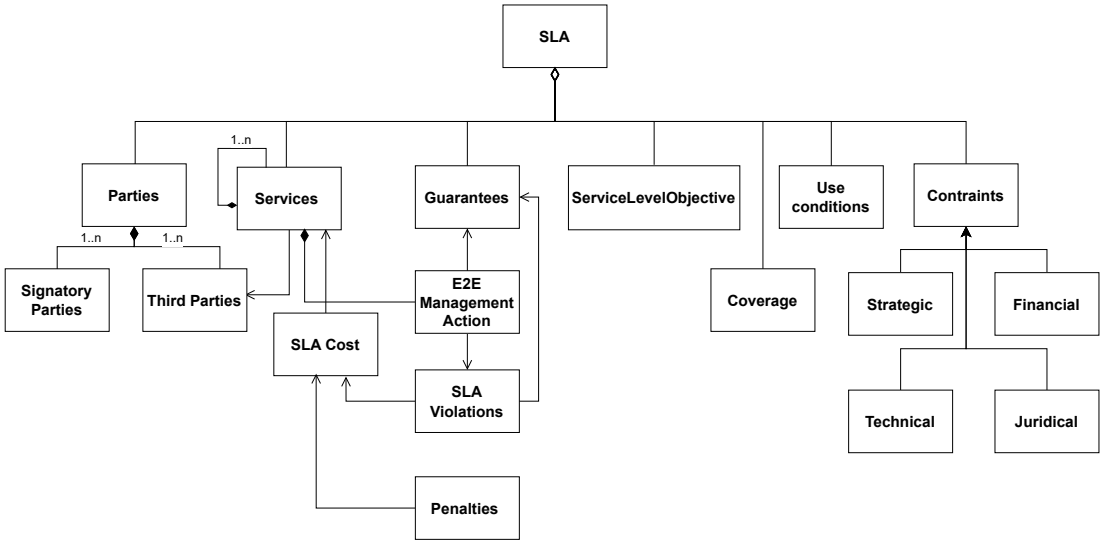


Figure 3.2: ETSI SLA Model [93]

WSLA IBM research introduces the Web Service Level Agreement (WSLA) [98], a framework aimed at defining and monitoring SLAs for web services, it is also applicable to any other environments such as cloud computing. The framework is considered as one of the most mature specifications for defining an SLA. WSLA consists of a flexible and extensible language based on XML and an architecture that takes into account multiple monitoring services. The WSLA is described using a metamodel, illustrated in Figure 3.3. It includes three sections: one section describes the parties, one section contains one or more service definitions, and one section defines the obligations. WSLA supports two types of actors: signatories, namely the service provider and the client, and trusted third parties. A service definition contains one or more service objects. A service object is an abstraction of a service. A service object can have one or more ”SLAParameters” to define associated guarantees. Each ”SLAParameter” is defined by a metric. This metric is calculated by defining a measurement directive or a function. The obligations section contains two types of obligations: a SLO and an action guarantee. An action guarantee is the promise to do something in a defined situation.

Listing 3.1 provides an example of an SLO writing with the WSLA. This SLO is provided by `End2EndServiceProvider`, and it guarantees that the SLA parameter `AverageResponseTime` must be less than 0.1 if the SLA parameter Transactions is less than 4500$.

Listing 3.1: Example of a Service Level Objective (SLO) using WSLA

```
1<ServiceLevelObjective name="NewSLO">
```

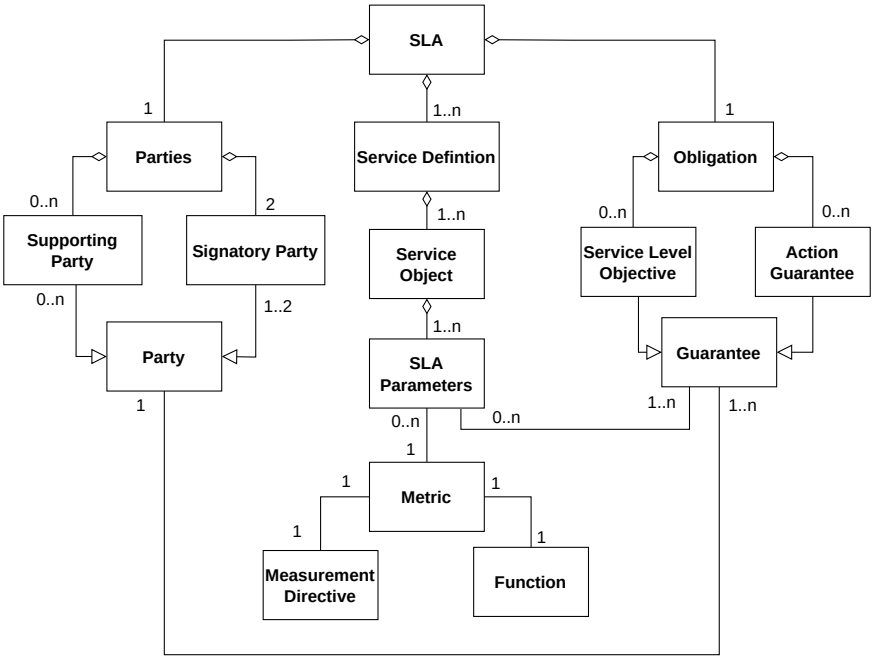


Figure 3.3: WSLA Meta-Model [98]

```
<Obliged>End2EndServiceProvider</Obliged>
<Expression>
   <Implies>
      <Expression>
         <Predicate xsi:type="Less">
         <SLAParamter>Transactions</SLAParameter>
         <Value>4500</Value>
         </Predicate>
      </Expression>
      <Expression>
         <Predicate xsi:type="Less">
         <SLAParamter>AverageResponseTime</SLAParameter>
         <Value>0.1</Value>
         </Predicate>
      </Expression>
   </Implies>
</Expression>
</ServiceLevelObjective>
```

WS-Agreement WS-Agreement [99] is a protocol proposed by the OpenGridForum (OGF). It defines a standard for the creation and specification of SLAs for web services. The Figure 3.4 illustrates the basic structure of an agreement according to WS-Agreement. The structure of an agreement consists of several key components. Each agreement is distinguished by a unique identifier. The Agreement

context holds metadata about the agreement, including details about the involved parties, validity, and references to the contract model. The Terms of the Agreement include Service Description Terms, which describe the offered service and its properties, Service References pointing to external services (though not handled in the default implementation). Guarantee Terms define constraints within the agreement, featuring a unique name, obligations (typically assigned to the Service Provider), and specifying which service elements they apply to. SLO set constraints on service attributes, and Business Values establish penalties and rewards associated with these objectives. This versatile structure allows for adaptable agreement management, even if some features are not fully implemented in the default setup. The standard proposes a web service protocol as a comprehensive tool for agreement management. It enables the presentation of potential agreement offers via templates, the creation of customized agreement proposals, negotiation within specific constraints, and the finalization of agreements between service providers and customers with detailed conditions and restrictions. Additionally, it offers monitoring capabilities to ensure the fulfillment of these agreements.

Listing 3.2 provides an example of SLA using the WS-Agreement. This SLA outlines an agreement between ”Customer” and ”ServiceProvider”, specifying service availability constraints.

Listing 3.2: Example of an SLA using WS-Agreement

```
1
2<?xml version="1.0" encoding="UTF-8"?>
3<wsag:Agreement xmlns:wsag="http://www.ogf.org/namespaces/ws-agreement"
4  AgreementId="example-agreement">
5
6  <wsag:Name>Example Agreement</wsag:Name>
7  <wsag:Context>
8    <wsag:AgreementInitiator>Customer</wsag:AgreementInitiator>
9    <wsag:ServiceProvider>ServiceProvider</wsag:ServiceProvider>
10   <wsag:ExpirationTime>2023-10-07T14:00:00</wsag:ExpirationTime>
11   <wsag:TemplateId>template001</wsag:TemplateId>
12   <sla:Service xmlns:sla="http://service.provider.eu">example-service</sla:Service>
13 </wsag:Context>
14 <wsag:Terms>
15   <wsag:All>
16     <wsag:ServiceDescriptionTerms Name="SDT" ServiceName="ServiceName"/>
17     <wsag:ServiceProperties Name="ServiceProperties" ServiceName="ServiceName">
18       <wsag:VariableSet>
19         <wsag:Variable Name="availability" Metric="xs:double">
20           <wsag:Location>metric1</wsag:Location>
21         </wsag:Variable>
22       </wsag:VariableSet>
23     </wsag:ServiceProperties>
24     <wsag:GuaranteeTerm Name="availability">
```

```
25        <wsag:ServiceScope ServiceName="ServiceName"/>
26        <wsag:ServiceLevelObjective>
27          <wsag:KPITarget>
28            <wsag:KPIName>AVAILABILITY</wsag:KPIName>
29            <wsag:CustomServiceLevel>
30              {"constraint" : "availability BETWEEN (0.99, 1)"}
31            </wsag:CustomServiceLevel>
32          </wsag:KPITarget>
33        </wsag:ServiceLevelObjective>
34      </wsag:GuaranteeTerm>
35    </wsag:All>
36  </wsag:Terms>
37</wsag:Agreement>
```

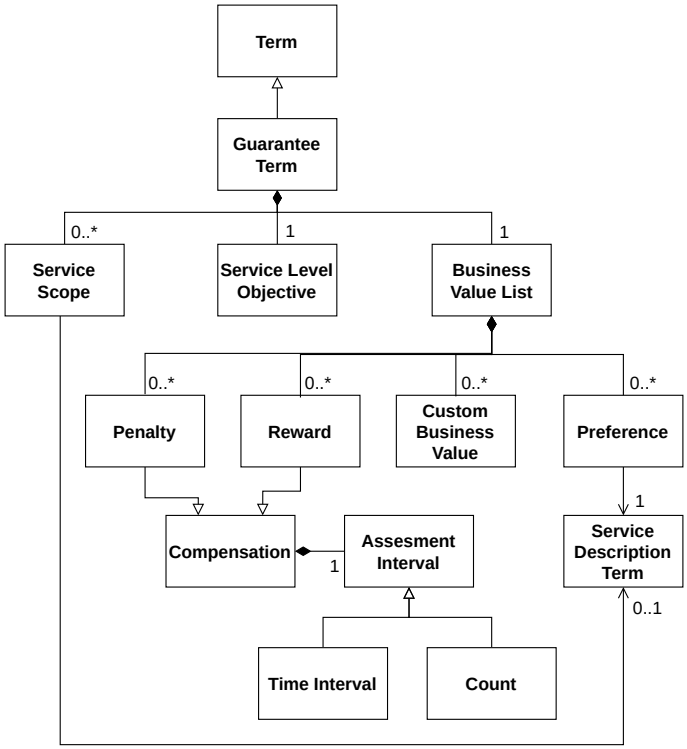


Figure 3.4: WS-Agreement Meta-Model [98]

The SLA@SOI The SLA@SOI project proposes the SLA* [100], an abstract syntax for SLAs that provides a highly expressive and extensible solution. It draws primary inspiration from WS-Agreement. This approach encourages the formalization of SLAs in any language for any service, eliminating the restrictions imposed by XML. As shown by the Figure 3.5, an SLA* is essentially a template SLA with an extended set of attributes that precisely specify the contract's validity. A template SLA comprises five key sections: the template SLA attributes, the agreement parties, the service descriptions, the

variable declarations, and the agreement terms that detail quality of service guarantees. In this abstract SLA syntax, parties are identified by their roles (provider and client). Service descriptions are established using interface declarations, assigning local identifiers to interfaces, whether they are functional interfaces or resource descriptions. Variable declarations are also provided to enhance readability and prevent content repetition. Furthermore, agreement terms are formalized into two categories: action guarantees and state guarantees. In addition to simple expressions, SLA* introduces a formal model for penalty formalization. The primary motivation behind this approach is to ensure openness and applicability across various domains, free from specific language, taxonomy, or technology dependencies.

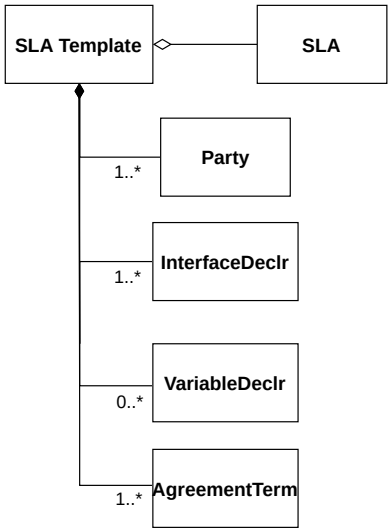


Figure 3.5: SLA* Meta-Model [100]

CSLA [95] is a language designed to address the needs of cloud SLA. CSLA focuses on adapting to the changing nature of clouds by adding SLA features that can handle violations. Figure 3.6 shows the metamodel of CSLA. It includes three fundamental sections, parties, validity, and templates. The validity section specifies the duration of the agreement's effectiveness and identifies the parties bound by it, distinguishing between signatory parties (comprising the service provider and service customer) and supporting parties, which may include trusted third parties. Templates, on the other hand, serve as structural frameworks for the SLA, featuring five key components. These encompass `Services Definition`, which describe the services offered, adhering to cloud service models like SaaS, PaaS, and IaaS; `Parameters`, defining relevant variables for Metric, `Monitoring`, and Schedule elements;

Guarantees, which include Scope, `Requirements`, Terms, and Penalties; `Billing`, detailing the billing method; and `Terminations`, which describe the procedure for ending the agreement.

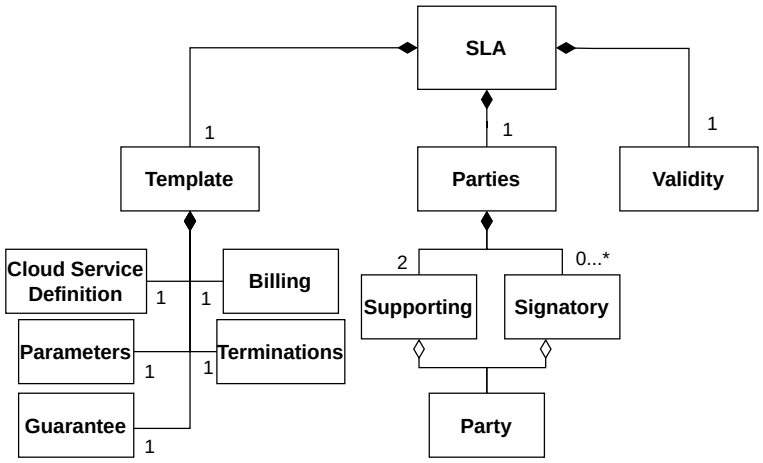


Figure 3.6: CSLA Meta-Model [95]

Listing 3.3 provides an example of an SLA using CSLA. This SLA, identified as "CSLA1," was agreed upon on 10-10-2023. It involves two main parties, a cloud provider, and a cloud consumer (Customer), each with their respective contact details. The terms within the SLA include two objectives, focusing on response time and availability, with specific metrics, thresholds, and monitoring parameters defined for each objective.

Listing 3.3: Example of an SLA using CSLA

```
1
2<csla:parties>
3    <csla:cloudProvider>
4        <csla:name>provider</csla:name>
5        <csla:contact>
6            <csla:address>address</csla:address>
7            <csla:email>email@email.fr</csla:email>
8            <csla:phoneNumber>+33 (0)0 00 00 00 00</csla:phoneNumber>
9        </csla:contact>
10   </csla:cloudProvider>
11   <csla:cloudConsumer>
12       <csla:name>Customer</csla:name>
13       <csla:contact>
14           <csla:address>France</csla:address>
15           <csla:email>email@email.fr</csla:email>
16           <csla:phoneNumber>+33 (0)0 00 00 00 00</csla:phoneNumber>
17       </csla:contact>
18   </csla:cloudConsumer>
19</csla:parties>
20<csla:terms>
```

```
21    <csla:term id="T1" operator="and">
22        <csla:item id="responseTimeTerm"/>
23        <csla:item id="availabilityTerm"/>
24    </csla:term>
25    <csla:objective id="responseTimeTerm" priority="1" actor="provider">
26        <csla:precondition policy="Required">
27            <csla:description>Data size less than 1 TB</csla:description>
28        </csla:precondition>
29        <csla:expression metric="Rt" comparator="lt" threshold="3" unit="second" monitoring=
              "Mon1" schedule="Sch1" Confidence="99" fuzzinessValue="0.2" fuzzinessPercentage=
              "10"/>
30    </csla:objective>
31    <csla:objective id="availabilityTerm" priority="2" actor="provider">
32        <csla:expression metric="Av" comparator="gt" threshold="98" unit="%"
33 monitoring="Mon2" Confidence="99" fuzzinessValue="1" fuzzinessPercentage="5"/>
34    </csla:objective>
35</csla:terms>
```

Wonjiga *et al.* [101] propose Extended CSLA (ECSLA), an extension of the CSLA language. ECSLA extends the original language in order to have a standard method to define security monitoring SLAs. This involves adding new features, including a new generic service, a structure to define security vulnerabilities, and a definition of security monitoring service.

SLAng SLAng [102] can specify horizontal SLAs (agreements between parties providing the same service) and vertical SLAs (agreements between subordinate pairs) between users and service providers. It is easily extensible to increase expressiveness and can be used to regulate possible agreements among different types of parties involved in the contract (e.g., application, web service, component, container, storage, and network). The language consists of three parts: service description (e.g., service location, provider information), contract statements (e.g., engagement duration, penalties), and service level specification (e.g., QoS metric descriptions). SLAng's syntax is defined using an XML schema, facilitating integration with existing service description languages. For example, SLAng can be combined with WSDL (Web Services Description Language) and BPEL (Business Process Execution Language), both of which are defined using XML schemas.

RBSLA Rule Based Service Level Agreement (RBSLA) [103] is a language based on RuleML [104] and logic programming. It allows for the implementation of SLAs in a machine-readable syntax for automated monitoring. RBSLA follows the design principle of RuleML, where new concepts are introduced in layers that enrich the RuleML core. It was designed to be compatible with the Semantic Web and other existing standards. The language itself is very expressive, but like other logic

programming-based languages, it can be challenging for non-experts to understand and use.

CSLAM CSLAM [105] is an SLA framework developed for the cloud environment. It has a language for defining an agreement between two parties and a management mechanism to deploy and monitor the SLA. It is based on WSLA with some changes and extensions to adapt it to the cloud environment. CSLAM is structured into three substructures: i) Parties: where the user and service provider are defined. ii) Obligations: where SLOs and guarantees are defined. iii) Cloud Major Service: this part defines the hierarchical structure of the cloud service and the parameters to be satisfied, where each parameter can define a cloud metric or use a function to define it. Each metric will be matched with one or more.

To address the lack of flexibility in representing SLAs, some authors have proposed representing them using ontologies. In [106], Glen Dobson *et al.* present QoSOnt, an ontology for quality of service primarily developed for applications in the field of service-centric systems. QoSOnt is designed to promote consensus on QoS concepts by providing a sufficiently generic model for reuse across multiple domains. QoSOnt consists of three layers, each of which is an ontology: i) Usage Domains: it links the QoS to be achieved with a specific domain, currently supporting network and system domains. ii) Attributes: it presents attributes related to the QoS to be achieved, for example: integrity, security, confidentiality, etc. iii) Base QoS: it represents a minimal set of concepts related to QoS, such as measurability, units, metrics, etc. Using ontologies, the authors of [107] propose a QoS model for the cloud that focuses on three dimensions: resource utilization by the service, service performance, and cost. Three properties are defined: constraints of each party, the influence between different parties, and the weight of each party. The proposed ontology focuses solely on QoS properties without detailing any SLA concepts. In [108], the authors propose an ontology for SLAs in the cloud environment. They illustrate how SLA monitoring and management can be simplified using semantic web languages such as OWL, RDF, and SPARQL. They also demonstrate, with a prototype, how SLA measurements can be automatically extracted from the legal service terms available on the Cloud Provider's website. Taher Labidi *et. al.* have proposed CSLAOnto [109], a generic and semantically rich SLA model based on ontologies. The authors follow the MethOntology [110] guideline, which consists of four steps: CSLAOnto specification, CSLAOnto Conceptualization, CSLAOnto Formalization, and CSLAOnto Validation. The model draws inspiration from WS-agreement and WSLA and supports open cloud computing interfaces. Its structure includes elements such as Context, ApplicationDomain, Version,

Parties, SignatoryParty, SupportingParty, and terms. A prototype has been proposed to validate the model, as well as the monitoring process where SLA evaluation and guarantee actions are automatically triggered.

One of the challenges that can arise when establishing an SLA in a multi-actor, multi-domain, dynamic environment with multiple levels of delegation is the interdependence between SLAs. SLAs can establish dependencies across different layers in a service stack, where actors in the hierarchy serve as client/providers. Violations in one layer can have repercussions on higher layers, necessitating automated solutions. The challenge lies in translating SLAs between these levels. In [111], the author describes this challenge and provides an overview of existing solutions. Two types of dependencies are identified: interdependencies, which pertain to dependencies among SLAs within the same level, and interdependencies, which concern dependencies among SLAs from different layers or levels, signifying vertical relationships between them. According to [111], various initiatives propose to handle interdependencies challenge. Karaenke *et al.* [112] presents a software architecture that streamlines SLA negotiation and SLA-based resource management within complex agreement hierarchies, fostering interoperability in heterogeneous distributed environments and seamless integration with existing service-oriented systems. Di Modica *et al.*. [113] propose enhancements to the WS-Agreement specification, enabling parties to re-negotiate and modify agreement terms during service provision, offering flexibility in scenarios involving multiple service providers to prevent rigid SLAs from negatively affecting the final service quality. Interdependencies introduce another challenge, namely, the translation of SLAs. This issue can be addressed through three primary approaches: monitoring, planning, and prediction. In terms of monitoring, there are various works such as MoDe4SLA [114], NITY [115], and LoM2HiS [116]. For instance, MoDe4SLA, a framework designed to identify complex dependencies within service compositions, helps explain the causes of SLA violations when services within the composition malfunction. In the planning domain, Chen *et al.* [117] suggest an SLA decomposition approach (known as translation top-down) to automate system design and monitoring to achieve high-level business objectives, involving the translation of high-level goals into manageable sub-goals using various low-level attributes and metrics. Regarding prediction, Wada *et al.* [118] propose an optimization framework called E3 to tackle the QoS-aware service composition problem in Service-Oriented Architecture. E3 employs a multiobjective genetic algorithm to heuristically and efficiently solve this problem. It can simultaneously consider multiple SLAs and produce a set of

Pareto solutions with equivalent quality to satisfy multiple SLAs.

3.2.4 Existing Profiles on the Internet of Things (IoT) Ecosystem

In the IoT ecosystem, there are specific IoT profiles that have been developed to address various challenges and requirements. These profiles offer standardized solutions for diverse needs such as seamless device communication, robust security and efficient device management. In the following, we are going to describe the main profiles such as MUD (Manufacturer Usage Description) and the SUIT (Firmware Update and Integrity Test) Manifest.

Software Updates for Internet of Things (SUIT) Manifest The IETF Software Updates for Internet of Things (SUIT) working group has set out to create a firmware update solution tailored to the unique needs of IoT devices [2]. This update process is engineered to safeguard against unauthorized and malicious firmware modifications, ensuring the integrity and confidentiality of firmware images. By doing so, it effectively reduces the susceptibility of devices to compromise and potential reverse-engineering attacks. Therefore, the group defines both a mechanism for transporting firmware images [119] and a manifest needed to securely update an IoT system [120]. The SUIT manifest plays a pivotal role in the update validation process, as they contain vital information. These documents serve as guides for making determinations regarding trust in the author, the integrity of the image, its applicability, storage considerations, and more. Table 3.2 summarizes mandatory and recommended elements. This table outlines essential attributes within the SUIT manifest for software updates. It distinguishes between mandatory and recommended attributes, including those like the `Version Identifier`, `Storage Location`, and `Payload Digest`, which are crucial for ensuring the security and integrity of the update process. Additionally, recommended attributes like `Vendor ID Condition` and `Class ID Condition` provide valuable context for device management.

Attribute	Status	Explanation
Vendor ID Condition	Recommended	Distinguish products from different vendors.
Class ID Condition	Recommended	Distinguish incompatible devices in a vendor's infrastructure.
Version identifier	Mandatory	Manifest format version, requiring a sequential number to prevent rollbacks.
Storage Location	Mandatory	Informs the device of the updated component.
Payload Digest	Mandatory	The digest of the payload to ensure authenticity.
Payload Format	Mandatory	Describes the format of the payload.
Size	Mandatory	The size of the payload in bytes.
Dependencies	Mandatory	A list of digest/URI pairs linking manifests that are needed to form a complete update.
Signature	Mandatory	The Signature element secures the manifest's content against changes and verifies the signer's authenticity.

Table 3.2: Mandatory and Recommended manifest attributes of the SUIT manifest.

Lightweight M2M (LwM2M) Data Model LwM2M is a protocol developed by OMA SpecWorks for remote device management in the IoT and other Machine-to-Machine (M2M) applications [3]. The application data is encapsulated using the Constrained Application Protocol (CoAP) [3]. In the LwM2M protocol, three entities are at play: LwM2M Clients on end devices communicate with servers to manage device resources via a standardized data model, identified uniquely by an Endpoint Client Name. The LwM2M Bootstrap Server initializes the data models and connections for clients during boot-up. LwM2M Servers maintain connections with clients, enabling reading and writing of exposed data models. The LwM2M data model follows a structured two-level tree format, distinguishing entities at each level through numerical identifiers. At the initial level, we encounter `Objects`, each representing a distinct data concept accessible via the LwM2M Client. These Objects are characterized by essential attributes, including their `Name` for describing the object, the `Object ID` as a numerical identifier, the `Instances` which can be categorized as Single or Multiple, and the `Mandatory` status indicating whether all LwM2M Client implementations must support the object. Additionally, there is the `Object Uniform Resource Name (URN)` format in the pattern of ObjectID/ObjectInstance/ResourceID. Moving to the second level, we find `Resource definitions` within each Object definition, which offer a comprehensive set of information about the resources. This includes the `ID` serving as the numerical identifier for the resource, a `Name` providing a concise resource description, `Operations` represented as R (read-only Resource), W (write-only Resource),

RW (writable Resource), E (executable Resource), or left empty. `Instances` can be classified as Single or Multiple, and `Mandatory` status signifies whether both the LWM2M Server and the LWM2M Client must support the Resource when marked as Mandatory, or ideally support it when marked as Optional. Further attributes encompass the `Type` indicating the resource's data type, `Range or Enumeration` specifying valid values, `Units` indicating measurement units for numerical values, and finally, `Description` furnishing a detailed resource description.

Manufacturer Usage Definition (MUD) profile The IoT's rapid expansion heightens cybersecurity risks, expanding attack surfaces and enabling severe threats. To address this, the Manufacturer Usage Description (MUD) [4], standardized in 2019 by the Internet Engineering Task Force (IETF), aims to control IoT device behavior for secure deployment despite device diversity and management challenges. MUD establishes an architecture and data model to control communication for specific devices. Manufacturers can define behavior profiles for devices, shaping communication endpoints through policies or Access Control Lists (ACLs) to reduce attack surfaces. The architecture supports profile enforcement within device networks. The uptake of MUD has sparked considerable interest in both research and standardization communities, particularly by the National Institute of Standards and Technology (NIST) as a solution for security threats and Denial-of-Service (DoS) attacks in IoT [121], and endorsed by the European Union Agency for Cybersecurity (ENISA) to enhance IoT security practices [122].

The foundational elements of the MUD architecture are designed to facilitate the deployment and utilization of a MUD file, detailing the device's behavior as defined by its manufacturer. The concept of the manufacturer, as outlined in [4], encompasses the entity or organization responsible for specifying the intended use of the device. Figure 3.7 illustrates these components, along with the primary interactions involved in acquiring a MUD file. The architecture encompasses a Thing, representing the IoT device, responsible for generating and transmitting a MUD URL. Additionally, a router grants network access to the device, while the MUD Manager initiates requests for MUD file retrieval based on the received MUD URL. Finally, the MUD File Server hosts the actual MUD files.

According to MUD specifications, IoT devices share a MUD URL with the MUD Manager, indicating the MUD file's location. The router facilitates this communication. The MUD Manager, using the MUD URL, requests the MUD file from the MUD file server, validating and parsing it upon receipt. Network components are then configured based on the file's restrictions.

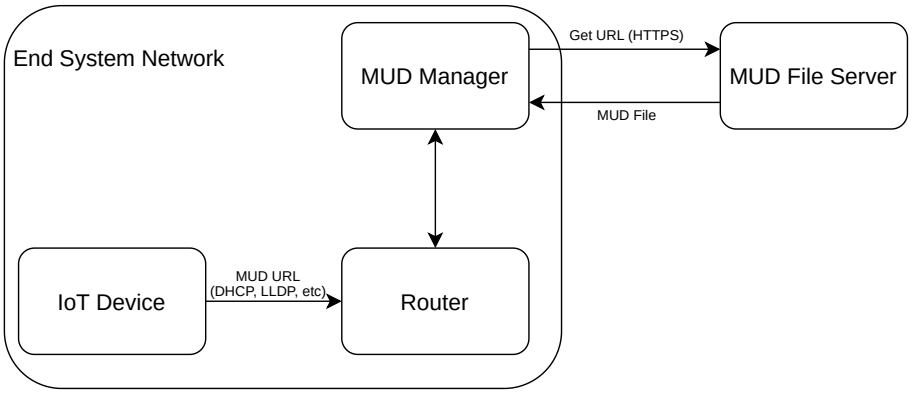


Figure 3.7: MUD architecture [4]

The MUD standard employs ACLs to regulate communications of IoT devices, using Yet Another Next Generation (YANG) to model network restrictions and JavaScript Object Notation (JSON) for serialization. The MUD model extends the YANG data model for ACLs, enhancing expressiveness. The 'mud' container in the model provides MUD file details like storage location ('mud-url') and generation time ('last-update'). The MUD data model's 'acls' container adds further restrictions, allowing/denying communication with specific IP addresses, ports, or devices from the same manufacturer ('manufacturer' and 'same-manufacturer').

An example of MUD file is presented in Listing 3.4. The file begins by specifying its version, hosting URL, last update timestamp, and a link to its digital signature for verification. Within the "acls" container, an ACL policy named "to-device-policy" is defined. In this policy, a specific rule labeled "allow-ntp" is detailed. This rule pertains to incoming traffic initiated from the device and allows traffic on User Datagram Protocol (UDP) port 123, associated with Network Time Protocol (NTP). The "actions" section of this rule designates that the controller enforces it, and it applies to traffic directed to the device.

Listing 3.4: Example of MUD file

```
{
 "mud-version": 1,
 "mud-url": "https://example.com/mudfile",
 "last-update": "2023-08-10T10:00:00Z",
 "mud-signature": "https://example.com/mudfile.sig",
 "acls": {
  "to-device-policy": {
   "acl-list": [
    {
```

```
10          "name": "allow-ntp",
11          "acces": [
12            {
13              "name": "allow-ntp-udp",
14              "matches": {
15                "ietf-mud:mud": {
16                  "direction-initiated": "from-device",
17                  "port": 123
18                }
19              },
20              "actions": {
21                "ietf-mud:controller": {
22                  "direction-initiated": "to-device",
23                  "order": 1
24                }
25              }
26            }
27          ]
28        }
29      ]
30    }
31  }
32}
```

Figure 3.8 illustrates the relationship between a MUD file and an IoT device's lifecycle. Following the phases defined by [123], we found some similarity with the Inspire-5Gplus manifest lifecycle. The device's journey begins with manufacturing, where the manufacturer creates a MUD file containing network access controls. During onboarding, the IoT device is installed, and the MUD file configures network components based on restrictions. In the operational phase, the MUD file enforces constraints through technologies like SDN. Software updates or vulnerabilities may trigger MUD file updates, ensuring secure device behavior.

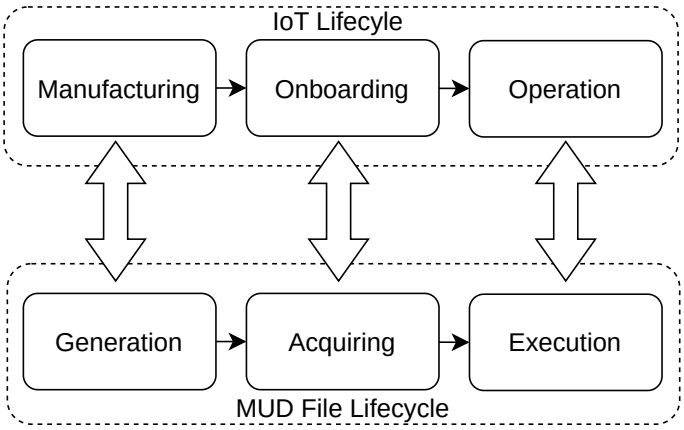


Figure 3.8: MUD & IoT lifecycle [123], The arrows in the diagram indicate the flow and relationship between different stages of both the IoT lifecycle and the MUD lifecycle

3.2.5 Existing profiles in the Network Function Virtualization (NFV) ecosystem

The NFV architecture relies on descriptors to abstractly represent and manage network functions and services within virtualized environments. These descriptors, standardized and consistent, facilitate automation and orchestration, enabling dynamic deployment and scaling of services. They define connectivity, resource requirements, and service chaining, promoting efficient resource allocation and flexibility. Descriptors streamline the lifecycle management, allowing for updates and vendor independence while enhancing operational efficiency and agility. Next, we explain the main descriptors in the NFV architecture: the Virtual Network Function Descriptor (VNFD) and the Network Service Descriptor (NSD). There are other descriptors like Virtual Link Descriptors (VLDs) and VNF Forwarding Graph Descriptors (VNFFGDs), which will briefly touch on without going into too much detail.

