TECHNOLOGICAL UNIVERSITY OF THE SHANNON

Adaptive Intent Realisation (AIR) – Inductive Intent Realisation Through NLP Enabled Intent Matching.

by

Joseph McNamara

Supervisor

Dr. Enda Fallon

Dr. Paul Jacob

PhD Thesis

in the
Faculty of Engineering & Informatics

September 2023

Declaration of Authorship

I, Joseph McNamara, declare that this thesis titled, 'Adaptive Intent Realisation (AIR) – Inductive Intent Realisation Through NLP Enabled Intent Matching' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this Institute.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this Institute or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____

Date: ____

TECHNOLOGICAL UNIVERSITY OF THE SHANNON

Abstract

There is a strong interest in designing systems that can simplify the interactions between humans and complex digital systems. Network Operators want more straightforward mechanisms to engage with their networks and inform their actions and goals. Intent was proposed to meet this challenge, but comes at a cost. Intent introduces large modelling efforts, requiring Network Operators to gain expertise in formal model notation and the integration of these models with their network. The cost is compounded by the speed which modern networks evolve, requiring constant adaption to maintain intent-driven features.

This work aims to leverage the concepts of intent-based management for private networks, without component and formal model expertise. This will be achieved through the coordination of three enablers, Adaptive Policy, Machine Learning and Intent. Adaptive Policy provides a flexible framework for context-aware decision making, utilising a state-based approach to policy execution. Machine Learning informs the decision making process to produce impact-aware responses based on closed-loop reporting. Intent structures the realisation process, how abstraction is handled through inductive processes to generate actionable output. This work is highly experimental, developed on site at the Network Management Lab in an Ericsson Product Development Unit based in Ireland. This work concludes with the Adaptive Intent Realisation (AIR) reference architecture successfully demonstrated in three use cases hosted in industrial grade private 5G networks.

Acknowledgements

I would like to thank my primary supervisor, Dr Enda Fallon, for his support throughout the duration of my study. Enda's encouragement and direction has enabled me to pursue areas of network communication and management that I am passionate about and he has challenged me both in terms of work complexity and volume where I have learned skills that I will use for the rest of my life.

I would also like to thank Dr. Liam Fallon, Dr. John Keeney and Dr. Sven Van Der Meer. Their advice and guidance was always well meaning and light hearted. Starting off in such an experienced and open group accelerated my learning and their genuine interest and enthusiasm for their work was infectious. I can honestly say that without their influence I would not have the same passion or enjoyment for my work.

I would like to thank all of my friends in the Network Management Lab in Ericsson, Athlone. Their guidance and encouragement has been extraordinary. The trust that they had in me, allowing me to get involved in large projects and listening to my input, made me feel like a valued member of the group. I can honestly say that the time spent with the Network Management Lab has been brilliant and I did not think that every week I would look forward to going to work.

Contents

Declaration of Authorship **i**

Abstract **ii**

Acknowledgements **iii**

List of Figures **viii**

List of Tables **x**

Abbreviations **xi**

1 Introduction **1**
- 1.1 Background . . . 2
- 1.2 Research Question . . . 7
  - 1.2.1 Core Research Question . . . 7
  - 1.2.2 Sub-Research Questions . . . 8
- 1.3 Contributions . . . 10
- 1.4 Publications Arising From This Work . . . 11

2 Literature Review **12**
- 2.1 Telecommunications . . . 13
  - 2.1.1 Fourth Generation 4G . . . 13
    - 2.1.1.1 MIMO . . . 13
    - 2.1.1.2 Frequency Domain Equalization . . . 14
    - 2.1.1.3 Positioning Reference Signals . . . 14
  - 2.1.2 Fifth Generation 5G . . . 15
    - 2.1.2.1 Network Softwarization - NFV/SDN . . . 15
    - 2.1.2.2 Network Slicing . . . 16
    - 2.1.2.3 Location Management Function . . . 18
    - 2.1.2.4 Beamforming . . . 18
    - 2.1.2.5 Management and Orchestration . . . 19
  - 2.1.3 Sixth Generation 6G . . . 20
    - 2.1.3.1 Drivers, Use Cases and Technologies . . . 20
- 2.2 Network Optimisation Strategies . . . 22
  - 2.2.1 Policy . . . 22

2.2.2 Closed Control Loop Systems . . . 23
2.2.3 Adaptive Policy . . . 24
2.3 Intent . . . 25
2.3.1 Goal-Oriented Policies . . . 25
2.3.2 Intent-Driven Interfaces . . . 26
2.3.3 Complete Intent-Driven Network Frameworks . . . 28
2.3.4 Intent Translation Techniques . . . 28
2.3.4.1 Template / Blueprint . . . 28
2.3.4.2 Mapping . . . 29
2.3.4.3 Refinement . . . 30
2.3.4.4 Network service descriptors . . . 30
2.3.4.5 Inference . . . 31
2.3.4.6 Keyword . . . 31
2.3.4.7 Machine Learning . . . 31
2.3.4.8 Semantics . . . 32
2.3.4.9 State Machine . . . 32
2.4 Machine Learning . . . 33
2.4.1 Supervised Learning . . . 33
2.4.1.1 Support Vector Machine . . . 34
2.4.1.2 Linear Regression . . . 34
2.4.1.3 Decision Tree . . . 34
2.4.2 Unsupervised Learning . . . 35
2.4.2.1 K-Means . . . 35
2.4.2.2 Auto Encoder . . . 35
2.4.2.3 One Class SVM . . . 35
2.4.3 Reinforcement Learning . . . 36
2.4.3.1 Markov Decision Process . . . 36
2.4.3.2 Q-Learning . . . 37
2.4.4 Natural Language Processing . . . 37
2.4.4.1 Lemmatization and Stemming . . . 37
2.4.4.2 Topic Modelling . . . 38
2.4.4.3 Keyword Extraction . . . 38
2.4.4.4 Knowledge Graphs . . . 38
2.5 Reference Architecture . . . 38
2.5.1 ONAP Platform Architecture . . . 39
2.5.1.1 ONAP Policy Framework . . . 41
2.5.1.2 Optimisation Framework . . . 42
2.5.2 ETSI OSM (Open Source MANO) . . . 44
2.5.3 Autonomous Networks . . . 45
2.5.3.1 TM Forum - Autonomous Networks Project . . . 46
2.5.3.2 ZSM - Intent-Driven Autonomous Networks . . . 47
2.5.3.3 NMRG - Intent-Based Networking . . . 51
2.5.4 Industrial Hosting . . . 52
2.5.4.1 5GENESIS . . . 53
2.5.4.2 5G-CLARITY . . . 54

3 Architectural Design 56

3.1 Implementation Architecture . . . 56
3.1.1 Adaptive Policy Execution . . . 56
3.1.2 Intent Engine . . . 57
3.1.3 Adaptive Intent Realisation (AIR) . . . 59
3.2 Evaluation . . . 63
3.2.1 OpenDaylight . . . 63
3.2.2 Mininet . . . 64
3.2.3 Policy Framework . . . 64
3.2.4 Slice Manager . . . 64
3.2.5 Positioning Server . . . 65
3.2.6 NFV Orchestrator . . . 66
3.3 Architecture Maturity . . . 66

4 Detailed Design and Evaluation **70**
4.1 A Testbed For Policy Driven Closed Loop Network Management . . . 71
4.1.1 Previous Work . . . 71
4.1.1.1 The APEX System . . . 71
4.1.2 Testbed Architecture . . . 74
4.1.3 Scenario . . . 76
4.1.3.1 Temporary Failure Scenario . . . 76
4.1.3.2 Prolonged Failure Scenario . . . 77
4.1.4 Implementation . . . 77
4.1.5 Results & Experiences . . . 81
4.2 An Adaptive Policy Approach to Video Quality Assurance . . . 86
4.2.1 System Architecture . . . 86
4.2.2 Testbed and Policy . . . 88
4.2.3 Preliminary Evaluation . . . 92
4.3 A Hybrid Machine Learning/Policy Approach to Optimise Video Path Selection . . . 95
4.3.1 System Architecture . . . 95
4.3.2 Implementation . . . 97
4.3.3 Preliminary Evaluation . . . 102
4.4 A Mechanism for Intent Driven Adaptive Policy Decision Making . . . 104
4.4.1 Intent Generation . . . 105
4.4.2 Intent Comparison . . . 106
4.4.3 Intent Resolution . . . 107
4.4.4 Usage Scenario . . . 108
4.5 A Flexible Interpreter For Intent Realisation . . . 110
4.5.1 Representations and Semantics . . . 110
4.5.1.1 Mathematical . . . 110
4.5.1.2 Model-based . . . 110
4.5.2 Flexible Interpretation . . . 112
4.5.2.1 Functionality Templates . . . 112
4.5.2.2 Intent Matching . . . 112
4.5.2.3 Translation . . . 113
4.5.3 Implementation . . . 114
4.5.4 Preliminary Results . . . 118

4.6 NLP Powered Intent Based Network Management for Private 5G Networks 120
4.6.1 Intent Engine 120
4.6.1.1 Intent design 121
4.6.1.2 Intent Matching Engine 122
4.6.1.3 South-bound provider integration 123
4.6.2 Demonstration 123

5 Conclusions and Future Work 128
5.1 Conclusions 128
5.2 Future Work 131

List of Figures

2.1 High Level Functional Architecture of ENI[1] 23
2.2 Intent driven management vs Policy driven management[2] 26
2.3 Open Network Automation Platform Architecture[3] 40
2.4 Traditional Optimisation Framework[4] 43
2.5 ONAP Optimisation Framework[4] 44
2.6 TMF Autonomous Network Architecture[5] 46
2.7 ZSM Reference Architecture[6] 48
2.8 Intent Management Entity[7] 49
2.9 Intent Management Entity in ZSM Reference Architecture[7] 50
2.10 ZSM Intent Control Loop[7] 50
2.11 High-Level View of an Autonomic Network[8] 51
2.12 Model of an Autonomic Node[8] 52
2.13 5G-CLARITY Architecture[9] 55

3.1 Intent Engine Concept[9] 58
3.2 Intent Interpreter V1 60
3.3 Intent Interpreter V2 61
3.4 Intent Interpreter V3 62
3.5 Initial Architectures Adopted by the Adaptive Policy testbed 67
3.6 Architecture and Results for Video Quality Assurance 67
3.7 Architecture and results for ML/Policy approach 68
3.8 Architecture and input for intent mechanism 68
3.9 Architecture and execution times for the intent interpreter 69
3.10 Architecture and intent driven UC execution times 69

4.1 APEX: Linking Policy to goals and context 72
4.2 Policy Model 72
4.3 APEX Architecture 73
4.4 The Closed Loop System Testbed Architecture 74
4.5 VPN Domain Network Topology 78
4.6 The Adaptive Policy as a Flowchart 81
4.7 Customer A and B Sharing two Links 82
4.8 Customer A and B Sharing two Links: Stream Quality 82
4.9 Customer A and B Sharing one Link 83
4.10 Customer A and B Sharing one Link: Stream Quality 83
4.11 Policy cuts Customer B Stream 84
4.12 Policy cuts Customer A Stream: Stream Quality 84
4.13 Policy cuts Customer B Stream: Stream Quality 85

4.14 Customer Stream Overview 85
4.15 System Architecture 87
4.16 Network Diagram containing packet loss information for links 89
4.17 Brute force analysis results 93
4.18 Current and Optimal Stream Comparison 94
4.19 System Architecture 95
4.20 Graphical Representation of a Perceptron 98
4.21 Weight Adjustment for 50 Cycle Test 103
4.22 Usage Scenario Intent 106
4.23 Intent Path Collection 107
4.24 Architecture 108
4.25 Network Traffic Data 109
4.26 Intent Matching 113
4.27 System Architecture 114
4.28 Functionality Template Event Message 115
4.29 Intent Realisation Timeframe 118
4.30 Internal Interpreter Execution Times 119
4.31 Intent based slice provisioning through POSTMAN based interface 124
4.32 Intent based NLoS identification, with object blocking line of sight 125
4.33 Intent based network service deployment, depicting the robot mounted 360° camera 126