Virtual Network Function Descriptor (VNFD) Before delving into the VNFD, let us provide an introduction to Virtualized Network Functions (VNFs). A VNF is a software-based representation of a distinct network service or functionality, such as a firewall, router, or load balancer. It operates within a virtualized environment on standard hardware, effectively decoupling it from proprietary hardware appliances. VNFs are under the management of orchestration platforms, facilitating dynamic instantiation, scaling, and termination in response to network demands. These functions can be interconnected to form intricate network services and interact with the underlying network infrastructure to manage data processing and forwarding. VNFs are characterized by their scalability,

flexibility, and agility, allowing seamless software updates, enhancements, and upgrades without necessitating alterations to the physical hardware. This technological innovation significantly elevates the efficiency, cost-effectiveness, and adaptability of contemporary networking solutions. VNFs are commonly distributed and implemented as packages designed for utilization on virtualization infrastructure. This package is called the VNF package, it encapsulates the VNFD along with the following files: the VNF Package manifest file, denoted by the extension *.mf*, it's structured as a name-value format, this file provides crucial details such as the VNF provider's ID, the VNF name, the creation date of the VNF, and the version of the VNF Package. The VNF package change history file, a text file documenting all modifications made to the VNF Package over time. In order to facilitate the validation of a VNF Package, testing files must be included. These files contain essential information required for conducting VNF tests, such as test descriptions. Additionally, a certificate file is required. If the manifest file is signed by the VNF provider, the VNF Package should contain a certificate file to validate the authenticity and integrity of the package [5].

The VNFD encapsulates the essential aspects that define both the deployment and operational behavior of a VNF, structured into three key components [5]: Firstly, the topology component offers a comprehensive depiction of the required nodes, typically represented as Virtual Machines (VMs), and establishes their interconnections and relationships. This element employs VNF Component (VNFC) and Virtual Deployment Units (VDUs) to encapsulate critical details such as memory allocation, disk size, and CPU specifications, thereby providing a precise blueprint for the functional environment. Secondly, the deployment aspects segment delves into a range of considerations vital for successful deployment. It encompasses deployment parameters, instantiation constraints, scaling mechanisms, and more. Additionally, the concept of deployment flavors is introduced, enabling tailored configurations based on specific deployment scenarios. For example, it might outline distinct requirements for supervisory nodes in larger-scale deployments. Lastly, the VNF Lifecycle Management (LCM) operations component furnishes a comprehensive description of management tasks and procedures throughout the lifecycle of the VNF. These operations are elucidated along with the relevant input parameters, forming a crucial guide for the effective management and orchestration of the VNF from its instantiation to termination and beyond. Table 3.3 offers a systematic exploration of the attributes within the VNFD. The table dissects essential factors: in the `VNF Profile` section, the table covers vital identifiers, product descriptions, provider information, and version details. Within

`VNF Deployment and Management`, the focus shifts to deployment nuances, constraints, resource prerequisites, lifecycle management, external interfaces, and security rules. Note that deployment flavors offer tailored configurations to match diverse network needs, optimizing performance and resources for enhanced flexibility and efficiency. Lastly, the `VNFC Information` section concentrates on VNFC components, highlighting unique identifiers, detailed descriptions, resource essentials, and internal connections within the VNFC.

Section	Subsection	Attribute(s)
VNF Profile	Unique VNFD Identifier	vnfdId
	VNF Description	vnfProductInfoDescription, vnfProductInfoName, vnfProductName
	VNF/VNFD Provider Description	vnfProvider
	Version	vnfSoftwareVersion, vnfdVersion
VNF Deployment and Management	Deployment Flavours	deploymentFlavour
	Deployment Constraints	vnfmInfo, intVirtualLinkDesc
	Required Resources	virtualComputeDesc, virtualStorageDesc,
	VNF Lifecycle	lifeCycleManagementScript, lcmOperationCoordination
	External Connection Interface	vnfExtCpd
	Security Rules	securityGroupRule
VNFC information	Unique VNFC Identifier	vdu.vduId
	VNFC Description	vdu.name, vdu.description
	Required Resources	vdu.virtualComputeDesc, vdu.virtualStorageDesc
	Internal Connection Interface	vdu.intCpd

Table 3.3: General characteristics of VNFD

The VNFD is expressed using the TOSCA simple profile NFV. Listing 3.5 showcases the VNFD for a virtual firewall. As shown, the *node_templates* include three nodes, namely the VDU1, the VL1 and the CP1. VDU1 is a virtual machine instance falling under the *tosca.nodes.nfv.vdu* type. It encompasses NFV compute capabilities, detailing CPU count, memory size, and disk dimensions, while also specifying properties like image and configuration parameters. VL1 describes a virtual link categorized as tosca.nodes.nfv.vl type. It defines attributes such as network name and vendor. CP1 corresponds to a connection point and is of type *tosca.nodes.nfv.cp*. It defines properties such as management settings and order. The requirements section establishes relationships with the VL1 and the VDU1.

Listing 3.5: Example of a VNFD file

```
1
2tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
3
4description: Virtual Firewall
```

```
5
6topology_template:
7  node_templates:
8    VDU1:
9      type: tosca.nodes.nfv.vdu
10     capabilities:
11       nfv_compute:
12         properties:
13           num_cpus: 1
14           mem_size: 256MB
15           disk_size: 0.5 GB
16     properties:
17       image: IPFire-0.4.0-x86_64-disk
18       config: |
19         param0: key1
20         param1: key2
21
22   CP1:
23     type: tosca.nodes.nfv.cp
24     properties:
25       management: true
26       order: 0
27
28     requirements:
29       - virtualLink:
30           node: VL1
31       - virtualBinding:
32           node: VDU1
33
34   VL1:
35     type: tosca.nodes.nfv.vl
36     properties:
37       network_name: nfv
38       vendor: openstack
```

ETSI [5] provides an informative use case describing the steps involving the VNF package during its transition from the VNF provider to the service provider. The steps identified for this use case are described in Figure 3.9. It involves a series of interconnected stages, each contributing to the successful integration and operation of the network function within a virtualized environment. The process begins with VNF Package building, where all necessary components, configurations, and interfaces are compiled into a package. Following this, VNF Package testing is conducted to verify its functionality, performance, and compatibility, ensuring it meets the desired specifications and requirements. Once

the testing phase is successfully completed, the next step is VNF Package validation and certification. This involves a comprehensive assessment to validate that the package aligns with industry standards, security protocols, and regulatory compliance. Certification adds a layer of trust and reliability to the package, affirming its suitability for deployment. With a certified VNF package in hand, the final stage is VNF installation. During this phase, the package is deployed within the NFVI using one or more VMs.

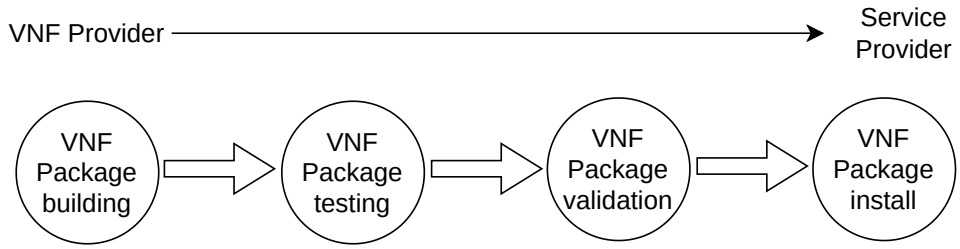


Figure 3.9: VNF Package Lifecycle [5]

Network Service Descriptor (NSD) As outlined in the ETSI specification [124], a Network Service (NS) is a composition of network functions, which can include VNFs or Physical Network Functions (PNF), all interconnected by Virtual Links (VLs). ETSI's standardization efforts extend to the Network Service Descriptor (NSD), which offers guidance for deploying and managing instances of an NS. The NSD contains or references a set of descriptors, including VNFDs, Virtual Link Descriptors (VLDs), and VNF Forwarding Graph Descriptors (VNFFGDs). A VLD provides information of a VL, including the deployment configurations available for VL instantiation and the VNFFGD references the VNFDs and VLDs for topology description [5]. Table 3.4 provides a comprehensive overview of the fundamental attributes within the NSD. The NSD Profile section delineates the essential identification and versioning attributes associated with an NSD. The `identifier` serves as unique identifiers, while the `nsdName` offers a brief description of the NSD. Additionally, the `designer` attribute encapsulates the description of the NSD provided by the designer, while the `version` indicates the specific version of the NSD. The NS management section focuses on efficiently handling the lifecycle of the NS. The `lifeCycleManagementScript` entails the script for managing the NSD's lifecycle, while the `autoScalingRule` defines rules governing its automated scaling. The `nsDf` attribute denotes the deployment flavor of the NSD. The last section highlights association. The `nestedNsdId` establishes a connection to another nested NSD, `vnfdId` references a VNFD, `pnfdId` connects to a Physical Network Function Descriptor (PNFD), and `vnfgd` relates to a Virtual Network Function Graph Descriptor

(VNFGD).

Section	Subsection	Attribute(s)
NSD Profile	Unique Identifier of NSD	nsdIdentifier
	NSD Description	nsdName
	NSD Designer Description	designer
	NS Version	version
NS Management	NSD Lifecycle	lifeCycleManagementScript, autoScalingRule
	Deployment Flavor	nsDf
References to other descriptors	Reference to NSD	nestedNsdId
	Reference to VNFD	vnfdId
	Reference to PNFD	pnfdId
	Reference to VNFGD	vnfgd

Table 3.4: General Characteristics of NSD

Listing 3.6 provides an example of an NSD. As shown, it's formulated within TOSCA Simple Profile for NFV. It leverages imports to integrate two VNFDs, denoted as VNFD1 and VNFD2. The "VNF1" node embodies a virtual network function of type *tosca.nodes.nfv.VNF1*, while "VNF2" represents another virtual network function under the *tosca.nodes.nfv.VNF2* type. To establish connectivity, "VL1" and "VL2" nodes stand for virtual links of type *tosca.nodes.nfv.VL*, with properties like network name and vendor defining their connection attributes.

Listing 3.6: Example of a NSD file

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0
imports:
  - VNFD1
  - VNFD2
topology_template:
  node_templates:
    VNF1:
      type: tosca.nodes.nfv.vnf1
      requirements:
        - virtualLink1: VL1
        - virtualLink2: VL2
    VNF2:
      type: tosca.nodes.nfv.vnf2
    VL1:
      type: tosca.nodes.nfv.vl
      properties:
        network_name: net0
        vendor: openstack
    VL2:
      type: tosca.nodes.nfv.vl
```

```
21      properties:
22        network_name: net_mgmt
23        vendor: openstack
```

3.2.6 Conclusion

The models and profiles presented in this state of the art provide means for various purposes, such as enhancing security, deploying a network component or service, and assisting in its management. Our objective is to address the gaps described in Table 3.5 which corresponds to the identified features for the Inspire-5Gplus manifest.

Responsibility Following our study, we noticed a lack of expression of commitment across the different profiles studied, thereby identifying responsible parties. Indeed, the VNFD, the NSD, and the SUIT manifest provide a field for identifying the provider, but it's not mandatory. Additionally, there is no specific field for each component lifecycle, thus delineating each one's responsibility. This criterion is therefore partially fulfilled.

Accountability No field explicitly specifies what each stakeholder needs to demonstrate. The accountability criterion is therefore not met; it's something that we really need to address with our model. This is crucial to cope with the nature of the cloud-edge-IoT environment.

Liability The liability is not expressed in any of the profiles.

Modularity The two descriptors VNFD and NSD use TOSCA, which is a modular metamodel as described in Chapter 2, which validates the modularity criterion. However, the other profiles—MUD, SUIT, and LwM2M—do not allow profile combination.

Genericity This criterion falls short due to the nature of each profile, each profile specifically describes a distinct component type. Consequently, this approach restricts their utility, as they do not accommodate the diversity or variability found across the Cloud-Edge-IoT continuum.

Features	VNFD	NSD	MUD profile & extension	SUIT manifest	LwM2M model
Responsibility	□	□	□	□	-
Accountability	-	-	-	-	-
Liability	-	-	-	-	-
Modularity	■	■	-	-	-
Genericity	-	-	-	-	-

■: the feature is supported
□: the feature is partially supported
-: the feature is not supported

Table 3.5: Compliance with Inspire-5Gplus manifest requirements 1

Following the study of prior works about the SLA, we showcased SLAs as a fundamental tool capable of establishing both liability and accountability. Through this study, we believe that SLAs will serve as a cornerstone for the first contribution. Indeed, it delineates liability by assigning clear responsibilities to each involved party. It outlines performance expectations and consequences if predefined service levels are not met, usually through penalties. Furthermore, the service provider can express what must be demonstrated, contributing to accountability. Lastly, termination clauses specify liabilities and responsibilities upon contract conclusion or termination. We identified WSLA and WS-Agreement as the most frequently used models. WS-Agreement stands out as the most flexible language in terms of contract definition. This means that users are free to define their terms as they desire. For this thesis, we used this SLA model for the criteria mentioned above.

3.3 Liability and Trust Metrics

3.3.1 Introduction

Liability and trust analysis in a multi-actor and dynamic architecture can be challenging due to the system complexity and the involvement of multiple actors. The research on liability in complex architecture as studied in this thesis is still an emerging field, and more studies are needed to fully understand and address the legal and technical challenges. Similarly, few papers address the research question of liability and trust metrics in such architecture. The literature review will focus on trust model, exploring how trust can be measured in the Cloud-Edge-IoT ecosystem, as well as the metrics of liability and accountability proposed in the scientific literature.

3.3.2 Trust Computation within the Cloud-Edge-IoT Continuum

Within the study by Govindaraj *et al.* [8], trust models in the cloud ecosystem are categorized into three groups: recommendation-based trust models, reputation-based trust model and SLA-based trust models. To calculate the metrics of liability and trust, we will rely on our responsibility model, which utilizes SLAs. Our primary interest lies in the SLA-based trust model, although we will briefly explore the others models. The recommendation-based model, termed indirect trust, relies on recommendations or experiences from others rather than direct interaction with the system. Some papers addressed the topic, for example, Singh *et al.* [125] introduced a trust assessment mechanism for cloud service providers, evaluating trust using three metrics: customer self-trust, third-party trust, and trust from friends towards the service provider. Rizvi *et al.* [126] proposed a trust model involving cloud service users, providers, and third-party auditors. This model ranks providers based on assessments conforming to Cloud Security Alliance (CSA) recommendations for cloud service registration. Reputation-based models rely on evaluating a system's trustworthiness based on their past behavior, interactions, or performance. Significant work on this topic includes Wang *et al.* in [127], where they propose a classification framework for trust and reputation systems in web services. It hinges on three crucial factors: centralization (presence of a central entity managing reputation), target (focus on individuals or resources), and scope of opinions (global or personalized, from the general population or a specific group). J.F.Borowski *et al.* [128] created a system combining reputation-based trust and agent-based safety mechanisms to prevent harmful failure ratings. It calculates trust through agent interactions,

where queries about peer status are exchanged, and unanswered responses flag a faulty agent. The overall trust rating derives from the average of these interactions. Papadakis-Vlachopapadopoulos *et al.* [129] introduces a platform for federated cloud providers that manages both SLAs and trust. The SLA service allows providers to specify performance criteria for cloud applications and assesses agreements, triggering notifications for violations. Additionally, a reputation-based trust service uses QoS and KPIs to depict provider reliability, designed for easy scaling in federated settings. Habib *et al.* [130] introduces a trust management system for cloud computing, assisting users in finding reliable service providers. It uses QoS attributes and combines reputation and recommendation techniques, employing various operators to handle trust data from multiple sources. Noor *et al.* describe in [131] CloudArmor, a Trust-as-a-Service (TaaS) framework that includes multiple functionalities. It introduces a protocol for credible feedback and user privacy, an adaptable credibility model for security, and an availability model for reliable decentralized trust management.

SLA-based trust model is a widely used method for building trust in cloud computing environments. It employs SLAs to establish and measure trust between providers and consumers. The authors in [9] claim that QoS monitoring and SLA verification is an important basis of trust management for cloud computing.

Chandrasekhar *et al.* [10] suggest a QoS monitoring technique and dynamic trust calculation method using a state-based approach to reduce network data. The trust calculation uses Markov Chain theory to identify steady, unsteady, or failure states. Valero *et al.* presents in [16] a trust framework for reliable stakeholder selection in a 5G marketplace. It includes a reputation-based model with four modules: Information gathering, Trust computation, Trust storage, and Continuous update. The Continuous update module introduces an SLA-driven reward and penalty system to adjust trust scores based on breach predictions, detections, and violations. In [132], the authors propose a trust model empowering Cloud Service Provider (CSPs) to assess trust for participation in reliable cloud federations. It relies on feedback and CSPs' SLAs, extracting Quality of Protection (QoP) attributes from SLA documents to gauge security and privacy levels. The model computes an aggregated trust value using this information.

Also, Alhamad *et al.* [133] propose a trust model for cloud services using SLA metrics. Their work defines SLA metrics for IaaS, PaaS, and SaaS, and presents a two-module architecture: the SLA agent module, which monitors SLA parameters, and the trust management module, which oversees trust

relationships using local experiences, external opinions, and SLA agent reports. Moreover, Chakraborty *et al.* [134] propose a quantitative trust model for cloud services that utilizes parameters extracted from SLA, such as CPU and memory capacity. Their model allows for assigning different degrees of importance to parameters and updates values based on interaction histories. Their framework includes systems and modules, such as the cloud consumer, SLA/other document, policy base, and trust calculator module. Chang *et al.* [135] propose a multidimensional trust model for Fog computing that considers application, peer, and auditor perspectives. Parameters like availability, response time, throughput, and security are used to assess Fog service provider trustworthiness, with adjustable weights for each perspective based on application requirements. Dimitrakos *et al.* present in [136] a trust model for assessing infrastructure providers' reliability in cloud computing. It calculates trust values based on SLA compliance, service provider ratings, and behavior, using an opinion model encompassing belief, disbelief, uncertainty, and base rate. No implementation or evaluation are provided for the models proposed in the previous papers. Moreover, the models primarily serve to compare cloud providers based on the services provided and the level of trust they instill. In contrast, the frameworks in question are more specifically tailored towards cloud computing and lack the generality present in the framework that we propose. As well the liability and trust indicators, our framework provides trends of SLA Violation Rate, which is not provided by other frameworks.

3.3.3 Liability metrics within the Cloud-Edge-IoT Continuum

Given the lack of literature regarding liability metrics within the Cloud-Edge-IoT continuum, we extended our research to related topics such as accountability. In [12], K.L Ryan *et al.* highlight the urgent need for research in cloud accountability. For that, they propose TrustCloud, a framework that addresses accountability in cloud computing via policy-based approaches. The framework includes five abstraction layers namely system layer that perform system file-centric logging, data layer that facilitates data-centric logging, workflow layer that performs audit-related data. These layers will ensure that data are logged to track how items are processed, accessed, stored, or transmitted. Additionally, these layers manage the seven phases of the cloud accountability life cycle, including planning policies, tracing, logging, safekeeping, reporting, auditing, and optimizing. The proposal lacks tangible implementation and evaluation. Furthermore, the authors do not offer a method to quantify the level of trust in a cloud provider. The accountability literature extensively covers the use

of data management tools to ensure data protection, privacy, security, and regulatory compliance. For example, Thiago Rodrigues *et al.* [137] proposed the Cloudacc framework to ensure accountability and trust in federated cloud environments. It combines cloud and blockchain technologies to create a distributed and transparent mechanism for cloud providers to record and share information about their services and operations. Also, the A4CLOUD project [11] proposes tools and models that provide users with greater control, transparency, and enforcement capabilities over the use and protection of their data in the cloud. Moreover, the authors of [138] propose a data-centric logging approach to improve accountability and security in cloud computing. Their four-stage framework includes standardizing data transaction definitions, real-time analysis for detecting security threats, and generating reports to help customers understand their data transactions. The ETSI GR NFV REL018 [139] defines principles for accountability management and presents a Quality Accountability Framework for ensuring the quality of NFV implementations and establishing accountability mechanisms. No paper has addressed the importance of providing indicators to handle the liability in the cloud architecture. In addition, previously mentioned works focus mainly on the accountability and transparency of cloud service providers in managing and protecting user data. In [140], the authors emphasize the necessity of clear guidelines for handling liability concerns amid service failures within critical cloud architecture. This involves establishing a legal framework. They advocate for a cloud architecture model that integrates resilience and multi-tenancy aspects. The authors suggest incorporating anomaly-based techniques to detect deviations in system and network behavior. Additionally, they recommend implementing monitoring and auditing tools to ensure lawful services while safeguarding privacy, and creating collaborative interfaces to enhance transparency in root-cause analysis without divulging sensitive operational details. However, their focus in this paper primarily centers on data protection issues, offering only a cloud infrastructure model without quantifiable metrics for trust and liability.

3.3.4 Monitoring and Detecting SLA Breaches

Given that our contribution introduces metrics conducting to the detection of SLA breaches. We proceed with an overview of the proposals. Several open-source SLA-oriented monitoring tools are available, each providing specific functionalities to track performance, manage alerts, and ensure compliance within SLA. García *et al.* propose CloudCompass [141] [142], an SLA-aware PaaS Cloud platform that manages resource lifecycles, extending the WS-Agreement SLA specification for Cloud

Computing. It enables Cloud providers with a versatile SLA model, accommodating higher-level metrics and flexible requirements from multiple actors. Additionally, it offers a framework for Cloud applications to dynamically correct QoS violations using cloud infrastructure elasticity. Wood *et al.* introduce Sandpiper [13] a framework that automates the monitoring, detection of hotspots, and the remapping or reconfiguring of VMs as needed. Its monitoring system aligns with our objective: to prevent and detect SLA violations by using threshold values to evaluate potential breaches.

Comuzzi *et al.* propose SLA@SOI [14], a framework that incorporates SLA-based monitoring and penalty management. This feature actively monitors agreed-upon service levels between service providers and end-users. If an SLA is breached, the framework takes measures to manage associated penalties or consequences. In [143], the authors propose the Cloud Application SLA Violation Detection architecture (CASViD), a framework that monitors and identifies breaches in application-level SLAs. It focuses specifically on resource management, scheduling, and deployment in a multi-customer Cloud environment, setting it apart from other monitoring architectures. Emeakaroha *et al.* propose in [144] a comprehensive framework called QoS-MONaaS for Quality of Service Monitoring as a Service. The framework enables the formalizaton of SLAs by defining essential performance indicators and establishing alert protocols to address SLA violations. J.Bendriss *et al.* present in [145] a framework for cognitive SLA enforcement of networking services involving VNFs and SDN controllers, using ANN. This framework is designed to efficiently manage and anticipate SLO breaches. The framework identify correlations in historical data and predict future resource usage, which helps optimize resource utilization and reduce the risk of SLA violations. To prevent violation of SLA, Haq *et al.* [146] propose a validation SLA framework, this framework enables the selection of services at the pre-SLA stage relaying on a hybrid PKI (Public Key Infrastructure) and reputation-based trust model to prevent SLA violation. Services reputation are updated after each SLA validation process.

3.3.5 Financial Exposure To Risk Metric

One of the metrics we calculate is a financial risk exposure metric. There is limited literature discussing the application of Financial Risk Analysis Techniques in conjunction with SLAs and service construction. Financial exposure to risk refers to the potential financial loss a business or individual faces due to adverse market movements, operational failures, or unexpected events. It represents the vulnerability of financial assets, investments, or operations to fluctuations in interest rates, currency

values, commodity prices, or other market variables. Antonopoulos *et al.* [15] discusses the application of financial risk analysis techniques to Grid Economics in order to ensure availability, capability, and liability in relation to financial applications. They construct their Grid SLA, with reference to WS-Agreement, by considering relative pricing of resources of different specifications and the associated risk of any items in the portfolio being unable to run or complete its task within a limited timeframe. The AssessGrid project [147] uses WS-Agreement for contract negotiation, considering a probability of failure (PoF) impacting price and penalty. The broker gathers SLAs from various providers, creating a ranked list based on price, penalty, and PoF aligned with user preferences. While aiming to optimize economic benefits, the end user still needs to compare and select SLA offers.

3.3.6 Conclusion

The frameworks proposed in the literature for computing trust metrics based on SLAs are not extensive, especially in recent publications. Most of the works date back to 2012/2015, lacking implementation and evaluation; they primarily focus on cloud services. They do not aim to offer a generic solution, as we intend and as imposed by the cloud-edge-IoT continuum environment. Additionally, the proposed works do not enable metric calculation across multiple services on a single host. Therefore, none of the works support the challenges of the cloud-edge-IoT continuum. The solution we propose can be applied as long as there is an objective and a method to measure that objective. It remains generic to both service and infrastructure.

The primary focus on accountability resides in the A4Cloud project. The tools, models, and guidelines proposed by the project heavily emphasize data protection in the cloud while enhancing transparency. While these subjects are crucial for accountability management, they differ from our contribution, which centers on metrics enabling the quantification of trust and liability.

In the state-of-the-art literature, there's a lack of metrics that enable a trust score for a service instance, a service class, and the service provider. One work, that of Valer *et al.* [16] focuses on selecting stakeholder services in a marketplace, but it remains specific to the 5G realm and doesn't offer the generalization we achieve with our contribution. Our proposal aims to provide an overall rating for a service.

Regarding the detection of SLA breaches, the tools presented primarily concentrate on the cloud, utilizing a common method of setting a threshold and checking if observations exceed it. In contrast,

our contribution aims to provide an enhanced visualization of SLA evolution. Using a two-dimensional map, we define different zones to facilitate a better interpretation of the SLAs' progression.

When it comes to the Financial Exposure To Risk metric, there appears to be a gap in the literature. Apart from Antonnoplous *et al.*'s work [15], there is no other proposal for such a metric. What sets our proposition apart is its capability to observe its evolution over time and calculate it at the application level.

Chapter 4

Contribution 1: TRAILS, Extending TOSCA NFV Profiles for Liability Management in the Cloud-Edge-IoT Continuum

Contents

4.1 Introduction **90**
4.2 Energy-aware Service-Level Agreements in 5G NFV architecture **90**
4.2.1 VNFD Energy Extension 91
4.2.2 Energy-aware Service Level Agreement template for Network communications 94
4.2.3 Exploring Use Cases: Practical Scenarios 95
4.2.4 Conclusion 97
4.3 TRAILS: Extending TOSCA NFV profiles for liability management in the Cloud-to-IoT continuum **98**
4.3.1 Introduction 98
4.3.2 The TRAILS Model 98
4.3.3 Illustrative Example 101
4.3.4 Evaluation 104
4.3.5 Discussion, Conclusion and Perspective 113

4.1 Introduction

In this chapter, we present the first contributions of the thesis. The state-of-the-art on SLAs resulted in a paper for the MSICC 2021 workshop. Section 4.2 will showcase this contribution, it is referred to as an introductory one, as it does not directly address the thesis problem statement.

Moving forward we introduce TRAILS (sTakeholder Responsibility, AccountabIlity, and Liability deScriptor), a responsibility model that captures the responsibilities of different parties in a supply chain. TRAILS extends the TOSCA NFV, by integrating existing profiles of the Cloud-Edge-IoT Continuum and providing a comprehensive description of the responsibilities, liabilities, and accountabilities of supply chain actors. The necessity for this model arises from the observation of a gap in the existing state of the art, where no such comprehensive model was found. This contribution aligns with the block FB.1 presented in the chapter 1 as it contribute to define accountability and liability relationship. This contribution was submitted and accepted at the Netsoft 2022 conference.

4.2 Energy-aware Service-Level Agreements in 5G NFV architecture

For the thesis's first contribution, we examined the potential for incorporating energy consumption into the negotiation of SLAs for a network service. In today's context, customers are increasingly concerned about the environmental impact of their activities. Consequently, they are actively seeking services that have minimal energy consumption and a low carbon footprint, particularly when some resources used to provide the service are situated on their premises. To effectively manage energy usage and demonstrate that the service is operated in an energy-efficient manner, it is essential for the service provider to include energy consumption targets in the SLAs negotiated with the customer. Furthermore, it is crucial that the VNFs listed in the service catalog provide information about their energy consumption. Currently, this parameter is absent from the catalog. As it stands, service providers are unable to commit to specific energy consumption levels because they lack access to this vital information.

This contribution can be divided into two main aspects. First, we enhance the VNFD by including energy consumption data, which assists network operators in creating energy-efficient network services. Second, we introduce an energy-aware SLA template that enables operators to make commitments regarding energy considerations, along with metrics for detecting any deviations from

these commitments. It's worth noting that existing works, like those mentioned in [148], [149], and [150], primarily focus on optimizing the placement of VNFs to minimize energy consumption in a network service composed of VNFs. In contrast, our research offers a solution to disclose the energy consumption of VNFs, allowing service providers to specify energy consumption characteristics in the SLA. Additionally, many existing works are related to cloud computing and would require adaptation to address network management concerns

While many studies have examined energy considerations in cloud computing, only a few closely align with our research. In [151], the authors introduce an energy-aware Service Level Agreement to balance energy efficiency with quality of service. [152] proposes a Green Service Level Agreement (GSLA) and an associated framework to optimize energy usage for cloud users. Laszewski *et al.* [153] propose GreenIT-SLA, an SLA template designed to enhance service eco-efficiency by integrating Green IT metrics into the SLA monitoring process. Additionally, [154] surveys Green SLAs in the IT industry, which primarily focus on energy, carbon footprint, green energy, and recycling in cloud computing environments.

4.2.1 VNFD Energy Extension

Our proposed extension for VNFD's energy considerations is founded on the guidelines outlined in [155]. It defines energy metrics and measurement methods for NFV components, including VNFs. It introduces two metric categories: energy efficiency metrics, quantified by the functional units of useful output relative to energy consumption, and resource efficiency metrics, assessed as the ratio of useful outputs to the resources consumed by the VNF. The measurement methods are tied to both power consumption and resource consumption. It outlines measurement conditions, offering recommendations for configuring the System Under Test (SUT), specifying environmental test conditions, and defining the required measurement instruments. The following formulas translate more formally what has been mentioned above :

- The VNF's energy efficiency ratio metric is defined as :

$$VNF\_EER = \frac{Usefuloutput}{Powerconsumption} \tag{4.1}$$

- The power consumption P is measured as follows :

$$P = P_{load} - P_{idle} \quad (Watt) \tag{4.2}$$

Where P_{load} is the power consumption of NFVI platform including the deployed VNF, and P_{idle} is the power consumption of NFVI platform without any VNF deployment.

- The VNF's resource efficiency ratio metric is defined as:

$$VNF\_RER = \frac{Usefuloutput}{Resourceconsumption} \tag{4.3}$$

.

- The resource consumption R is measured as follows :

$$R = R_{load} - R_{idle} \tag{4.4}$$

Where R_{load} is the resource consumption of NFVI platform including the deployed VNF, and R_{idle} is the resource consumption of NFVI platform without any VNF deployment.

In addition, we propose the following metric in order to compute energy consumption :

$$\mathrm{E} = \mathrm{P} \times \mathrm{T} \quad (Joule) \tag{4.5}$$

Where P is the power consumption and T is the time.

The energy consumption is measured in Joule or KWh and the consumed resources refer to the virtual resources allocated to the VNF, which can be CPU, memory, storage, and network. The useful output of the VNF depends on the types of VNFs, that can be throughput (e.g., bps, pps) for data plane VNF, or capacity (e.g., subscribers, sessions) for control plane VNF. We introduce these metrics as an extension to the VNFD using a methodology illustrated in Figure 4.1. Each VNF runs on an NFVI and is scaled to provide different levels of service capacity by scaling one or more of its VNFCs. This capacity level depends on the provider of the VNF. Hence, the measurement tests must be performed with reference to an NFVI and with a given capacity level. This extension of the VNFD dedicated to energy aspects allows the operator to obtain information about the consumption features of the VNF as displayed by the manufacturer. To increase confidence in this information, the operator may rely on consolidation mechanisms based on experiences with this particular VNF or other VNFs

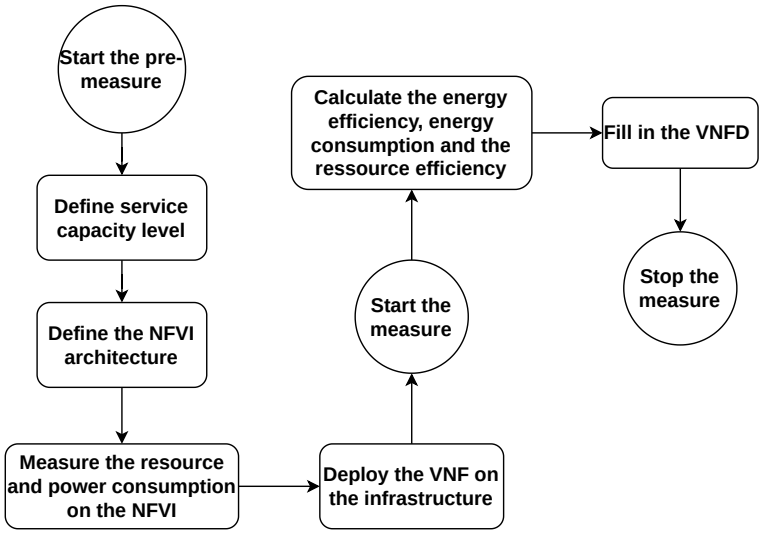


Figure 4.1: VNFD Extension Methodology

of the same type from the same manufacturer, tests performed on the VNF or reputation from third parties experiences and opinions.

Expanding the VNFD involves adhering to the guidelines outlined in [5]. According to these specifications, we must furnish the following details for each attribute: the attribute name, whether it's mandatory (M), optional (O), or conditional (C), the range of occurrences (e.g., 1, 0...N), the data type of the attribute values, and a concise attribute description. The manufacturer includes the NFVI reference, which serves as a description of the NFV infrastructure used for conducting measurement tests. Details regarding this attribute are provided in Table 4.1. The manufacturer is also required to specify the capacity level utilized in these measurement tests, as demonstrated in Table 4.2. Attributes associated with energy metrics, including Energy Consumption, Energy Efficiency, Resource Efficiency, and measurement duration, are presented in Table 4.3.

Attribute	Description	
NFVI reference	Qualifier	M
	Cardinality	1...N
	Content	String
	Description	Textual description of the NFV infrastructure

Table 4.1: NFVI reference attribute

Attribute	Description	
Capacity level	Qualifier	M
	Cardinality	1...N
	Content	String
	Description	Textual description of the capacity level of the VNF set up during the measure

Table 4.2: Capacity level attribute

Attributes	Descriptions	
Time	Qualifier	M
	Cardinality	1...N
	Content	Time
	Description	Duration of the measures expressed in hour
Energy consumption	Qualifier	M
	Cardinality	1...N
	Content	Float
	Description	The energy consumption (in Joule or kW/h) of the VNF recorded after the measurements
Energy efficiency	Qualifier	M
	Cardinality	1...N
	Content	Float
	Description	The energy efficiency of the VNF recorded after the measurements
Resource efficiency	Qualifier	M
	Cardinality	1...N
	Content	Float
	Description	The resource efficiency of the VNF recorded after the measurements

Table 4.3: The Energy attributes

4.2.2 Energy-aware Service Level Agreement template for Network communications

The template that we propose is an extended version of the network communication SLA [93]. We propose to add two fields. The first field is the energy aware SLA ID which is a unique identifier to differentiate between contracts. The second field is the energy aware offer, which will make possible to express the customer's demands in relation to energy consumption. From that, the customer can accept, renegotiate or reject offers related to energy. The metrics we propose to detect any deviation from the commitment are related to the energy attributes added to the VNFD, they are listed in Table 4.4. These parameters can be measured by the service provider or by the client himself. For

this, he can rely on a subcontractor assessing the measurements.

Metric name	Description	Unit
Total energy consumption	Total of energy consumed by the VNFs provided by the operator	Joule or KWH^{-1}
Total energy efficiency	Total energy efficiency of VNF provided by the operator	Mbps/Joule
Total resource efficiency	Total resource efficiency of VNF provided by the operator	Mbps/MHz

Table 4.4: Energy-aware SLA metrics

4.2.3 Exploring Use Cases: Practical Scenarios

We propose a practical use case to demonstrate the implementation of the energy-aware SLA and the VNFD extension. We consider a hospital center that aims to offer a remote monitoring service to its patients' rooms. The objective is to collect and securely store data in a private cloud for future analysis and processing. To deliver this comprehensive service, the hospital has acquired a dedicated slice from an infrastructure provider (slice provider). The hospital's slice needs to be enhanced with components tailored to its healthcare operations, while also utilizing network components for efficient and high-speed communication. The infrastructure operator supplies the network components, exclusively using VNFs. The hospital center is ecologically aware and practices a serious environmental policy. In addition, it needs to comply with its annual energy budget. Hence, it is looking for a commitment from the slice provider that includes the energy consumption of the components being supplied and hosted in his premises. Therefore, the operator must use the energy aware SLA proposed. To be able to respect his commitment, the operator has to be aware of the energy consumption of the VNF that he purchased from a manufacturer. Thus, the manufacturer must add energy information into the VNFD following the methodology proposed. In support of the service, the operator furnishes four network components to the hospital center. These components include a security function vFirewalls, a traffic analysis function vDPI, an edge encryption function vEncryption, and a routing function with vGateways. These VNFs are supplied to the operator by a manufacturer. The monitoring system, on the other hand, is provided by the hospital center. Within this setup, two capacity levels are defined: a minimum capacity level, where the number of VNFCs is set to one, and a maximum capacity level,

where the number of VNFCs is increased to three.

The hospital center's objective is to provide 500 patient rooms with a target throughput of 2 Mbit/s per room. The hospital center requires a commitment regarding the energy consumption of the network components, stipulating that their combined energy consumption must not exceed 998.4 KWh per year. Note that energy and resource efficiency are not considered in this context. Following negotiations between the operator and the hospital center, they establish an energy-aware SLA with ID "1." Both the hospital center and the operator are listed as signatory parties. The SLA terms include continuous service availability (24/7), a minimum data rate of 1 Gbit/s, and a resolution time of less than one hour for any issues. Energy consumption, as previously defined, is integrated into the SLA terms. In cases of non-compliance, the operator must pay the hospital center a penalty of $10,000.

The manufacturer has provided the VNF and therefore the VNFD to the operator. In order to add the energy consumption information into the VNFD, the manufacturer performed various measurement tests, the results are summarized in the table 4.5. The duration of the test is one hour. The capacity level corresponds to the throughput that each VNF can offer. The minimum capacity level is equal to 1 Gbit/s while the maximum is 2 Gbit/s.

VNF	Capacity level	Power consumption (watts)	Energy consumption (kwh)
vFirewall	Min	18	0.018
	Max	40	0.04
vDPI	Min	20	0.02
	Max	50	0.05
vEncryption	Min	39	0.039
	Max	90	0.09
vGateway	Min	9	0.009
	Max	20	0.020

Table 4.5: Results of measurement tests

Multiple combinations of VNF capacity levels can fulfill the requirements for data rate and energy consumption. For instance, opting for the minimum capacity level for all VNFs can satisfy the demands. This is supported by the following calculation:

$$24 \times 365 \times (0.0180 + 0.02 + 0.039 + 0.009) = 753.36kWh$$

This use case demonstrates that, with the help of the VNFD energy extension, the operator is able to reference the right VNF in order to fulfill the commitment. It's includes energy demand thanks to the energy aware SLA.

4.2.4 Conclusion

In summary, in this section, we have presented our initial thesis contribution, which comprises two key elements. Firstly, we expanded the VNFD to incorporate energy-related data, encompassing details on energy consumption, energy efficiency, and resource efficiency. Secondly, we introduced an energy-aware SLA template to enable the operator to make commitments regarding energy-related aspects. The primary aim is to empower the operator to select the most suitable VNFs based on their energy consumption, ensuring compliance with commitments made to consumers through energy-aware SLAs. These two fundamental concepts have been applied and examplified within a real-world use case. The next step could involve suggesting an enhancement to the MANO orchestrator to consider energy information and, in turn, optimize the placement of VNFs to reduce network service energy consumption.

4.3 TRAILS: Extending TOSCA NFV profiles for liability management in the Cloud-to-IoT continuum

4.3.1 Introduction

As seen in the literature review, there is currently no model available for describing the responsibility, accountability, and liability within a multi-actor, multi-domain service involving different legal entities and the potential for multiple levels of delegation. Furthermore, Sharif *et al.* [156], Pan *et al.* [157], and Atzori *et al.* [158] advocate for the necessity of achieving consistent service management across cloud, IoT, and NFV, requiring combining existing descriptors.

We introduce TRAILS (sTakeholder Responsibility, AccountabIlity, and Liability DeScriptor) to enhance responsibility management within the Cloud-Edge-IoT continuum. TRAILS extends the TOSCA NFV profiles, addressing the need to unify existing profiles in the Cloud-IoT-Edge ecosystem while incorporating a description of supply chain responsibilities, accountability, and liability. This model aims to achieve uniform and liability-aware service management involving IoT devices, fog, edge, and cloud nodes. The TRAILS archive adheres to the CSAR format, widely adopted by many cloud service providers, tools, and communities. The immediate implication of such a Model is that it empowers service providers to discern the commitments made by various stakeholders during the creation of a service that spans from the cloud to IoT, involving multiple legal entities and participants, and potentially featuring various levels of delegation. This model aligns with FB.1 and serves as the foundational component for liability-aware management.

4.3.2 The TRAILS Model

The model is presented through the TRAILS metamodel and its associated grammar (in Appendix A). A complete example is presented in section 4.3.3

We chose to extend TOSCA because it is a modelling language for defining portable deployment and automated management of services which is already commonly used to manage VNFs, NSs [159] and even IoT devices[160] [161]. It is also modular and enables to compose multiple components, thus achieving Inspire-5Gplus modularity requirement.

ETSI specifies an NFV specific data model using TOSCA metamodel. TRAILS extends this model to include responsibility, accountability, and liability (Figure 4.2 block B). For this, we

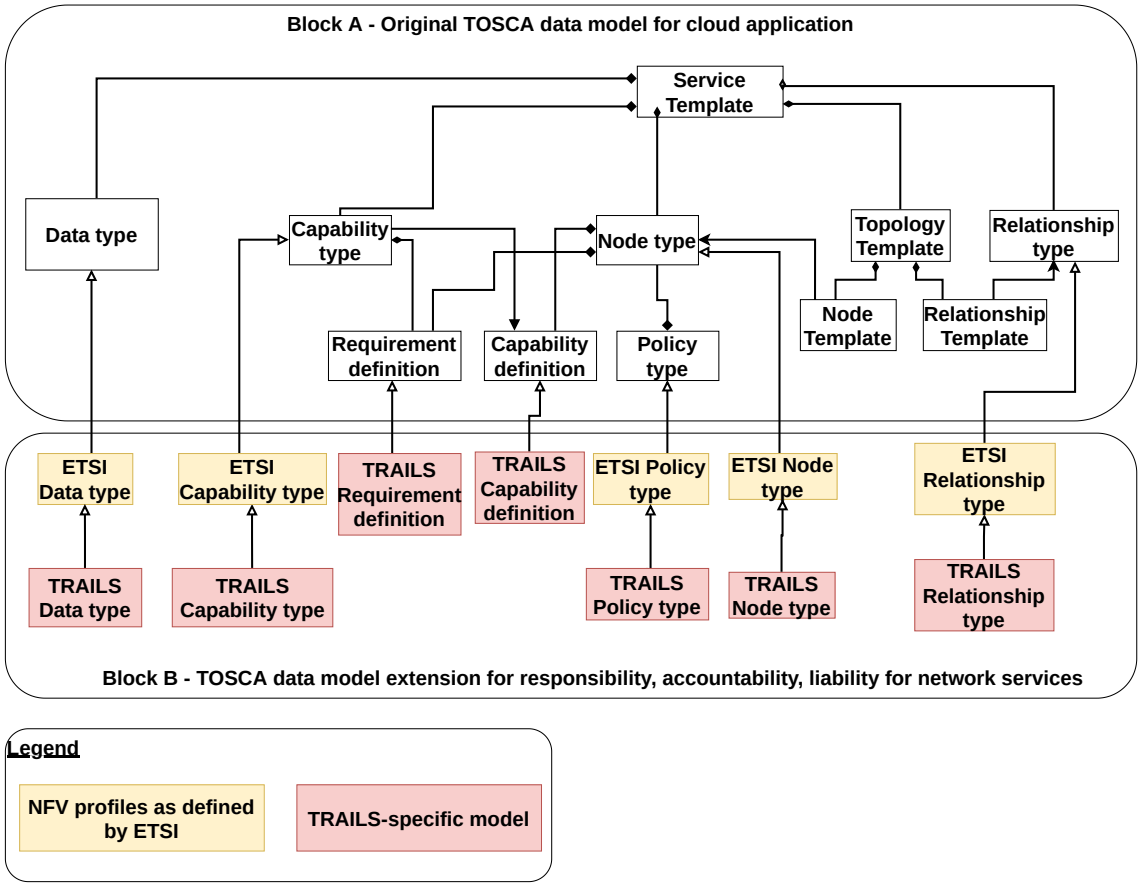


Figure 4.2: Extension of the TOSCA NFV metamodel

introduce multiple elements. First, we updated `TRAILS Data type` to include the semantics required to describe a MUD profile, SUIT manifest or OpenAPI file. Second, `TRAILS Capability type`, which describes capabilities related for example to security service. Third, `TRAILS policy type` describes the operation limitation which is a restriction imposed by an administrator before referencing the component. Fourth, `TRAILS Requirement definition`, which describes services requirements, for example security requirements provided from the extended MUD profile. Fifth, *TRAILS Relationship type*, which binds two TRAILS's nodes through a relationship and finally `TRAILS node`. To build a TRAILS CSAR archive, three new directories are required. The directory `Files` includes profiles and descriptors that can be referenced in the TRAILS data structure, which facilitates the reuse of existing profiles. The directory `Certificates` contains all authors' certificates and `Signature` that includes file's signature. Finally, the file `Manifest.mf` file which lists all the files in the archive, the certificate of the LeadAuthor and the CSAR's signature.

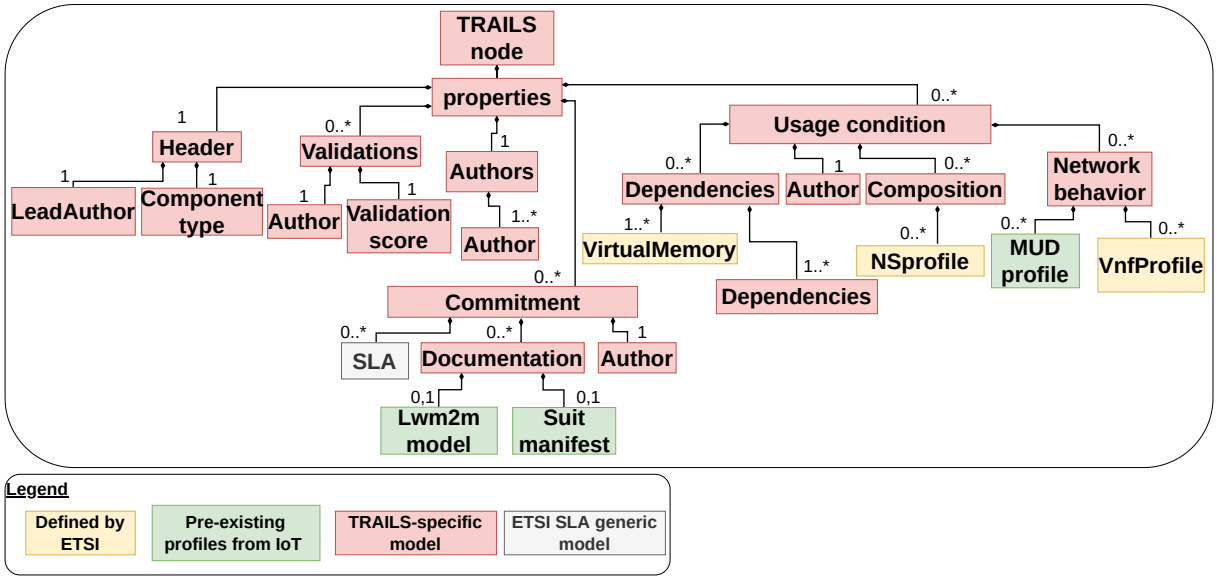


Figure 4.3: High-level structure of a TRAILS node

Following Djikstra's separation of concern concept [162], we designed TRAILS node data structure, depicted in Figure 4.3 so that each type of TRAILS node property describes a specific aspect of the component. The `header` provides an overview of the component or service by identifying its type, model and the entity which bears overall responsibility, the `LeadAuthor.Validation` indicates when the component was validated, by whom, the scope and the outcome of the validation. The `Authors` property lists all stakeholders of the component. The property `Commitment` describes the features promised by a given stakeholder, such as the SLA. and the attributes of the service described in the `Documentation`. The property `Usage condition` defines which conditions should be fulfilled to benefit at best of the component's features, such as the hardware and software dependencies, the way subservices should be combined or the component's expected network behavior. Together, `Commitment` and `Usage condition` describe complementary aspects of liability.

Liability is ensured by the fact that properties are signed by their author or responsible party using a public/private key pair managed through a Public Key Infrastructure (PKI). We distinguish authors which take responsibility of a specific property and `LeadAuthors` which integrate multiple components and properties provided by other actors. As such, authors only sign the properties that it commits to, whereas `LeadAuthors` sign all the properties in the scope of the integration it performed. To achieve this, we separate claims and properties in files that authors can sign individually. Then we

regroup them in a CSAR archive that is signed by the relevant `LeadAuthor`.

The separation of concern is also demonstrated by the fact that TRAILS model can be used to study a network component under the angle of its topology or its responsibility chains. The topology view is a directed graph where each component is a node and each link describes a connectivity link between two nodes. The responsibility view is a directed graph with a root (the final `LeadAuthor` which proposes the modeled service). Each vertex represents a couple of an Author and a Claim. Each directed edge represents a responsibility of an actor towards another one. `Commitments` are represented by an edge from a supplier towards its customer, whereas `Usage conditions` are represented by an edge from a customer to its supplier.

4.3.3 Illustrative Example

This section provides an illustrative example of TRAILS using YAML syntax. This example describe a streaming service hosted in the MEC and provided by Orange. The streaming service comprises three sub-services: the infrastructure responsible for hosting the service and provided by `Orange`, the streaming service that generates the video stream provided by `StreamInc`, the dashboard service that takes the video stream as input and displays it, provided by `Dash`, and the orchestration service, which orchestrates the various subcomponents, provided by `Orange`.

As stated in the background, TOSCA introduces the concept of substitution mapping to describe subsystems. In Listing 4.1, we describe the `MEC Streaming Service` node. In the header, we find the identity of the lead author `Orange`, who is responsible for the entire service. In the `authors` section, we list the authors involved in the service. In the `commitments` section, we have a list of SLOs to which Orange commits to the client, such as Mean Packet Loss Ratio, Mean Initial Time for Critical Mode, and Mean Ratio of Time Functions that are Not Isolated In Critical Mode. WS-agreement is the SLA model used for this service. The service can provide a virtual link to an external service (specified in the capabilities).

Listing 4.1: TRAILS example - The Streaming Service 1

```
1topology_template:
2  substitution_mappings:
3    node_type: tosca.nodes.nfv.TRAILS.MEC-streaming-service
4    properties:
5        header:
```

```
6           component_type: Network Service
7           lead_author:
8             country: France
9             name: Orange
10            role: MEC Service Provider
11          model: MEC
12          system_info: MEC
13          title: MEC Streaming Service
14          url: https://www.orange.fr/MEC-Streaming-Service
15          version: '1'
16        capabilities:
17          external_virtual_link:
18            properties:
19              protocol: IPv4
20
21          authors:
22          - country: France
23            name: Orange
24            role: Service provider
25          - country: France
26            name: StreamInc
27            role: Component Provider
28          - country: France
29            name: Dash
30            role: Component provider
31          commitments:
32          - author:
33              country: France
34              name: Orange
35              role: MEC Service Provider
36            sla:
37            - sla_model: WS-Agreement
38              sla_name: MEC_SLA
39              slo:
40              - slo_max_value: 0.1
41                slo_min_value: 0.0
42                slo_name: 'Mean Packet Loss Ratio'
43                slo_type: MPLR_MEC_O
44              - slo_max_value: 15.0
45                slo_min_value: 10.0
46                slo_name: 'Mean Initial Time for Critical Mode'
47                slo_type: MITCM_MEC_O
48                slo_unit: sec
49              - slo_max_value: 5.0
50                slo_min_value: 0.0
```

```
51              slo_name: 'Mean Time To Detect Tampering or incorrect location of
                    Function'
52              slo_type: MTTD_TILF_MEC_0
53              slo_unit: sec
54            - slo_max_value: 0.1
55              slo_min_value: 0.0
56              slo_name: 'Mean Ratio of Time Functions are Not isolated In Critical
                    mode'
57              slo_type: MRT_FNIC_MEC_0
58            - slo_max_value: 20.0
59              slo_min_value: 0.0
60              slo_name: 'Mean Observation Report Request Response Time'
61              slo_type: MORRT_MEC_0
62              slo_unit: sec
```

Following this, each sub-service is detailed by its provider. In the follow, we will exclusively present the streaming sub-service (Listing 4.2), as the remaining sub-services adhere to a similar structure. This service is supplied by `StreamInc`, as delineated in the header. Notably, this service does not include SLOs. It does, however, have a prerequisite for a containerization and orchestration service (indicated in the requirement) and it provides a virtual link to another service (indicated in the capabilities).

Listing 4.2: TRAILS example - The Streaming Service 2

```
1streaming-service:
2   type: tosca.nodes.nfv.TRAILS.streaming-service
3   properties:
4     capabilities:
5       virtual_link:
6         properties:
7           protocol: IPV4
8       authors:
9       - country: France
10        name: StreamInc
11        role: Component Provider
12      header:
13        component_type: Software
14        lead_author:
15          country: France
16          name: StreamInc
17          role: Component Provider
18        model: Streaming Service
19        system_info: Streaming Service
```

```
20          title: Streaming Service
21          url: https://www.StreamInc.fr/streaming-service
22          version: '1'
23      requirements:
24      - container:
25      - orchestrator:
```

Figure 4.4 illustrates the structure of the TRAILS archive for the streaming service. It includes the TRAILSs for the three sub-services. The `File` directory host the service's SLA. The `Certificates` and `publicKey` directories contain the certificates and public keys of the stakeholders Orange, StreamInc, and Dash. `Definitions` hold the TRAILS grammar files, as well as the `Instance File`, which serves as the main file containing the service description and references to the TRAILS of the sub-services.

```
TRAILS-MEC-Streaming-Service/
├── Certificates/ ..............................Stakeholder's certificate folder.
├── Definitions/ ...............................TRAILS's Grammar and instance
│                                               file.
├── Files/ .....................................Referenced files directory.
├── PublicKey/ .................................Stakeholder's publicKey folder.
├── Signatures/ ................................Signature folder.
├── TRAILS-Dashboard.zip/ ......................Dashbord service TRAILS.
├── TRAILS-MEC-Infrastructure.zip/ .............MEC Infrastructure TRAILS.
└── TRAILS-Orchestration.zip/ ..................Orchestrator service TRAILS.
```

Figure 4.4: TRAILS MEC Streaming Service Directory Tree

4.3.4 Evaluation

To evaluate our proposal, we show that TRAILS complies with Inspire-5Gplus manifest requirements. We illustrate how TRAILS and LASM, the Liability-Aware Security Manager specifically developed for evaluation purposes, can be used to take into account responsibility and accountability and liability in the management of a service in the Cloud-Edge-IoT continuum. This is examplified through a use case. Afterwards, we evaluate its semantics by describing how we used TRAILS to model existing network components and services. We also evaluate the impact of using TRAILS on scalability by evaluating its impact on convergence and stability.

Table 4.6 provides a comparison between TRAILS and the profiles studied in the state-of-the-art, considering the criteria outlined in the Inspire-5Gplus manifest. Our analysis of the current state-of-

the-art reveals that a majority of existing models fall short in meeting all the specified requirements, including those related to responsibility, accountability, liability, modularity, and generality.

Features	VNFD	NSD	MUD profile & extension	SUIT manifest	LwM2M model	TRAILS
Responsibility	□	□	□	□	-	■
Accountability	-	-	-	-	-	■
Liability	-	-	-	-	-	■
Modularity	■	■	-	-	-	■
Genericity	-	-	-	-	-	■

■: the feature is supported
□: the feature is partially supported
-: the feature is not supported

Table 4.6: Compliance with Inspire-5Gplus manifest requirements 2

TRAILS fulfills the genericity criteria because it can be used for IoT devices, VNFs and NSs and leverages commonly used profiles that are relevant for each domain such as SUIT, MUD profiles, MUD extensions, VNF and NS descriptors. TRAILS traces the responsibilities of each actor involved in the supply chain. Several stakeholders involved in the creation of one service can define their responsibilities independently of each other. Supply chain providers can define responsibilities for themselves and their users. If users accept to use the service described by TRAILS, they can define responsibilities for themselves and include it as a new composite service. In this case, a composite TRAILS can be generated. As such, TRAILS fulfills at the same time the responsibility and modularity criteria. It should be noted that TRAILS also provides traceability of services. Liability is expressed in TRAILS by SLAs, given their penalties and triggers. The signature of commitments, as well as the usage conditions, contribute to achieving the liability criteria. At the same time, TRAILS ensures accountability by including SLA in the properties committed by each actor. In particular, the SLI, which provides evidence that results have or have not been achieved before ensuring full accountability.

TRAILS complies with the Inspire-5Gplus manifest lifecycle described in Chapter 3 Section 3.2.2 Figure 3.1. During the manufacturing phase, the TRAILS profiles of multiple building blocks can be aggregated to form the profile of a new service. During the testing phase, validators can describe in TRAILS additional features, controls, or usage conditions. During the referencing phase, a service operator can add operation limitations to comply with internal policies before adding the component to its catalog. All these characteristics can then be to perform liability-aware service management.

The TRAILS outlined in this contribution does not establish standardization for the data that service providers can incorporate into the TRAILS profile during the deployment and operational phases.

TRAILS and LASM For this contribution, we focus on the LRS module and the ontology. It is responsible for the management of the network component and service catalog. This module includes an ontology that offers tools for reasoning about responsibility, accountability, and liability aspects associated with network components. The LRS handles synchronization between the database and the ontology, making it the sole entity that exposes an interface to external services. Implementation-wise, the LRS is realized as REST web services using the Django Rest framework, while the ontology is developed with owlready2, a Python module for ontology-oriented programming. We used SWRL and SQWRL presented in the chapter 2 to express respectively referencing policies and queries on the ontology content. LRS centralizes all external requests and queries. For example, when the administrator adds a new component, LRS first validates the compliance of the profile to the TRAILS model by verifying the directory pattern, signatures, topology and syntax. Then, it requests the ontology to evaluate the associated TRAILS profile with regard to a referencing policy. LRS stores TRAILS profiles in a database and associates them to a status, either ”not evaluated”, ”Accepted” or ”Rejected”.

Use case description A Service Provider (SP) deploys a service on an infrastructure spanning from the Cloud to an IoT campus and managed by a Slice Provider (SLP). SLP subcontracts the management and monitoring of SP’s IoT campus to the SubContractor (SC). Under normal conditions, SLP routes the packets collected from SP’s devices in the IoT campus to SP’s Cloud Delivery Network (CDN) application. SLP operates SP’s slice with a basic assurance level where he commits for example to ensure a low loss of packets and an optimized level of energy consumption. In case an anomaly in the IoT devices is monitored, the contract between SP and SLP stipulates that SP shall put in place a video streaming service with a level of assurance high (e.g. providing proof of transit by specific nodes, high level of availability of the video streaming solution, guaranteed end-to-end isolation of the video streaming feed) to control and confirm the potential threat.

TRAILS and LASM assist SLP’s administrator at three different stages, respectively the referencing of a network component, the component selection for orchestration or root cause analysis. During the referencing stage of a network component, SLP can reference subcontracting solutions and ensure beforehand that they comply with its cybersecurity policies. In some cases, SLP cybersecurity policies

will impose operational measures. If components are not compliant, SLP may decide to renegotiate a contract with the SC. The TRAILS profile of SC's IoT monitoring service can then be included in the contract between SLP and SC. Similarly, the TRAILS profiles of SLP's base service and video streaming service can be included in the contract between SP and SLP. Listing 4.3 illustrates three rules from the operator's policy. The first rule indicates that any TRAILS for a component with a validation score "high" will be referenced. The second rule defines a restriction about the energy consumption, formulated thanks to the energy-aware SLA. It formally reflects the following statement: any network component that has as an energy consumption above 0.0018kw/h has the status "Rejected". In the last rule, the administrator assigns a scaling policy to a specific VNF model for which cybersecurity tests showed the need of scaling up resources such as CPU, RAM, energy. The scaling policy is based on the Anomaly Detection System (ADS) designed by *Lazri et al.* [163]. The system identifies the behaviors of a VNF before it leads to an SLA violation, which would enable to adopt proactive measures before a violation actually happens. This third rule will modify TRAILS by adding a new policy associated with the defined operation limitation, as shown in Listing 4.4.

Listing 4.3: Operator's security policy

```
1
2  - swrl_rule 1:
3       name: R-High level of assurance
4       src :"TRAILS(?t), validation(?v)
5            , validation_score(?v,'high')
6            , has_validation(?t,?v)
7            -> value_Status(?t,'Accepted')"
8   - swrl_rule 2:
9       name: R-Restrictions on energy consumption
10      src :"TRAILS(?t), SLA(?s)
11           , has_slo_type(?s,'energy')
12           , has_slo_value(?s,?x)
13           , lessThan(?x,0.018)
14           , has_sla(?t,?s)
15           -> value_Status(?t,'Accepted')"
16 - swrl_rule 3:
17      name: OL-Scaling policy
18      src: "TRAILS(?t)
19           , model(?m,'VSRX-Juniper')
20           , has_model(?t,?m)
21           -> value_Status(?t, 'Accepted')
22           , policy(?p)
23           , action(?a,'scaling_policy')
```

```
24              , has_action(?p,?a)
25              , has_policy(?t,?p)"
```

Listing 4.4: TRAILS's operation limitation policy

```
1
2OperationLimitationPolicy:
3   Description:
4   ...
5   ...
6   Trigger:
7    Event:
8    ...
9    Condition:
10   ..
11   Action:
12   patch:
13    description:
14    implementation: /scripts/patch.sh
15   scaling:
16    description:
17    implementation: /scripts/scaling_policy.sh
18   configure:
19    description:
20    implementation: /scripts/black_list.sh
21   ...
22   ...
```

With LASM and TRAILS, SLP can select the components with the right characteristics to create services that comply with the contract binding SLP and SP. In our use case, SLP selects, using the query shown in Listing 4.5, an IoT-camera with a high level of assurance, an SLA with an availability objective of 99%, and an SLI availability metric measured by the SC.

Listing 4.5: Component selection query

```
1   - sqwrl_rule 1:
2     name: Component selection
3     src: "TRAILS(?t)
4          , validation(?v)
5          , SLA(?s)
6          , validation_score(?v, 'high')
7          , has_validation(?t,?v)
8          , model(?m, 'IoT-camera')
9          , has_model(?t,model)
```

```
10          , has_sli_type(?s, 'Availability metric')
11          , has_slo_type(?s, 'Availability')
12          , has_slo_value(?t,99)
13          , has_sla(?t,?s)
14          -> sqwrl:select(?t)"
```

LRS and TRAILS can complement a Root Cause Analysis (RCA) Service, which identifies the most probable cause of an issue by estimating the liabilities with the help of TRAILS. LRS is not intended to impose automated penalties, but to provide estimations for potential negotiations carried out by SLP's jurists. For this, SLP can query the ontology searching for a component, feature, responsible party involved in the issue. For example, the query represented in Listing 4.6 search whether there is an actor which commits on the throughput.

Listing 4.6: Author identification query