List of Tables

4.1 Weight adjustment for cycles 1-5 102
4.2 Average intent execution time per use case. 126

Abbreviations

5G	Fifth **G**eneration
COMPA	**C**ontrol **O**rchestration **M**anagement **P**olicy **A**nalytics
APEX	**A**daptive **P**olicy **EX**ecution
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
HAS	**H**TTP **A**daptive **S**treaming
GPML	**G**eneral **P**olicy **M**odelling **L**anguage
CIM	**C**ommon **I**nformation **M**odel
ECA	**E**vent **C**ondition **A**ction
ISP	**I**nternet **S**ervice **P**roviders
OPEX	**O**perating **EX**pense
IBM	**I**nternational **B**usiness **M**achines
MAPE-K	**M**onitor **A**nalyse **P**lan **E**xecute **K**nowledge
FOCALE	**F**oundation **O**bservation **C**omparison **A**ction**L**earning **E**nvironment
SDN	**S**oftware **D**efined **N**etworking
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
MPEG	**M**oving **P**icture **E**xperts **G**roup
DASH	**D**ynamic **A**daptive **S**treaming over **H**TTP
RTP	**R**eal-time **T**ransport **P**rotocol
RTSP	**R**eal-**T**ime **S**treaming **P**rotocol
OTT	**O**ver **T**he **T**op
LTE	**L**ong **T**erm **E**volution
SAND	**S**erver **A**nd **N**etwork assisted **D**ASH
3G	Third **G**eneration
4G	Fourth **G**eneration

IoT	**I**nternet **o**f **T**hings
eMBB	**e**nhanced **M**obile **B**road**B**and
mMTC	**m**assive **M**achine-**T**ype **C**ommunications
URLLC	**U**ltra-**R**eliable and **L**ow-**L**atency **C**ommunications
E2E	**E**nd-to-**E**nd
FH	**F**ront**H**aul
BH	**B**ack**H**aul
CN	**C**ore **N**etwork
WDM	**W**avelength-**D**ivision **M**ultiplexing
NFV	**N**etwork **F**unction **V**irtualisation
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IFIP	**I**nternational **F**ederation for **I**nformation **P**rocessing
ONAP	**O**pen **N**etworking **A**utomation **P**latform
MEC	**M**obile **E**dge **C**omputing
VNF	**V**irtual **N**etwork **F**unction
MOS	**M**ean **O**pinion **S**core

Chapter 1

Introduction

It is not often possible to remove complexity from our lives, instead we mitigate it through automation and digital systems. The modern person does not need to have the same level of skill than those of the previous generation, because we have designed tools to simplify the experience while maintaining an acceptable performance. As this is true for the progression from the bow and arrow to the crossbow, so it is for network management to autonomous networks. This work aims to reduce the complexity of modern network management through the coordination of technology enablers. In section 1.1 a background is provided on important network management concepts and the role of these concepts in modern networks. This includes a detailed explanation of network management, rules-based decision making, the evolution of networks, the expanded role of policy in these networks and ways in which policy can influence future networks through the selection and actioning of optimisation strategies. In section 1.2 the core research question is presented followed by reasoning for the importance and applicability of the study. This question is then compartmentalised into three sub-questions. In section 1.3 the contributions of the work to date are presented, categorising them in to major and minor contributions. In section 1.4 all publications related to this work are listed as full citations.

1.1 Background

The term Network Management encompasses duties related to the administration and management of a networked system. Hegering describes network management as encompassing all measures which ensure the effective and efficient operations of a system, in accordance with corporate goals [10]. Clemm's description elaborates on these measures as activities, methods, procedures and tools applicable to the operation, administration, maintenance and provisioning of networked systems [11]. Network Management has four key objectives. To ensure the system is running smoothly by monitoring and changing configuration parameters for resources and services. To reduce management complexity by adopting tools for automation and rules based decision making. To provide reliable services by supporting means for fault management and repair. To track and report network resource and service usage for cost consciousness.

Rules based decision making is often referred to as policy. Policy has played an important role in network management systems by applying rules to manage complex network behaviors. As networks became more complex, policy adapted to mitigate network complexity while simplifying policy implementation through standardization, modeling and the use of patterns in policy execution. However as networks evolve further and 5G networks are realised, policy will be required to evolve to relieve this complexity. Adaptive policy has been proposed as a solution to manage this new level of complexity. Adaptive Policy EXecution (APEX) is a carrier-grade, production ready environment capable of scalable policy authoring, deployment and execution. The environment was based on Universal Policy Theory (UPT) implementing immutable policy infrastructure according to the Universal Policy Execution Environment (UPEE), both UPT and UPEE are described in [12]. For policy to become adaptive policy several requirements must be achieved, these requirements were discussed in [13] and in [14] in the context of a Self Organising Network (SON) use case. APEX was introduced in [15] and a mobile network security use case for adaptive management was described in [16]. Novel implementation for context aware policies with distributed context within APEX was detailed in [17]. Policy evolution and requirements were discussed in [18] including closed control loop management, a novel policy model and underlying resource modeling. APEX was released as open source by Ericsson in January 2018 and is currently adopted as a new Policy Decision Point (PDP) in the ONAP Policy Framework.

As networks evolved to become open and programmable this allowed for the realization of more dynamic and adaptable networks. However this evolution also forced the systems managing these networks to be equally dynamic and adaptive. Network controllers have the capability to mediate the programmable access to these networks and policy based management can be used to drive these controllers. For policies to perform this task effectively they must first support a more holistic view of the network should be managed. This means users, services and context should be recognised and considered when analyzing the performance of networks. Adopting a closed loop automation (e.g. COMPA[19], ONAP[20]) approach where the network, network controller and policy are coordinated to manage and optimise the network can allow for more autonomic networks where user and service aware network configuration can be adaptively controlled, orchestrated and managed. This approach produces two apparent problems, dynamic network which are dynamically managed and driven by adaptive high level goals are difficult to plan and manage creating networks that are not correctly dimensioned or demonstrate unanticipated behaviors once deployed. Also dynamic systems that are not adequately modeled are hard to simulate or evaluate analytically before deployment or require adjusting after deployment.

The amount of traffic on modern networks is continuously increasing, the Ericsson Mobility Report [21] acknowledges video as the most prevalent service used on mobile networks today, averaging 14 exabytes per month in the third quarter of 2017. This is likely to increase further as popular immersive video formats generate 4 to 5 times the traffic of standard video [22]. Assuming continuous growth, video will account for 75% of all mobile data traffic by 2023. This demand can be relieved by the arrival of 5G networks, promising larger data rates, low latency, higher bandwidth and a new level of energy efficiency realised through the implementation of networking concepts such as Virtualised Network Functions (VNF), Network Slicing and Mobile Edge Computing (MEC) to name a few. These concepts contribute to the 5G promise, but also add more complexity in regard to both network and video management. To combat this we have see a drive toward automation, adaptive policy and machine learning paradigms in an effort to mitigate this complexity.

A common scenario can be described as streaming video from a number of content providers using specific applications or embedded in HTML pages, all on demand. New usage scenarios use video as a facilitator for a complete, purpose driven user experience.

An example is discussed in [23] in respect to worldwide sporting events. In this example video plays an important part in enhancing the event experience comprising all aspects of the event for participating and remote viewers. Increased deployment of video-enabled devices and better QoS promised by 5G networks should create a technical environment facilitating many new, unforeseen usage scenarios.

Approaches and solutions are available for optimising particular network nodes, cross-layer optimization, end-to-end optimization, and optimization of OTT flows. The developed techniques include pre-selected and best-effort quality, variable bit-rate based on network conditions or client Quality of Service (QoS) parameters, single or hierarchical caching of videos, optimization of local or intermediate buffers, Quality of Experience (QoE) driven methods, and of course hybrids using two or more of the above. In [24] the authors present a cross-layer optimization algorithm using cache and buffer methods, aiming for fast video delivery. The algorithm is evaluated in a mobile network using relaying stations. Typically, video optimization is done *out-of-network* using transcoding, transrating, time-shifting, and pacing [25]. *In-network* optimization techniques are becoming popular, utilising available information of the underlying network. An optimization of radio base stations for multi-user video streaming is detailed in [25].

HTTP-based Adaptive Streaming (HAS) is an *in-network* technique that allows for better resource utilization using multi-layer information to deliver and if required adapt the best possible video stream given network conditions. A survey of HAS can be found in [26]. A few limitations remain: solutions are client-driven, hard to direct by network operators, and not policy-driven. Combining QoE with HAS promises to overcome them. In [27] the authors present in-network QoE management for video streams. This approach provides an interface for the network operator to steer the optimization process, thus facilitating policy control. In [28] the authors add fairness to the QoE management using client-transparent proxies. In February 2017, ISO has published the MPEG Server and Network Assisted Dynamic Adaptive Streaming over HTTP (SAND DASH) standard for video streaming over the Internet[29]. A programmer's introduction to and a demo of SAND can be found on Github [30]. In [31], the authors develop a multi-server multi-coordinator framework, which helps to model groups of clients accessing spatially distributed edge servers for replicated video content.

The rapidly changing capabilities and performance challenges of emerging telecommunications network requires a re-evaluation of traditional network configuration paradigms. Typically, networks adopted a policy based management system to govern their behaviour. This policy based approach has simplified the complex task of managing a network by specifying a set of preconfigured rules based on known scenarios[32]. However the constant development of new technology along with the rise of virtual networks, NFV and SDN has resulted in an exponential increase in complexity for network management systems[33]. This has created issues for the current policy based approach. Traditional policy based management systems encode logic to select from a set of predefined options rather than dynamically make a context-aware adaptive decision. Therefore, as changes occur and the scale of management tasks increase, existing management systems inevitably become static and brittle as they get more complicated[34]. Ericsson has developed the APEX (Adaptive Policy EXecution) engine to addresses this issue[15].

Intent based networking has been proposed as an approach to reduce complexity in network configurations [35]. The concept takes a modeling language and uses it to describe an abstract view of an intended network model [36]. Intent is currently topical in research, viewed as a mechanism to enhance network flexibility and management, however there lacks a unified definition for Intent Driven Networks [37]. This suggests intent requires more maturity before a consensus can be achieved and it can progress towards standardization. The authors of [38] indicated that intent based networking has not evolved since 2015 in regard to frameworks, platforms and tools however advances in artificial intelligence such as Natural Language Understanding are expected to increase its adaption in the future. Adopting Natural Language Understanding within an intent based system can produce abstract rules for the structuring of intent information. This would enable flexible representations of intent but is not a core requirement for the realization of these systems. Policy has played a core role in the realization of many intent driven mechanisms, often used to create actionable responses from abstract network statements [39]. This has been seen at multiple levels of the network from intent driven forwarding rules for programmable switches [40] to virtual network management platforms [41] and the orchestration of dynamic service chaining of Virtual Network Function [42]. Today we see more and more systems support intent languages, frameworks and interfaces. The incorporation of these approaches provide elements of abstraction between the request and the execution. However mapping is still required to get from the abstract to the real. This mapping often introduces elements of rigidity into the system.

Intent is a declarative approach to specifying goal oriented statements for the simplification and abstraction of complex network management [39]. Intent has often been coupled with SDN (Software Defined Networking) as a means to facilitate complex network operations through a simple objective orientated interface [43]. Recently we have seen Intent featured in a variety of scenarios such as path optimization [44], data protection [45] and cloud management and orchestration [46] [47]. The most prevalent incorporation of Intent is seen in the area of SDN, through the development of Intent frameworks for the Open Network Operating System[1] (ONOS) and OpenDaylight[2] (ODL) platforms. Intent based modeling languages such as IB-NEMO have also been adopted by these platforms, integrating a North Bound Interface for SDN controllers to allow the injection of Intent into these systems. The language has also been extended through research to allow more flexibility for Intent descriptions [35].

The development of these components and features in the open source environment and the expansion of capabilities through research increases the accessibility of Intent driven network management for network operators. However a large proportion of this research is focused on the creation of mechanisms to achieve the realization of Intent described goals in a closed off environment. Intent driven interfaces are designed specifically for components within the research scenario. This approach to intent would require every component in the system to support large shared models to provide understanding of intent between components. As a result the system is inflexible, requiring a large amount of work to introduce new components and functionality to the system. The goal of this work is to tackle the inflexibly and extensive modeling challenges through development of a flexible intent interpreter that utilises already existing component models.

Large standardisation efforts have been seen in relation to intent and its role in autonomous networks. The ETSI Zero-touch network and Service Management (ZSM) group aims to accelerate the definition of required architecture and solutions for full end-to-end automation of network and service management. In Decemeber 2020, a study began to investigate intent as a key enabler in autonomous network and service management within the ZSM framework. Work item ZSM-011 is currently in early

[1]`https://opennetworking.org/onos/`
[2]`https://www.opendaylight.org/`

draft, but aims to provide guidelines on utilising intent-driven management interfaces between framework consumers and management domains. The Network Management Research Group (NMRG) has included intent-based networking as one of their three priorities for investigation for a five year period between 2017 and 2022. During this time they have motivated intent as a research topic, identified several research challenges and contributed in the areas of intent classification[48], concept definitions[49] and service assurance for intent-based networking architecture[50].

1.2 Research Question

In this section we will detail the core research question of our study, followed by a number of sub-questions that arose as our knowledge and understanding developed.

1.2.1 Core Research Question

Is it possible to leverage intent-based management for private networks, without adopting ontology based model specifications?

Policy has played a key role in network management, as networks have evolved to become more dynamic and complex traditional policy has become static leading to the adoption of adaptive policies for dynamic decision making. The features which provide the adaptive functionality to policy also allow for the incorporation of machine learning paradigms such as straight forward neural networks. Typically machine learning is applied in post, generating a model that can be applied to new data to provide an insight. Adaptive policy can take relatively lightweight learning algorithms and apply them within the policy environment, learning from the network behavior as context information and quickly applying insights in the form of real-time decisions. Taking these decisions and implementing them through network management, an optimization component can introduce fast context-aware network optimization and repair. Taking this execution further, by leveraging an intent-based approach, both the north and south bound interfaces to this system can be abstracted. The user could then engage with the system, while being agnostic of the underlying components. This would remove a previously required expertise as the specificity of the components is no longer a consideration of the user resulting in a more straightforward experience. This is important functionality when dealing with modern dynamic networks as it will mitigate some of the complexity for the user while introducing more informed real-time decisions, increasing the possible application and responsibility of policy. The flexibility provided by adaptive policy, boosted by the insights of machine learning, exposing a user friendly intent-based interface would allow potential new technology enabled use cases to benefit from straightforward intuitive integration and execution.

1.2.2 Sub-Research Questions

Is it possible to design, develop and validate intent-based action generation using inductive operation discovery?

Our work presents a flexible approach to intent driven systems through an interpreter for intent realization. The mechanism of the interpreter is demonstrated through interactions with mock slice manager functions running on an open source, functions as a service platform. The interpreter accepts intent messages containing a request in the form of an English sentence, along with required parameters for the request. This information is then used to identify the appropriate functions available to the interpreter. These functions are described in the Functionality Template model. Once identified, the Functionality Template guides the building of an action in response to the intent message. This work was then extended into the Adaptive Intent Realisation (AIR) reference architecture. The machine learning processes were refined and integrated with feedback mechanisms. Utilising the reporting concepts outlined in Intent a running context of the system is maintained to inform future decisions.

Is it possible to leverage machine learning paradigms to evaluate policy decisions within the decision making process?

Our work introduces control logic for the APEX engine that implements a directed feed forward neural network to enable network path selection for multimedia streaming applications. Further control logic is introduced that calculates a MOS (Mean Opinion Score) to evaluate the current performance of the service, which is fed back to the neural network to complete the closed loop optimization. Initial results illustrate how service specific metrics can be used to inform adaptive control within the context of a real network control scenario implemented by the APEX engine. This was expanded to employ similar evaluation processes for in-the-moment decisions. Allowing the system to learn and adapt from the outcomes of previous decisions, while also building an awareness of potential impacts to performance or conflicts with existing goals.

Is it possible to improve user experience through adaptive decision making in low level network control, with respect to QoS capabilities and business goals?

Expanding from our previous work we proposed adaptive policy as an integrated, unified video optimization approach. Identifying key requirements as:

- the measurement of QoS capabilities of the network,
- translation of capabilities into Service Level Agreement (SLA) and QoE specifications,
- a closed loop monitoring and repair policy system.

Meeting these requirements we would then move towards a mathematical model for policy that governs the network for service assurance, a very good candidate being ΔQ (based on the model developed in [51]). We described our initial work in probing a unified approach to video quality assurance that aims to integrate (virtually) any of the usage scenarios discussed above with the existing in-network video optimization techniques we described in the context of 5G mobile networks. To address the requirements of the approach we specified a system architecture and conducted experiments which:

- generate Mean Opinion Score (MOS) to evaluate network path quality,
- deploy adaptive policy for network path consideration,
- implement decisions through the network controller.

Is it possible to test and evaluate management operations for dynamic networks in a manner that reflects the real environment before deployment onto live systems?

Our work details a realistic test bed that emulates the network, interfaces with real controllers using real policies and carries real traffic. Therefore, when the policy changes, their effects can be analyzed in a realistic manner. This test bed has a number of key requirements:

- it must be lightweight,
- support realistic behavior such as actual analytics frameworks, policy systems and real network controllers,
- should be open to allow for integration of new components,
- capable of fast loop times,
- support fast reproducible experiments, allowing rapid evaluation of different management strategies.

While there are numerous network simulators and emulators, several candidate network controllers and many network analytics platforms and rule/policy systems, there lacks an existing reusable test bed for evaluation of closed loop network management that meet our requirements. We developed our test bed with particular focus on allowing for quick evaluation of alternative network policies executing in APEX [15]. With this test bed policy authors can deploy and evaluate their policies on an emulated network and monitor the real time affect their policy has on the network.

1.3 Contributions

In this section we list the contributions of this work. The contributions are broken into major and minor:

Major Contributions

1. The specification of an intent based architecture for the execution of translation and inference policies evaluated by predictive ML models.
2. The specification of an adaptive architecture for execution of cognitive policies within the COMPA closed-loop architecture.

Minor Contributions

1. The generalisation of intent definition, policy composition and execution mechanism for an Adaptive Policy environment.

2. An industry proven evaluation environment with usability parameters for end users.

1.4 Publications Arising From This Work

[J. McNamara et al., "NLP Powered Intent Based Network Management for Private 5G Networks," in IEEE Access, vol. 11, pp. 36642-36657, 2023, doi: 10.1109/ACCESS.2023.3265894.]

[J. McNamara, E. Aumayr, L. Fallon and E. Fallon, "A Flexible Interpreter For Intent Realisation," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-6, doi: 10.1109/NOMS54207.2022.9789910.]

[J. McNamara, L. Fallon and E. Fallon, "A Mechanism for Intent Driven Adaptive Policy Decision Making," 2020 16th International Conference on Network and Service Management (CNSM), 2020, pp. 1-3, doi: 10.23919/CNSM50824.2020.9269073.]

[J. McNamara, L. Fallon and E. Fallon, "A Hybrid Machine Learning/Policy Approach to Optimise Video Path Selection," 2019 15th International Conference on Network and Service Management (CNSM), 2019, pp. 1-5, doi: 10.23919/CNSM46954.2019.9012667.]

[J. McNamara, S. d. van Meer, L. Fallon, J. Keeney and E. Fallon, "An Adaptive Policy Approach to Video Quality Assurance," 2018 14th International Conference on Network and Service Management (CNSM), 2018, pp. 363-367.]

[J. McNamara, J. Keeney, L. Fallon, S. van der Meer and E. Fallon, "A testbed for policy driven closed loop network management," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1-6, doi: 10.1109/NOMS.2018.8406144.]

Chapter 2

Literature Review

This chapter describes telecommunication technologies, network management strategies, the evolution of intent and advanced analytics through machine learning paradigms. These technologies and strategies play a large role in our modern networks. Legacy telecommunication technologies are still used in our modern networks due its iterative development and network management services engage with these technologies. Fourth Generation (4G), Fifth Generation (5G) and Sixth Generation (6G) mobile networks are described in section 2.1. Network management strategies such as policy, control loops and the adaptive policy approach are described in section 2.2 with a focus on the limitations of the approaches and the advancements made to produce adaptive network management solutions while reducing operation complexity. The evolution of intent is described in section 2.3 followed by detailed descriptions of current translation techniques. Machine learning techniques are described in section 2.4, these paradigms are expected to play a larger role on network management decision-making through standalone analytical components or embedded in policy driven components. In section 2.5 primary reference architectures are presented followed by detailed structures of open-source network management platforms and autonomous networks. 5G platforms developed under the H2020 program in parallel with this research are also presented with contributions relevant to this research.

2.1 Telecommunications

Telecommunications is the term used to describe the exchanging of information such as voice, data and video over wired and wireless mediums. The Fourth Generation of mobile networks is described in subsection 2.1.1, highlighting key technologies that enabled new use cases and application for mobile network devices. The Fifth Generation of mobile networks is described in subsection 2.1.2. This section highlights the core technologies of modern networks, resulting in increased complexity. The Sixth Generation of mobile networks is described in subsection 2.1.3. This section gives a brief overview of 6G through drivers, potential use cases and proposed technologies. Although still an abstract concept, 6G is expected to bring further complexity as a debt to improved performance.

2.1.1 Fourth Generation 4G

4G was designed to provide appropriate speeds for popular applications at the time such as HD video. As common use progressed beyond emails, messaging applications and sharing photos, networks needed to improve both speed and capacity to meet new demands. This was achieved through a number of architectural changes to the network along with the introduction of new technologies. This section presents two of the key features in Fourth Generation mobile networks:

2.1.1.1 MIMO

Multiple Input and Multiple Output is widely considered a key technology in 4G. It provides significant increases in throughput and link range while not requiring an increase in both bandwidth or power to transmit. Massive MIMO, that is MIMO deployed on large scale networks, can utilise multiplexing to largely increase bandwidth efficiency [52]. Energy efficiency is also achieved through an increase in the number of antennas, allowing for a more directed signal towards the User Terminal (UE) [53].

2.1.1.2 Frequency Domain Equalization

Orthogonal Frequency Division Multiplexing (OFDM) is used to encode data on multiple carriers, it emerged as a leading physical layer technology in wireless communications [54]. This technology was used to develop Orthogonal Frequency Division Multiple Access (OFDMA) which is a multi user approach to OFDM. OFDM is used to encode data on multiple carriers and by distributing these carriers into subsets, that can be accessed on a user by user basis, multiple users can transmit small amounts of data using different subsets simultaneously. [55]

2.1.1.3 Positioning Reference Signals

The ability to position outdoor devices through Global Navigation Satellite Systems (GNSS) [56] [57] has sparked new avenues of research for indoor positioning systems with precision. Today positioning technologies can be divided into radio and non-radio frequency based positioning. An example of non-radio frequency based positioning is the use of sensor data from mobile smart phones [58] [59] [60] [61]. However there are also more specialised mechanisms such as optical based positioning or magnetic field positioning.

Release-9 of LTE networks saw the support of positioning feature standardised under 3GPP with the introduction of positioning methods. Advanced techniques were also investigated for Release 16 of NR [62] and 3GPP NR positioning study conclusions were presented in [63]. In [64] the authors concluded that 3GPP NR systems should support solutions of observed time difference of arrival (OTDOA), uplink time difference of arrival (U-TDOA), angle information, multi-cell round trip time (Multi-RTT), and enhanced cell-ID (E-CID). Downlink based positioning is supported through the Positioning Reference Signal (PRS). PRS has a more regular structure and larger bandwidth when compared to LTE. This allows for more precise correlation and time of arrival (ToA) estimation. The position of the UE can be determined by comparing the ToA of a certain device to a number of base stations in the area.