```
1
2   - sqwrl_rule 2:
3     name: Author identification
4     src: "TRAILS(?t), author(?a)
5         , propertyDescription(?p)
6         , features(?f,'debit')
7         , has_features(?p,?f)
8         , has_propertyDescription(?t,?p)
9         -> sqwrl:select(?t,?a)"
```

Semantic To validate the semantic, we modeled a cellular blood pressure monitor IoT device from SmartMeter, an IoT Management Service (IMS) provided by Amazon Web Service (AWS), a Content Delivery Network Service (CNDS) provided by IBM and a Virtual Network Edge Service (VNES) provided by Equinix. Based on iBloodPressure's user manual [164], we filled in the TRAILS `Header` with the full name of the device, the model, and a description of the device which are indicated in the section `Introduction` of the manual. In TRAILS `validation` field, we list standards with which the device complies as stated in the section `Complied Standards List` of the manual. As usage condition, we referenced in the field `Network behavior` the MUD file of the device generated by [165]. Based on AWS user and developer guide for the IMS [166], we retrieve general information to fill the TRAILS `Header` and listed in the section `validation` the complied standards indicated in the AWS IoT services and compliance such as International Standards Organization 27001 (ISO). The developer guide provides an OpenAPI file, we referenced it in the field `Documentation`. We then referenced the

terms of the SLA indicated in [166] in the section `Commitment`. Similarly, we built a TRAILS for a CDN based on the information provided by IBM in [167]. We found the general information for `Header` under the section `About Content Delivery Networks`. We then listed the fact that IBM CND is PCI DSS compliant in the TRAILS `Validation` section. we filled in the TRAILS `commitment` with the OpenAPI file provided in `CDN API reference` section. Then, we built an example of TRAILS for an NS basing ourselves on the offer proposed by Equinix of a virtual network service provider [168]. We referenced Equinix `OpenAPI` file and an SLA file written with WS-Agreement in TRAILS `Commitment` section. The NSD of the service is referenced in the `Usage condition` field to define usage condition and more particularly the protocol needed for the transport of the packets. For each example, there are at minimum two actors listed in the TRAILS, a validator, and the Service Provider. For each author, we computed the signature of the claims to which it commits. Then, we generated a CSAR archive signed by each `LeadAuthor`. Finally, we composed all the services to build a new offer which corresponds to the use case described above. Figures 4.5 and 4.6 show respectively the topology of the composed service and its corresponding responsibilities share. In comparison with TOSCA NFV profiles, TRAILS brings extra values such as the `Security Service` requirement and capability, which binds the `Virtual Gateway Service` and the `Virtual Firewall Service` through security relationship, the generic capacity that allows describing all the services provided by the SP using a unique model and the ability to highlight the responsibilities of each actor involved in the supply chain. In terms of memory size, the TOSCA NFV profiles of NS service reaches 36 bytes compared to 57 bytes for the TRAILS profiles, which represents an increase of 58.33%.

As described in the use case, our target implementation requires the RCA module or the MANO to query the LRS in order to get a list of components which comply with specific criteria. So we expect that our impact on scalability will mostly correspond to the overhead required to perform a query. To quantify the impact of TRAILS on scalability, we break down this property into convergence, the time required to find a solution, and stability, how well the system performs when it is confronted to a large amount of data. All the experiments were performed five times on an Intel® Xeon® W-2133 Processor with 32 GBytes of available RAM, and the results presented below corresponds to the average times measured over the 5 experiments.

We measure the impact of using TRAILS rather than TOSCA NFV by comparing the time required to query ontology-1, an ontology compatible with TOSCA NFV, and ontology-2, an ontology

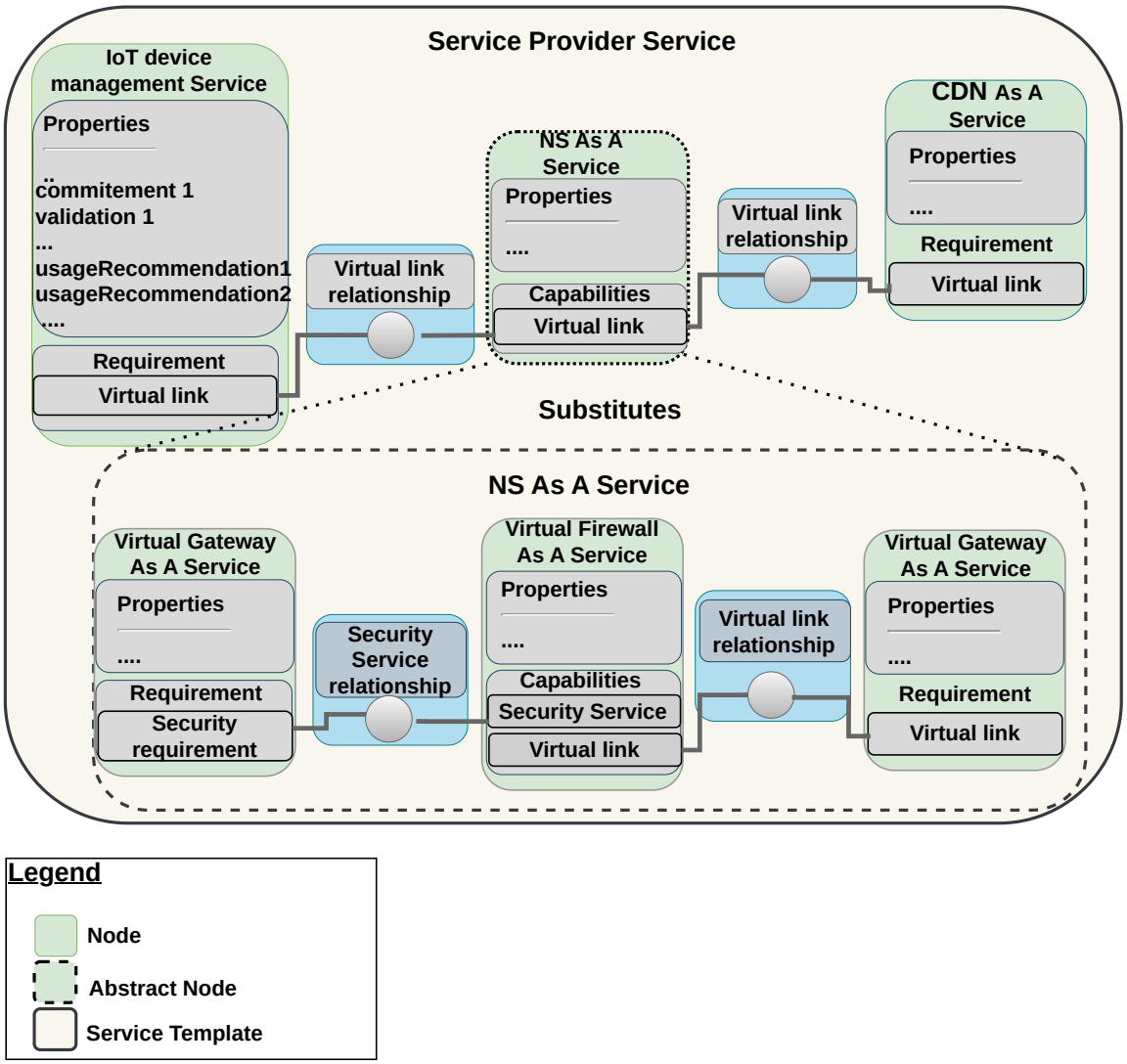


Figure 4.5: TRAILS's topologic view

compatible with TRAILS. Each of them were populated with two individuals. Ontology-1 contains two TOSCA NFV files, which describe a VNF and an NS. Ontology-2 contains two TRAILS that describes a VNF and a TRAILS that describes an NS. The required time to query ontology-1 to retrieve the VNF is 0.18 seconds and NS is 0.67 seconds, whereas the same queries took respectively 0.21 seconds and 1.23 seconds o LOS-2. This represents an increase of 17% for the VNF and 84% for the NS.

Stability To evaluate stability, we measure the evolution of the computation time to respond to a request which has a solution and a request without a solution, depending on the size of the ontology. For this purpose, we progressively populated the ontology with clones of the TRAILS described in the

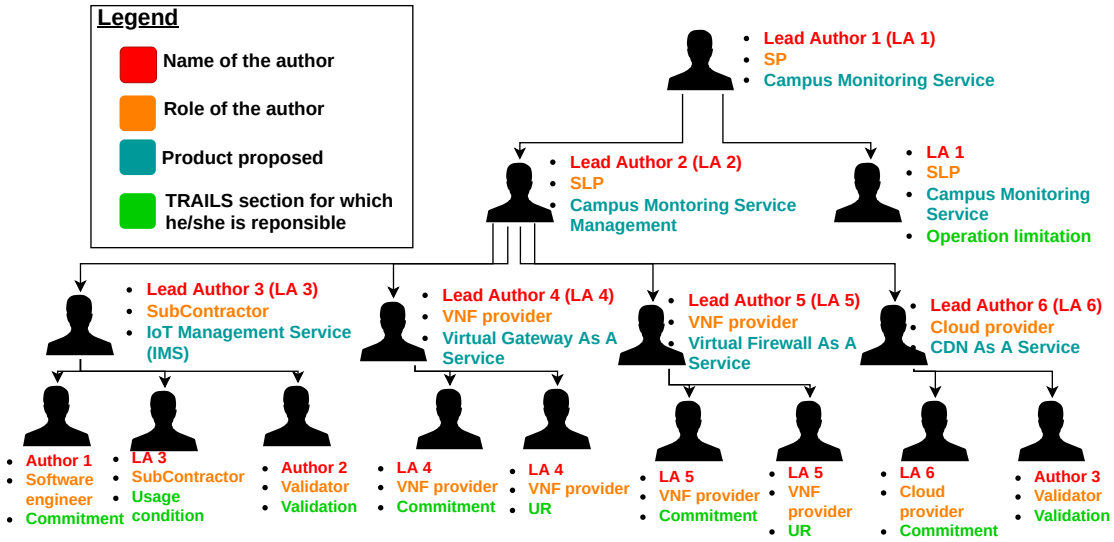


Figure 4.6: TRAILS's responsibility view

use case that we modified so that the ontology considers they are unique individuals. At each step, we added 100 TRAILS until we reached 1000 TRAILS (54,18MB) since MANOs can have around this number of components in their catalog. The results displayed in Figure 4.7 suggest that the time necessary to compute both types of requests follows linearly the size of the ontology.

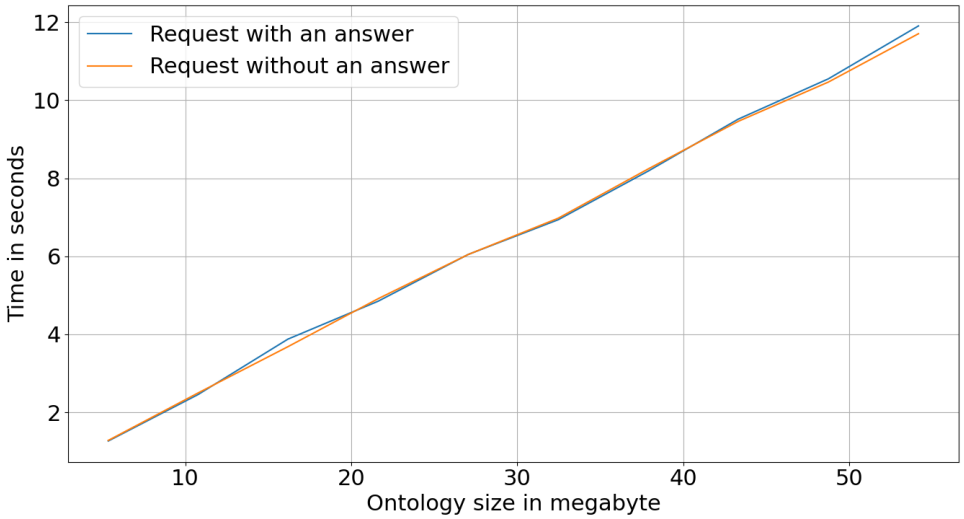


Figure 4.7: Time evolution to perform a request to the regard to the size of the ontology

4.3.5 Discussion, Conclusion and Perspective

By design, TRAILS reflects the clauses of a contract as both are composed by obligations and conditions of use, measurable objectives, rewards and penalties.

TRAILS profiles can be used as a preventive measure, describing security policies and configurations to enhance the security of the device or to limit its attack surface, as the MUD, for example, is intended to do. Indeed, they can be generated as part of existing IoT certification process such as the industry-backed schemes GSMA IoT Security Assessment (IoTSA), the PSA Certified IoT Security Framework (PSA) or Eurosmart IoT Certification Scheme (IoTCS) or a state-backed certification frameworks such as Australia's IoT Security Trust Mark (STM). Indeed, the responses to the IoTSA or PSA questionnaires and IoTCS security profile can be mapped to the properties section of TRAILS and signed by vendors or evaluators, depending on whether the certification relies on self or third party assessment. Current certification schemes mainly focus on the certification and evaluation processes, obtaining a security measurement of the device [169]. By linking the generation of the profile with the cybersecurity certification process, we benefit from the information obtained during the evaluation, recommending security measurements that could cope with the security issues detected. The usage conditions and controls described in TRAILS can be also used as a preventive measure by analyzing during the operation time if the device is behaving as expected. In case of a deviation of the conditions imposed, it can be understood as a possible attack and appropriate measures should be applied. In the same way, if a service depending on the device (services to which the device accesses or receives information) has been compromised by a threat, fast mitigation is a key to avoid major consequences. However, patches and updates delivered by the manufacturer can take days or even months. TRAILS provides a dynamic way to reconfigure the device, applying the needed countermeasures to protect it until the service is recovered from the attack. In particular, we can deny the access of the device to the compromised service and/or redirect the requests of the device to other similar and reliable services. To our knowledge, no certification scheme evaluates the trustworthiness of network function validation. This task is traditionally performed by network operators through internally defined processes. Regarding performance evaluation, vendors provide for each network function, required resources that should be allocated to achieve a given service performance. For security assessment, security auditing is also conducted by operator security teams. In the recent last years, the Network Equipment Security Assurance Scheme (NESAS) has been created by 3GPP and GSMA to accelerate

the industrialization of network function security evaluation. NESAS group aims at defining a baseline security level that should be guaranteed by every network function vendor. Moreover, the group is responsible for defining test case scenarios to be validated by network vendors that belong to NESAS. While NESAS proved its benefit to both vendors and operators but also authorities as it provides an overall framework for security evaluation, an equivalent initiative that targets virtual network functions is still missing.

In addition to the software nature of virtual network functions, the multiplication of actors in the deployment of virtual networks makes it more challenging to define an overall framework for NFV validation. Indeed, in contrast to the legacy network ecosystem where the hardware is tightly coupled with the software, the operation of virtual network functions involves multiple actors including infrastructure providers, network vendors, and service operators. Security and performance evaluation in such a context requires strong liability management mechanisms. Given that stakeholders sign their claims, TRAILS requires a Public Key Infrastructure and certificate. This is not the case today for ETSI NFV and would require setting up an organization to manage. Well-known and trusted organisms such as Global Platform, GSMA or ETSI could register supply chain actors and manage a Public Key Infrastructure. We propose to follow the example of the MUD file service hosted by Global Platform[1] or the eSIM certificate provisioning by GSMA[2]. TRAILS can be used in assurance continuity workflows as a way to rapidly share with users updated usage conditions in the case where vulnerabilities are disclosed. This scheme specifically includes an assurance continuity workflow in case a vulnerability is disclosed. In terms of performance, we showed that the TRAILS model adds a significant amount of information, which may significantly impact convergence, especially if the ontology is populated with complex multi-actor and multi-layer network services. However, we also showed that the system seems stable, given that the time required to perform a query seems to follow linearly the size of the ontology. Further works could examine whether it is possible to optimize the ontology or the data structure to improve these performances.

[1] https://globalplatform.org/iotopia/mud-file-service/
[2] https://www.gsma.com/esim/gsma-root-ci/

Chapter 5

Contribution 2: The Liability and Trust Metrics

Contents

5.0.1	Introduction	116
5.0.2	LASM Analysis Service (LAS) Architecture	116
5.0.3	Instance Trust Score (ITS)	119
5.0.4	MicroService Trust Score (MTS) and Service Provider Trust Score (SPTS)	119
5.0.5	Temporal Evolution on the Self-Organized Map of the ITS and the SLA Violation Risk	120
5.0.6	Financial Exposure to Penalty Risk (FEPR)	121
5.0.7	Evaluation and Result	121
5.0.8	Conclusion, Discussion and Future Work	140

5.0.1 Introduction

Previously, we introduced TRAILS, a descriptor designed to overcome the constraints of existing profiles in the Cloud-Edge-IoT domain, particularly in the context of describing responsibility, accountability, and liability within the supply chain. This initial introduction serves as a cornerstone for our second contribution, the establishment of liability and trust metrics. In fact, to enable a liability-aware management of services, it is essential to furnish metrics that offer a quantifiable means of measuring and monitoring liability-related elements. The absence of such metrics poses a significant challenge in evaluating the present status, enhancements, or setbacks in liability management. This contribution aligns with FB.2 as the metrics provide evidence for liability. This section clarify our second contribution. The metrics are generated by a module within the LASM, the LAS. This module adopts a framework structure. To clarify the tool's functionality, we employed the example of the microservices architecture commonly employed within the cloud-edge-IoT continuum. In the following sections, we delve into the LAS framework, its architecture, the generated metrics, and their evaluations. This contribution was submitted and accepted with major revisions at IEEE Transactions on Network and Service Management (IEEE TNSM) journal paper special issue on networks, systems and services operations and management through intelligence.

5.0.2 LASM Analysis Service (LAS) Architecture

This section describes the internal architecture of the proposed LAS as well as the trust and liability metrics it computes. As shown by Figure 5.1, the LAS uses labelled data sets provided by risk management experts, and the SLAs committed by Service Providers to generate three categories of metrics: Commitment Trust Score, Financial Exposure, and Commitment Trends. The LAS calculates three types of Commitment Trust Scores, namely Microservice Instance Trust Score (ITS), Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS). To achieve this, it uses the MLP and k-means described in Chapter 2. The LAS computes the Financial Exposure to Penalty Risk (FEPR) inspired from financial exposure metric calculated in the field of investments. Finally, two types of Commitments Trends are generated. Using SOM, the LAS tracks the changes of the ITS and the SLA Violation Risk over time. This generates two other outputs, namely the Instance Trust Score Trend-Variation (ITS-TV) and the SLA Violation Risk Trend-Variation (SVR-TV). Next, we outline the LAS step by step, breaking down each block for clarity.

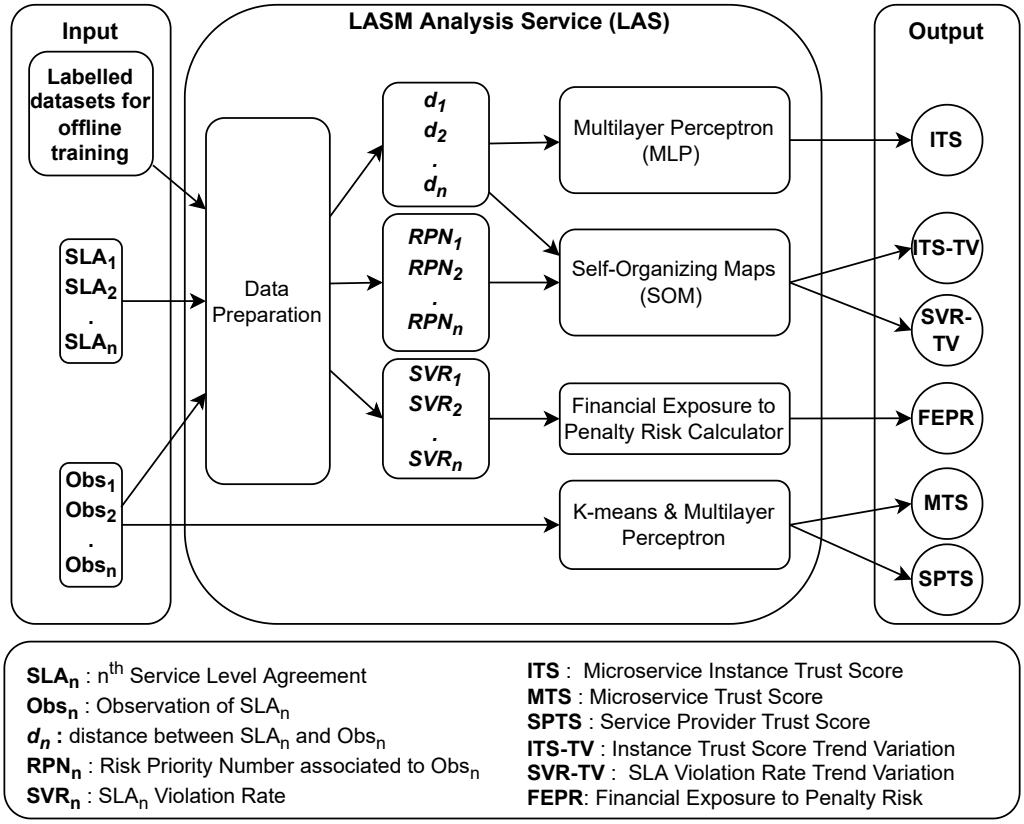


Figure 5.1: Overview of LASM Analysis Service (LAS)

Data Preparation The Data Preparation module has two roles. First, it prepares the labelled dataset to train the MLP and the SOM models. Second, it generates the input used by the models in the operational phase based on the SLAs collected from TRAILS descriptors and the observations which correspond to the values of the SLI collected by GRALAF. Preparing labelled datasets for training, each item of the dataset records the value of all the SLIs acquired at periodic intervals or when an SLA violation occurs. The data are labelled with the risk management expert's evaluation ("high", "medium", "low") of the given observation. The MLP requires a labelled dataset which contains a balanced number of normal and abnormal situations. However, we expect that, in most cases, Service Providers and their microservices tend to fullfil their commitments which would result in an imbalanced dataset problem, potentially affecting learning and predictions.To address this issue, we adopted the SMOTETomek algorithm as presented in Chapter 2. The dataset is split into training and testing sets, with a ratio of 75% and 25%. It was randomly divided several times until it was verified that the testing set represented behaviors that were unseen before. Finally, we use the GridSearchCV

methodology to set the hyperparameters of the MLP, including the number of hidden layers, the number of nodes in each layer, the activation function, the learning rate, and the solver. This method does an exhaustive search over specified parameter values for the MLP. As recommended by the methodology, we first define our GridSearchCV strategy by specifying the expected scores, then we determine the cross-validation splitting strategy as Time Series Split.

Distance between committed SLA and observation The data preparation module produces a vector D, consisting of a series of distances d, which measures the degree of compliance with the SLA. The computation of these distances depends on the specific characteristics of the SLA. For example, the following distance is suitable for an SLA which penalizes under-performance:

$$d_i = \frac{SLA_i - Obs_i}{max(Obs_i)} \tag{5.1}$$

i is a unique pairing, with SLA_i as the value for the i^{th} SLA and Obs_i as the corresponding i^{th} observation value.

Severity of deviation between committed SLA and observations Severity is measured on a scale ranging from 0 to $NCat$, indicating the severity level of the distance between the committed SLA_i and the related observation Obs_i.

$$s_i = \begin{cases} 0, & R_0(SLA_i, Obs_i) \\ 1, & R_1(SLA_i, Obs_i) \\ \dots & \\ N, & R_N(SLA_i, Obs_i) \end{cases} \tag{5.2}$$

s_i depends on the relationship between SLA_i and Obs_i as determined by various relational conditions R_0,R_1,...,R_N. Each condition $R(SLA_i, Obs_i)$ involves comparing SLA_i and Obs_i using relational operators.

SLA Violation Rate The Data Preparation module computes the SVR for each SLA_i as follows:

$$SVR_{i,l} = \frac{1}{TN} \sum_{t=0}^{TN} f(s_{i,t}, l) \tag{5.3}$$

l ranges from severity 0 to $NCat$. SVR is calculated for each l value. TN represents an observation time frame, while the function f quantifies severity occurrences within a time range TN. It is defined as follows:

$$\mathbf{f(x, l)} = \begin{cases} 1, & x = l \\ 0, & x \neq l \end{cases} \tag{5.4}$$

Risk Priority Number The data preparation module computes a Risk Priority Number (RPN) for each observation i and for each SLA_i based on the severity of deviation s_i:

$$RPN_i = s_i * SVR_{i,l} \tag{5.5}$$

5.0.3 Instance Trust Score (ITS)

Our objective is to classify each observation acquired by GRALAF. We opt for the MLP due to its proficiency in handling multi-classification problems, being a renowned neural network for such tasks. The input of the MLP is the distance vector D. The number of nodes in the input layer corresponds to n, the size of the vector D. The number of nodes in the output layer corresponds to the number of class of trusts we define. In our case, there are three classes of trust, namely "High level of trust", "Medium level of trust" and "Low level of trust". We also use the function `Softmax` in the output layer.

To adapt to the dynamic and ever-changing nature of our environment, we train and test our model offline and deploy it in the production environment. Periodically, we retrain the model using newly labeled and validated data and evaluate its performance before redeploying it with metrics such as accuracy, precision, recall, F1-score, the confusion matrix and the ROC curve using the one-vs-rest method, with the "High level of trust" designated as the class of focus. Furthermore, if we detect any issues during monitoring, such as drift, we may accelerate the retraining process.

5.0.4 MicroService Trust Score (MTS) and Service Provider Trust Score (SPTS)

The aim is to determine the level of trust in a microservice and the providers of that service from multiple observations of the service instance at a specific time point, the level of trust in the microservice and the providers of the service. (Note that one provider can offer multiple classes, and one class may involve several providers). The method for computing the MTS involves using the k-means algorithm, a Vector Quantization (VQ) technique. This algorithm is a popular clustering technique due to its simplicity and ability to scale large data sets. We have opted to use k-means algorithm because it is relatively easy to implement and is applicable to numeric and continuous data.

The k-means algorithm is used to perform VQ on several observations of a commitment on the same instance of a microservice. Let n be the number of instances of a microservice and o_j be the observation of SLA_j. At an instant T, we measure the observation o for the n microservice

instances. These measurements form the observation vector O. Then we represent the observations by a prototype (*centroid*) using k-means. The vector O and the number of cluster k are the parameters of the algorithm. To determine k, we use the Elbow Method [170], which assists in determining the ideal number of clusters in datasets. This method plots cluster numbers against a performance metric, pinpointing the optimal cluster count where further additions don't notably enhance the model. As output, the algorithm gives the codebook as output. Using the codebook, we map the code to *centroid* in order to obtain the prototype observation. The prototype observation and the commitment SLA_j are processed to obtain the distance d. This process is repeated for all commitments made on the microservice, and the resulting vector D is presented to the MLP model to obtain the trust class.

For the SPTS, we need to make some modifications to the existing methodology. Specifically, instead of inputting the observations into the k-means algorithm, we input the MTS of the relevant service provider's microservice. These values are encoded using one-hot encoding techniques. The resulting prototype generated by the k-means algorithm represents the SPTS.

5.0.5 Temporal Evolution on the Self-Organized Map of the ITS and the SLA Violation Risk

Our objective is to perform a real-time observation of the ITS and the SLA Violation Risk to have an efficient tracking of the metric dynamics. Also, we want to determine when the metrics are entering a non-desired state represented by a "forbidden" area and a "warning" area on a map. For that purpose, we use a special class of ANN called Self-Organizing Maps (SOM). The following gives an overview on how it is applied.

For each class of service, two SOM maps are created, i.e., one for each metric. Let map_1 be the one related to ITS and map_2 the one related to the SLA Violation Risk. The process is the same for both metrics. It is composed of a training phase (Step 1, 2 and 3) and a operation phase (Step 4) explained below. We used `minisom` for implementation of the SOM algorithm, which is a Python-based tool to train and construct SOM maps.

Step 1. The data acquired from measurements are transformed into input data for the SOM map. Specifically, we compute the D vector (the input for the map_1) and the RPN vector (the input for the map_2).

Step 2. The SOM is trained with all available data. The parameters for training, including the map's

dimension, learning rate, and neighborhood coefficient, are chosen based on empirical benchmarks. The quality of the resulting mapping is assessed using metrics including the quantization error, the topographic error, the silhouette score, the distortion and the neighborhood preservation [171].

Step 3. After the training phase, a label is assigned to each neuron in the grid. This label corresponds to a particular class, determined by analyzing the u-matrix and the component planes representation. Two types of area are defined: the "forbidden" area, which corresponds to neurons being labeled "Low level of trust" and "High level of risk" for map_1 and map_2, respectively and the 'warning' area which correspond to neurons being labeled "Medium level of trust" and "Medium level of risk" for map_1 and map_2, respectively.

Step 4. During the operational phase, the inputs are projected onto the map, the sequence of node in time that forms a trajectory on the map depicting the movement of the metrics. A detailed alert message is triggered if an input is projected in a restricted area.

5.0.6 Financial Exposure to Penalty Risk (FEPR)

Financial Exposure to Penalty Risk (FEPR) is a term that comes from the financial and risk management world. It is used to measure the amount of money that an investor might lose on an investment. In our context, we use it to quantify the financial risk a microservice architecture provider integrating multiple microservice components is exposed to when it offers a service to a customer. It is defined as

$$FEPR_{i,j} = \sum_{i=0}^{3} SVR_{i,j,l} * (rew_l - pen_l) \tag{5.6}$$

where rew corresponds to the reward that the microservice architecture provider earns if it honours its commitments, and pen the penalty the microservice architecture provider must repay if it does not meet its commitments. The SVR is the SLA Violation Rate.

5.0.7 Evaluation and Result

To evaluate our contribution, we showcase a practical application of the LAS through two use cases. For both, we defined several scenarios to highlight the characteristics of our contribution. The LAS was deployed on a Kubernetes container platform, with the help of Python library such as Numpy, Pandas, Scikit-Learn and Matplotlib. In the following, we describe the use cases and exhibit the

results obtained.

5.0.7.1 Use case n°1 - PacketFabric SLA

Use Case Description In the first use case, we work with synthetic data that we generated based on the SLA of PacketFabric [172]. We assume that PacketFabric provides a service which consists of deployment and management of network services. This use case also illustrates that the LAS is applicable in scenarios beyond microservices.

Service Level Agreement PacketFabric commits to the following service level metrics which are denoted as SLA_i with its related observation $\mathcal{O}_i$ [172]:

- Network availability: Deliver availability of at least 99.988% in the network → SLA: SLA_0, related observation: O_0.
- Latency: Deliver a network service with an end-to-end latency lower than 95ms → SLA: SLA_1, related observation: O_1.
- Packet loss: Deliver a network service with a network packet loss across the network lower than 0.14% → SLA: SLA_2, related observation: O_2.

The *Core Network Availability*, *Latency Metric Extended* and *Loss Metric Exceeded* tables provided in [172] summarize the levels of SLA penalties and the corresponding penalties that the service provider is eligible to receive if SLA_0, SLA_1 and SLA_2 are not met, respectively. The SP deploys the LAS in order to evaluate the network service using the Liability and Trust metrics.

Dataset for training phase For this use case, we used synthetic data to overcome the lack of real-world data. For dataset creation, we relied on the SLA details from packet fabric as our starting point. Utilizing this information, we crafted a distribution function, employing both a Gaussian Mixture Distribution (GMD) and a uniform distribution, to mimic the SLA attributes. Consequently, our dataset mirrors the SLA specifications of the packetFabric service. The methodology used consists of three steps. The first step involves generating five datasets with five GMDs that have the same mean but different variances. The second step involves drawing a uniform number of samples from these five datasets to create the final dataset. Finally, the final dataset is timestamped.

We illustrate this methodology for the Core Network Availability service level. The first step will be to generate the five datasets. The GMD takes the following form:

$$X \sim 0.90 * \mathcal{N}(\mu_1, \sigma) \ + 0.04 * \mathcal{N}(\mu_2, \sigma) \ + 0.03 * \mathcal{N}(\mu_3, \sigma) \ + 0.01 * \mathcal{N}(\mu_4, \sigma) \\ + 0.005 * \mathcal{N}(\mu_5, \sigma) \ + 0.005 * \mathcal{N}(\mu_6, \sigma) \tag{5.7}$$

The means μ_1, μ_2, μ_3, μ_4, μ_5, and μ_6 are computed as the average between each interval in the Core Network Availability table. They are $\mu_1 = 0.9999$, $\mu_2 = 0.9995$, $\mu_3 = 0.99673$, $\mu_4 = 0.991$, μ_5=0.986, and $\mu_6 = 0.961$, and are the same for all five GMDs.

However, the standard deviation σ values differ for each dataset. Specifically, for dataset n°1, $\sigma_{[1,2,3,4,5,6]} = 0.001$, for dataset n°2, $\sigma_{[1,2,3,4,5,6]} = 0.005$, for dataset n°3, $\sigma_{[1,2,3,4,5,6]} = 0.01$, for dataset n°4, $\sigma_{[1,2,3,4,5,6]} = 0.05$, and for dataset n°5, $\sigma_{[1,2,3,4,5,6]} = 0.1$.

To create the final dataset, we draw samples from the five datasets using a uniform distribution and timestamp the resulting synthetic time-series dataset. This process is repeated for the other two service levels, resulting in a final synthetic dataset of 4230 samples.

Datasets for operational phase Six datasets were created to evaluate our metrics. The first dataset is used in our evaluation of ITS, ITS-TV, and SVR-TV. The other five datasets were created to evaluate MTS and SPTS. They represent data generated by five instances of two difference microservices provided by a unique Service Provider. The first dataset is illustrated in first three rows of Table 5.1 for SLA_0, SLA_1 and SLA_2, respectively. The dataset spans over six Time Windows (TW), each having a different scope.

	TW0	TW1	TW2	TW3	TW4	TW5
$\mathcal{O}_0$	Comply	0.0048% ↓	0.0048% ↓	0.01288% ↓	4.1% ↓	Comply
$\mathcal{O}_1$	Comply	Comply	63% ↑	Comply	69% ↑	Comply
$\mathcal{O}_2$	Comply	Comply	Comply	Comply	43% ↑	Comply
ITS	High	Medium	Low	Low	Low	High

Table 5.1: Time Windows and Corresponding ITS (Comply: as expected, ↓ : less than expected, ↑ : higher than expected)

Nine case studies are included in the last five datasets, and they are all denoted using the notation $O_{x,y,z}$, where x denotes the observation index, y the instance number and z the microservice number, respectively. The following presents these nine case studies:

- **CS1** : All the observations behave as expected.
- **CS2** : $O_{1,1,1}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS3** : $O_{2,1,1}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS4** : $O_{2,1,1}$ result in 10% of MRC penalty and $O_{3,2,1}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS5** : $O_{1,1,1}$ result in 10% of MRC penalty, $O_{2,2,1}$ result in 10% of MRC penalty and $O_{3,3,1}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS6** : $O_{1,1,1}$ result in 20% of MRC penalty and $O_{1,1,2}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS7** : $O_{2,2,1}$ result in 20% of MRC penalty, $O_{2,3,1}$ result in 20% of MRC penalty and $O_{2,1,2}$ result in 30% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS8** : $O_{3,2,1}$ result in 20% of MRC penalty, $O_{2,3,1}$ result in 30% of MRC penalty and $O_{3,1,2}$ result in 20% of MRC penalty. The remaining observations conform to the expected behavior.

5.0.7.2 Results

5.0.7.2.1 Instance Trust Score

Offline phase results First, the dataset for the training phase is labelled, resampled, scaled and split into training and testing sets. The hyperparameters determined by the GridSearchCV algorithm are the learning rate of 0.0001, a single hidden layer comprising 15 neurons, and the hyperbolic tangent function as the activation function. We then start the evaluation with the confusion matrix (Figure 5.2). As we can see, only three samples have been incorrectly predicted (predicted Medium rather than Low Trust). Then, we evaluate the model using the metrics defined in Chapter 2 and show the results in Table 5.2. The classifier performs with high precision and recall scores for all three classes, along with high specificity and F1-scores. Additionally, the geometric means for all three classes are also high, indicating that the classifier is unbiased towards any particular class.

	Precision	Recall	Specificity	F1-score	Geo. mean
High Trust	1.00	1.00	1.00	1.00	1.00
Medium Trust	0.99	1.00	1.00	1.00	1.00
Low Trust	1.00	0.99	1.00	1.00	1.00

Table 5.2: Classification Results

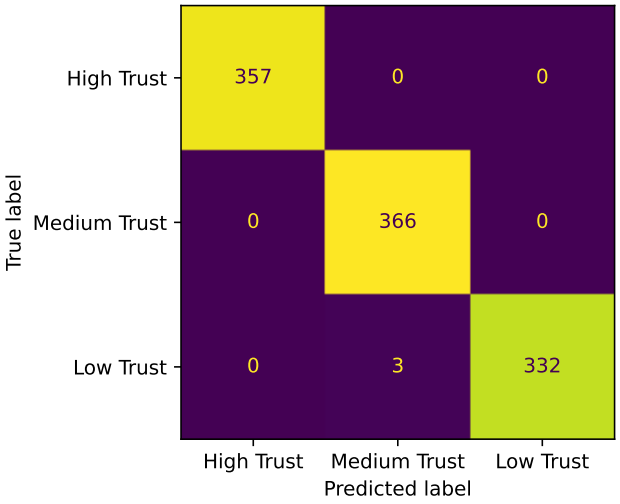


Figure 5.2: Confusion matrix

Afterwards, we plotted the ROC curve (Figure 5.3) and determined the AUC (Area under the ROC Curve). The AUC value of 1 indicates that the model can effectively distinguish instances that belong to the target class with a high level of accuracy.

Operational phase results The goal is to assess whether the ITS progress for each TW aligns with the anticipated trend of the three observations over time. The results displayed in the fourth row of Table 5.1 shows that the LAS correctly evaluates the ITS over time. Indeed, the ITS is *High* at **TW0** and **TW5**. This fits with the observations as the three SLAs are met. At **TW1**, the LAS indicates that the ITS is *Medium*. This is accurate because it is clear that during this TW the SLA_1 and the SLA_2 are met but the $\mathcal{O}_0$ deviates slightly from the committed value SLA_0 which is enough to bring down the ITS to *Medium* as it is built as such. During **TW2**, **TW3** and **TW4** the ITS is *Low*. This can be explained as at **TW2** the $\mathcal{O}_0$ deviates slightly and $\mathcal{O}_1$ deviates moderately which brings the ITS down to *Low*. At **TW3** we can observe that $\mathcal{O}_0$ deviates totally and at **TW4** the three observations totally

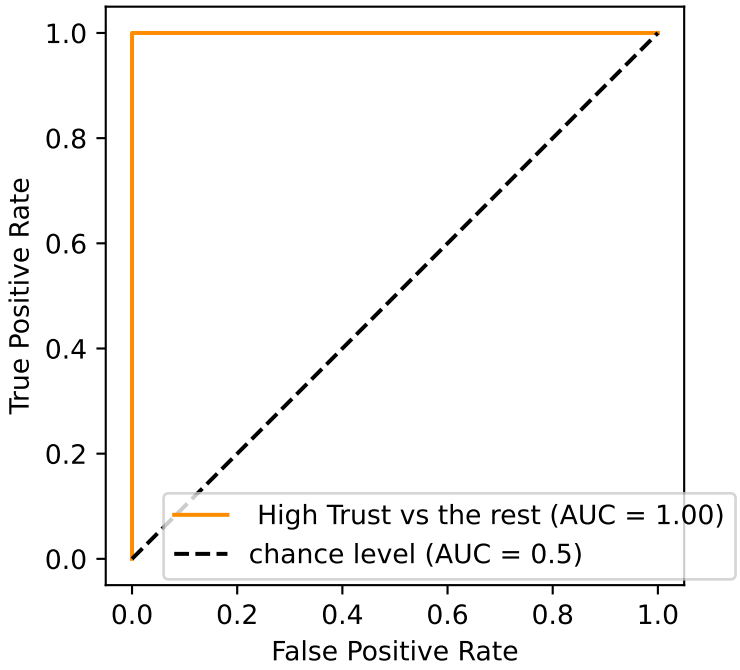


Figure 5.3: One-vs-Rest ROC curves: High Trust vs (Medium Trust & Low Trust)

deviate.

5.0.7.2.2 Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS)

Table 5.3 summarizes the MTS and SPTS for this use case in operational phase. In CS1, all observations behave as expected, resulting in a *High* rating for CTS 1, CTS 2, and SPTS. In CS2, CTS 1 is rated *Medium* due to a slight deviation in class instance availability, which is enough according to the *Core Network Availability* table. However, CTS 2 is *High* resulting in a *High* rating for SPTS. In CS3, all observations in CTS 1, CTS 2, and SPTS are *High* with only minor latency deviations in the first class. In CS4, CTS 1 is *Medium* due to slightly high latency in the first instance and packet loss in the second instance. However, CTS 2 is *High* resulting in a *High* rating for SPTS. In CS5, CTS 1 is rated *Low* due to three deviating observations: availability in the first instance, latency in the second instance, and packet loss in the third instance. CTS 2 is *High* resulting in a *Medium* rating for SPTS. In CS6, CTS 1 and 2 are rated *Medium* due to a slight deviation in availability for the first instance of each class, resulting in a *Medium* rating for SPTS. In CS7, CTS 1 is rated *Low* due to deviating latency in the second and third instances. CTS 2 is rated *Medium* due to excessively high latency in

the first instance, resulting in a *Medium* rating for SPTS. In CS8, CTS 1 is rated *Medium* due to a slight deviation in error rate and in the latency for the second and third instance, CST 2 is *High* because the deviation in latency is quite minor, resulting in a *High* SPTS.

	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8
CTS 1	H	M	H	M	L	M	L	M
CTS 2	H	H	H	H	H	M	M	H
SPTS	H	H	H	H	M	M	M	H

Table 5.3: MTS and SPTS (L: Low, M: Medium, H: High)

5.0.7.2.3 Trend Variations of Instance Trust Score and SLA Violation Rate

For both of our maps, we set the learning rate to 0.0001 to ensure a stable SOM map. Neuron weights were randomly initialized, and the Gaussian function was chosen as the neighborhood function due to its common usage. We chose to create a rectangular map with dimensions of 15 by 15 for the ITS-TV map and 10 by 10 for the SVR-TV map.

Metrics	ITS-TV map	SVR-TV map
Quantization error	0.03	0.02
Topographic error	0.08	0.03
Silhouette score	0.65	0.78
Distortion	0.71	0.65
Neighborhood preservation	0.86	0.75

Table 5.4: SOM - Evaluation

Table 5.4 presents the evaluation results of two SOM maps. The low quantization and topographic errors (0.03/0.08 for ITS-TV map, 0.02/0.03 for SVR-TV map) indicate that the SOM maintained the spatial and topological relationships of input data points, accurately. The high silhouette scores (0.65 for the ITS-TV map, 0.78 for the SVR-TV map) show excellent clustering quality, and the low distortion values (0.71 for the ITS-TV map, 0.65 for the SVR-TV map) suggest tightly packed data points in each cluster. The high neighborhood preservation score (0.86 for ITS-TV map, 0.75 for SVR-TV map) demonstrates effective preservation of spatial relationships between neighboring data points.

Interpretation of the ITS-TV Map. After an initial analysis of the u-matrix and the components plane, we have arrived at a comprehensive interpretation which is presented in detail in Figure 5.4. Our analysis has led us to identify a total of six distinct areas on the map, which can be broadly categorized into three forbidden areas and three warning areas. Each of these areas is associated with a specific color code and a detailed description of the anomaly observed in that area. The uncolored neurons indicate an area where there are no observations associated with them. The normal area where no anomaly is detected is in green. The first forbidden area, in red, corresponds to cases where the SLA_0 is 3% above the committed value. In the second forbidden area, in blue, the SLA_2 exceeds the commitment by 26%. In the third forbidden area, in black, the SLA_1 is 32% higher than the committed value. In the first warning area, in orange, the SLA_1 is 14% higher than the committed value. In the second warning area, in yellow, the SLA_0 is 1% higher than the committed value. Finally, the third warning area, in cyan, the SLA_2 is 16% above the committed value.

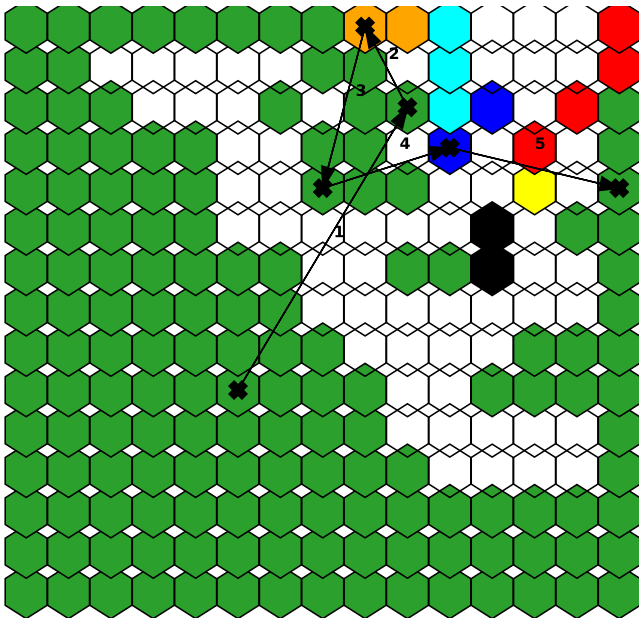


Figure 5.4: ITS-TV Map with data insight.

Operational Phase for the ITS-TV Map. To illustrate how the map can be practically employed, we present the following scenario: the service begins functioning as expected, the input data is then projected in the green area. However, the latency rises from 94ms to 95.5ms. The input data is then projected in the first warning area, which raises an alert. Then, the service returns to its normal state. Next, the packet loss rate increases to reach 0.2% and the input data is projected into the third

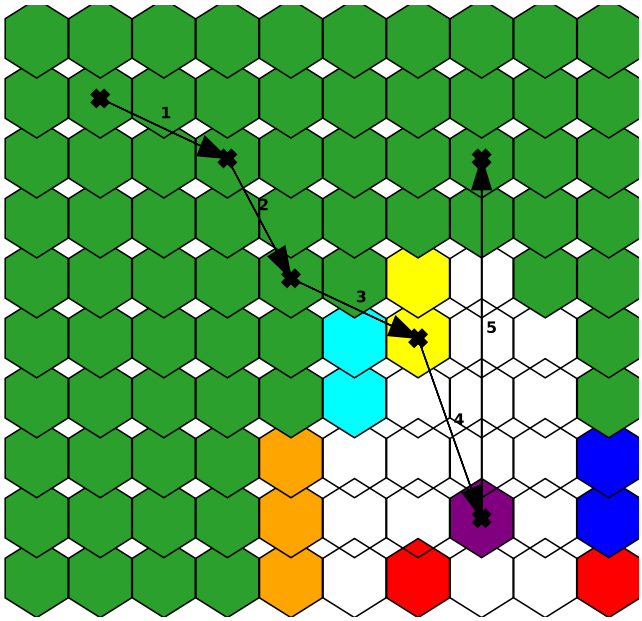


Figure 5.5: SVR-TV Map with data insight.

forbidden area. Eventually, the Network Service returns to its normal behavior, which brings the input data into the green zone. The path created by the various inputs is visualized Figure 5.4

Interpretation of the SVR-TV Map. Figure 5.5 presents an overview of the distribution of the training dataset on the SOM map. Our analysis has led us to identify three forbidden areas and three warning areas. As previously, the uncolored neurons mean no associated observations in that area, the normal area is in green. The first forbidden area, in blue, corresponds to a probability of over 58% that SLA_1 will be violated. The second forbidden area, in purple, corresponds to a probability of over 55% that SLA_2 will be violated. The third forbidden area, in red, corresponds to a probability of over 60% that SLA_0 will be violated. The first warning area, in orange, corresponds to a probability of over 27% that SLA_0 will be violated. The second warning area, in cyan, corresponds to a probability of about 13% that SLA_1 will be violated. Finally, the third warning area, in yellow, corresponds to a probability of about 18% that SLA_2 will be violated.

Operational Phase for the SVR-TV Map. To demonstrate the practical application of the map, we consider the following scenario: once the service is up and running as anticipated, the input data is then mapped onto the green region. However, the packet loss rate start to slightly increase, the input data is then projected into the third warning area and an alert is raised. The packet loss rate continues to increase, and in turn the risk that this SLA will be violated increases, so the data are

projected into the second forbidden area, an alert indicating that the high probability of this SLA being violated is raised. Finally, the Network Service resumes its normal behavior, resulting in the input data being brought back into the green zone.

5.0.7.2.4 Financial Exposure to Penality Risk

Highlight. In the following, we use the FEPR metric to analyze the PacketFabric use case and examine the relationship between the increased risk of SLA failure and the corresponding FEPR. For that purpose, we apply penalties for availability, latency, and packet loss rate, as defined in [172], using the same scenario used previously to demonstrate the variation in SLA Violation Trend.

Result. At the start of the monitoring period, the FEPR metric is at 0$, indicating that the service is operating as expected. However, as time goes on, there is a gradual degradation in packet Loss rate, which increases the risk of SLA violations and causes the FEPR to decline to -200$ and eventually to -1200$. The situation worsens, further raising the risk of SLA violations and resulting in a sharp increase in the FEPR to -2200$. The service eventually returns to normal operation, reducing the risk of SLA violations and causing the FEPR to go back to 0$.

5.0.7.3 Use case n°2 - Edgex SLA:

Use Case Description The testbed for the second use case is illustrated in Figure 5.6. It is developed by the university of Zurich. The Edgex is used for IoT device management and is an open source software framework that offers device and application interoperability at the IoT edge. Apart from the main microservices from Edgex, we also deployed some additional services. `MQTT-broker` service serves as an intermediary between the IoT devices supporting MQTT and the `MQTT-device` service. they deployed multiple instances of a microservice application that emulates IoT devices, sends random sensor data periodically and can be controlled via MQTT. To send the received sensor data to an external server which hosts Fledge, a data exporter service called `exporter-fledge` is also deployed in the Edgex environment. Fledge is an open source framework and community focused on IoT devices for the industrial edge. Locust is an open source performance testing tool capable of simulating a large number of concurrent users, and it is used to generate traffic on the Edgex ecosystem. GRALAF periodically queries Prometheus, a widely-used open-source system for gathering, storing, and querying metrics. Prometheus is configured to scrape the metrics every 60 seconds and to store them in a

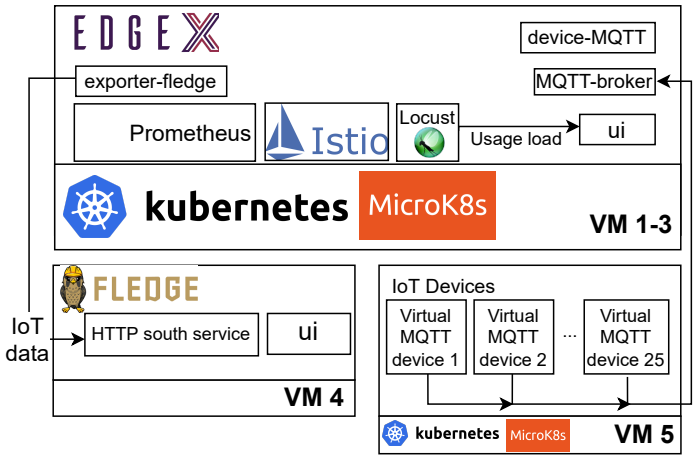


Figure 5.6: Test environment for Use Case 2, VM indicates in which virtual machine the services are deployed

time-series database that is organized by SLA name.

For the infrastructure setup, we use five virtual machines running on an OpenStack cloud infrastructure. A MicroK8s based Kubernetes cluster, which hosts Edgex services, Locust and all other required system services like Prometheus and Istio are deployed using three of the VMs. These three virtual machines have 4 vCPUs, 8 GB of RAM, and 160 GB of SSD storage. Istio provides traffic related metrics such as response time and error rate, while Prometheus scrapes all the metrics from the available providers like Kubernetes infrastructure service and Istio. A Fledge server is hosted by one VM, and 25 MQTT-based virtual IoT device applications are deployed in a MicroK8s environment on the other VM. These two virtual machines have the following resource assignments: 1 vCPU, 2GB of RAM, and 120GB of SSD storage. The service-VM mapping for this use case is illustrated in Fig. 5.6.

Target Service. Edgex service is divided into four services, specifically the *core*, *supporting*, *system management* and *devices* services. Each service is composed of one or several microservices. Each service or microservice is provided by a service/microservice provider. For the evaluation, we focused on the core service, namely the `core-metadata` microservice. It communicates with other microservices such as `core-command`, `UI` and `device-mqtt`.

Service Level Agreement. An SLA is established between the two parties where the provider of the `core-metadata` microservice committed to the following service level metrics:

- Service Availability: Deliver availability of at least 99% for the service → SLA: SLA'_0, related observation: O'_0.
- Service Latency: Deliver a service with a latency lower than 100ms → SLA: SLA'_1, related observation: O'_1.
- Service Error Rate: Deliver a service with an error rate lower than 0.5 → SLA: SLA'_2, related observation: O'_2.

Table 5.5 lists the penalty charges the consumer of the core microservice is entitled to receive if the commitment is not met with a Monthly Recurring Charge (MRC) of 10000$.

Availability	Penalty	Latency	Penalty	Error rate	Penalty
>=99.862% <99.988%	10% of MRC	10% above SLA	10% of MRC	10% above SLA	10% of MRC
>=99.445% <99.862%	20% of MRC	20% above SLA	20% of MRC	25% above SLA	25% of MRC
>=98.889% <99.445%	30% of MRC	40% above SLA	30% of MRC	50% above SLA	50% of MRC
>=98.334% <98.889%	40% of MRC	60% above SLA	40% of MRC	75% above SLA	40% of MRC
>=96.667% <98.334%	60% of MRC	75% above SLA	50% of MRC	100% above SLA	60% of MRC
<96.667%	100% of MRC	100% above SLA	60% of MRC		

Table 5.5: SLA penalties for Edgex: Availability, Latency and Error Rate

Datasets for the operation phase Four datasets were created to evaluate our metrics. The first dataset is used in our evaluation of ITS, ITS-TV, SVR-TV. The last three datasets were created to evaluate MTS and SPTS. They represent data of three instances of one microservice provided by a unique service provider. The first dataset is illustrated in the first three row of Table 5.6 for SLA'_0, SLA'_1 and SLA'_2 respectively. It spans over six TW, each having a different scope.

The last three datasets contain a total of nine case studies, each designated with the notation $O_{x,y}$ where x represents the observation index and y indicates the microservice number. Below are the details of these nine case studies:

- **CS0**: All the observations behave as expected.

	TW0	TW1	TW2	TW3	TW4	TW5
$\mathcal{O}_0$	Comply	Comply	0.0053% ↓	Comply	0.0053% ↓	Comply
$\mathcal{O}_1$	Comply	13% ↑	Comply	48% ↑	Comply	10% ↑
$\mathcal{O}_2$	Comply	Comply	Comply	Comply	13% ↑	9% ↑
ITS	High	Medium	Low	Low	Low	Low

Table 5.6: Time Windows and Corresponding ITS (Comply : as expected, ↓ : less than expected, ↑ : higher than expected)

- **CS1**: $O_{0,1}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS2**: $O_{1,1}$ result in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS3**: Each observation $O_{0,1}$ and $O_{1,1}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS4**: Each observation $O_{0,1}$ and $O_{1,1}$ results in 10% of MRC penalty. $O_{2,1}$ result in 25% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS5**: Each observation $O_{0,1}$ and $O_{0,2}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS6**: Each observation $O_{0,1}$, $O_{0,2}$ and $O_{1,1}$ results in 10 % of MRC penalty. $O_{1,2}$ iresult in 25% of MRC penalty. The remaining observations conform to the expected behavior.
- **CS7**: Each observation $O_{0,1}$, $O_{0,2}$ and $O_{1,1}$ results in 10% of MRC penalty. Additionally, $O_{1,2}$, $O_{2,1}$ and $O_{2,2}$ results in 20% of MRC penalty each. The remaining observations conform to the expected behavior.
- **CS8**: Each observation $O_{1,1}$, $O_{1,2}$ and $O_{1,3}$ results in 10% of MRC penalty. The remaining observations conform to the expected behavior.

5.0.7.4 Results

5.0.7.4.1 Instance Trust Score

Offline phase results. Prometheus may produce raw data that contains missing values. To handle

this, we use the Multivariate Imputer method outlined in [173]. We labelled, resampled, scaled and split. The GridSearchCV procedure yielded identical values for learning rate and activation function. However, some differences were noted in the values of other parameters such as the number of hidden layers (two instead of one), and the size of the hidden layers (25 instead of 15). We evaluate the model on the testing dataset. The confusion matrix is presented in Figure 5.7. We can see that only three samples were misclassified. From the confusion matrix, we calculate the evaluation metrics presented in Table 5.7.

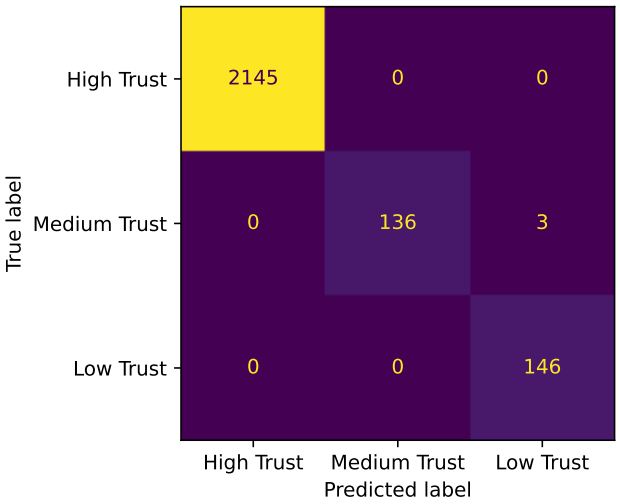


Figure 5.7: Confusion matrix

	Precision	Recall	Specificity	F1-score	Geo.mean
High Trust	1.00	1.00	1.00	1.00	1.00
Medium Trust	1.00	0.98	1.00	0.99	0.99
Low Trust	0.98	1.00	1.00	1.00	1.00

Table 5.7: Classification Results.

The classifier's performance exhibits a precision of 1.00 for the first class, 1.00 for the second class, and 0.99 for the last class, indicating a low false positive rate. Moreover, the model displays a high accuracy in identifying positive cases for all three classes, as evidenced by a recall of 1.00 for the first class, 0.98 for the second class, and 1.00 for the last class. Additionally, the specificity is high for all three classes (1.00), signifying that the classifier excels at identifying negative cases. The F1-scores for

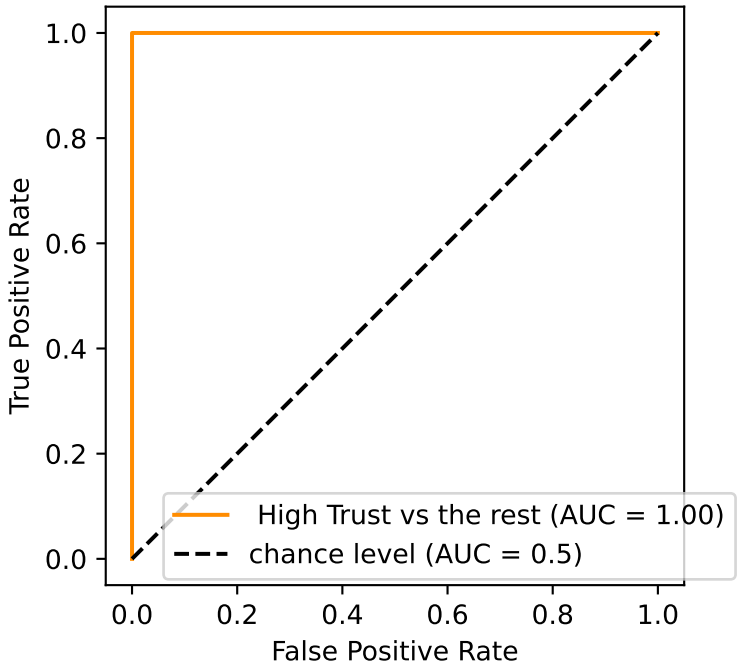


Figure 5.8: One-vs-Rest ROC curves: High Trust vs (Medium Trust & Low Trust)

all three classes are high (1.00, 0.99, and 0.99), indicating that the classifier can accurately identify most positive cases while minimizing false positives. Furthermore, a high F1-score suggests that the classifier is effectively balancing the trade-off between the three classes. Similarly, the geometric means are high for all three classes (1.00, 0.99, and 1.00), indicating that the classifier can correctly identify both positive and negative cases without showing bias towards any particular class.

Finally, we generated a ROC curve (refer to Figure 5.8) and calculated AUC, with a result value of 1, indicating the model's precise identification of target class instances.

Operational phase results. To assess the performance of the offline model for online classification, the results presented in the last row of Table 5.6 demonstrate that the LAS is successful in accurately assessing the ITS over time. Indeed, at **TW0**, the ITS is classified as *High* since all three commitments were met during that period, aligning with our expectations. However, at **TW1**, the ITS is deemed *Medium* due to the unexpected behavior of O'_1, resulting in a decrease in the ITS. This deviation is not considered critical, and the Latency is not significant based on the criteria outlined in Table 5.5. Following this disruption, the ITS gradually improves as all three commitments are met. At **TW2**, the ITS drops to *Low,* since even a minor variation in the availability of O'_0 is deemed crucial according to

the delimitation provided in Table 5.5. From that moment, the ITS will remain low, indeed at **TW3**, the ITS is **Low** since the latency is far too high compared to the commitment. At **TW4**, the ITS is *Low* since both O'_0 and O'_1 deviate from their expected behavior. At **TW5**, the ITS falls to *Low* as both O'_1 and O'_2 deviate slightly from their anticipated behavior. Finally, the ITS returns to *High* as all three observations behave as expected.

5.0.7.4.2 Microservice Trust Score (MTS) and Service Provider Trust Score (SPTS)

We report the results in Table 5.8. Since the provider only offers one class in this particular case, the score of the provider is equivalent to the score of the class. We observed that for CS1, all observations behaved as expected, leading to a CTS rating of *High*. However, for CS2, the availability of the microservice on cluster0 slightly deviated, resulting in a CTS rating of *Medium*. Similarly, for CS3, the availability and latency did not behave as expected on cluster0, resulting in a CTS rating of *Low*. For CS4, the three commitments on cluster0 were not met, leading to a CTS rating of *Low*, indicating that the issue was with the cluster0 and not the microservice. For CS5, the microservice availability on both cluster0 and cluster1 slightly deviated, leading to a CTS rating of *Low*. In CS6, unexpected behavior was observed in O_1 and O_2 in two separate clusters, namely cluster0 and cluster1, resulting in a CTS rating of *Low*. Similarly, in CS7, unexpected behavior was observed in O_1, O_2, and O_3 in two separate clusters, resulting in a CTS rating of *Low*. Finally, for CS8, the latency of the service did not behave as expected in all three clusters, leading to a CTS rating of *Medium*. Based on these observations, we can conclude that the microservice has an issue with delivering good latency.

	CS1	CS2	CS3	CS4	CS5	CS6	CS7	CS8	CS9
CTS	H	M	L	L	L	L	L	M	M
SPTS	H	M	L	L	L	L	L	M	M

Table 5.8: Microservice Trust Score and Service Provider Trust Score (L: Low, M: Medium, H: High)

5.0.7.4.3 Trend Variations of Instance Trust Score and SLA Violation Rate

For both maps, the parameters are the same as for the ITS-TV map of use case n°1. We evaluate the maps with the same metrics. Table 5.9 summarizes the results for the two SOM maps.

As shown, the quantization error and topographic error are both relatively low (0.013 and 0.02 for the first map and 0.23 and 0.026 for the second map), indicating that the SOM preserves the

Metrics	ITS-TV map	SVR-TV map
Quantization error	0.013	0.23
Topographic error	0.02	0.026
Silhouette score	0.65	0.78
Distortion	0.75	0.65
Neighborhood preservation	0.92	0.75

Table 5.9: SOM: Evaluation

topological and spatial relationships between the input data points. The silhouette score is relatively high (0.65 for the first map and 0.78 for the second map), indicating that the clustering obtained by the SOM is of good quality. Distortion is 0.75 for the first map and 0.65 for the second map, which suggests that the data points within each cluster are tightly packed around their cluster center. Finally, the neighborhood preservation is 0.92 and 0.75 which indicates a high level of preservation.