2.1.2 Fifth Generation 5G

5G will expand mobile technologies, supporting new and diverse devices and services. 5G promises more capacity and lower latency enabling new use cases such as larger scale IoT and self driving cars. This section presents three key technologies in Fifth Generation mobile networks.

2.1.2.1 Network Softwarization - NFV/SDN

An overall approach for designing, implementing, deploying, managing and maintaining network equipment and/or network components by software programming [65]. It enables flexibility, adaptability, and complete network reconfiguration on the fly based on timely requirements and behaviours by considering cost and process optimization in the overall maintenance of the network life-cycle [66]. Traditional network services are based on proprietary appliances to provide different network services. In these purpose built hardware systems and accompanying architectures the devices are vertically integrated and have distributed control intelligence[67]. This introduces limitations in regard to management, configuration and flexible implementation of high level network policies [68].

A possible solution to this is the use of Network Function Virtualisation (NFV). The virtualisation of network functions enables characteristics like flexible provisioning, deployment and centralised management. Current network services rely on proprietary appliances and different network devices that are diverse and purpose-built [69]. This allows network operators and service providers to move away from purpose-built hardware instead implementing network functions in a virtualised environment running on standard servers and off the shelf hardware. Cost is also taken into consideration when encouraging the move to NFV. Proprietary hardware and applications are expensive to create and maintain. Both capital expenditure (CapEx) and operating expenditure (OpEx) can be reduced through the decoupling the application from the respective hardware [70] [71]. NFV supports the multi tenancy of network and service functions utilising the same physical hardware systems for distinct services, application and tenants. NFV was a key factor to allow the virtualisation of networks [70] [71], breaking the connection of network functions from dedicated hardware and packaging the functionality into a software package runnable on commercial off the shelf equipment [72] [73] [74].

New technologies and the evolution of existing technologies has identified an issue with traditional networks, primarily a lack of flexibility. One way to address this was the development of Software Defined Networking and the incorporation of abstractions in the network architectures [75]. With NFV decoupling network functions from dedicated hardware, Software Defined Networking (SDN) aims to decouple the control plane from the data plane[76]. This is achieved by introducing elements of programmability into the network in the form of SDN controllers and commonly associated Openflow switches [77]. With this approach routing tables and forwarding rules can be programmatically generated [78]. The northbound interface allows SDN controllers to be incorporated into network control mechanisms. Guided by the control layer, high level goals can be realised through the controllers southbound interface. Southbound APIs interface with data plane devices allowing forwarding devices to be controlled in a programmable, goal oriented, flexible manner. [79]

Although NFV and SDN are completely different concepts with unique aims, when implemented in a cooperative manner they provide a new level of functionality and flexibility to modern networks. NFV aims to decrease cost through the virtualisation of network functions allowing for fast provisioning and deployment, while SDN aims to provide flexibility through a programmable network architecture by decoupling the control plane from the data plane. Today we see many examples of virtualised SDN controllers such as the Apache licensed Floodlight SDN controller and ONAP's OpenDaylight controller. This allows for NFV enabled dynamic and optimised controller placement with the flexible control provided by SDN.

2.1.2.2 Network Slicing

The arrival of 5G networks enabled the integration of cross-domain network through the enabling of logical slices of the network allowing distinct domain and technologies to create tenant / service specific networks resulting in virtual networks that are highly optimised to fulfil specific use cases such as video streaming. Network Slicing will realise an end-to-end (E2E) vision from the mobile edge through the mobile transport layer, including front-haul (FH) and back-haul (BH) segments, resolving before the core network (CN). The 5G architecture utilises slicing which decouples Virutal Network Functions (VNFs) and Software Defined Networking (SDN) from the physical infrastructure through resource abstraction technologies.

This technology allows for fully decoupled end-to-end (E2E) networks sitting on a shared physical infrastructure, as a result well-known resource-sharing technologies such as multiplexing will be implemented on the orchestration layer as oppose to the network layer. This change will improve Quality of Experience (QoE) for subscribers as well as increasing the network operability for Internet Service Providers (ISPs) and network operators. [80]

The concepts, use cases and requirements for management of network slicing in 5G mobile networks is presented in 3GPP Technical Specification [81]. Business level requirements are described along with high level features such as Self-Organising Networks (SON) and closed-loop SLA assurance. Many use cases are described for network slicing involving coordinated support from Network Operators and Communication Service Providers.

Other standardisation bodies such as ETSI have produced Group Specifications [82] detailing E2E network slicing management solutions and related management interfaces. These include E2E network slicing including provisioning, performance assurance and fault management of an E2E slice instance across multiple management domains.

Network slicing introduces new and interesting capabilities to the network. However slicing a physical infrastructure into multiple virtual networks will require a form of management framework to handle slice configuration. More importantly given the ability to generate service-specific slices these slices may have conditions or requirements in terms of QoS (Quality of Service) and depending on network conditions will require reconfiguration. Previously we discussed the drive toward Autonomous Networks and the importance of the closed control loop. Utilising a closed control loop such as COMPA, would allow Adaptive Policy to influence slice reconfiguration through informed decisions as Adaptive Policy would have both network and customer information available at decision time.

2.1.2.3 Location Management Function

3GPP positioning is realised through a location server in the LTE domain or a Location Management Function typically found in 5G networks. This server/function gathers and processes data related to the positioning of devices in the network. This information can then be forwarded to management components/functions to inform operations in improve performance/reliability/energy efficiency etc. One or more positioning methods, provided by the PRS, may be called or combined to report device locations depending on the scenario and required accuracy [83].

In current research, the location where this data is computed is under discussion. Some systems have delegated these calculations to take place and be stored on the device while others perform these calculations on the cloud. Both of these approaches may result in degradation of performance through incurred delay, battery usage and network resources [84]. Hybrid approaches combining device and cloud may also affect accuracy through the decoupling of the final position calculation and the storage of training data sets. The authors of [85] demonstrated that offloading position information to the edge mitigated computing complexity and reduced energy consumption at the Base Station. This approach also encourages private network operators to leverage these calculations for their own networks, as they can perform the calculations quickly, without requiring their data to leave their network and can continue to comply with a 3GPP standard.

2.1.2.4 Beamforming

Beamforming is a signal processing technique used for directional signal transmission or reception. Spatial selectivity/directionality is achieved by using adaptive transmit/receive beam patterns. When transmitting, a beamformer controls the phase and relative amplitude of the signal at each transmitter antenna to create a pattern of constructive and destructive interference in the wavefront. When receiving, signals from different receiver antennas are combined in such a way that the expected pattern of radiation is preferentially observed. [86]

Beamforming can be achieved in the digital baseband, analog baseband, or RF front end. With digital baseband beamforming and multiple RF chains, it is possible to transmit multiple streams of data simultaneously, thus enabling SDMA or MIMO operation. [87]

In general, digital beamforming provides a higher degree of freedom and offers better performance at the expense of increased complexity and cost due to the fact that separate FFT/IFFT blocks (for OFDM systems), digital-to-analog converters (DACs), and analog-to-digital converters (ADCs) are required per each RF chain. [88]

2.1.2.5 Management and Orchestration

The authors of [89] highlight Open Standardisation as an enabler in 5G network management and orchestration. SDN and NFV enabled networks to be programmable and dynamically instantiated. Detailed in IETF RFC 6241 [90], NETCONF provided configuration and management while being transport protocol independent meaning access was not restricted. NETCONF also allowed administrators to set variables from features Configuration and operation data was separated which allowed administrators to set variables from features like statistics and alarms. Also, transaction operations were supported to ensure the completion of configuration tasks. NETCONF defined mechanisms for the configuration and access of network elements, the information was described through the YANG data modelling language. YANG abstracted the configuration of devices and services for the network administrator. It also simplified configuration management by supporting validation features for input data. IETF is a leader in the standardisation of these models for network management.

Similar to NETCONF, the RESTCONF protocol provides a programmatic interface for create, read, update and delete (CRUD) operations that can access data defined in YANG based on Hypertext Transfer Protocol (HTTP) transactions. This allows web-based applications to access a variety of information models along with remote procedure call (RPC) operations of the networking device in a flexibile manner. The purpose is then similar to the one described for NETCONF. Regarding the orchestration of services and the management of VNF lifecycles, Topology and Orchestration Specification for Cloud Applications (TOSCA) was proposed as viable approach.

TOSCA is detailed in OASIS Standard [91]. This specification provides a language to describe service components and their relationships using a service topology, and it provides for describing the management procedures that create or modify services using orchestration processes. TOSCA templates are designed to support describing both NS descriptors (NSDs) and VNF descriptors (VNFDs). TOSCA is a service-oriented description language to describe a topology of cloud-based web services, their components, relationships, and the processes that manage them, all by the usage of templates. TOSCA supports large areas of service configurations such as resource requirements, VNF life-cycle management and the definition of workflows and FCAPS management of VNFs. This allows implementations of given behaviors to be instantiated and a service template. The system details the structure of different topologies, in template form, as a collection of node templates with relationships. The approach creates a directed graph template. Node and relationship templates specify the properties and the operations (via interfaces) available to manipulate components. The inclusion of the relationships between the nodes in the topology template allows the orchestrator to interpret the order of instantiation through the dependencies of nodes. TOSCA templates can also inform other life-cycle management operations such as scaling.

NETCONF, YANG and TOSCA complement each other in their approaches to network configuration and management. The TOSCA templates can manage the life-cycle of VNFs while dynamic configuration at run-time can be realised through YANG models over the NETCONF/RESTCONF transport protocols. The interplay is facilitated by architectural propositions like the integrated SDN control for tenant-oriented and infrastructure-oriented actions in the framework of NFV. [89]

2.1.3 Sixth Generation 6G

6G plans to expand further on mobile technologies, utilising higher frequencies while supporting higher capacity and lower latency. In this section we will detail some of the key areas of 6G through the lens of drivers, use cases and technologies.

2.1.3.1 Drivers, Use Cases and Technologies

6G brings with it the potential for new and exciting applications. The authors of [92] highlight the limitations of existing 5G mobile networks for emerging applications such as Internet of Everything, Holographic Telepresence, collaborative robots, and space and deep-sea tourism. A large list of additional applications of 6G are presented in survey papers such as [93] and the volume of work already present in this area shows that the roles of mobile networks will continue to expand as performance improves.

Trustworthiness has been proposed as a key driver for 6G mobile networks. It can be broken down into a list of important areas such as: privacy, security, integrity, resilience, reliability, availability, accountability, authenticity and device independence. Each of these areas present unique challenges for a range of engineering disciplines.[94]

Sustainability has been a key driver of many technological advances in recent years. On a social level we have become more environmentally aware and have made conscience efforts towards cheaper and greener alternatives. This change is also reflected in industry where large research efforts have been put into topics like energy efficiency. 6G provides opportunities to produce more sustainable networks in the future. The authors of [95] aim to disseminate the latest theoretical and experimental works in the domain of energy-efficient communication and computing technologies towards enabling massively connected, fully intelligent and sustainable green 6G networks.

Another driver of 6G networks is the expansion of AI and Smart systems in an effort to achieve a simplified life. The performance and capability improvements expected from 6G may enable existing smart systems to provide a wider range of more powerful services to it users. These existing smart systems that were realised in modern 5G networks are now being proposed in the 6G environment. The authors of [96] looked at key 6G technologies and present a layered network approach to resource allocation problems in dense city networks. The authors of [97] looked at the technologies of 6G and detailed the potential advances in healthcare. These include services like smart monitoring of remote patients through intelligent wearable devices, wireless communication of scans with AI assisted diagnosis and the utilisation of smart vehicles in emergency services. The authors of [98] produced a survey paper which investigated the applications of ML technologies in the vehicular network toward the future 6G intelligentized network. Focusing on the distinctive challenges in the existing vehicular communication, networking, and security, and investigated the corresponding ML-based solutions

Current white papers on future networks provide some insight into the industry's perspective on 6G. The authors of [99] provide a detailed description of research activities globally along with prominent industry use cases and technology scenarios such as ubiquitous services, uniform coverage and smart agriculture. The authors of [100] highlight the technology elements of 6G including the role of intent-based management in cognitive networks.

2.2 Network Optimisation Strategies

In this section we will provide detail on the network management tools used as part of our research. Policy is a straightforward rules based approach to network management through Event, Condition, Action sequence decision making. Control Loops were utilised to add autonomic features to the network through the use of a feedback of actions and monitoring. Adaptive Policy then introduced a context aware and dynamic decision making mechanism which addressed some of the limitations of the control loop. Finally Intent which has evolved from a policy approach to a fully fledged modern network management framework.

2.2.1 Policy

This section describes policy's role in network management and the progress of policy to stay flexible while providing informed and context aware decisions for the volatile and complex modern network. The section introduces traditional approaches to policy, the limitations of the approach and the requirements of Adaptive policy.

Policy was introduced into network management to provide decision making capabilities in the area of fault management. Policies are typically modelled and there are a number of structures such as General Policy Modelling Language (GPML)[101] and Common Information Model (CIM)[102]. The IETF specified the CIM in standards through RFC 3060[103] and is a prominent reference in policy modeling. The terminology for Policy-Based Management is defined in RFC 3198[104]. Regardless of the modelling language used policy tends toward a pattern of execution. This can be described as Event Condition Action (ECA) where an event has triggered the execution of the policy, a condition is applied to the trigger generating an action as output[105]. ECA is still prominent policy-based management technique today. ETSI produced a Group Report detailing a gap analysis in the area of context-aware policy management[106]. This report analysed the work done in various SDOs and open source consortia on policy-based modelling. ECA is detailed as an example of imperative policy types and linked to the IETF RFC 8328[107], a data model for policy abstraction in a network management environment.

Internet Service Providers (ISP) have adopted policy as a means to reduce Operating Expense (OPEX). Through policy decision making, ISPs can reduce the number of human-in-the-loop interactions letting the network run in a more autonomic manner. Autonomous Networks, networks that operate with little to no required human intervention would reduce OPEX costs even further expanding the role and responsibilities of policy in the network. Closed loop control systems have be proposed as a mechanism to achieve autonomous networks.

2.2.2 Closed Control Loop Systems

ETSI has produced a Group Report detailing prominent control loop architectures that can be used in modular system design[108]. The purpose of the Group Report was to inform a Group Specification for the Experiential Networked Intelligence (ENI) System Architecture detailed in [109]. The Group Specification defines a high-level functional abstraction of the ENI System Architecture. The terminology for the ENI System is specified in [110].

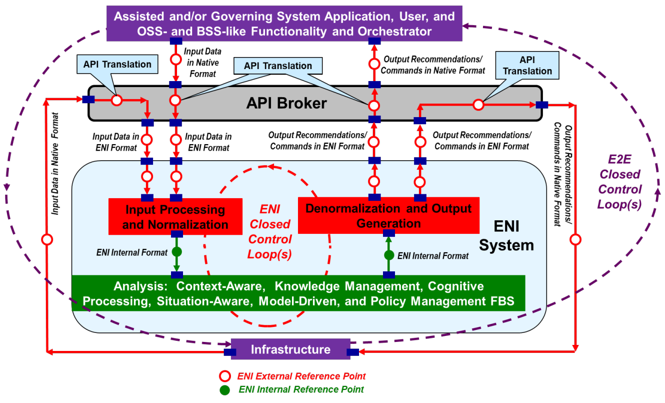

Assisted and/or Governing System Application, User, and OSS- and BSS-like Functionality and Orchestrator
API Translation
Input Data in Native Format
API Translation
Output Recommendations/ Commands in Native Format
API Translation
API Broker
Input Data in ENI Format
Input Data in ENI Format
Output Recommendations/ Commands in ENI Format
Output Recommendations/ Commands in ENI Format
Input Data in Native Format
Output Recommendations/ Commands in Native Format
E2E Closed Control Loop(s)
Input Processing and Normalization
ENI Closed Control Loop(s)
Denormalization and Output Generation
ENI System
ENI Internal Format
ENI Internal Format
Analysis: Context-Aware, Knowledge Management, Cognitive Processing, Situation-Aware, Model-Driven, and Policy Management FBS
Infrastructure
ENI External Reference Point
ENI Internal Reference Point

Figure 2.1: High Level Functional Architecture of ENI[1]

IBM introduced the Monitor Analyse Plan Execute Knowledge (MAPE-K) control loop in 2005 which detailed an architectural plan for autonomic computing [111]. Using this approach monitor, analyse, plan and execute share a knowledge pool which allows for more informed decisions. The FOCALE autonomic network architecture was introduced in [112] allowing business rule orientated resource and service allocation, self monitoring, self adaptation, modelled data verification and had the capability to add new data dynamically. Ericsson introduced the Control Orchestration Management Policy Analytics (COMPA) framework in [113]. This approach broke the network down into separate roles and introduced a second COMPA loop to manage that area of the network. This nesting of COMPA loops provided another layer of automation to the network allowing high level COMPA loops to communicate goals to lower level COMPA loops where changes can be executed and monitored. However the Policy of COMPA has displayed a number of limitations. These limitations are listed in [14]:

- Aggregates of elements may exhibit behaviour not predictable from knowledge of individual behaviours
- Causal determinacy is still limited by simple statistical analysis and rudimentary correlation approaches
- No ability of the system to “go beyond” static knowledge and procedures.
- No or limited feedback within the loop (leading to incorrect/sub-optimal decisions
- Only taking context into account at the decision point
- No tie back to the business or system goal in a dynamic manner.

As policy plays a key role in all control loops these limitations would have a negative impact on the viability of autonomous networks. This has motivated the development of Adaptive Policy.

2.2.3 Adaptive Policy

ETSI has produced a Group Report detailing a study on policy models in NFV-MANO[114]. This report primarily focused on TMF, IETF and ONAP policy models. The ONAP Policy model used in this study was Adaptive Policy EXecution. APEX was the only policy model deemed to have satisfied or considered all recommendations of the NFV-MANO.

The authors in [14] discussed the need for telecommunication control and management policies to evolve to cope with the flexibility and dynamicity of network components. They highlighted the deficiencies and limitation with policy and proposed Adaptive Policy as a means to address these shortfalls. The described Adaptive Policy has three characteristics:

- Context-aware decision making
- Can change decision-making behaviour based on external activity
- Can change decision-making behaviour based on internal activity

Instead of viewing policy as an Event Condition Action (ECA) the authors divide the policy into four states with clear separation of concerns. These states are referred to as Match Establish Decide Act (MEDA). These can then be executed on a flexible state machine with interchangeable state logic carrying behaviour. The Adaptive Policy EXecution (APEX) engine would allow for easy deployment and controlled policy execution. Notable improvements were concrete models for context and new mechanisms for conflict detection. The integration of meta-data and context was outlined as future work for advanced conflict detection. In [15] the authors present the APEX engine. The policy engine was capable of running Adaptive Policies which could adapt to changes in goals, environment and incoming event context. The advantage over ECA policies was the incorporation of meta-data which enabled the conflict detection on a per state per execution basis. APEX addressed the deficiencies and limitations found in the incorporation of policy into the COMPA control loop allowing policy to play a much larger role in the development of Autonomous networks.

2.3 Intent

Intents can be thought as high-level business or operational targets that a system should meet, without specification on how it is achieved[115]. In this section we will detail the evolution of intent in three stages. Intent started as goal-oriented policies for network operators. Later intent was used in the interfaces of network components creating intent-driven interfaces. Today intent has expanded to a framework and extensive standardisation efforts from a number of different bodies has made it a viable network management approach.

2.3.1 Goal-Oriented Policies

The goal-oriented policies of the 90's was the first iteration of intent in network management. Policies allowed for user involvement with minimum user interaction [116].

Examples of these policies can be found in [117] which details a variety of goal-oriented policies for handling congestion on ATM and IP Networks. As the policies developed they introduced more intent qualities such as pro-activeness, these goal-oriented policies could take the initiative in its decision making [118]. Further development would enable policy management goals to govern the overall management objective and activities [119]. Goal-oriented policy authors had to used translation to get the intents into actionable per device configurations [120]. This translation requirement may have motivated to move away from intent as part of policy execution and towards intent as an interface. The translation of the goal described in the intent to the devices which carry out actions to achieve it is something that we still see today in intent realisation. Modern policy-driven implementations have focused on abstraction through modelling. A straightforward example of this is the Simplified Use of Policy Abstractions (SUPA) [107]. This approach proposed a data model which utilised network resource data models defined by other working groups or Standards Development Organizations (SDOs).

2.3.2 Intent-Driven Interfaces

A Technical Report produced by 3GPP describes, intent driven management concept, intent driven management scenarios, and recommendation for the way forward on standardization expression of the intent in normative phase [2]. The study addresses different standardised reference interfaces for different users such as the communication service provider, communication service consumer and network operator

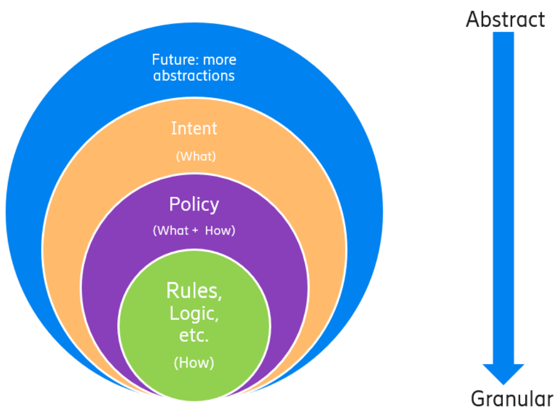


Figure 2.2: Intent driven management vs Policy driven management[2]

The study also provides additional information in the comparison of policy-driven and intent-driven management. The representation shows the encapsulation of policy within the intent, highlighting the level of abstraction expected with each approach. The level of abstraction vary between intent-driven interfaces. This can be seen when comparing the interfaces of different SDOs. TM Forum provide a specification which defines a set of operations that should be offered in manage intent and intent-driven interactions in a consistent manner [121]. This specification lists Intent Entities with relevant associations. This is a more abstracted approach compare to the Intent Driven Action and Intent Driven Object described in the 3GPP Technical Report.

Intent-driven interfaces were largely found as the NBIs of Software Defined Network controllers. When these interfaces started they suffered from missing NBI standards around intent and as a result many interfaces were vendor specific. The Open Networking Foundation (ONF) established a Work Group (WG) to manage the development of NBI in controllers. From this many SDN controllers developed intent driven interfaces including Floodlight, OpenDayLight, ONOS and NIC. The IB-NEMO language was then created to translate intents into network processes and services. These projects addressed missing NBI standards for controllers at the time. [122]

The IB-NEMO language was a transaction based Northbound API that supported descriptions of proactive intents. However, functionality was limited to updating network paths at at specified times. As a result the IB-NEMO language was extended to enable dynamic configuration, allowing for the network environment to automatically update in response to changes in network conditions.[35]

Modern intent driven interfaces for network controllers have moved away from the language approach towards ML enabled translation. An example of this is the use of intent refinement to convert intents from declarative language to a machine-readable policy. This is achieved through an intelligent intent refinement system based on natural language processing and deterministic finite automation. Intents are then translated into policies where they are deployed to the infrastructure layer through the standardised OpenFlow protocol.[123]

The use of a standardised protocol in the south bound interfaces allowed for a straightforward uniform translation of intent to actions which are understandable to any controller. When managing a range of different network components and services through intent, an action vocabulary is required for the translation to take place. To allow more of the network to become intent driven, a standard or higher level framework must provide the semantics for the translated intent to be validated against.