Interpretation of the ITS-TV Map. After an initial analysis of the u-matrix and the components plane, we conclude an interpretation presented in Figure 5.9. We defined five forbidden areas and four warning areas. Each area is accompanied by a color code and a description of the anomaly. In the first forbidden area, in brown, the SLA'_0 exceeds the commitment by 4%. In the second forbidden area, in purple, the SLA'_2 exceeds the commitment by 21%. In the third forbidden area, in red, the SLA'_1 with `core-command` is 32% higher than the committed value. In the fourth forbidden area, in pink, the SLA'_1 with `UI` is 49% higher than the committed value. In the fifth forbidden area, in cyan, SLA'_1 with `device-mqtt` is 52% higher than the committed value. In the first warning area, in orange, the SLA'_1 with `core-command` is about 14% higher than the committed value. In the second warning area, in yellow, the SLA'_1 with `UI` is 15% higher than the committed value. In the third warning area, in blue, the SLA'_0 is 1% above the committed value. In the fourth warning area, in gray, the SLA'_2 is 8% higher than the committed value.

Operational Phase for the ITS-TV Map. To show how the map can be employed in practical situations, we establish a scenario. Initially, the service is functioning normally as expected. However, over time, the error rate gradually deteriorates from 0 to 0.53 and eventually to 0.75, while the latency improves from 98.98ms to 96.23ms and then to 95.14ms. The input data are then directed toward the Warning area n°4, an alert signaling the issue is triggered. Later on, the service returns to normal but with a slight increase in latency. However, it is suddenly disrupted, due to an increase in latency with the `core-command` microservice, The input data enter the forbidden area n°3 and an alert is triggered. Finally, the service returns to normal and behaves as expected. The path created by the various

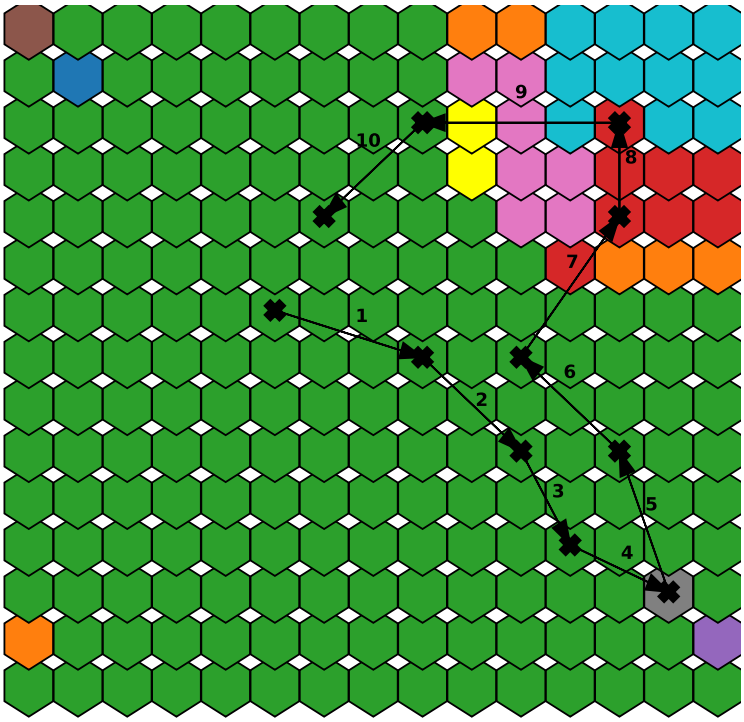


Figure 5.9: ITS-TV Map with data insight.

inputs is visualized in Figure 5.9.

Interpretation of the SVR-TV Map. As illustrated in Figure 5.10, we defined five forbidden areas and three warning areas have been identified. This first forbidden area, in blue, corresponds to a probability of over 60% that SLA'_1 with `core-command` will be violated. In the second forbidden area, in brown, the probability that SLA'_0 will be violated is over 55%. In the third forbidden area, in red, there is a probability of over 60% that SLA'_2 will be violated. In the fourth forbidden area, in black, there is a probability of over 62% that SLA'_1 with `UI` will be violated. In the fifth forbidden area, in purple, there is a probability of over 62% that SLA'_1 with `device-mqtt` will be violated. In the first warning area, in orange, there is a probability of about 22% that SLA'_0 will be violated. In the second warning area, in cyan, there is a probability of about 14% that SLA'_1 will be violated. In the third warning area, in yellow, the probability of violating SLA'_2 is about 16%.

Operational Phase for the SVR-TV Map. To demonstrate how the map can be applied in practical situations, We establish the following scenario: the service performs according to expectations initially, the input data is projected in the green area. Then there is a slight deterioration in latency. The projected data gradually shifts toward the forbidden area n°1. After the fifth projection, an alert

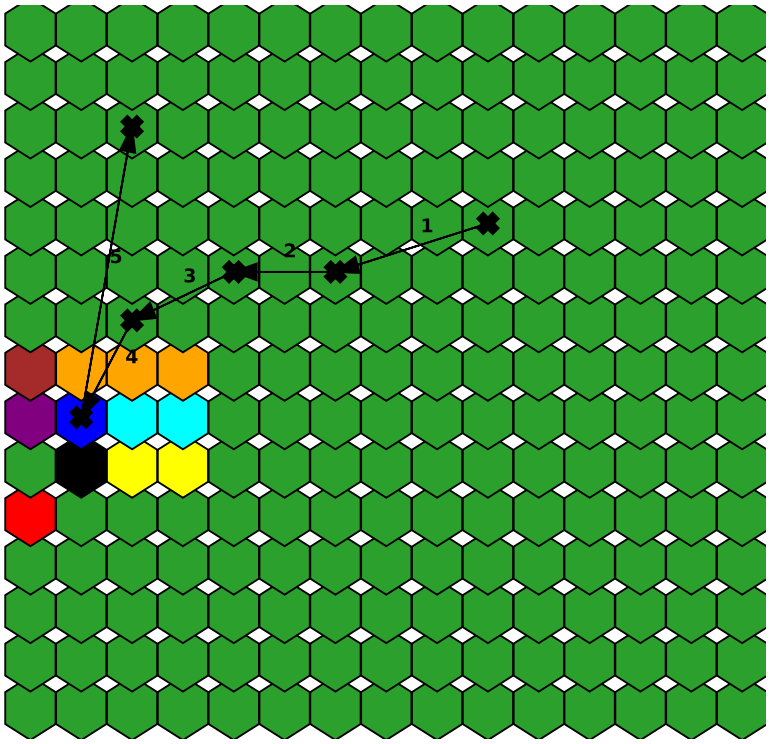


Figure 5.10: SVR-TV Map with data insight.

is triggered as the input data is projected into the forbidden area n°1. At the end, the neuron is projected to the green area as the service resumes normal operation. The trajectory created by the inputs data can be visualized in Figure 5.10.

5.0.7.4.4 Financial Exposure to Performance Risk

Highlight. The FEPR makes sense for the composition of microservices, not for an isolated microservice. For that, we first perform an SLA composition process with all the microservices using decision rules such as `maximum` and `minimum`. We reused the scenario defined for the operation phase of the SVR-TV map. The penalties for the availability, the latency and the error rate are defined in Table 5.5.

Result. Initially, the FEPR remains at 0$ as the service operates as expected. However, over time, there is a slight degradation in latency, which raises the risk of SLA violations and causes the FEPR to decline to -100$ and eventually to -1000$. The error rate also starts to deteriorate, further increasing the risk of SLA violations and resulting in an elevation of the FEPR to -1100$ and -2000$. Eventually, the service returns to normal operation, which reduces the risk of SLA violations and leads to a decrease in the FEPR to -900$ and back to 0$.

5.0.8 Conclusion, Discussion and Future Work

In this section we introduce through the LASM Analysis Service (LAS), a framework for analyzing liability and trust in multi-actor dynamic environment, three types of liability and trust metrics: Commitment Trust Scores, which assess the trust that an instance, all instances of a service, or all services of a provider will perform as expected based on SLA commitments; Financial Exposure, which measures the potential monetary loss for the overall service architecture provider with the current composition of services; and Commitment Trends, which monitors trends of SLA Violation Rates and Instance Commitment Trust to predict violations. We used a microservice architecture in order to demonstrate the LAS. The framework has been implemented on a Kubernetes platform. We apply our framework to evaluate two services in different scenarios and case studies, namely a network service which simulates the behavior of the packet-fabric service and Edgex service for IoT edge computing. The results demonstrate the effectiveness of the LAS in accurately computing the trust and liability metrics for both use cases.

We finalize this contribution number by discussing some practical remarks. Beginning with the ITS metric, the MLP demonstrated favorable outcomes for both use cases and accurately identified the trust level in various scenarios. Nevertheless, a significant drawback of utilizing this method is the necessity to train a model for each service class. Furthermore, since we operate in a dynamic environment, it is crucial to retrain the model to consider variations in the fundamental data distribution, adapt to novel patterns and trends, and enhance the model's overall effectiveness and precision. These two adverse points can be computationally expensive and time-consuming.

During the evaluation, both MLP models had a size of approximately 16 MB. Using an Intel® Xeon® W-2133 Processor with 32GB RAM, the first use case model required 42 seconds to train, while the second model took slightly longer, about 55 seconds. This training time is considered reasonable.

For CTS and STS, we tried to use case studies to scan the most convincing cases. However, we only had three clusters at our disposal for the Edgex use case, so we did not have enough data to show the usefulness of k-means algorithm. Also, it can be interesting to explore alternative clustering algorithms that are better suited, for example k-mode [174] or hierarchical clustering [175]. Additionally, it may be helpful to validate the clustering results using other metrics and visualizations, such as silhouette plots, to ensure that the clustering is meaningful and useful for the specific use case.

One of the main limitations of using SOM in trending the variation of the ITS and the SLA Violation Risk is their inability to adapt to changes in the underlying data distribution over time. This means that if the input data changes significantly or new data is introduced, the original SOM may no longer accurately represent the data and its performance may degrade. Another limitation of SOM in dynamic environments is their sensitivity to initial conditions and the specific parameters used during training. This implies that the resulting SOM may not always converge to the optimal solution. To overcome these limitations, researchers have proposed various modifications to the SOM algorithm, such as incorporating adaptive learning rates, incorporating online learning techniques, and using incremental training approaches [176]. These modifications can be integrated into a new version of the LAS as future work.

For the FEPR, we manage to demonstrate the correlation between the SLA Violation Risk and the Financial Risk Exposure with the two use cases. However, it is important to note that this metric should be presented in a clear and understandable manner, and any potential biases or limitations of the metric should be thoroughly addressed and discussed. Additionally, the metric should be used as a supplement to, rather than a replacement for, human judgement and expertise in the legal field.

As part of our future work, we didn't address microservice dependencies, which could influence responsibility-related indicators. We plan to investigate this aspect further. We also aim to explore additional metrics from the SOM map and test the LAS in various use cases beyond Network Services and microservices, like a 5G service involving IoT and Edge computing services.

Chapter 6

Conclusion

Contents

6.1 Synthesis of the Context and Problem Statement **144**
6.2 Contributions **145**
6.3 Perspectives **146**

6.1 Synthesis of the Context and Problem Statement

The IoT's rapid expansion is revolutionizing industries through innovations like Smart Home Automation and Autonomous Vehicles, supported by the integration of Cloud and Edge Computing with microservices. This creates a complex Cloud-Edge-IoT continuum, marked by cross-domain collaboration and a multi-layered responsibility structure.

A key challenge in the Cloud-Edge-IoT continuum is the management of liability. Indeed, in such a diverse and interconnected environment, determining who is responsible and liable for various aspects of the service delivery becomes complex. This complexity arises from several factors such as the multiple stakeholders: the continuum involves a range of entities like service providers, infrastructure providers, device manufacturers, and application developers. Each has distinct roles and responsibilities, making it difficult to ascertain who is accountable in the event of a service failure or security breach. The dynamic environment, the continual evolution of technology and services in the IoT landscape, means that responsibilities and roles are not static. They can change as new services are developed and deployed, further complicating liability management. Interoperability challenges, with various technologies and platforms involved, ensuring integration while maintaining clear lines of responsibility, is a hard task.

In cases of cyberattacks or service failures, identifying the liable party is crucial for legal and financial recovery. The distributed nature of services in the Cloud-Edge-IoT continuum makes it challenging to pinpoint where the fault lies and who bears the financial burden. SLAs in such environments tend to be intricate. Ensuring compliance with these agreements, and determining liability when SLAs are breached, requires a nuanced understanding of the entire service delivery chain.

To effectively address the challenges of responsibility management in this kind of environment, it is essential to develop responsibility management mechanisms that can adapt to the dynamic and multi-actor nature of the Cloud-Edge-IoT continuum.

6.2 Contributions

We address the challenge of responsibility management through two key aspects: first, by defining the responsibilities within the supply chain, and second, by proposing metrics for liability and trust in order to effectively quantify and assess compliance.

The cornerstone of this research is the development of TRAILS (sTakeholder Responsibility, AccountabIlity, and Liability deScriptor), a novel descriptor and its accompanying ontology. TRAILS tracks a component or service throughout its lifecycle, enabling all supply chain participants to outline their commitment, addressing a significant gap in existing models. This innovative approach not only delineates the roles and responsibilities of stakeholders but also lead to more transparent and accountable services.

We developed the descriptor by enhancing an already existing descriptor in the Cloud-Edge-IoT continuum, specifically the TOSCA NFV, which is used to deploy VNFs. Our enhancement ensures that the descriptor is generic, suitable for any component or service within the continuum. TRAILS aligns with the Inspire-5Gplus manifest's criteria, including key aspects such as responsibility, accountability, liability, and being both generic and modular. These features are critical for effective responsibility management in the Cloud-Edge-IoT continuum. We also developed a user-friendly interface for TRAILS, enabling stakeholders to easily comprehend and engage with the descriptor, and to construct a TRAILS descriptor from scratch.

Further, this thesis presents the LAS, a framework that generates metrics for liability and trust. These metrics are divided into three categories: Commitment Trust Scores, Financial Exposure, and Commitment Trends. Developed using machine learning and neural network techniques, they provide a comprehensive method for evaluating liability and trust. This includes assessing the reliability of service commitments and estimating potential financial risks. These metrics form a strong tool for service providers to accurately assess and manage the liabilities and responsibilities of a dynamic and multi-actor environment. This framework employs the TRAILS descriptor to clearly outline the responsibilities of supply chain stakeholders, set objectives, and define penalties for not meeting these responsibilities. Additionally, the framework utilizes GRALAF, a Root Cause Analysis (RCA) tool developed by the University of Zurich, which is used to monitor services targeted by the LAS, providing the necessary observations for metric calculations.

This thesis has led to the development of the Liability-Aware Security Manager (LASM) framework. This tool assists administrators in making informed decisions to ensure service commitments are met. LASM is composed of several modules: the first, LASM Visualized Service (LVS), is dedicated to displaying services and their related data. The second module, LASM Referencing Service (LRS), organizes a catalog of network components and their associated TRAILS profiles. It uses an ontology to facilitate the evaluation of new components' TRAILS in accordance with a referencing policy, or to search for profiles with specific attributes. The third module, LASM Analysis Service (LAS), is designed to evaluate various metrics concerning trust, responsibility, or the reputation of components and authors. Lastly, the LASM Creation Service (LCS) assists administrators in creating TRAILS profiles. The LRS, the LVS and the LCS were implemented for the first contribution and the LAS for the second contribution. A presentation showcasing the first contribution was delivered at the Orange Salon De La Recherche 2022, while another presentation featuring the second contribution was presented at the ACM MobiCom 2023 conference.

6.3 Perspectives

In the short term, our plans for TRAILS involve some enhancements, primarily by incorporating new descriptors to more effectively manage responsibility and liability. A key part of this enhancement is the integration of the SBOM descriptor, which provides a detailed inventory of all software components in a product, including their versions and origins. This integration aims to foster greater transparency within software supply chains. By merging TRAILS with SBOM, we are working towards establishing a comprehensive system for supervising supply chains, effectively combining software component tracking with a clear outline of stakeholder responsibilities. Additionally, we are exploring ways to expand TRAILS' compatibility with various SLA languages. Currently, TRAILS is compatible with WS-Agreement, but we plan to extend its compatibility to include other SLA languages like CSLA or WSLA. This expansion may involve the development of an interpreter that can adapt to different SLA formats, thereby enhancing TRAILS' versatility and applicability in diverse settings.

In the long term, we plan to delve into service instantiation using the TRAILS descriptor, particularly in cloud infrastructures. This may involve adding new instantiation-specific fields to TRAILS, drawing inspiration from existing descriptors like Kubernetes manifests or VNFDs. Another ambitious goal is to update the ontology to include real-time liability and trust metrics, which would

facilitate ontology-based searches for service components, using trust metrics as part of the criteria.

In the short term for the LAS, our focus is on extracting new metrics from the SOM map. Key research inquiries involve understanding the implications of service trajectories on the map for trust and liability assessment. Specifically, we are exploring how to interpret the trustworthiness of a service that frequently moves between different zones. This also extends to risk mitigation strategies – determining the significance of these movements and whether they warrant raising alerts. Also, regarding financial exposure, we can extract information about whether to implement a countermeasure. For instance, if the cost of the countermeasure is less than the potential loss, it is preferable to implement it. This calculation can be automated within the LAS, triggering a message advising whether to proceed with the countermeasure.

In the long term, conducting further evaluations with more complex services is a strategic approach. This involves applying the framework to a wider range of services that have higher complexity in terms of their operations, dependencies, and stakeholder interactions. Evaluating the tool in these varied and intricate environments would provide deeper insights into its scalability and adaptability. It would also reveal areas where the tool may require refinement or additional features to handle complex service scenarios effectively. Additionally, experimenting with other ML algorithms, such as dynamic SOM, which allow on-line and continuous learning on both static and dynamic data distributions, could be highly beneficial. Dynamic SOMs are particularly well-suited for situations where the data environment is constantly evolving. Unlike static models, dynamic SOMs can adapt to changes over time, making them more aligned with the dynamic nature of service environments. Implementing these in the context of the tool could enhance its ability to learn from and adapt to new data, patterns, and changes in the service landscape. This would potentially lead to more accurate predictions and more effective management of liabilities and trust in services. Overall, these long-term strategies aim to evolve the tool into a more versatile and powerful solution for managing complex service environments.

Bibliography

[1] J. Ortiz, R. Sanchez-Iborra, J. Bernal Bernabe, A. Skarmeta, C. Benzaid, T. Taleb, P. Alemany, R. Muñoz, R. Vilalta, C. Gaber, j.-p. Wary, D. Ayed, P. Bisson, M. Christopoulou, G. Xilouris, E. Montes de Oca, G. Gür, G. Santinelli, V. Lefebvre, and D. Lopez, *INSPIRE-5Gplus: intelligent security and pervasive trust for 5G and beyond networks*, Aug. 2020.

[2] "Software Updates for Internet of Things (suit)." [Online]. Available: https://datatracker.ietf.org/wg/suit/about/

[3] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task Force, Request for Comments RFC 7252, Jun. 2014. [Online]. Available: https://datatracker.ietf.org/doc/rfc7252

[4] E. Lear, R. Droms, and D. Romascanu, *RFC 8520 - Manufacturer Usage Description Specification*, IETF Std., Mar. 2019.

[5] "Etsi gs nfv-ifa 011 : Network functions virtualisation (nfv) management and orchestration vnf packaging specification."

[6] "ETSI GS NFV 002 V1.2.1 : Network Functions Virtualisation (NFV); Architectural Framework." [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[7] *TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0*. [Online]. Available: http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.html#_Toc421872036

[8] P. Govindaraj, "A Review on Various Trust Models in Cloud Environment," *JOURNAL OF ENGINEERING SCIENCE AND TECHNOLOGY REVIEW*, vol. 10, pp. 213–219, Mar. 2017.

[9] J. Huang and D. M. Nicol, "Trust mechanisms for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 2, no. 1, p. 9, Apr. 2013. [Online]. Available: https://doi.org/10.1186/2192-113X-2-9

[10] A. Chandrasekar, K. Chandrasekar, M. Mahadevan, and P. Varalakshmi, "QoS Monitoring and Dynamic Trust Establishment in the Cloud," in *Advances in Grid and Pervasive Computing*, ser. Lecture Notes in Computer Science, R. Li, J. Cao, and J. Bourgeois, Eds. Berlin, Heidelberg: Springer, 2012, pp. 289–301.

[11] S. Pearson, V. Tountopoulos, D. Catteddu, M. Südholt, R. Molva, C. Reich, S. Fischer-Hübner, C. Millard, V. Lotz, M. G. Jaatun, R. Leenes, C. Rong, and J. Lopez, "Accountability for cloud and other future Internet services," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, Dec. 2012, pp. 629–632.

[12] R. K. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. S. Lee, "TrustCloud: A Framework for Accountability and Trust in Cloud Computing," in *2011 IEEE World Congress on Services*, Jul. 2011, pp. 584–588.

[13] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration."

[14] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, "Establishing and Monitoring SLAs in Complex Service Based Systems," in *2009 IEEE International Conference on Web Services*, Jul. 2009, pp. 783–790. [Online]. Available: https://ieeexplore.ieee.org/document/5175897

[15] N. Antonopoulos, G. Exarchakos, M. Li, and A. Liotta, Eds., *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications.* IGI Global, 2010. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-61520-686-5

[16] J. M. J. Valero, V. Theodorou, M. G. Pérez, and G. M. Pérez, "Sla-driven trust and reputation management framework for 5g distributed service marketplaces," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–13, 2023.

[17] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "TOSCA: Portable Automated Deployment and Management of Cloud Applications," in *Advanced Web Services*, A. Bouguettaya, Q. Z. Sheng, and F. Daniel, Eds. New York, NY: Springer, 2014, pp. 527–549. [Online]. Available: https://doi.org/10.1007/978-1-4614-7535-4_22

[18] F. van Lingen and Yannuzzi, "The unavoidable convergence of nfv, 5g, and fog: A model-driven approach to bridge cloud and edge," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 28–35, 2017.

[19] "Landing page," publication Title: EUCloudEdgeIOT. [Online]. Available: https://eucloudedgeiot.eu/

[20] Orange et inria renforcent leur partenariat stratégique avec le lancement d'un laboratoire commun dédié au continuum ■ cloud to IoT ■ | inria. [Online]. Available: https://www.inria.fr/fr/cloud-orange-inria-partenariat-strategique-laboratoire-commun

[21] "Funding & tenders." [Online]. Available: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/horizon-cl3-2023-cs-01-01;callCode=null;freeTextSearchKeyword=HORIZON-CL3-2023-CS-01;matchWholeText=true;typeCodes=0,1,2,8;statusCodes=31094501,31094502;prog=

[22] J. Hong, Y.-G. Hong, X. d. Foy, M. Kovatsch, E. Schooler, and D. Kutscher, "IoT Edge Challenges and Functions," Internet Engineering Task Force, Internet Draft draft-irtf-t2trg-iot-edge-02, num Pages: 30. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-t2trg-iot-edge-02

[23] "System architecture for the 5g system (5gs)." [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

[24] "(7) (PDF) The Cloud-to-Thing Continuum Opportunities and Challenges in Cloud, Fog and Edge Computing: Opportunities and Challenges in Cloud, Fog and Edge Computing." [Online]. Available: https://www.researchgate.net/publication/342746158_The_Cloud-to-Thing_Continuum_Opportunities_and_Challenges_in_Cloud_Fog_and_Edge_Computing_Opportunities_and_Challenges_in_Cloud_Fog_and_Edge_Computing

[25] R. Shivaswamy, R. Shailendra, and S. Velayutham, "Challenges for Convergence of Cloud and IoT in Applications and Edge Computing," Apr. 2022, pp. 644–662.

[26] A. R. Biswas and R. Giaffreda, "IoT and cloud convergence: Opportunities and challenges," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Mar. 2014, pp. 375–376.

[27] C. Gaber, G. Arfaoui, Y. Carlinet, N. Perrot, L. Valleyre, M. Lacoste, J.-P. Wary, Y. Anser, R. Artych, A. Podlasek, E. Montesdeoca, V. Hoa La, V. Lefebvre, and G. Gür, "The Owner, the Provider and the Subcontractors: How to Handle Accountability and Liability Management for 5G End to End Service," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*. Vienna Austria: ACM, Aug. 2022, pp. 1–7. [Online]. Available: https://dl.acm.org/doi/10.1145/3538969.3544465

[28] N. Zahan, E. Lin, M. Tamanna, W. Enck, and L. Williams, "Software Bills of Materials Are Required. Are We There Yet?" *IEEE Security & Privacy*, vol. 21, no. 2, pp. 82–88, Mar. 2023, conference Name: IEEE Security & Privacy.

[29] C. Gaber, J. S. Vilchez, G. Gür, M. Chopin, N. Perrot, J.-L. Grimault, and J.-P. Wary, "Liability-Aware Security Management for 5G," in *2020 IEEE 3rd 5G World Forum (5GWF)*, Sep. 2020, pp. 133–138.

[30] M. Wilson, D. E. de Zafra, S. I. Pitcher, J. D. Tressler, and J. B. Ippolito, "Information technology security training requirements :: a role- and performance-based model," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-16, 1998. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-16.pdf

[31] A. E. Oldehoeft, "Foundations of a security policy for use of the National Research and Educational Network," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST IR 4734, 1992. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir4734.pdf

[32] T. W. House, "FACT SHEET: Biden-Harris Administration Announces National Cybersecurity Strategy," Mar. 2023, publication Title: The White House. [Online]. Available: https://www.whitehouse.gov/briefing-room/statements-releases/2023/03/02/fact-sheet-biden-harris-administration-announces-national-cybersecurity-strategy/

[33] F. B. Schneider, "Accountability for Perfection," *IEEE Security & Privacy*, vol. 7, no. 2, pp. 3–4, Mar. 2009.

[34] B. Lampson, "Privacy and securityUsable security: how to get it," *Communications of the ACM*, vol. 52, no. 11, pp. 25–27, Nov. 2009. [Online]. Available: https://dl.acm.org/doi/10.1145/1592761.1592773

[35] Z. Xiao, N. Kathiresshan, and Y. Xiao, "A survey of accountability in computer networks and distributed systems," *Security and Communication Networks*, vol. 9, no. 4, pp. 290–315, 2016, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.574. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.574

[36] M. Y. Vardi, "Accountability and Liability in Computing." [Online]. Available: https://cacm.acm.org/magazines/2022/11/265836-accountability-and-liability-in-computing/fulltext

[37] R. Anderson and T. Moore, "Information Security Economics – and Beyond."

[38] D. Berry, "Appliances and Software: The Importance of the Buyer's Warranty and the Developer's Liability in Promoting the Use of Systematic Quality Assurance and Formal Methods," Oct. 2000.

[39] D. Ryan and C. Heckman, "Two views on security software liability. Let the legal system decide," *IEEE Security & Privacy*, vol. 1, no. 1, pp. 70–72, Jan. 2003, conference Name: IEEE Security & Privacy.

[40] lafourcade, "LISE - [Verimag]." [Online]. Available: https://www-verimag.imag.fr/LISE.html?lang=en

[41] B. Chun and A. Bavier, *Decentralized Trust Management and Accountability in Federated Systems*, Jan. 2004, vol. 37.

[42] A. Yumerefendi and J. Chase, "The role of accountability in dependable distributed systems," Jun. 2005.

[43] H. Nissenbaum, "Accountability in a computerized society," *Science and Engineering Ethics*, vol. 2, no. 1, pp. 25–42, Mar. 1996. [Online]. Available: https://doi.org/10.1007/BF02639315

[44] M. Felici and C. Fernández-Gago, Eds., *Accountability and Security in the Cloud: First Summer School, Cloud Accountability Project, A4Cloud, Malaga, Spain, June*

2-6, 2014, Revised Selected Papers and Lectures, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, vol. 8937. [Online]. Available: https://link.springer.com/10.1007/978-3-319-17199-9

[45] "About Us – INSPIRE-5Gplus." [Online]. Available: https://www.inspire-5gplus.eu/about-us/

[46] J. Ortiz, R. Sanchez-Iborra, J. B. Bernabe, A. Skarmeta, C. Benzaid, T. Taleb, P. Alemany, R. Muñoz, R. Vilalta, C. Gaber, J.-P. Wary, D. Ayed, P. Bisson, M. Christopoulou, G. Xilouris, E. M. de Oca, G. Gür, G. Santinelli, V. Lefebvre, A. Pastor, and D. Lopez, "INSPIRE-5Gplus: Intelligent Security and Pervasive Trust for 5G and beyond Networks," ser. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020, event-place: Virtual Event, Ireland.

[47] N. Dragoni, F. Massacci, T. Walter, and C. Schaefer, "What the heck is this application doing? – A security-by-contract architecture for pervasive services," *Computers & Security*, vol. 28, no. 7, pp. 566–577, Oct. 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404809000716

[48] W. K. Wong, S. O. Cheung, T. W. Yiu, and H. Y. Pang, "A framework for trust in construction contracting," *International Journal of Project Management*, vol. 26, no. 8, pp. 821 – 829, 2008.

[49] G. Gür, V. Lefebvre, L. Morel, C. Benzaid, and A. G. Skarmeta, "WP4, T4.2 Trust, Trust Service Level Agreement, TSLA, Trustworthiness," 2019.

[50] M. Felici and S. Pearson, "Accountability, Risk, and Trust in Cloud Services: Towards an Accountability-Based Approach to Risk and Trust Governance," in *2014 IEEE World Congress on Services*, Jun. 2014, pp. 105–112.

[51] D. Catteddu, M. Felici, G. Hogben, A. Holcroft, E. Kosta, R. Leenes, C. Millard, M. Niezen, D. Nuñez, N. Papanikolaou, S. Pearson, D. Pradelles, C. Reed, C. Rong, J. Royer, D. Stefanatou, and T. Wlodarczyk, "Towards a model of accountability for cloud computing services," May 2013. [Online]. Available: https://www.semanticscholar.org/paper/Towards-a-model-of-accountability-for-cloud-Catteddu-Felici/4987aa74f760962cf1ab2673c21766477d00bf6b

[52] D. Nuñez, "Metrics for Accountability."

[53] M. Abrams and J. Kropf, "The Global Information Accountability Project at Five Years," May 2014, place: Rochester, NY Type: SSRN Scholarly Paper. [Online]. Available: https://papers.ssrn.com/abstract=2510933

[54] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner, "OpenTOSCA – A Runtime for TOSCA-Based Cloud Applications," in *Service-Oriented Computing*, ser. Lecture Notes in Computer Science, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds. Berlin, Heidelberg: Springer, 2013, pp. 692–695.

[55] "TOSCA Simple Profile for Network Functions Virtualization (NFV)—Version 1.0." [Online]. Available: http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html

[56] M. C. Suárez-Figueroa, R. García Castro, B. Villazon Terrazas, and A. Gomez-Perez, "Essentials In Ontology Engineering: Methodologies, Languages, And Tools," Oct. 2011.

[57] F. Manola and E. Miller, "Rdf primer: W3c recommendation," *Decision Support Systems - DSS*, Jan. 2004.

[58] B. Motik, P. Patel-Schneider, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler, and M. Smith, "OWL 2 Web Ontology Language: Structural Specification and Functional-Style," *Journal of Pragmatics - J PRAGMATICS*, vol. 27, Jan. 2008.

[59] "RDF 1.1 XML Syntax." [Online]. Available: https://www.w3.org/TR/rdf-syntax-grammar/

[60] A. Polleres, "SPARQL," in *Encyclopedia of Social Network Analysis and Mining*, R. Alhajj and J. Rokne, Eds. New York, NY: Springer, 2014, pp. 1960–1966. [Online]. Available: https://doi.org/10.1007/978-1-4614-6170-8_124

[61] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," World Wide Web Consortium, W3C Member Submission, 2004.

[62] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means Clustering Algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979, publisher: [Wiley, Royal Statistical Society].

[63] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[64] S. Narayan, "The generalized sigmoid activation function: Competitive supervised learning," *Information Sciences*, vol. 99, no. 1, pp. 69–82, Jun. 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025596002009

[65] K. Banerjee, V. P. C, R. R. Gupta, K. Vyas, A. H, and B. Mishra, "Exploring Alternatives to Softmax Function," Nov. 2020, arXiv:2011.11538 [cs]. [Online]. Available: http://arxiv.org/abs/2011.11538

[66] Y. H. Hu and J.-N. Hwang, Eds., *Handbook of neural network signal processing*, ser. Electrical engineering and applied signal processing series. Boca Raton: CRC Press, 2002.

[67] H. J. Kelley, "Gradient theory of optimal flight paths," *Ars Journal*, vol. 30, no. 10, pp. 947–954, 1960.

[68] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[69] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[70] "RMSProp - Cornell University Computational Optimization Open Textbook - Optimization Wiki." [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=RMSProp

[71] "Mean Squared Error," in *The Concise Encyclopedia of Statistics*. New York, NY: Springer, 2008, pp. 337–339. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_251

[72] A. Mao, M. Mohri, and Y. Zhong, "Cross-Entropy Loss Functions: Theoretical Analysis and Applications," Jun. 2023, arXiv:2304.07288 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2304.07288

[73] T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," Mar. 2020, arXiv:2003.05689 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2003.05689

[74] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[75] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization."

[76] D. S. Soper, "Hyperparameter Optimization Using Successive Halving with Greedy Cross Validation," *Algorithms*, vol. 16, no. 1, p. 17, Jan. 2023, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1999-4893/16/1/17

[77] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565

[78] T. Fawcett, "Introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, Jun. 2006.

[79] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990, conference Name: Proceedings of the IEEE.

[80] M. Attik, L. Bougrain, and F. Alexandre, *Self-organizing Map Initialization*, Sep. 2005, pages: 362.

[81] J. Vesanto, "Data Exploration Process Based on the Self-Organizing Map," iSBN: 9789512258970. [Online]. Available: http://lib.tkk.fi/Diss/2002/isbn9512258978/

[82] D. Polani, "Measures for the organization of self-organizing maps," in *Self-Organizing Neural Networks: Recent Advances and Applications*, ser. Studies in Fuzziness and Soft Computing, U. Seiffert and L. C. Jain, Eds. Physica-Verlag HD, pp. 13–44. [Online]. Available: https://doi.org/10.1007/978-3-7908-1810-9_2

[83] "(6) (PDF) Neighborhood Preservation in Nonlinear Projection Methods: An Experimental Study." [Online]. Available: https://www.researchgate.net/publication/2365851_Neighborhood_Preservation_in_Nonlinear_Projection_Methods_An_Experimental_Study

[84] J. Ortigosa-Hernández, I. Inza, and J. A. Lozano, "Measuring the class-imbalance extent of multi-class problems," *Pattern Recognition Letters*, vol. 98, pp. 32–38, Oct. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786551730257X

[85] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002.

[86] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1322–1328, iSSN: 2161-4407. [Online]. Available: https://ieeexplore.ieee.org/document/4633969

[87] M. Beckmann, N. F. F. Ebecken, and B. S. L. Pires de Lima, "A KNN Undersampling Approach for Data Balancing," *Journal of Intelligent Learning Systems and Applications*, vol. 07, no. 04, pp. 104–116, 2015. [Online]. Available: http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jilsa.2015.74010

[88] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004. [Online]. Available: https://doi.org/10.1145/1007730.1007735

[89] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing Training Data for Automated Annotation of Keywords: a Case Study," in *WOB*, 2003.

[90] O. Kalinagac, W. Soussi, and G. Gür, "Graph Based Liability Analysis for the Microservice Architecture," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022.

[91] X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, "DAGs with NO TEARS: Continuous Optimization for Structure Learning," 2018. [Online]. Available: https://arxiv.org/abs/1803.01422

[92] J. Ortiz, R. Sanchez-Iborra, J. B. Bernabe, A. Skarmeta, C. Benzaid, T. Taleb, P. Alemany, R. Muñoz, R. Vilalta, C. Gaber, J.-P. Wary, D. Ayed, P. Bisson, M. Christopoulou, G. Xilouris,

E. M. de Oca, G. Gür, G. Santinelli, V. Lefebvre, A. Pastor, and D. Lopez, "Inspire-5gplus: Intelligent security and pervasive trust for 5g and beyond networks," ser. ARES '20. New York, NY, USA: Association for Computing Machinery, 2020.

[93] "User group; quality of ict services; part 3: Template for service level agreements (sla)."

[94] L.-J. Jin, V. Machiraju, and A. Sahai, "Analysis on service level agreement of web services," Jan. 2002.

[95] Y. Kouki and T. Ledoux, "CSLA : a Language for improving Cloud SLA Management," Apr. 2012, p. 586. [Online]. Available: https://hal.science/hal-00675077

[96] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of Quality of Experience of Video-on-Demand Services: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[97] Y. Bo, Y. Lin, and M. L. Ru, "Quality of Protection in Web Service: An Overview," in *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, Oct. 2011, pp. 495–498.

[98] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, *Web Service Level Agreement (WSLA) Language Specification*, Jan. 2003.

[99] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web services agreement specification (ws-agreement)," *Global Grid Forum*, vol. 2, 01 2004.

[100] B. Koller, H. Frutos, and G. Laria, "Service Level Agreements in BREIN," Aug. 2010, pp. 157–165.

[101] A. T. Wonjiga, L. Rilling, and C. Morin, "Defining Security Monitoring SLAs in IaaS Clouds: the Example of a Network IDS."

[102] D. D. Lamanna, J. Skene, and W. Emmerich, *SLAng: A language for defining service level agreements*, Jun. 2003.

[103] A. Paschke, "RBSLA A declarative Rule-based Service Level Agreement Language based on RuleML," in *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, vol. 2, Nov. 2005, pp. 308–314.

[104] G. Governatori and A. Rotolo, "Modelling Contracts Using RuleML," Jan. 2004.

[105] M. Torkashvan and H. Haghighi, "CSLAM: A framework for cloud service level agreement management based on WSLA," in *6th International Symposium on Telecommunications (IST)*, Nov. 2012, pp. 577–585.

[106] G. Dobson, R. Lock, I. Sommerville, and Ian Sommerville, "QoSOnt: a QoS Ontology for Service-Centric Systems," in *31st EUROMICRO Conference on Software Engineering and Advanced Applications*. Porto, Portugal: IEEE, 2005, pp. 80–87. [Online]. Available: http://ieeexplore.ieee.org/document/1517730/

[107] G. Chen, X. Bai, X. Huang, M. Li, and L. Zhou, "Evaluating services on the cloud using ontology QoS model," in *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, Dec. 2011, pp. 312–317.

[108] K. P. Joshi and C. Pearce, "Automating Cloud Service Level Agreements Using Semantic Technologies," in *2015 IEEE International Conference on Cloud Engineering*. Tempe, AZ, USA: IEEE, Mar. 2015, pp. 416–421. [Online]. Available: http://ieeexplore.ieee.org/document/7092954/

[109] T. Labidi, A. Mtibaa, and H. Brabra, "CSLAOnto: A Comprehensive Ontological SLA Model in Cloud Computing," *Journal on Data Semantics*, vol. 5, Sep. 2016.

[110] M. Fernandez, A. Gomez-Pearez, and N. Juristo, "Methontology: From Ontological Art Towards Ontological Engineering," p. 8.

[111] Y. Kouki, "Approche dirigée par les contrats de niveaux de service pour la gestion de l'élasticité du "nuage"."

[112] P. Karaenke, A. Micsik, and S. Kirn, "Adaptive SLA Management along Value Chains for Service Individualization," Jan. 2009.

[113] G. Di Modica, V. Regalbuto, O. Tomarchio, and L. Vita, "Dynamic re-negotiations of SLA in service composition scenarios," in *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)*, Aug. 2007, pp. 359–366, iSSN: 2376-9505. [Online]. Available: https://ieeexplore.ieee.org/document/4301099

[114] L. Bodenstaff, A. Wombacher, M. Reichert, and M. C. Jaeger, "Monitoring Dependencies for SLAs: The MoDe4SLA Approach," in *2008 IEEE International Conference on Services Computing*, vol. 1, Jul. 2008, pp. 21–29. [Online]. Available: https://ieeexplore.ieee.org/document/4578445

[115] G. Spanoudakis, C. Kloukinas, and K. Mahbub, "The runtime monitoring framework of SERENITY," in *Information Security Series*, Mar. 2009, vol. 45, pp. 213–237, journal Abbreviation: Information Security Series.

[116] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, *Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments*, Jun. 2010, journal Abbreviation: Proceedings of the 2010 International Conference on High Performance Computing and Simulation, HPCS 2010 Pages: 54 Publication Title: Proceedings of the 2010 International Conference on High Performance Computing and Simulation, HPCS 2010.

[117] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "Translating Service Level Objectives to lower level policies for multi-tier services," *Cluster Computing*, vol. 11, pp. 299–311, Sep. 2008.

[118] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E^3: A Multiobjective Optimization Framework for SLA-Aware Service Composition," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 358–372, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/5710869/

[119] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A Firmware Update Architecture for Internet of Things," Internet Engineering Task Force, Request for Comments RFC 9019, Apr. 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-suit-architecture

[120] B. Moran, H. Tschofenig, and H. Birkholz, "A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices," Internet Engineering Task Force, Request for Comments RFC 9124, Jan. 2022. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-suit-information-model

[121] "Securing Home IoT Devices Using MUD \textbar NCCoE." [Online]. Available: https://www.nccoe.nist.gov/projects/securing-home-iot-devices-using-mud

[122] "Good Practices for Security of Internet of Things in the context of Smart Manufacturing." [Online]. Available: https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot

[123] O. Garcia-Morchon, S. Kumar, and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges," Internet Engineering Task Force, Request for Comments RFC 8576, Apr. 2019. [Online]. Available: https://datatracker.ietf.org/doc/rfc8576

[124] "Network functions virtualisation (nfv) release 3; management and orchestration; network service templates specification."

[125] "Trust evaluation in cloud based on friends and third party's recommendations | IEEE Conference Publication | IEEE Xplore." [Online]. Available: https://ieeexplore.ieee.org/document/6799600

[126] S. Rizvi, J. Ryoo, Y. Liu, D. Zazworsky, and A. Cappeta, "A centralized trust model approach for cloud computing," in *2014 23rd Wireless and Optical Communication Conference (WOCC)*, May 2014, pp. 1–6, iSSN: 2379-1276. [Online]. Available: https://ieeexplore.ieee.org/document/6839923

[127] Y. Wang and J. Vassileva, "Toward Trust and Reputation Based Web Service Selection: A Survey," *International Transactions on Systems Science and Applications - ITSSA*, vol. 3, Jan. 2007.

[128] J. Borowski, K. Hopkinson, J. Humphries, and B. Borghetti, "Reputation-based trust for a cooperative agent-based backup protection scheme," in *2012 IEEE Power and Energy Society General Meeting*, Jul. 2012, pp. 1–1, iSSN: 1944-9925.

[129] K. Papadakis-Vlachopapadopoulos, R. S. González, I. Dimolitsas, D. Dechouniotis, A. J. Ferrer, and S. Papavassiliou, "Collaborative SLA and reputation-based trust management in cloud federations," *Future Generation Computer Systems*, vol. 100, pp. 498–512, Nov. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167739X18329248

[130] S. Habib, S. Ries, and M. Mühlhäuser, *Towards a Trust Management System for Cloud Computing*, Nov. 2011.

[131] T. H. Noor, Q. Z. Sheng, L. Yao, S. Dustdar, and A. H. Ngu, "CloudArmor: Supporting Reputation-Based Trust Management for Cloud Services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 367–380, Feb. 2016, conference Name: IEEE Transactions on Parallel and Distributed Systems.

[132] A. Kanwal, R. Masood, and M. A. Shibli, "Evaluation and establishment of trust in cloud federation," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2557977.2558023

[133] M. Alhamad, T. Dillon, and E. Chang, "SLA-Based Trust Model for Cloud Computing," in *2010 13th International Conference on Network-Based Information Systems*, Sep. 2010, pp. 321–324, iSSN: 2157-0426.

[134] S. Chakraborty and K. Roy, "An SLA-based Framework for Estimating Trustworthiness of a Cloud," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Jun. 2012, pp. 937–942, iSSN: 2324-9013.

[135] V. Chang, J. Sidhu, S. Singh, and R. Sandhu, "SLA-based Multi-dimensional Trust Model for Fog Computing Environments," *Journal of Grid Computing*, vol. 21, no. 1, p. 4, Dec. 2022.

[136] P. S. Pawar, M. Rajarajan, S. K. Nair, and A. Zisman, "Trust Model for Optimized Cloud Services," in *Trust Management VI*, T. Dimitrakos, R. Moona, D. Patel, and D. H. McKnight, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 374, pp. 97–112, series Title: IFIP Advances in Information and Communication Technology. [Online]. Available: http://link.springer.com/10.1007/978-3-642-29852-3_7

[137] T. G. Rodrigues, "Cloudacc: a cloud-based accountability framework for federated cloud," Sep. 2016. [Online]. Available: https://repositorio.ufpe.br/handle/123456789/18590

[138] R. K. L. Ko, M. Kirchberg, and B. S. Lee, "From system-centric to data-centric logging - Accountability, trust & security in cloud computing," in *2011 Defense Science Research Conference and Expo (DSR)*, Aug. 2011, pp. 1–4.

[139] "ETSI GS NFV-REL 005 : Network functions virtualisation (nfv); accountability; report on quality accountability framework."

[140] M. Scholler, R. Bless, F. Pallas, J. Horneber, and P. Smith, "An Architectural Model for Deploying Critical Infrastructure Services in the Cloud," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*. Bristol, United Kingdom: IEEE, dec 2013, pp. 458–466. [Online]. Available: http://ieeexplore.ieee.org/document/6753832/

[141] Andrés, "cloudcompaas-common," Sep. 2013, original-date: 2013-09-10T11:21:00Z. [Online]. Available: https://github.com/angarg12/cloudcompaas-common

[142] A. García García, I. Blanquer Espert, and V. Hernández García, "SLA-driven dynamic cloud resource management," *Future Generation Computer Systems*, vol. 31, pp. 1–11, Feb. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1300215X

[143] V. C. Emeakaroha, T. C. Ferreto, M. A. S. Netto, I. Brandic, and C. A. F. De Rose, "CASViD: Application Level Monitoring for SLA Violation Detection in Clouds," in *2012 IEEE 36th Annual Computer Software and Applications Conference*. Izmir, Turkey: IEEE, Jul. 2012, pp. 499–508. [Online]. Available: http://ieeexplore.ieee.org/document/6340204/

[144] O. Adinolfi, R. Cristaldi, L. Coppolino, and L. Romano, "QoS-MONaaS: A Portable Architecture for QoS Monitoring in the Cloud," in *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*. Naples: IEEE, Nov. 2012, pp. 527–532. [Online]. Available: http://ieeexplore.ieee.org/document/6395139/

[145] J. Bendriss, I. G. B. Yahia, P. Chemouil, and D. Zeghlache, "AI for SLA Management in Programmable Networks," 2017.

[146] I. U. Haq, R. Alnemr, A. Paschke, E. Schikuta, H. Boley, and C. Meinel, "Distributed Trust Management for Validating SLA Choreographies," in *Grids and Service-Oriented Architectures for Service Level Agreements*, P. Wieder, R. Yahyapour, and W. Ziegler, Eds. Boston, MA: Springer US, 2010, pp. 45–55.

[147] K. Voss, K. Djemame, I. Gourlay, and J. Padgett, "AssessGrid, Economic Issues Underlying Risk Awareness in Grids," in *Grid Economics and Business Models*, D. J. Veit and J. Altmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4685, pp. 170–175, iSSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-540-74430-6_14

[148] S. Kim, S. Park, Y. Kim, S. Kim, and K. Lee, "VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV," *Cluster Computing*, vol. 20, no. 3, pp. 2107–2117, Sep. 2017.

[149] O. Soualah, M. Mechtri, C. Ghribi, and D. Zeghlache, *Energy Efficient Algorithm for VNF Placement and Chaining*, May 2017.

[150] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining." [Online]. Available: http://arxiv.org/abs/1710.02611

[151] N. Joy, K. Chandrasekaran, and Binu A., "Energy Aware SLA with Classification of Jobs for Cloud Environment," *Procedia Computer Science*, vol. 70, pp. 740–747, 2015.

[152] S. Goyal, S. Bawa, and B. Singh, "Green Service Level Agreement (GSLA) framework for cloud computing," *Computing*, vol. 98, no. 9, pp. 949–963, Sep. 2016.

[153] G. von Laszewski and L. Wang, "GreenIT Service Level Agreements," in *Grids and Service-Oriented Architectures for Service Level Agreements*, Aug. 2010, pp. 77–88.

[154] I. Ahmed, H. Okumura, and K. Arai, "Analysis on Existing Basic Slas and Green Slas to Define New Sustainable Green SLA," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 12, p. 9, 2015.

[155] "Etsi es 203 539 : Environmental engineering (ee); measurement method for energy efficiency of network functions virtualisation (nfv) in laboratory environment."

[156] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang, "A survey of network virtualization techniques for internet of things using sdn and nfv," *ACM Comput. Surv.*, vol. 53, no. 2, apr 2020.

[157] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2018.

[158] L. Atzori, J. L. Bellido, R. Bolla, G. Genovese, A. Iera, A. Jara, C. Lombardo, and G. Morabito, "SDN&NFV contribution to IoT objects virtualization," *Comput. Networks*, vol. 149, pp. 200–212, Feb 2019.

[159] Y.-M. Hung, S.-C. Chien, and Y.-Y. Hsu, "Orchestration of nfv virtual applications based on tosca data models," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 219–222.

[160] A. C. F. da Silva, U. Breitenbücher, K. Képes, O. Kopp, and F. Leymann, "OpenTOSCA for IoT: Automating the Deployment of IoT Applications based on the Mosquitto Message Broker," in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT'16. Association for Computing Machinery, Nov. 2016, pp. 181–182, event-place: New York, NY, USA.

[161] F. Li, M. Vögler, M. Claeßens, and S. Dustdar, "Towards automated iot application deployment by a cloud-based approach," in *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, 2013, pp. 61–68.

[162] E. W. Dijkstra, in *Selected Writings on Computing: A Personal Perspective*.

[163] C. Sauvanaud, M. Kaâniche, K. Kanoun, K. Lazri, and G. Da Silva Silvestre, "Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned," *Journal of Systems and Software*, vol. 139, pp. 84–106, May 2018.

[164] "Resources." [Online]. Available: https://smartmeterrpm.com/resources/

[165] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as MUD: Generating, validating and applying IoT behaviorial profiles (technical report)." [Online]. Available: http://arxiv.org/abs/1804.04358

[166] AWS IoT device management SLA.

[167] Content delivery network docs.

[168] "Cloud Edge Services & Edge Network Management \textbar Equinix." [Online]. Available: https://www.equinix.com/services/digital-infrastructure-services/network-edge

[169] S. N. Matheu, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, "A Survey of Cybersecurity Certification for the Internet of Things," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–36, 2 2021.

[170] D. Marutho, S. Hendra Handaka, E. Wijaya, and Muljono, "The determination of cluster number at k-mean using elbow method and purity evaluation on headline news," in *2018 International Seminar on Application for Technology of Information and Communication*, 2018, pp. 533–538.

[171] F. Forest, M. Lebbah, H. Azzag, and J. Lacaille, "A Survey and Implementation of Performance Metrics for Self-Organized Maps," Nov. 2020. [Online]. Available: http://arxiv.org/abs/2011.05847

[172] "PacketFabric's Terms and Conditions." [Online]. Available: https://packetfabric.com/terms-and-conditions

[173] S. v. Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate Imputation by Chained Equations in R," *Journal of Statistical Software*, vol. 45, pp. 1–67, Dec. 2011. [Online]. Available: https://doi.org/10.18637/jss.v045.i03

[174] M. Carreira-Perpiñán and W. Wang, "The K-modes algorithm for clustering," Apr. 2013. [Online]. Available: http://arxiv.org/abs/1304.6478

[175] "Hierarchical cluster analysis - cecil c. bridges, 1966." [Online]. Available: https://journals.sagepub.com/doi/10.2466/pr0.1966.18.3.851

[176] N. Rougier and Y. Boniface, "Dynamic self-organising map," *Neurocomputing*, vol. 74, no. 11, pp. 1840–1847, May 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231211000713

Appendix A

TRAILS Grammar

Listing A.1: TRAILS Grammar