2.3.3 Complete Intent-Driven Network Frameworks

Today intent has found a home in the pursuit of Autonomous Networks. Large standardisation bodies have extensively modelled intent as the primary communication technology at the business and service levels. TM Forum have proposed intent-driven APIs with the aim of including these interfaces in the Autonomous Networks project. Intent in Autonomous Networks is documented in three stages: Intent Modeling [124], Intent Extension Models [125] and Intent Lifecycle Management and Interfaces [126]. These standardised intent models are divided into common and extension types, utilising ontology frameworks to allow expressivity in the intent. The interfaces of autonomous network components are moving away from the common static XML based APIs seen in traditional networks. Autonomous networks have adopted a RDF data model approach to component interfaces, viewing the machine reasoning and inference capabilities as powerful tools in the intent realisation process. [127]

2.3.4 Intent Translation Techniques

Intent Translation involves the translation of abstract high level requests into low level operations or configurations that can be carried out on the system. Intent Translation does not involve the execution of the intents, instead focused on adapting the intent into a usually standardised format/structure such as YANG/NETCONF models. In the following sections we highlight a range of popular Intent Translation Techniques in the current state of the art.

2.3.4.1 Template / Blueprint

Template / Blueprint is a straightforward technique and popular due to its commonality with existing network management concepts. Modelling and Templates is a tried and tested approach in network management. This approach looks to create collections of workflows to bring about different network states. These workflows accept a variety of parameters as a way to fine tune and adjust these states to bring about the desired outcome. The authors of [128] describe a database of Service Function Chains which fulfill a wide combination of Quality of Service and Security metrics. The authors of [129] describe how single VNF instances can be templated with adaptable parameters for configuration. The authors of [130] shows how VNF configurations may change in response to intent described requirements such as SLA. However in [131] the authors also acknowledge the predefined nature and expertise required for the creation and maintenance of relevant scripts. This approach is not only VNF focused, the authors of [132] show the amalgamation of various distinct templates for network components and functions into a grand model. Users would have engaged with the network through these predefined templates. This approach meets many of the requirements of intent-based management as it allows the user to engage with the network holistically.

2.3.4.2 Mapping

Mapping looks to break down large intents into smaller mappable segments. Mapping techniques evaluate these segments against predefined network policies. The authors of [133] utilise a mapping procedure to associate intents with network descriptors. The authors of [134] identify network services to fulfill intent goals while leveraging network functions and requirements to fulfill intent scope. The authors of [135] distributed the mapping process across application, protocol and device levels, this allowed for different policies to triggered depending on where the mapping process took place. The authors of [136] linked intent labels with network specific parameters or classes, these key value pairs could then be one to one mapped in the network. The authors of [137] adopt a parallel mapping mechanism as a means to detect resource and dependency conflicts between network functions. In [138] the authors highlighted the benefit of a policy database containing all relevant network function and network information. This could then be used to inform the mapping process giving a clearer view of the available network policies. This approach can also include information on Quality of Service features to be incorporated into the intent mapping [139]. The authors of [140] acknowledge that intents may be fulfilled in several ways, thus an interactive mechanism is adopted allowing users to select from a range of network configurations. This selection may also be stored in policy database for future reference [141]. This approach identifies the importance of breaking down large intent models into more digestible pieces and also acknowledges instances where more than one viable approach is possible to address intent goals.

2.3.4.3 Refinement

Intent Refinement views intents as a tree like structure rather than straightforward declaration. As a result an iterative approach is adopted to process the intent tree. A publication as part of this work is present in this category of Intent Translation. We adopted a recursive function to navigate the intent tree and identify keywords on the leaves of the tree structure [142]. The depth in these tree like structures can affect the context of the embedded information with each step. In [143] a similar step approach is shown in the construction of networks, where first step identifies endpoints, second finds appropriate subnets, third assigns IP addresses, fourth configures the VLANs and finally fifth checks Access Control Lists to realise the network. Similar to the mapping approach, users may wish to verify parameters generated as part of the translation process [144].

2.3.4.4 Network service descriptors

Network Service Descriptors (NSD) have become a popular translation technique for intents dealing with Service Function Chains and network slices. NSD specification can be found here [145]. NSD is a standardised deployment template for NFV Orchestrators (NFVO). NFVOs are used for the management and deployment of network services. In a straightforward approach intents could be used to describe VNFs, which would then call a mapping technique to translate the intent into implementation details for the NSD to be stored in the network function catalogue. The translation mechanism can then build the NSD so it is readable by the NFVO [146]. However in a scenario where VNFs are not explicitly included in the intent, the translation mechanism must extract them from the network service [147] or intent soft goals [148]. This information can be leveraged to identify relevant VNFs and extract corresponding VNF descriptors. This allows flexible intent driven configuration for large parts of the NFVO, to the point where connectivity constraints can be expressed through intent [149]

2.3.4.5 Inference

Templating and mapping translation mechanisms are often quite static in nature, often due to their rule based implementations. Translation mechanisms can become more efficient if allowed to learn and improve over time [144]. Adapting due to previous experiences can benefit translation mechanisms when identifying missing parameters or correlated elements. In [150] the authors adapted this approach with NSDs to improve selection of VNFDs by looking at previous executions and designs. In [151] the authors looked at encryption for security applications and adapted based on bandwidth requirements. This adaptability can improve the performance of intent based systems, for example an endpoint connection scenario. The translation mechanism can adapt between establishing a network route between two points [152] or initialising a virtual link forwarding policy [153]. This adaptability based on historical information can make intent based systems more robust while also improving performance over time.

2.3.4.6 Keyword

Often associated with Natural Language Processing, translation mechanisms based on keywords utilise model representations of labels and correlate them with actions. In scenarios where keywords may reference several potential elements, priority can be assigned based on service characteristics [154]. These systems maintain a classification of tags and service types which are associated with service templates and rules [155]. This approach benefits when coupled with an inference based approach described in the previous section. Inference would allow for correlations between keywords improving parser performance [156].

2.3.4.7 Machine Learning

Machine Learning can provide the classification characteristics of the keyword approach without the manual effort require to maintain or expand this functionality. Machine Learning provides techniques to allow intent classification while reducing required manual input [157]. The IDON Framework employs a neural net to identify relationships between intentions and service characteristics [158]. Machine Learning also brings NLP concepts and techniques which can be utilised in the translation of intents. An NLP translation mechanism should provide syntactic and semantic analysis to achieve accurate interpretation [159]. The authors of [160] utilise word embedding and convolutional neural networks to produce similarity scores between keywords. The authors of [161] utilise a sequence to sequence learning model through a recurrent neural network to predict keywords and correlations within the intent. These types of translation mechanisms are then linked to powerful AI driven network management components. The authors of [162] translate their intents into a chain of VNFs to instantiate a network service, the parameters of which are generated through a Deep Reinforcement Learning model. Additional generalised NLP based approaches will be detailed in subsection 2.4.4.

2.3.4.8 Semantics

Resource Description Framework (RDF) is heavily associated with semantic based approaches due to its ability to detail relationships between data elements. It has been used to document combinations of network actions and conditions which can be merged with services and parameters defined through intent [163]. RDF has been combined with refinement translation mechanisms containing more detailed information as more layers are added to the model. The authors of [164] describe this with basic information found in the first layer of the model while specific tooling and network information is embedded. RDF also works well with other semantic languages as shown in [165], where Web Ontology Language (OWL) can provide additional information around specific data models.

2.3.4.9 State Machine

State Machine based translation is based on Deterministic Finite Automata (DFA). DFA can perform a lexical analysis of an intent breaking it up into tokens which can then be applied to states. The authors of [47] describe DFA policy model which contains three states to manage an intent request. Each state is responsible for the translation of tokens resulting in a complete network policy. In [166] the authors ensure that default policy states can be produced if an appropriate states cannot be found. This approach while strict in their application utilises intent grammar which is easily interpreted by users. Functionality can be expanded to integrate with network management tools through the development of selection policies.

2.4 Machine Learning

The first machine learning related algorithms appeared in the 1970s but were limited by the technology of the time. Modern computing power allows us to apply machine learning to more complex problems and when accompanied by the vast amount of data being captured and stored, machine learning can be applied to a wider range of domains. Today machine learning is utilise is many different domains including:

- Security Heuristics - Expert based analysis determining the susceptibility of a system towards particular threat/risk using various decision rules or weighing methods such as Multi-Criteria analysis (MCA).
- Image Analysis - The study of images to extract useful information through pattern recognition, digital geometry and signal processing.
- Deep Learning - A multi-layer approach to progressively extract higher level features from raw data.
- Object Recognition and Prediction - A collection of computer vision tasks involved in identifying objects in a digital image and building a model for identification of similar objects in further images.
- Pattern Recognition - The automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.[167]

Machine learning learns to perform a task based on a large amount of training data to generate a model. This model can then be applied to new data which it has not encountered before and carry out the task for which it was trained. [168]

This training can be implemented using a wide range of scenarios depending on the types of available training data, the method by which training data is received and the test data used to evaluate the generated model.

2.4.1 Supervised Learning

The learner receives a set of labeled examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification, regression, and ranking problems. Instances of supervised learning include Regression, Decision Tree, Random Forest, KNN, Logistic Regression etc.

2.4.1.1 Support Vector Machine

Used as a means of data classification. Support Vector Machine constructs a hyperplane or set of hyper-planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection[169]. Common applications of SVM are classification of text, hypertext, images

2.4.1.2 Linear Regression

Regression models a target prediction value based on independent variables. It is mostly used for identifying relationships between characteristics [170]. Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

Linear regression takes external factors into account with dependant and multiple independent variables, using this data the forecast value of the next period of time can be calculated [171].

2.4.1.3 Decision Tree

Decision Tree is one such modern solution to the decision making problems by learning the data from the problem domain and building a model which can be used for prediction supported by the systematic analytics[172]. Decision Trees are the most popular classification methods as they are easy to understand and has a high accuracy in decision making [173]. However decision trees are not very robust and struggle with "missingness" or "missing data" which has resulted in considerable study in the area to address this issue [174].

2.4.2 Unsupervised Learning

The learner exclusively receives unlabeled training data, and makes predictions for all unseen points. Since in general no labeled example is available in that setting, it can be difficult to quantitatively evaluate the performance of a learner. Instances of unsupervised learning include Apriori algorithm and K-means.

2.4.2.1 K-Means

One of the popular clustering methods using the partition nodes into clusters is K-means. K-means is unsupervised learning algorithm, developed by McQueen in 1967 [175]. K-means clustering is a mathematical, non-manageable, non-deterministic, iterative method. In k-means clustering, sensors are divided into K clusters, so each sensor node belonging to one cluster based on some similarity. The each cluster contain a centroid or cluster head. In the k-means, a clustering input is how many clusters are required and output also desired numbers of clusters. [176]

2.4.2.2 Auto Encoder

Auto-encoder neural network is a well-known unsupervised feature representation method that learns low-dimensional hidden variables with the minimum reconstruction error to the input matrix. An auto-encoder neural network can be divided into two symmetric steps: encoder and decoder. The former aims to learn a low-dimensional feature representation of the input data, whereas the latter is to recover the data with the minimum reconstruction error. An encoder can be regarded as a feed-forward bottom-up step, while a decoder can be viewed as a feedback top-down generative step. In order to capture the certain geometrical structures of the input data, a number of variations of auto-encoder have emerged. [177]

2.4.2.3 One Class SVM

OCSVM is a generalization of SVM in the area of unsupervised machine learning. It is mainly used for processing outliers detection of data. Only normal data samples are needed, which takes the origin after the kernel function projection as the negative class of SVM, searches for a hyperplane to separate the normal samples from the origin, and maximizes the structural risk to ensure that the hyperplane is maximal from the sample and contains enough normal samples. [178]

2.4.3 Reinforcement Learning

The training and testing phases are also intermixed in reinforcement learning. To collect information, the learner actively interacts with the environment and in some cases affects the environment, and receives an immediate reward for each action. The object of the learner is to maximise his reward over a course of actions and iterations with the environment. However, no long-term reward feedback is provided by the environment, and the learner is faced with the exploration versus exploitation dilemma, since he must choose between exploring unknown actions to gain more information versus exploiting the information already collected. An example of reinforcement learning is the Markov Decision Process In practice, many other intermediate and somewhat more complex learning scenarios may be encountered. [179]

2.4.3.1 Markov Decision Process

Markov decision processes, also referred to as stochastic dynamic programs or stochastic control problems, are models for sequential decision making when outcomes are uncertain. The Markov decision process model consists of decision epochs, states, action, rewards and transition probabilities. Choosing an action in a state generates a reward and determines the state at the next decision epoch through a transition probability function. Policies or strategies are prescriptions of which action to choose under an eventually at every future decision epoch. Decision makers seek policies which are optimal in some sense. An analysis of this model includes:

- providing conditions under which there exist easily implementable optimal policies,
- determining how to recognise these policies,
- developing and enhancing algorithms for computing them,
- establishing convergence of these algorithms.[180]

2.4.3.2 Q-Learning

One of the early breakthroughs in reinforcement learning was the development of an off-policy TD control algorithm known as Q-learning [181]. In this case, the learned action-value function (Q) directly approximates the optimal action-value function, independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state–action pairs are visited and updated. However, all that is required for correct convergence is that all pairs continue to be updated. This is a minimal requirement in the sense that any method guaranteed to find optimal behavior in the general case must require it. [182]

2.4.4 Natural Language Processing

Natural Language Processing (NLP) aims to retrieve the syntactic, semantic, and sentimental aspects of spoken/written language referred to as natural language. NLP is a prevalent research topic producing solving problems such as Neural Machine Translation, Name Entity Recognition and Sentiment Analysis. NLP specialises in taking raw data in the form of text and translating it into a mathematical format such as vectors and matrices. Models are then designed to process the mathematical formats which can then be translated back into text. [183]

2.4.4.1 Lemmatization and Stemming

Stemming and lemmatization are NLP preprocessing techniques to handle different versions of words [184]. Processing produces comprehensible, predictable, and analyzable text [185]. Stemming and lemmatization are categorised under Information Retrieval [186] [187]. Both techniques group similar words based on root or canonical citation form [185]. Stemming reduces words to their most common stem maintaining the semantic meaning [185] [188] [189] [190]. However, grammatical rule are not maintain in this process [191]. Lemmatization maintains the actual word [192]. Lemmatization removes ending to reduce the word to its dictionary form to remove variations [193] [194]. The suffix is removed or replaced to bring it to is base which is referred to as lemma [185] [188].

2.4.4.2 Topic Modelling

Topic Modelling is an unsupervised machine learning approach which generates a weighted list of words based on text input [195]. Topics are interpretable and can be used to generate topic based representations of bodies of text. Topic Modelling is a popular text analysis tool used in applications such as exploratory text analysis [196], information retrieval [197], natural language processing [198], and topic discovery [199].

2.4.4.3 Keyword Extraction

Keyword extraction is the process of identifying key words and phrases from a body of text with the aim of providing the user with quick insights into the text. Popular keyword extraction tasks include opinion mining [200], text summarization [201], and text categorization [202]. Current popular keyword extraction models are compared and evaluated based on their recall and precision, however this can fluctuate between data sets [203] [204].

2.4.4.4 Knowledge Graphs

Knowledge Graph is a semantic network of objects typically stored as a graph database and visualised in a graph like structure [205] [206] [207]. In 2012, Google adopted Knowledge Graphs to assist in search queries [208]. A knowledge graph consists of nodes and edges. Nodes represent a body, this could be an object, resource or information and edges represent the relationship between two nodes [209]. Knowledge graphs were inspired by the semantic web [210]. Knowledge graphs leverage existing standards such as RDF and OWL, but are less formal than ontology based semantic web [210].

2.5 Reference Architecture

This section describes a range of network management architectures from different domains. The first group of reference architectures relate to open source, an area of increased growth and utilisation in network management. In subsection 2.5.1 the ONAP Platform Architecture is described with focus on the platforms Policy and Optimisation frameworks. In subsection 2.5.2 the ETSI OSM (Open Source MANO) is described with focus on its role as a widely adopted network service orchestrator. The second group of reference architectures is described subsection 2.5.3. This group relates to leading Standard Development Organisations in the area of Autonomous and Intent-Based Networking, such as TM Forum, Zero touch network & Service Management (ZSM) and the IRTF Network Management Research Group (NMRG). In subsection 2.5.4 the industrial influence on this work is detailed with an example of how research is leveraged in the industrial domain.

2.5.1 ONAP Platform Architecture

The ONAP platform is an open source network automation platform which enables product-independent capabilities for design, creation and life-cycle management of network services. ONAP provides vendor-agnostic policy-driven services with analytical and lifecycle management capabilities. Through ONAP, network operators can orchestrate both physical and virtual network functions allowing operators to utilise existing network investments while providing an open platform to motivate further development of VNFs in the network management space.

ONAP has provided blueprints of key network use cases which the industry and open source communities have tested. The experiences of testing these use cases has fueled the platform development ensuring a trusted framework at final release.

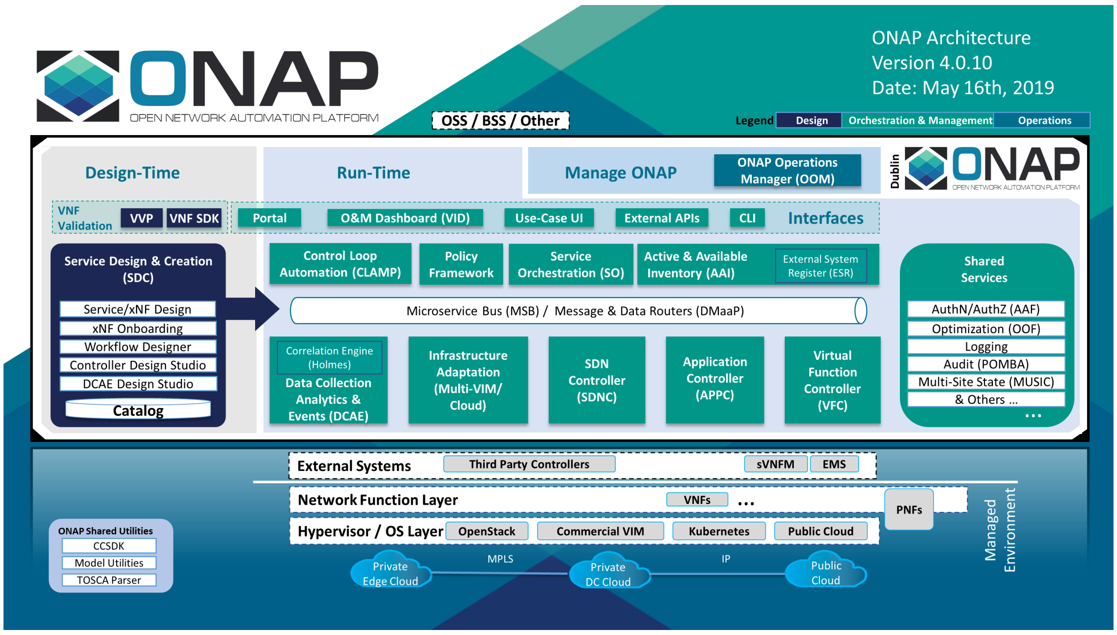


Figure 2.3: Open Network Automation Platform Architecture[3]

Deployed as a cloud native application with a variety of instantiated services, ONAP requires an advanced management system to navigate the pre and post deployment stages. This role is fulfilled by the ONAP Operations Manager (OOM). This component orchestrates the life-cycle management and monitoring of underlying components. As the OOM is integrated with the micro-services bus it has access to registration and discovery features along support for external API and SDKs. The OOM can also provide scalability and resiliency enhancements to managed components. Kubernetes is utilised to proved CPU efficiency and platform deployment.

ONAP provides a model driven runtime environment with analytical feedback capabilities to support closed loop automation and service optimization. Tooling is also provided for service designers and opertors with access to design time and run time environments through the Portal Framework with supported role based access.

The design time environments provide development tools, techniques and repositories for defining resources, services and products. This includes policy design and implementation, as well as an SDK with tools for VNF supplier packaging and validation.

The run time environment executes the content from the design environment such as the Active & Available Inventory (A& AI) component. The component provides real time monitoring of resources and services allowing for service assurance in a dynamic environment where virtual resource are continuously go through stages of deployment and teardown. Run time services are in constant communication with closed loop monitoring and analytics modules to provide insights to service designers for optimisation opportunities.

The platform is responsible for providing the interfaces which allow separate components and frameworks to communicate. The policy framework described in subsubsection 2.5.1.1 outlines the goals and functionality of the framework. The optimization framework is described in subsubsection 2.5.1.2 detailing the ONAP platform's approach to optimization and repair.

2.5.1.1 ONAP Policy Framework

The ONAP Policy Framework provides an environment for policy design, deployment, and execution. The Policy Framework contains the primary decision making components of the ONAP system. It enables the platform operator to specify, deploy, and execute the governance of the features and functions in their ONAP system, be they closed loop, orchestration, or more traditional open loop use case implementations. The Policy Framework supports Policy Driven Operational Management during ONAP control loop execution. Orchestration and and control use cases can utilise the Policy Framework to execute their policies instead of embedding them in their applications.

The Policy Framework is deployment agnostic, separating policy execution and policy enforcement. This allows for flexibility in what applications or functions the policy is impacting in a given use case. For example policies can be deployed along side applications as a means to improve performance, alternatively policies can be deployed as a separate executing entity in the form of a container. The Policy Framework separates policies from the underlying platform. The framework supports development, deployment, and execution of any type of policy due to its metadata (model) driven approach. This is to ensure maximum flexibility as to support modern rapid development ways of working such as DevOps.

2.5.1.2 Optimisation Framework

ONAP Optimisation Framework (OOF) is a policy and model driven framework used to create optimisation applications for a wide range of use cases. OOF adopts a typical optimization construct, this is an example provided by ONAP documentation:

- Objective: Maximize/minimize a metric, measured by appropriate key performance indicators (KPIs)
- Technology and operating constraints, such as:
  - Parameter change limits (such as power)
  - Frequency of changes permitted
  - Number of parameters that can be changed simultaneously
  - Data latencies (typically in percentile)
  - DC compute, network, storage, energy capacity
  - Location based and time based energy cost

In this instance the objective metric to be adjusted could be for example, throughput or retainability which you would wish to maximise, alternatively it could be interference levels or cost which you would wish to minimise.

The development of OOF was based on the following core ideas:

- Most optimization problems can be solved in a declarative manner using a high-level modelling language.
- Recent advances in open source optimization platforms allow the solution process to be mostly solver-independent.
- By leveraging the library of standard/global constraints, optimization models can be rapidly developed.
- By developing a focused set of platform components a policy-driven, declarative system that allows ONAP optimization applications be composed rapidly can be realised and managed easily
  - Policy and data adapters

  - Execution and management environment
  - Curated "knowledge base" and recipes to provide information on typical optimization examples and how to use the OOF

- More importantly, by providing a way to support both "traditional" optimization applications and model-driven applications, users have a choice to adapt the platform based on their business needs and skills/expertise.