```
1
2tosca_definitions_version: tosca_simple_yaml_1_2
3description: TRAILS types definitions version 0.1
4metadata:
5  template_name: TRAILS_types
6  template_author: Orange
7  template_version: 0.1
8
9
10
11data_types:
12
13  tosca.datatypes.nfv.direction:
14    description: Identify which way are we talking about
15    derived_from: string
16    constraints:
17      - valid_values: [to-device, from-device]
18
19  tosca.datatypes.nfv.mud-grouping:
20    description: MUD related information, as specified by RFC-8520.
21    properties:
22      mud-version:
23        description: This is the version of the MUD specification. This memo
              specifies version 1.
24        type: string
25        required: true
26      mud-url:
27        description: This is the MUD URL associated with the entry found in a MUD
              file.
28        type: string
29        required: true
30      last-update:
31        description: This is intended to be when the current MUD file was generated.
               MUD Managers SHOULD NOT checkfor updates between this time plus cache
              validity
32        type: timestamp
33        required: true
34      mud-signature:
35        description: A URI that resolves to a signature as described in this
              specification.
36        type: string
37        required: false
38      cache-validity:
39        description: The information retrieved from the MUD server is valid for
```

```
            these many hours, after which it should be refreshed. N.B. MUD manager
            implementations need not discard MUD files beyond this period.
40        type: integer
41        constraints:
42          - in_range: [1,168]
43        required: false
44        default: 48
45      is-supported:
46        description: This boolean indicates whether or not the Thing is currently
            supported by the manufacturer.
47        type: boolean
48        required: true
49      systeminfo:
50        description: A UTF-8 description of this Thing. This should be a brief
            description that may be displayed to the user to determine whether to
            allow the Thing on the network.
51        type: string
52        required: false
53      mfg-name:
54        description: Manufacturer name, as described in the ietf-hardware YANG
            module.
55        type: string
56        required: false
57      model-name:
58        description: Model name, as described in the ietf-hardware YANG module.
59        type: string
60        required: false
61      firmware-rev:
62        description: firmware-rev, as described in the ietf-hardware YANG module.
            Note this field MUST NOT be included when the device can be updated but
            the MUD-URL cannot.
63        type: string
64        required: false
65      software-rev:
66        description: software-rev, as described in the ietf-hardware YANG module.
            Note this field MUST NOT be included when the device can be updated but
            the MUD-URL cannot.
67        type: string
68        required: false
69      documentation:
70        description: This URL points to documentation that relates to this device
            and any classes that it uses in its MUD file. A caution MUD managers
            need not resolve this URL on their own, but rather simply
71          provide it to the administrator. Parsing HTML is not an intended function
              of a MUD manager.
```

```
72        type: string
73        required: false
74      extensions:
75        description: A list of extension names that are used in this MUD file. Each
              name is registered with the IANA and described in an RFC.
76        type: list
77        entry_schema:
78          type: string
79          constraints:
80            - min_length: 1
81            - max_length: 40
82      from-device-policy:
83        description: The policies that should be enforced on traffic coming from the
              device. These policies are not necessarily intended to be enforced at a
              single point, but may be rendered by the controller to any relevant
              enforcement points in the network or elsewhere.
84        type: tosca.datatypes.nfv.access-lists
85      to-device-policy:
86        description: The policies that should be enforced on traffic going to the
              device. These policies are not necessarily intended to be enforced at a
              single point, but may be rendered by the controller to any relevant
              enforcement points in the network or elsewhere.
87        type: tosca.datatypes.nfv.access-lists
88
89  tosca.datatypes.nfv.access-list:
90    description: Each entry on this list refers to an ACL that should be present in
          the overall access list data model. Each ACL is identified by name and type.
91    properties:
92      name:
93        description: The name of the ACL for this entry.
94        type: string
95        required: false
96
97  tosca.datatypes.nfv.access-lists:
98    description: The access lists that should be applied to traffic to or from the
          device.
99    properties:
100     access-list:
101       description: Each entry on this list refers to an ACL that should be present
              in the overall access list data model. Each ACL is identified by name
              and type.
102       type: list
103       entry_schema: tosca.datatypes.nfv.access-list
104
105 tosca.datatypes.nfv.mud:
```

```
106   description: MUD-specific matches.
107   properties:
108     manufacturer:
109       description: A domain that is intended to match the authority section of the
              MUD URL. This node is used to specify one or more manufacturers a
             device should be authorized to access.
110       type: string
111       required: false
112     same-manufacturer:
113       description:
114         This node matches the authority section of the MUD URL
115         of a Thing. It is intended to grant access to all
116         devices with the same authority section.
117       type: string
118       required: false
119     model:
120       description: Devices of the specified model type will match if they have an
             identical MUD URL.
121       type: string
122       required: false
123     local-networks:
124       description: IP addresses will match this node if they are considered local
             addresses. A local address may be a list of locally defined prefixes and
              masks that indicate a particular administrative scope.
125       type: string
126       required: false
127     controller:
128       description: This node names a class that has associated with it zero or
             more IP addresses to match against. These may be scoped to a manufacturer
              or via a standard URN.
129       type: string
130       required: false
131     my-controller:
132       description: This node matches one or more network elements that have been
             configured to be the controller for this Thing, based on its MUD URL.
133       type: string
134       required: false
135
136 tosca.datatypes.nfv.matches:
137   description: adding abstractions to avoid need of IP addresses
138   properties:
139     mud:
140       description: MUD-specific matches.
141       type: tosca.datatypes.nfv.mud
142
```

```
143 tosca.datatypes.nfv.MUD_profil:
144   description: #
145   properties:
146     name:
147       description :Name of the mud profil file
148       required :true
149     direction:
150       description: Identify which way are we talking about
151       type: tosca.datatypes.nfv.direction
152       required :false
153     mud-grouping:
154       description: MUD related information, as specified by RFC-8520.
155       type: tosca.datatypes.nfv.mud-grouping
156       required: false
157     access-lists :
158       description: The access lists that should be applied to traffic to or from
             the device.
159       type :tosca.datatypes.nfv.access-lists
160       required: false
161     mud:
162       description :MUD-specific matches
163       type :tosca.datatypes.nfv.mud
164       required: false
165     matches:
166       description: adding abstractions to avoid need of IP addresses
167       type: tosca.datatypes.nfv.matches
168       required: false
169
170
171
172
173 tosca.datatypes.nfv.author:
174     derived_from: tosca.datatypes.Root
175     description: Give a set of information about the author involved in the
          creation of the manifest.
176     properties:
177       name:
178         type: string
179         description:
180         required: true
181       role:
182         type: string
183         description:
184         required: true
185       country:
```

```
186         type: string
187         description:
188         required: true
189       mail :
190         type :string
191         required :false
192       url:
193         type: string
194         description:
195         required: false
196
197
198 tosca.datatypes.nfv.header:
199   derived_from: tosca.datatypes.Root
200   description: Hold manifest's metadata.
201   properties:
202     title:
203       type: string
204       description: Title of the manifest
205       required: false
206     lead_author:
207       type: tosca.datatypes.nfv.author
208       description: Identifying the author who is responsible for the present
              manifest.
209       required: true
210     version:
211       type: string
212       description: Version of the present manifest. MUST be modified at each
              update.
213       required: true
214     url:
215       type: string
216       description: Url where the last updates of the manifest can be found.
217       required: false
218     last_update:
219       type: timestamp
220       description: Date and time where the manifest was last updated. MUST be
              modified at each update.
221       required: false
222     cache-validity:
223       type: integer
224       description: The period of time in hours that a network management station
              MUST wait since its last retrieval before checking for an update.
225       required: false
226     system_info:
```

```
227       type: string
228       description: Humain readable description of the component described by the
              present manifest.
229       required: false
230     component_type:
231       type: string
232       description: Describes the type of Component. A SimpleObject is a physical
              component without hosting capacities which exposes fixed ressources (e.g.
               an IoT device) or HostingObject or VNF.
233       required: true
234     model:
235       type: string
236       description: Identifies the model or the version of the Component.
237       required: true
238
239 tosca.datatypes.nfv.validation:
240   derived_from: tosca.datatypes.Root
241   description: Give a set of information about the validation process.
242   properties:
243     author:
244       type: tosca.datatypes.nfv.author
245       description: Specify the entity responsable for the validation process.
246       required: true
247     last_validation:
248       type: timestamp
249       description: Specify the date-time of the last validation.
250       required: true
251     validation_compliance :
252       type: string
253       description: Security score asigned to the component.
254       required: false
255     validation_scope:
256       type: string
257       required: false
258
259
260 tosca.datatypes.nfv.documentation:
261   derived_from: tosca.datatypes.Root
262   description: #add description
263   properties:
264     doc_type:
265       type: string
266       description: Indicates which type of documentation is defined
267       required: true
268       constraints:
```

```
269         - valid_values: ["openAPI","lwm2m-model", "test-file", "API-guide", "
                Features", "SUIT-manifest", "User-guide", "Datasheet", "Deployment-
                guides"]
270     openAPI:
271       type: tosca.datatypes.nfv.openAPI
272       description: Machine-readable interface file for describing, producing,
              consuming, and visualizing RESTful service.
273       required: false
274     lwm2m-model:
275       type: tosca.datatypes.nfv.lwm2m-model
276       description: # add description
277       required: false
278     suit-manifest:
279       type: tosca.datatypes.nfv.suit-manifest
280       description: # add description
281       required: false
282     url:
283       type: string
284       description: Url points to the documentation.
285       required: false
286     doc_description:
287       type: string
288       description: Humain-readable description of the documentation provided.
289       required: false
290
291
292 tosca.datatypes.nfv.performanceMetric:
293   derived_from: tosca.datatypes.Root
294   description: Contains information about the measurment performance method.
295   properties:
296     name:
297       type: string
298       description: The name of the performance performance metric
299       required: false
300     interface:
301       type: string
302       description: The interface used to calcul the performance metric.
303       required: false
304     metric:
305       type: string
306       description: The metric used to calcul the performance.
307       required: false
308     SLI:
309       type: string
310       description: Reference to the service level indicator of the metric
```

```
      required: false


tosca.datatypes.nfv.slo:
  derived_from:
  description:
  properties:
    slo_name:
      type :string
      description :Name of the SLO
      required :true
    slo_type:
      type: string
      description : Type of the SLO
      required: true
      constraints:
        - valid_values :[energy,communication,availability,cpu_speed]
    slo_min_value:
      type :float
      description: Min value of the SLO
      required: true
    slo_max_value :
      type :float
      description: Max value of the SLO
      required :true

tosca.datatypes.nfv.sla:
  derived_from: tosca.datatypes.Root
  description: Contains information about the service level agreement
  properties:
    sla_name:
      type: string
      description: Name of the SLA
      required: true
    sla_model:
      type: string
      description: The type of the SLA
      required: true
      constraints:
        - valid_values :['WS-Agreement','ETSI-SLA-Model']
    slo:
      type: list
      description: Identifies the service level objectives commits to deliver
      required :true
      entry_schema:
```

```
356         type: tosca.datatypes.nfv.slo
357
358
359
360
361
362
363 tosca.datatypes.nfv.commitment:
364   derived_from: tosca.datatypes.Root
365   description: Give a set of information about the property of the component.
366   properties:
367       title:
368         type: string
369         description: The title of the property description. MUST be unique.
370         required: true
371       author:
372         type: tosca.datatypes.nfv.author
373         description: Identify the author of the property description.
374         required: true
375       documentations:
376         type: list
377         description: Identifies the documentation files of the component.
378         required: false
379         entry_schema :
380           type: tosca.datatypes.nfv.documentation
381       performanceMetrics:
382         type: list
383         description: Identifies the performance metrics of the component.
384         required: false
385         entry_schema:
386           type: tosca.datatypes.nfv.performanceMetric
387       sla:
388         type: list
389         description: Identifies the service level agreement commits to deliver.
390         required: false
391         entry_schema:
392           type: tosca.datatypes.nfv.sla
393
394
395 tosca.datatypes.nfv.recommendedPatch:
396   derived_from: tosca.datatypes.Root
397   description: Give recommendations in terms of vulnerability and weakness
        correction.
398   properties:
399     name:
```

```
400       type: string
401       description: Name of the patch.
402       required: true
403     patch_type:
404       type: string
405       description: Type of the patch.
406       required: false
407       constraints:
408         - valid_values: ["vulnerability_patch","weakness_patch"]
409     patch_description:
410       type: string
411       description: Humain-readable description of the vulnerability or the
              weakness.
412       required: false
413     risk:
414       type: string
415       description: The level of risk the the device is exposed to.
416       required: false
417       constraints:
418         - valid_values: ["low", "medium", "high"]
419     patch:
420       type: string
421       description: Url points to the patch.
422       required: true
423
424
425 tosca.datatypes.nfv.VNFD:
426   derived_from: tosca.datatypes.Root
427   description: # add description
428   properties:
429     descriptor_id:
430       type: string
431       description: Globally unique identifier of the VNFD
432       required: true
433
434
435 tosca.datatypes.nfv.NSD:
436   derived_from: tosca.datatypes.Root
437   properties:
438     descriptor_id:
439       type: string
440       description: Identifier of this NS descriptor
441       required: true
442
443 tosca.datatypes.nfv.VNF:
```

```
444   derived_from: tosca.nodes.Root
445   description: The generic abstract type from which all VNF specific node types
          shall be derived to form, together with other node types, the TOSCA service
          template(s) representing the VNFD
446   properties:
447     descriptor_id: # instead of vnfd_id
448      type: string # UUID
449      description: Identifier of this VNFD information element. This attribute
             shall be globally unique
450      required: true
451     descriptor_version: # instead of vnfd_version
452      type: string
453      description: Identifies the version of the VNFD
454      required: true
455     provider: # instead of vnf_provider
456      type: string
457      description: Provider of the VNF and of the VNFD
458      required: true
459     product_name: # instead of vnf_product_name
460      type: string
461      description: Human readable name for the VNF Product
462      required: true
463     software_version: # instead of vnf_software_version
464      type: string
465      description: Software version of the VNF
466      required: true
467     product_info_name: # instead of vnf_product_info_name
468      type: string
469      description: Human readable name for the VNF Product
470      required: false
471     product_info_description: # instead of vnf_product_info_description
472      type: string
473      description: Human readable description of the VNF Product
474      required: false
475     vnfm_info:
476      type: list
477      required: true
478      description: Identifies VNFM(s) compatible with the VNF
479      entry_schema:
480       type: string
481     localization_languages:
482      type: list
483      description: Information about localization languages of the VNF
484      required: false
485      entry_schema:
```

```
      type: string #IETF RFC 5646string
    default_localization_language:
     type: string #IETF RFC 5646string
     description: Default localization language that is instantiated if no
         information about selected localization language is available
     required: false
    lcm_operations_configuration:
     type: tosca.datatypes.nfv.VnfLcmOperationsConfiguration
     description: Describes the configuration parameters for the VNF LCM
         operations
     required: false
    monitoring_parameters:
     type: map # key: id
     entry_schema:
      type: tosca.datatypes.nfv.VnfMonitoringParameter
     description: Describes monitoring parameters applicable to the VNF.
     required: false
    flavour_id:
     type: string
     description: Identifier of the Deployment Flavour within the VNFD
     required: true
    flavour_description:
     type: string
     description: Human readable description of the DF
     required: true
    vnf_profile:
     type: tosca.datatypes.nfv.VnfProfile
     description: Describes a profile for instantiating VNFs of a particular NS DF
          according to a specific VNFD and VNF DF
     required: false

tosca.datatypes.nfv.dependencies:
  derived_from: tosca.datatypes.Root
  description: #add description
  properties:
    name:
      type: string
      description: Name of the dependencies.
      required: false
    dependency_type:
      type: string
      description: Type of the dependencies.
      required: true
      constraints:
        - valid_values: ["Hardware" ,"Software"]
```

```
528     dependency_description:
529       type: string
530       description: Humain-readable description of the dependency.
531       required: false
532     virtual_network_interface_requirements:
533       type: tosca.datatypes.nfv.VirtualNetworkInterfaceRequirements
534       description: Describes requirements on a virtual network interface.
535       required: false
536     virtual_memory:
537       type: tosca.datatypes.nfv.VirtualMemory
538       description: Supports the specification of requirements related to virtual
              memory of a virtual compute resource.
539       required: false
540     virtual_cpu_pinning:
541       type: tosca.datatypes.nfv.VirtualCpuPinning
542       description: Supports the specification of requirements related to the
              virtual CPU pinning configuration of a virtual compute resource.
543       required: false
544     virtual_cpu:
545       type: tosca.datatypes.nfv.VirtualCpu
546       description: Supports the specification of requirements related to virtual
              CPU(s) of a virtual compute resource.
547       required: false
548     vnf:
549       type: tosca.datatypes.nfv.VNF
550       description: # add description
551       required: false
552
553 tosca.datatypes.nfv.composition:
554   derived_from: tosca.datatypes.Root
555   description: #add description
556   properties:
557     connectivity:
558       type: string
559       description: Specifies the type of connectivity required for the link.
560       required: false
561     max_instance:
562       type: integer
563       description: Specifies the maximum number of instances that can be included
              in the composition.
564       required: false
565     Connectivity_type:
566       type: tosca.datatypes.nfv.ConnectivityType
567       description: Describes additional connectivity information of a virtualLink.
568       required: false
```

```
    NSD:
      type: tosca.datatypes.nfv.NSD
      description:  # add description
      required: false

tosca.datatypes.nfv.networkBehavior:
  derived_from: tosca.datatypes.Root
  description: #add description
  properties:
    file_type:
      type: string
      description: Indicates the type of the file.
      required: true
      constraints:
        - valid_values: ["MUD profil", "VNFD"]
    MUD_profil:
      type: tosca.datatypes.nfv.MUD_profil
      required: false
    VNFD:
      type: tosca.datatypes.nfv.VNFD
      required: false


tosca.datatypes.nfv.usageRecommendation:
  derived_from: tosca.datatypes.Root
  description: Contains recommendation on term of security, resources required and
      dependencies.
  properties:
    title:
      type: string
      description: The title of the usage description. MUST be unique.
      required: true
    author:
      type: tosca.datatypes.nfv.author
      description: Identify the author of the usage description.
      required: true
    dependencies:
      type: list
      description: Give description about the dependencies needed by the device in
          ordre to operate.
      required: false
      entry_schema :
         type: tosca.datatypes.nfv.dependencies
    recommended_patchs:
      type: list
```

```
612       description: Give recommendations in terms of vulnerability and weakness
              correction.
613       required: false
614       entry_schema:
615         type: tosca.datatypes.nfv.recommendedPatch
616     compositions:
617       type: list
618       description: specify the requirements for the composition of several
              components described in the present manifest.
619       required: false
620       entry_schema:
621         type: tosca.datatypes.nfv.composition
622     networkBehavior:
623       type: tosca.datatypes.nfv.networkBehavior
624       description: contains a description of the network behavior of the device.
625       required :false
626
627######################
628
629capability_types:
630  tosca.capabilities.nfv.exposedSecurityService:
631    derived_from: tosca.capabilities.Node
632    description: Describes the capabilities related to the security service.
633    properties:
634      security_type:
635        type: string
636        description: The type of security service
637        required: false
638        constraints:
639          - valid_values: [confidentiality, integrity,authentication,"DDOS
                protection", "signature-generation", "signature-verification", Antispam
                , Antivirus, DLP, DPI, Honeypot, Identifies, IPS, NAT, "Packet Filter
                Firewall", "Parental Control", "VPN Gateway", WAF, "zero-day protection
                ", "Signature-based detection"]
640      keys:
641        type: string
642        description: Key type
643        required: false
644        constraints:
645          - valid_values: [EC, RSA, oct]
646      algorithm:
647        type: string
648        description: Algorithm used
649        required: false
650        constraints:
```

```
651         -valid_values: [SHA256, SHA384, RSA256, RSA384, RSA512, AES256]
652     length:
653       type: string
654       description: Key length in bits.
655       required: false
656     key_ops:
657       type: string
658       description: Key operations.
659       required: false
660       constraints:
661         -valid_values: [sign,verify,encrypt,decrypt,wrapKey,unwrapKey,deriveKey,
              deriveBits]
662
663 tosca.capabilities.nfv.domainEvidenceCollector:
664   derived_from: tosca.capabilities.Node
665   description: Describes the capabilities related to domain evidence collector
666   properties:
667     rate:
668       type :float
669
670 tosca.capabilities.nfv.criticalMode:
671   derived_from: tosca.capabilities.Node
672   description: Describes the capabilities related to critical mode
673   properties:
674     activation:
675       type :string
676
677 tosca.capabilities.nfv.container:
678   derived_from: tosca.capabilities.Node
679   description: Describes the capabilities related to container
680   properties:
681     cpu:
682       type :float
683     cpuFreq :
684       type :float
685     memory :
686       type: float
687     disk :
688       type: float
689
690 tosca.capabilities.nfv.E2E_evidence_collection:
691   derived_from: tosca.capabilities.Node
692   description: Describes the capabilities related to domain evidence collector
693   properties:
694     rate:
```

```
695       type :float
696
697 tosca.capabilities.nfv.evidenceCollector:
698   derived_from: tosca.capabilities.Node
699   description: Describes the capabilities related to evidenceCollector
700   properties:
701     rate:
702       type :float
703
704 tosca.capabilities.nfv.VNFmanager:
705   derived_from: tosca.capabilities.Node
706   description: Describes the capabilities related to VNF manager
707   properties:
708     cpu:
709       type :float
710     cpuFreq :
711       type :float
712     memory :
713       type: float
714     disk :
715       type: float
716
717 tosca.capabilities.nfv.VIM:
718   derived_from: tosca.capabilities.Node
719   description: Describes the capabilities related to VIM
720   properties:
721     cpu:
722       type :float
723     cpuFreq :
724       type :float
725     memory :
726       type: float
727     disk :
728       type: float
729
730 tosca.capabilities.nfv.deployImg:
731   derived_from: tosca.capabilities.Node
732   description: Describes the capabilities related to deployImg
733   properties:
734     image:
735       type :string
736
737 tosca.capabilities.nfv.bindable:
738   derived_from: tosca.capabilities.Node
739   description: Describes the capabilities related to bindable
```

```
740   properties:
741     bind:
742       type : string
743
744
745 tosca.capabilities.nfv.orchestrator:
746   derived_from: tosca.capabilities.Node
747   description: Describes the capabilities related to orchestrator
748   properties:
749     cpu:
750       type : float
751     cpuFreq :
752       type : float
753     memory :
754       type: float
755     disk :
756       type: float
757
758
759
760 tosca.capabilities.nfv.VirtualLink:
761   derived_from: tosca.capabilities.Node
762   description: Describes the capabilities related to VNF manager
763   properties:
764     protocol:
765       type : string
766
767 tosca.capabilities.nfv.cameraActivation:
768   derived_from: tosca.capabilities.Node
769   description: Describes the capabilities related to the camera activation.
770   properties:
771     mode :
772       type : string
773
774
775
776
777
778
779#############################################################################
      RELATIONSHIP TYPE
      ##########################################################################################

780relationship_types:
781 tosca.relationships.nfv.securityLinkto:
```

```
782   derived_from: tosca.relationships.Root
783   description: Represents an association relationship between two manifest
784   valid_target_types: [tosca.capabilities.nfv.exposedSecurityService]
785
786
787
788
789
790########################################################################## NODE TYPE
       ########################################################################
791
792
793node_types:
794  tosca.nodes.nfv.TRAILS.E2E_service:
795    derived_from: tosca.nodes.Root
796    properties:
797      header:
798        type: tosca.datatypes.nfv.header
799        description: Metadata of the manifest.
800        required: true
801      validations:
802        type: list
803        description: Hold a list of "validation-object". Each object give a set of
             information about the validation process.
804        required: false
805        entry_schema:
806          type: tosca.datatypes.nfv.validation
807      authors:
808        type: list
809        description: Hold a list of 'author-Object'. Each object give a set of
             information about the author involved in the creation of the manifest.
810        required: true
811        entry_schema:
812          type: tosca.datatypes.nfv.author
813      commitments:
814        type: list
815        description: Hold a list of 'commitment-Object'. Each object contains a set
             of information about the property of the device.
816        required: false
817        entry_schema:
818          type: tosca.datatypes.nfv.commitment
819      usageRecommendations:
820        type: list
821        description: Hold a list of 'usageRecommendation-Object'. Each object
             contains recommendation on term of security, resources required and
```

```
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - external_virtual_link :
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [1,1]
  capabilities:
    external_virtual_link :
      type: tosca.capabilities.nfv.VirtualLinkable

#####################################################

tosca.nodes.nfv.TRAILS.5G-core:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
```

```
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - critical-mode:
        capability: tosca.capabilities.nfv.criticalMode

    - virtual_link :
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [1,1]
  capabilities:
    virtual_link :
      type: tosca.capabilities.nfv.VirtualLinkable
    domainEvidence-collector:
      type: tosca.capabilities.nfv.domainEvidenceCollector

################################################

tosca.nodes.nfv.TRAILS.E2E_management:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
```

```
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - domainEvidence-collector:
        capability: tosca.capabilities.nfv.domainEvidenceCollector
    - virtual_link :
        capability: tosca.capabilities.nfv.VirtualLinkable
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [1,1]
    - container :
        capability: tosca.capabilities.nfv.container

  capabilities:
    external_virtual_link :
      type: tosca.capabilities.nfv.VirtualLinkable
    E2E_evidence_collection:
      type: tosca.capabilities.nfv.E2E_evidence_collection
    critical-mode :
      type: tosca.capabilities.nfv.criticalMode

######################

tosca.nodes.nfv.TRAILS.RAS:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
```

```
946       description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
947       required: false
948       entry_schema:
949         type: tosca.datatypes.nfv.validation
950     authors:
951       type: list
952       description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
953       required: true
954       entry_schema:
955         type: tosca.datatypes.nfv.author
956     commitments:
957       type: list
958       description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
959       required: false
960       entry_schema:
961         type: tosca.datatypes.nfv.commitment
962     usageRecommendations:
963       type: list
964       description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
965       required :false
966       entry_schema:
967         type: tosca.datatypes.nfv.usageRecommendation
968
969   requirements:
970     - evidence:
971         capability: tosca.capabilities.nfv.evidenceCollector
972
973   capabilities:
974     domainEvidence :
975       type: tosca.capabilities.nfv.domainEvidenceCollector
976
977
978##################
979
980 tosca.nodes.nfv.TRAILS.argoCD:
981   derived_from: tosca.nodes.Root
982   properties:
983     header:
984       type: tosca.datatypes.nfv.header
985       description: Metadata of the manifest.
```

```
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  capabilities:
    image :
      type: tosca.capabilities.nfv.deployImg


##################

tosca.nodes.nfv.TRAILS.deep-attestation:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
```

```
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - bind:
        capability: tosca.capabilities.nfv.bindable

  capabilities:
    evidence :
      type: tosca.capabilities.nfv.evidenceCollector


####################
tosca.nodes.nfv.TRAILS.kubernetes:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
```

```
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - img:
        capability: tosca.capabilities.nfv.deployImg

  capabilities:
    VNFmanger:
      type: tosca.capabilities.nfv.VNFmanager
    VIM:
      type :tosca.capabilities.nfv.VIM

#############
tosca.nodes.nfv.TRAILS.NFVI:
  derived_from: tosca.nodes.Root
```

```
1106  properties:
1107    header:
1108      type: tosca.datatypes.nfv.header
1109      description: Metadata of the manifest.
1110      required: true
1111    validations:
1112      type: list
1113      description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1114      required: false
1115      entry_schema:
1116        type: tosca.datatypes.nfv.validation
1117    authors:
1118      type: list
1119      description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1120      required: true
1121      entry_schema:
1122        type: tosca.datatypes.nfv.author
1123    commitments:
1124      type: list
1125      description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1126      required: false
1127      entry_schema:
1128        type: tosca.datatypes.nfv.commitment
1129    usageRecommendations:
1130      type: list
1131      description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1132      required :false
1133      entry_schema:
1134        type: tosca.datatypes.nfv.usageRecommendation
1135
1136  requirements:
1137    - VIM:
1138        capability: tosca.capabilities.nfv.VIM
1139
1140  capabilities:
1141    container :
1142      type: tosca.capabilities.nfv.container
1143    bindable:
1144      type :tosca.capabilities.nfv.bindable
1145
```

```
1146###########
1147  tosca.nodes.nfv.TRAILS.sec-by-orc:
1148    derived_from: tosca.nodes.Root
1149    properties:
1150      header:
1151        type: tosca.datatypes.nfv.header
1152        description: Metadata of the manifest.
1153        required: true
1154      validations:
1155        type: list
1156        description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1157        required: false
1158        entry_schema:
1159          type: tosca.datatypes.nfv.validation
1160      authors:
1161        type: list
1162        description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1163        required: true
1164        entry_schema:
1165          type: tosca.datatypes.nfv.author
1166      commitments:
1167        type: list
1168        description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1169        required: false
1170        entry_schema:
1171          type: tosca.datatypes.nfv.commitment
1172      usageRecommendations:
1173        type: list
1174        description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1175        required :false
1176        entry_schema:
1177          type: tosca.datatypes.nfv.usageRecommendation
1178
1179    requirements:
1180      - critical-mode:
1181          capability: tosca.capabilities.nfv.criticalMode
1182
1183
1184    capabilities:
1185      orchestrator :
```

```
      type: tosca.capabilities.nfv.orchestrator

###############

 tosca.nodes.nfv.TRAILS.VNF:
   derived_from: tosca.nodes.Root
   properties:
     header:
       type: tosca.datatypes.nfv.header
       description: Metadata of the manifest.
       required: true
     validations:
       type: list
       description: Hold a list of "validation-object". Each object give a set of
           information about the validation process.
       required: false
       entry_schema:
         type: tosca.datatypes.nfv.validation
     authors:
       type: list
       description: Hold a list of 'author-Object'. Each object give a set of
           information about the author involved in the creation of the manifest.
       required: true
       entry_schema:
         type: tosca.datatypes.nfv.author
     commitments:
       type: list
       description: Hold a list of 'commitment-Object'. Each object contains a set
           of information about the property of the device.
       required: false
       entry_schema:
         type: tosca.datatypes.nfv.commitment
     usageRecommendations:
       type: list
       description: Hold a list of 'usageRecommendation-Object'. Each object
           contains recommendation on term of security, resources required and
           dependencies.
       required :false
       entry_schema:
         type: tosca.datatypes.nfv.usageRecommendation

   requirements:
     - container:
         capability: tosca.capabilities.nfv.container
     - virual_link :
```

```
1226          capability: tosca.capabilities.nfv.VirtualLinkable
1227          relationship: tosca.relationships.nfv.VirtualLinksTo
1228          occurrences: [1,1]
1229      - orchestrator:
1230          capability: tosca.capabilities.nfv.orchestrator
1231      - VNFmanager:
1232          capability: tosca.capabilities.nfv.VNFmanager
1233
1234    capabilities:
1235      virtual_link :
1236        type: tosca.capabilities.nfv.VirtualLink
1237
1238###############
1239
1240  tosca.nodes.nfv.TRAILS.IoT-campus:
1241    derived_from: tosca.nodes.Root
1242    properties:
1243      header:
1244        type: tosca.datatypes.nfv.header
1245        description: Metadata of the manifest.
1246        required: true
1247      validations:
1248        type: list
1249        description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1250        required: false
1251        entry_schema:
1252          type: tosca.datatypes.nfv.validation
1253      authors:
1254        type: list
1255        description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1256        required: true
1257        entry_schema:
1258          type: tosca.datatypes.nfv.author
1259      commitments:
1260        type: list
1261        description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1262        required: false
1263        entry_schema:
1264          type: tosca.datatypes.nfv.commitment
1265      usageRecommendations:
1266        type: list
1267        description: Hold a list of 'usageRecommendation-Object'. Each object
```

```
            contains recommendation on term of security, resources required and
            dependencies.
1268        required :false
1269        entry_schema:
1270          type: tosca.datatypes.nfv.usageRecommendation
1271
1272    requirements:
1273      - critical-mode:
1274          capability: tosca.capabilities.nfv.criticalMode
1275
1276      - external_virtual_link :
1277          capability: tosca.capabilities.nfv.VirtualLink
1278
1279    capabilities:
1280      virtual_link :
1281        type: tosca.capabilities.nfv.VirtualLink
1282      domainEvidence:
1283        type: tosca.capabilities.nfv.domainEvidenceCollector
1284
1285
1286###############
1287  tosca.nodes.nfv.TRAILS.camera:
1288    derived_from: tosca.nodes.Root
1289    properties:
1290      header:
1291        type: tosca.datatypes.nfv.header
1292        description: Metadata of the manifest.
1293        required: true
1294      validations:
1295        type: list
1296        description: Hold a list of "validation-object". Each object give a set of
            information about the validation process.
1297        required: false
1298        entry_schema:
1299          type: tosca.datatypes.nfv.validation
1300      authors:
1301        type: list
1302        description: Hold a list of 'author-Object'. Each object give a set of
            information about the author involved in the creation of the manifest.
1303        required: true
1304        entry_schema:
1305          type: tosca.datatypes.nfv.author
1306      commitments:
1307        type: list
1308        description: Hold a list of 'commitment-Object'. Each object contains a set
```

```
            of information about the property of the device.
1309        required: false
1310        entry_schema:
1311          type: tosca.datatypes.nfv.commitment
1312      usageRecommendations:
1313        type: list
1314        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1315        required :false
1316        entry_schema:
1317          type: tosca.datatypes.nfv.usageRecommendation
1318
1319    requirements:
1320      - camera:
1321          capability: tosca.capabilities.nfv.cameraActivation
1322    capabilities:
1323      virtual_link :
1324        type: tosca.capabilities.nfv.VirtualLink
1325
1326##############
1327
1328  tosca.nodes.nfv.TRAILS.IOT-server:
1329    derived_from: tosca.nodes.Root
1330    properties:
1331      header:
1332        type: tosca.datatypes.nfv.header
1333        description: Metadata of the manifest.
1334        required: true
1335      validations:
1336        type: list
1337        description: Hold a list of "validation-object". Each object give a set of
                information about the validation process.
1338        required: false
1339        entry_schema:
1340          type: tosca.datatypes.nfv.validation
1341      authors:
1342        type: list
1343        description: Hold a list of 'author-Object'. Each object give a set of
                information about the author involved in the creation of the manifest.
1344        required: true
1345        entry_schema:
1346          type: tosca.datatypes.nfv.author
1347      commitments:
1348        type: list
```

```
1349        description: Hold a list of 'commitment-Object'. Each object contains a set
                of information about the property of the device.
1350        required: false
1351        entry_schema:
1352          type: tosca.datatypes.nfv.commitment
1353      usageRecommendations:
1354        type: list
1355        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1356        required :false
1357        entry_schema:
1358          type: tosca.datatypes.nfv.usageRecommendation
1359
1360
1361    capabilities:
1362      bind:
1363        type: tosca.capabilities.nfv.bindable
1364      container:
1365        type: tosca.capabilities.nfv.container
1366
1367#################
1368
1369  tosca.nodes.nfv.TRAILS.IoT-gateway:
1370    derived_from: tosca.nodes.Root
1371    properties:
1372      header:
1373        type: tosca.datatypes.nfv.header
1374        description: Metadata of the manifest.
1375        required: true
1376      validations:
1377        type: list
1378        description: Hold a list of "validation-object". Each object give a set of
                information about the validation process.
1379        required: false
1380        entry_schema:
1381          type: tosca.datatypes.nfv.validation
1382      authors:
1383        type: list
1384        description: Hold a list of 'author-Object'. Each object give a set of
                information about the author involved in the creation of the manifest.
1385        required: true
1386        entry_schema:
1387          type: tosca.datatypes.nfv.author
1388      commitments:
```

```
1389        type: list
1390        description: Hold a list of 'commitment-Object'. Each object contains a set
                of information about the property of the device.
1391        required: false
1392        entry_schema:
1393          type: tosca.datatypes.nfv.commitment
1394      usageRecommendations:
1395        type: list
1396        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1397        required :false
1398        entry_schema:
1399          type: tosca.datatypes.nfv.usageRecommendation
1400
1401    requirements:
1402      - external_virtual_link :
1403          capability: tosca.capabilities.nfv.VirtualLink
1404          relationship: tosca.relationships.nfv.VirtualLinksTo
1405          occurrences: [1,1]
1406      - virtual_link :
1407          capability: tosca.capabilities.nfv.VirtualLink
1408          relationship: tosca.relationships.nfv.VirtualLinksTo
1409          occurrences: [1,1]
1410    capabilities:
1411      virtual_link :
1412        type: tosca.capabilities.nfv.VirtualLink
1413
1414###################
1415
1416  tosca.nodes.nfv.TRAILS.IoT-MMT:
1417    derived_from: tosca.nodes.Root
1418    properties:
1419      header:
1420        type: tosca.datatypes.nfv.header
1421        description: Metadata of the manifest.
1422        required: true
1423      validations:
1424        type: list
1425        description: Hold a list of "validation-object". Each object give a set of
                information about the validation process.
1426        required: false
1427        entry_schema:
1428          type: tosca.datatypes.nfv.validation
1429      authors:
```

```
1430        type: list
1431        description: Hold a list of 'author-Object'. Each object give a set of
                information about the author involved in the creation of the manifest.
1432        required: true
1433        entry_schema:
1434          type: tosca.datatypes.nfv.author
1435      commitments:
1436        type: list
1437        description: Hold a list of 'commitment-Object'. Each object contains a set
                of information about the property of the device.
1438        required: false
1439        entry_schema:
1440          type: tosca.datatypes.nfv.commitment
1441      usageRecommendations:
1442        type: list
1443        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1444        required :false
1445        entry_schema:
1446          type: tosca.datatypes.nfv.usageRecommendation
1447
1448    requirements:
1449      - container:
1450          capability: tosca.capabilities.nfv.container
1451      - virtual_link :
1452          capability: tosca.capabilities.nfv.VirtualLink
1453      - evidence-collector:
1454          capability: tosca.capabilities.nfv.evidenceCollector
1455      - critical-mode:
1456          capability: tosca.capabilities.nfv.criticalMode
1457
1458    capabilities:
1459      virtual_link :
1460        type: tosca.capabilities.nfv.VirtualLink
1461      evidence:
1462        type :tosca.capabilities.nfv.evidenceCollector
1463      domainEvidence:
1464        type :tosca.capabilities.nfv.domainEvidenceCollector
1465      camera:
1466        type: tosca.capabilities.nfv.cameraActivation
1467
1468##########################
1469  tosca.nodes.nfv.TRAILS.IoT-RCA:
1470    derived_from: tosca.nodes.Root
```

```
1471  properties:
1472    header:
1473      type: tosca.datatypes.nfv.header
1474      description: Metadata of the manifest.
1475      required: true
1476    validations:
1477      type: list
1478      description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1479      required: false
1480      entry_schema:
1481        type: tosca.datatypes.nfv.validation
1482    authors:
1483      type: list
1484      description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1485      required: true
1486      entry_schema:
1487        type: tosca.datatypes.nfv.author
1488    commitments:
1489      type: list
1490      description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1491      required: false
1492      entry_schema:
1493        type: tosca.datatypes.nfv.commitment
1494    usageRecommendations:
1495      type: list
1496      description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1497      required :false
1498      entry_schema:
1499        type: tosca.datatypes.nfv.usageRecommendation
1500
1501  requirements:
1502    - container:
1503        capability: tosca.capabilities.nfv.container
1504        relationship: tosca.relationships.nfv.securityLinkto
1505    - virtual_link :
1506        capability: tosca.capabilities.nfv.VirtualLink
1507        relationship: tosca.relationships.nfv.VirtualLinksTo
1508        occurrences: [1,1]
1509  capabilities:
1510    virtual_link :
```

```
      type: tosca.capabilities.nfv.VirtualLink
    evidence:
      type: tosca.capabilities.nfv.evidenceCollector

##################

tosca.nodes.nfv.TRAILS.sniffer:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

  requirements:
    - virtual_link :
```

```
        capability: tosca.capabilities.nfv.VirtualLink
        relationship: tosca.relationships.nfv.VirtualLinksTo
        occurrences: [1,1]
  capabilities:
    virtual_link :
      type: tosca.capabilities.nfv.VirtualLink

##################
tosca.nodes.nfv.TRAILS.systemic:
  derived_from: tosca.nodes.Root
  properties:
    header:
      type: tosca.datatypes.nfv.header
      description: Metadata of the manifest.
      required: true
    validations:
      type: list
      description: Hold a list of "validation-object". Each object give a set of
          information about the validation process.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.validation
    authors:
      type: list
      description: Hold a list of 'author-Object'. Each object give a set of
          information about the author involved in the creation of the manifest.
      required: true
      entry_schema:
        type: tosca.datatypes.nfv.author
    commitments:
      type: list
      description: Hold a list of 'commitment-Object'. Each object contains a set
          of information about the property of the device.
      required: false
      entry_schema:
        type: tosca.datatypes.nfv.commitment
    usageRecommendations:
      type: list
      description: Hold a list of 'usageRecommendation-Object'. Each object
          contains recommendation on term of security, resources required and
          dependencies.
      required :false
      entry_schema:
        type: tosca.datatypes.nfv.usageRecommendation

```

```
1591    requirements:
1592      - bind:
1593          capability: tosca.capabilities.nfv.bindable
1594
1595    capabilities:
1596      evidence :
1597        type: tosca.capabilities.nfv.evidenceCollector
1598
1599##########
1600
1601  tosca.nodes.nfv.TRAILS.dashboard:
1602    derived_from: tosca.nodes.Root
1603    properties:
1604      header:
1605        type: tosca.datatypes.nfv.header
1606        description: Metadata of the manifest.
1607        required: true
1608      validations:
1609        type: list
1610        description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1611        required: false
1612        entry_schema:
1613          type: tosca.datatypes.nfv.validation
1614      authors:
1615        type: list
1616        description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1617        required: true
1618        entry_schema:
1619          type: tosca.datatypes.nfv.author
1620      commitments:
1621        type: list
1622        description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1623        required: false
1624        entry_schema:
1625          type: tosca.datatypes.nfv.commitment
1626      usageRecommendations:
1627        type: list
1628        description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1629        required :false
1630        entry_schema:
```

```
1631          type: tosca.datatypes.nfv.usageRecommendation
1632
1633    requirements:
1634      - container:
1635          capability: tosca.capabilities.nfv.container
1636      - orchestrator:
1637          capability: tosca.capabilities.nfv.orchestrator
1638      - VNFmanger:
1639          capability: tosca.capabilities.nfv.VNFmanager
1640      - virtual_link :
1641          capability: tosca.capabilities.nfv.VirtualLink
1642          relationship: tosca.relationships.nfv.VirtualLinksTo
1643          occurrences: [1,1]
1644    capabilities:
1645      virtual_link :
1646        type: tosca.capabilities.nfv.VirtualLink
1647
1648################
1649
1650  tosca.nodes.nfv.TRAILS.MEC_infra:
1651    derived_from: tosca.nodes.Root
1652    properties:
1653      header:
1654        type: tosca.datatypes.nfv.header
1655        description: Metadata of the manifest.
1656        required: true
1657      validations:
1658        type: list
1659        description: Hold a list of "validation-object". Each object give a set of
                 information about the validation process.
1660        required: false
1661        entry_schema:
1662          type: tosca.datatypes.nfv.validation
1663      authors:
1664        type: list
1665        description: Hold a list of 'author-Object'. Each object give a set of
                 information about the author involved in the creation of the manifest.
1666        required: true
1667        entry_schema:
1668          type: tosca.datatypes.nfv.author
1669      commitments:
1670        type: list
1671        description: Hold a list of 'commitment-Object'. Each object contains a set
                 of information about the property of the device.
1672        required: false
```

```
1673        entry_schema:
1674          type: tosca.datatypes.nfv.commitment
1675      usageRecommendations:
1676        type: list
1677        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1678        required :false
1679        entry_schema:
1680          type: tosca.datatypes.nfv.usageRecommendation
1681
1682    requirements:
1683      - VIM:
1684          capability: tosca.capabilities.nfv.VIM
1685    capabilities:
1686      evidence :
1687        type: tosca.capabilities.nfv.evidenceCollector
1688      container:
1689        type: tosca.capabilities.nfv.container
1690
1691################
1692
1693  tosca.nodes.nfv.TRAILS.sec-by-orc_mec:
1694    derived_from: tosca.nodes.Root
1695    properties:
1696      header:
1697        type: tosca.datatypes.nfv.header
1698        description: Metadata of the manifest.
1699        required: true
1700      validations:
1701        type: list
1702        description: Hold a list of "validation-object". Each object give a set of
                information about the validation process.
1703        required: false
1704        entry_schema:
1705          type: tosca.datatypes.nfv.validation
1706      authors:
1707        type: list
1708        description: Hold a list of 'author-Object'. Each object give a set of
                information about the author involved in the creation of the manifest.
1709        required: true
1710        entry_schema:
1711          type: tosca.datatypes.nfv.author
1712      commitments:
1713        type: list
```

```
1714        description: Hold a list of 'commitment-Object'. Each object contains a set
                of information about the property of the device.
1715        required: false
1716        entry_schema:
1717          type: tosca.datatypes.nfv.commitment
1718      usageRecommendations:
1719        type: list
1720        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1721        required :false
1722        entry_schema:
1723          type: tosca.datatypes.nfv.usageRecommendation
1724
1725    requirements:
1726      - container:
1727          capability: tosca.capabilities.nfv.container
1728      - critical-mode :
1729          capability: tosca.capabilities.nfv.criticalMode
1730      - evidence:
1731          capability: tosca.capabilities.nfv.evidenceCollector
1732    capabilities:
1733      VNFmanger :
1734        type: tosca.capabilities.nfv.VNFmanager
1735      orchestrator:
1736        type :tosca.capabilities.nfv.orchestrator
1737      domainEvidence:
1738        type: tosca.capabilities.nfv.domainEvidenceCollector
1739      VIM:
1740        type: tosca.capabilities.nfv.VIM
1741
1742##################
1743
1744  tosca.nodes.nfv.TRAILS.streaming-service:
1745    derived_from: tosca.nodes.Root
1746    properties:
1747      header:
1748        type: tosca.datatypes.nfv.header
1749        description: Metadata of the manifest.
1750        required: true
1751      validations:
1752        type: list
1753        description: Hold a list of "validation-object". Each object give a set of
                information about the validation process.
1754        required: false
```

```
1755        entry_schema:
1756          type: tosca.datatypes.nfv.validation
1757      authors:
1758        type: list
1759        description: Hold a list of 'author-Object'. Each object give a set of
                information about the author involved in the creation of the manifest.
1760        required: true
1761        entry_schema:
1762          type: tosca.datatypes.nfv.author
1763      commitments:
1764        type: list
1765        description: Hold a list of 'commitment-Object'. Each object contains a set
                of information about the property of the device.
1766        required: false
1767        entry_schema:
1768          type: tosca.datatypes.nfv.commitment
1769      usageRecommendations:
1770        type: list
1771        description: Hold a list of 'usageRecommendation-Object'. Each object
                contains recommendation on term of security, resources required and
                dependencies.
1772        required :false
1773        entry_schema:
1774          type: tosca.datatypes.nfv.usageRecommendation
1775
1776
1777    requirements:
1778      - container:
1779          capability: tosca.capabilities.nfv.container
1780      - orchestrator:
1781          capability: tosca.capabilities.nfv.orchestrator
1782      - VNFmanger:
1783          capability: tosca.capabilities.nfv.VNFmanager
1784      - virtual_link :
1785          capability: tosca.capabilities.nfv.VirtualLink
1786          relationship: tosca.relationships.nfv.VirtualLinksTo
1787          occurrences: [1,1]
1788    capabilities:
1789      virtual_link :
1790        type: tosca.capabilities.nfv.VirtualLink
1791
1792
1793##################
1794
1795  tosca.nodes.nfv.TRAILS.server:
```

```
1796    derived_from: tosca.nodes.Root
1797    properties:
1798      header:
1799        type: tosca.datatypes.nfv.header
1800        description: Metadata of the manifest.
1801        required: true
1802      validations:
1803        type: list
1804        description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1805        required: false
1806        entry_schema:
1807          type: tosca.datatypes.nfv.validation
1808      authors:
1809        type: list
1810        description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1811        required: true
1812        entry_schema:
1813          type: tosca.datatypes.nfv.author
1814      commitments:
1815        type: list
1816        description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1817        required: false
1818        entry_schema:
1819          type: tosca.datatypes.nfv.commitment
1820      usageRecommendations:
1821        type: list
1822        description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1823        required :false
1824        entry_schema:
1825          type: tosca.datatypes.nfv.usageRecommendation
1826
1827
1828    capabilities:
1829      container :
1830        type: tosca.capabilities.nfv.container
1831
1832###################
1833
1834  tosca.nodes.nfv.TRAILS.MEC:
1835    derived_from: tosca.nodes.Root
```

```
1836    properties:
1837      header:
1838        type: tosca.datatypes.nfv.header
1839        description: Metadata of the manifest.
1840        required: true
1841      validations:
1842        type: list
1843        description: Hold a list of "validation-object". Each object give a set of
              information about the validation process.
1844        required: false
1845        entry_schema:
1846          type: tosca.datatypes.nfv.validation
1847      authors:
1848        type: list
1849        description: Hold a list of 'author-Object'. Each object give a set of
              information about the author involved in the creation of the manifest.
1850        required: true
1851        entry_schema:
1852          type: tosca.datatypes.nfv.author
1853      commitments:
1854        type: list
1855        description: Hold a list of 'commitment-Object'. Each object contains a set
              of information about the property of the device.
1856        required: false
1857        entry_schema:
1858          type: tosca.datatypes.nfv.commitment
1859      usageRecommendations:
1860        type: list
1861        description: Hold a list of 'usageRecommendation-Object'. Each object
              contains recommendation on term of security, resources required and
              dependencies.
1862        required :false
1863        entry_schema:
1864          type: tosca.datatypes.nfv.usageRecommendation
1865
1866    requirements:
1867      - critical-mode:
1868          capability: tosca.capabilities.nfv.criticalMode
1869      - virtual_link :
1870          capability: tosca.capabilities.nfv.VirtualLinkable
1871          relationship: tosca.relationships.nfv.VirtualLinksTo
1872          occurrences: [1,1]
1873    capabilities:
1874      virtual_link :
1875        type: tosca.capabilities.nfv.VirtualLinkable
```

```
1876      domainEvidence:
1877        type : tosca.capabilities.nfv.domainEvidenceCollector
1878
1879
1880
1881
1882
1883interface_types:
1884  tosca.interfaces.nfv.operationLimitation:
1885    derived_from: tosca.interfaces.Root
1886    operation_limitation:
1887      description: Invoke before instanciation.
1888
1889
1890policy_types:
1891  tosca.policies.nfv.operationLimitation:
1892    derived_from: tosca.policies.Root
1893    description: The operationLimitation is a policy type that describe restriction
           imposed by the infrastructure administrator before the component is
           referenced in the operator's catalog.
1894    targets: [tosca.nodes.nfv.manifest]
1895    triggers: [tosca.triggers.nfv.operationLimitation]
```

Appendix B

ACM MobiCom2023 Poster

Demonstrating Liability and Trust Metrics for Multi-Actor, Dynamic Edge and Cloud Microservices

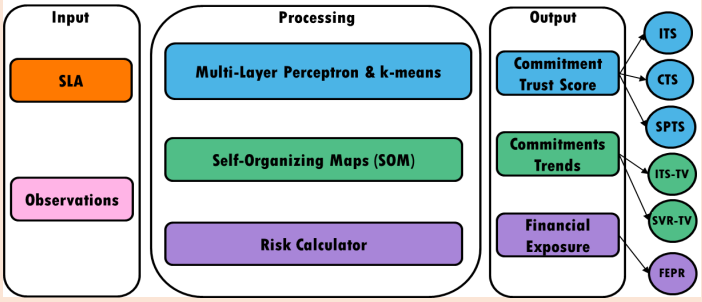
Yacine Anser, Romain Cajeat, Chrystel Gaber, Jean-Philippe Wary, Samia Bouzefrane, Méziane Yacoub, Onur Kalinagac, Gürkan Gür

Abstract

Transitioning edge and cloud computing in 5G networks towards service-based architecture increases their complexity as they become even **more dynamic and intertwine more actors or delegation levels**. We present the **Liability-aware security manager Analysis Service (LAS)**, a framework that computes liability and trust indicators for service-based architectures. Based on the commitments of Service Providers (SPs) and real-time observations collected by a Root Cause Analysis (RCA) tool GRALAF, the LAS computes three categories of liability and trust indicators, specifically, a Commitment Trust Score, Financial Exposure, and Commitment Trends.

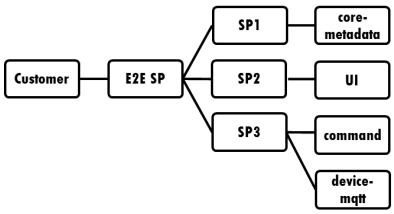
Contribution – LAS

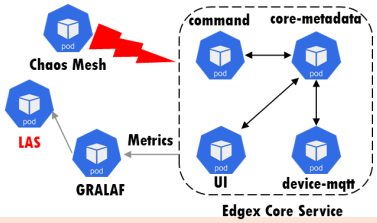
- Instance Trust Score (ITS)
- Class Trust Score (CTS)
- Service Provider Trust Score (SPTS)
- ITS-Trend Variation (ITS-TV)
- SLA Violation Rate – Trend Variation (SVR-TV)
- Financial Exposure to Penality Risk (FEPR)



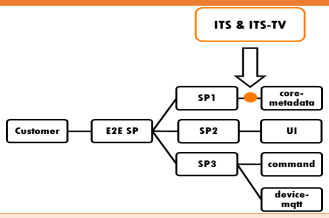
Running use case

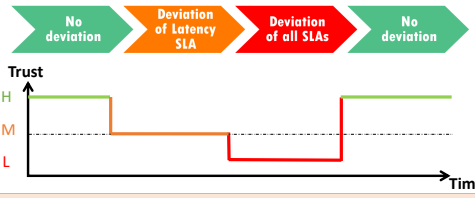
Observing three SLAs :
1. Availability
2. Latency
3. Error Rate

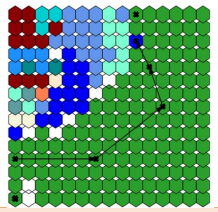




Use case 1: Following the Instance Trust Score and the ITS Trend Variation

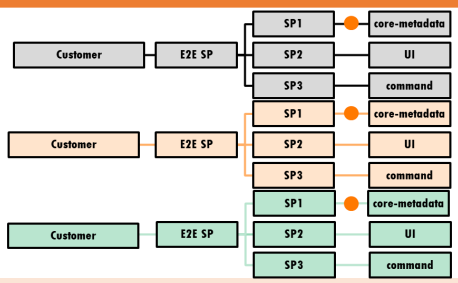


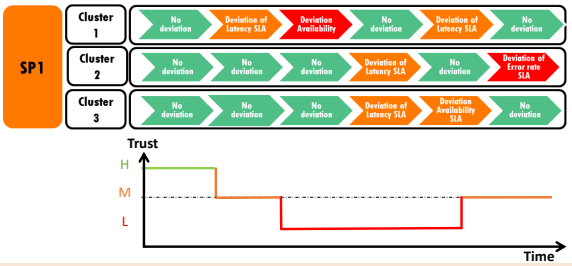




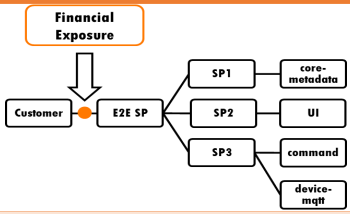
Use case 2: Infer trust level on Component Class & Service Provider

Three instances of Edgex





Use case 3: Following the Financial Exposure of the E2E Service



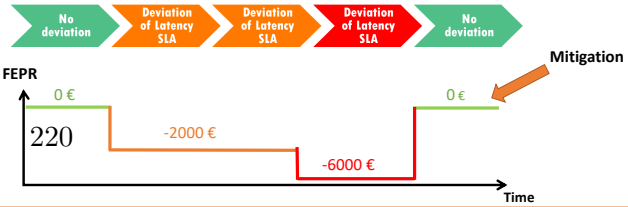


Figure B.1: ACM MobiCom2023 Poster Presenting the Liability-Aware Analysis Service