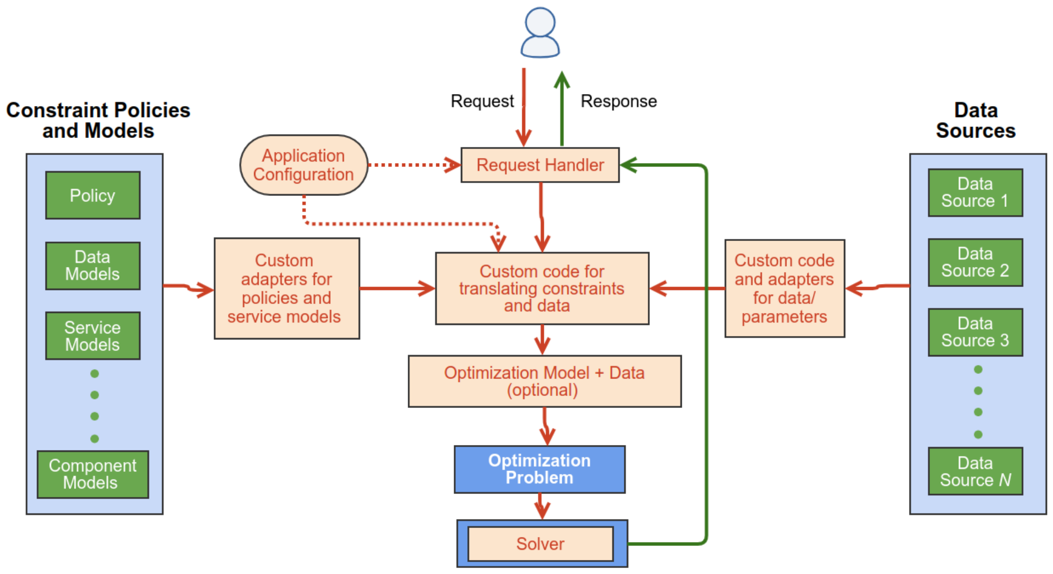

Constraint Policies and Models
Policy
Data Models
Service Models
Component Models
Request
Response
Application Configuration
Request Handler
Custom adapters for policies and service models
Custom code for translating constraints and data
Custom code and adapters for data/ parameters
Optimization Model + Data (optional)
Optimization Problem
Solver
Data Sources
Data Source 1
Data Source 2
Data Source 3
Data Source N

FIGURE 2.4: Traditional Optimisation Framework[4]

Traditionally, optimization applications are designed to perform a specific task to achieve an optimisation goal. Large amounts of application code are problem specific meaning changes in the problem require code changes in various components of the application. This approach results in large development cycles for changes in requirements. The Optimization Framework provides platform level functionality reducing the number of these code changes.

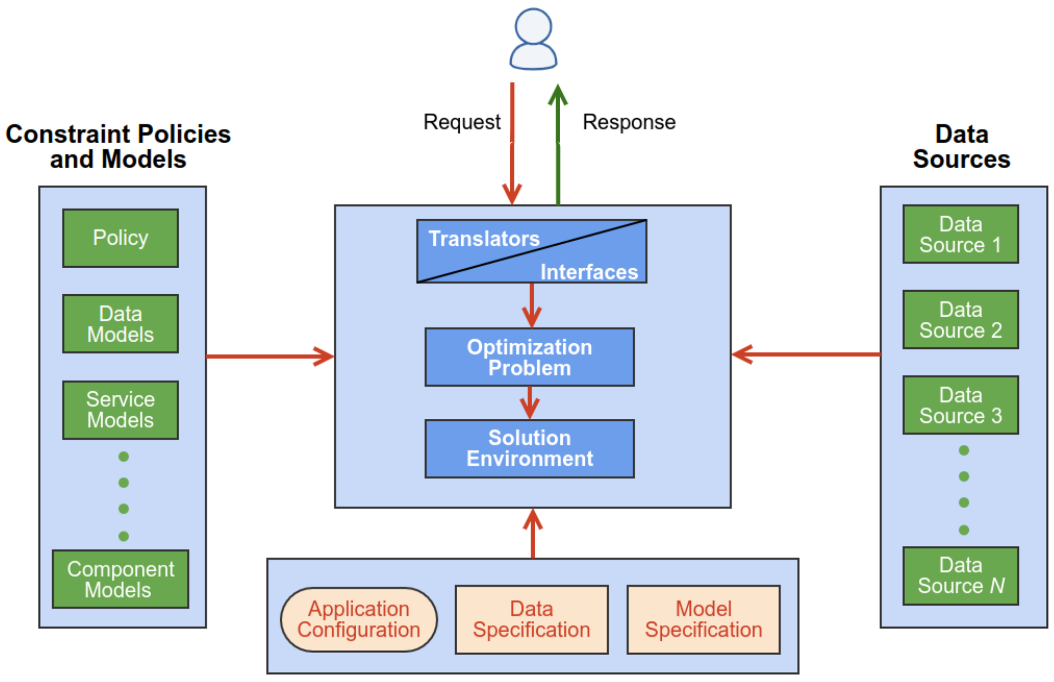


FIGURE 2.5: ONAP Optimisation Framework[4]

OOF allows for the quick development of new optimization applications through a policy driven, robust and scalable optimization framework. These optimization applications are independent from the implementation of underlying optimization modules allowing for re-usability by addressing the problems associated with application specific optimization codes, adapter libraries and configuration logic.

The policy driven approach reduces inconsistencies and duplication by making constraints and other policies available across services Optimisation constraints can be specified as policies which can be configured by service designer or operators. The model driven approach allows Data formats and API calls to be modeled, allowing for errors to be identified early in the design stages.

2.5.2 ETSI OSM (Open Source MANO)

ETSI OSM (Open Source MANO) is a community-driven production-quality E2E Network Service Orchestrator. This widely adopted orchestrator supports the modelling and automation of real telco-grade services in complex production environments. The popularity of OSM has accelerated the maturation of NFV technologies and standards allowing VNF vendors to test and validate on commercial NFV infrastructures through an open orchestrator. OSM minimizes integration efforts through an infrastructure-agnostic Information Model, standard compliant unified northbound interface and support for cross-domain network services and network slice life-cycle management.

OSM provides a powerful Information Model, fully aligned with ETSI NFV standard [211], capabale of modeling and automation of the full life-cycle of network functions, services and slices. This covers the initial deployment, daily operation and monitoring of each of these network technologies in an infrastructure-agnostic manner. This allows a single model to be instantiated or deployed in a large variety of environments without the requirement of domain-specific translation.

OSM provides a unified northbound interface, based on ETSI NFV standard [212], which supports the full operation of system functionality along with the managed services and slices. The OSM NBI provides, as a service, the necessary abstractions for client systems to control, operate and supervise the life-cycles of network services (NS) and network slice instances (NSI) without having to expose complex elements of the underlying system.

OSM supports an extended concept of network service allowing it to span across the virtual, physical and transport domains. This support allows the system to control the full life-cycle of network services as they interact with virtual, physical and hybrid network functions in a common manner with on demand transport connections. In addition, this flexibility has also enabled the life-cycle management of network slices allowing OSM to assume the role of Slice Manager if required. This concept if further extended to support an integrated operation.

2.5.3 Autonomous Networks

Autonomous networks aim to leverage AI decision making capabilities to inject autonomy into the network. SDOs have initiated focus groups such as ITU-T Focus Group on Autonomous Networks (FG-AN)[213]. The primary objective of the Focus Group is to provide an open platform to perform pre-standards activities related to this topic and leverage the technologies of others where appropriate. TM Forum and ZSM have proposed their reference architectures towards autonomus networks. Other bodies such as NMRG have detailed their Autonomic Networking Infrastructure (ANI). TM Forum and ZSM have made considerable contributions to their autonomous platforms which will be detailed in the following sections. In subsubsection 2.5.3.1 the autonomous network architecture of TM Forum is detailed highlighting some of the incorporated intent concepts. In subsubsection 2.5.3.2 the distributed network architecture of ZSM is detailed with a focus on loosely coupled management functions. In subsubsection 2.5.3.3 the autonomic architecture of NMRG is detailed emphasising the role of control loops and features of autonomic nodes.

2.5.3.1 TM Forum - Autonomous Networks Project

TM Forum proposed the Autonomous Networks Reference Architecture which provides a layered yet 'composable' framework for autonomous operations. The reference architecture aligns with the architectural principles outlined in [5]. The operational layers of the system are fully decoupled, these are consumer to business, business to service and service to resource. Open interfaces are utilised at all integration points between domains with a focus on intent-driven interfaces. Intelligent components are layered and distributed near the functional blocks in the system. This approach supports fast decision-making adaptive controls within the system, these logical units are referred to as Autonomous Domains. The complexity in the system is decreased through the distribution of these Autonomous Domains as they can be federated horizontally and vertically.

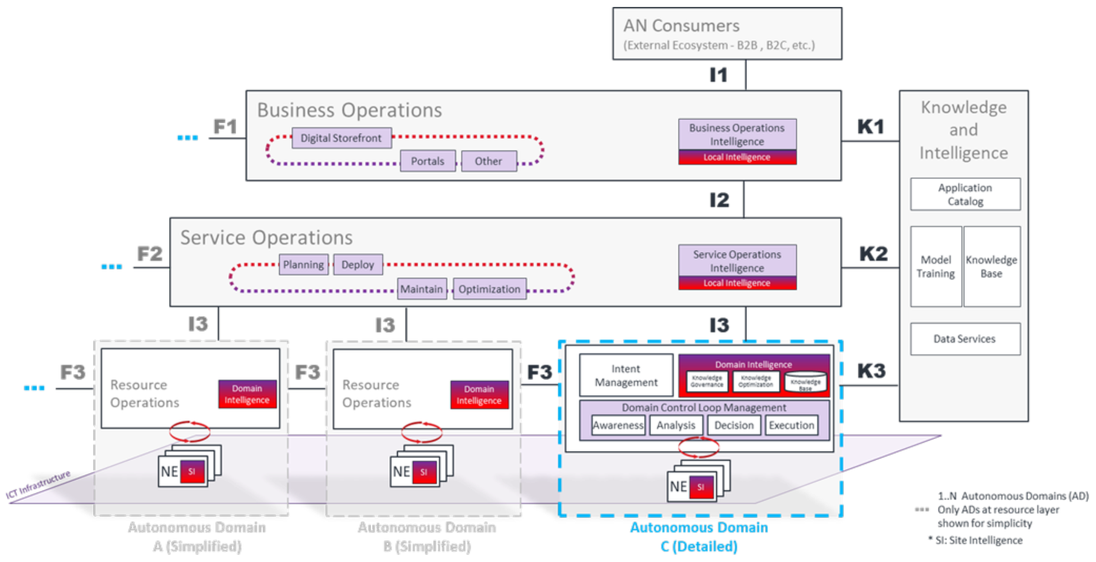


FIGURE 2.6: TMF Autonomous Network Architecture[5]

In Figure 2.6 a complete building block is presented for Autonomous Domain C. These blocks are repeated at all three layers in the network architecture. Autonomous Domain C shows some of the inner mechanisms and functions that comprise an Autonomous Domain. The horizontal domains of each layer cooperate in a federated manner. This allows for the Service Operations layer to perform a cross-domain collaboration of the three domains presented in the figure, resulting in an end-to-end coordination of the resource operation domains.

The automation of operations supports the fast deployment requirements of new 5G services. Efficiency is a large contributing factor in enabling operators to provide the best user experience, with full life-cycle automation of the network and maximum utilization of the network resources. This approach relies on the coordination of automated closed loops driven by intent in each of the operational layers to achieve autonomous behaviour. Control loops in independent domains are required to exchange information as part of the systems autonomous behaviour. This reference architecture utilised simplified open interfaces to communicate between layers, adopting more of an intent driven approach and moving away from the data-centric and parameter-heavy payloads seen in current networks. However, the simplification of the open interfaces is restricted by the autonomous capabilities of the domain which actuates the change.

Intent-driven autonomous systems allow for the intents of users to be inferred instead of the processing of a detailed instruction. These systems should provide an adaptive and robust communication with the user. An intent interface may support domain specific terminology and semantics, enabling a dialog style interaction. This interaction is referred to as an "interview" style interface where the handling system enquires for additional information from the user, clarification on translated intents, recommendation on alternatives or changes to the intent. TMF921A provides a set of interfaces to support intent-driven interactions[121].

2.5.3.2 ZSM - Intent-Driven Autonomous Networks

ZSM also follows the industry trend of moving away from tightly coupled management systems, towards flexible sets of management services. Similar to the architectures proposed by TM Forum[214] and 3GPP[215]. ZSM define architectural building blocks which can be generated through composition and inter-operation patterns to construct sets of complex management services. These services are then instantiated in different domains to manage other components and services.

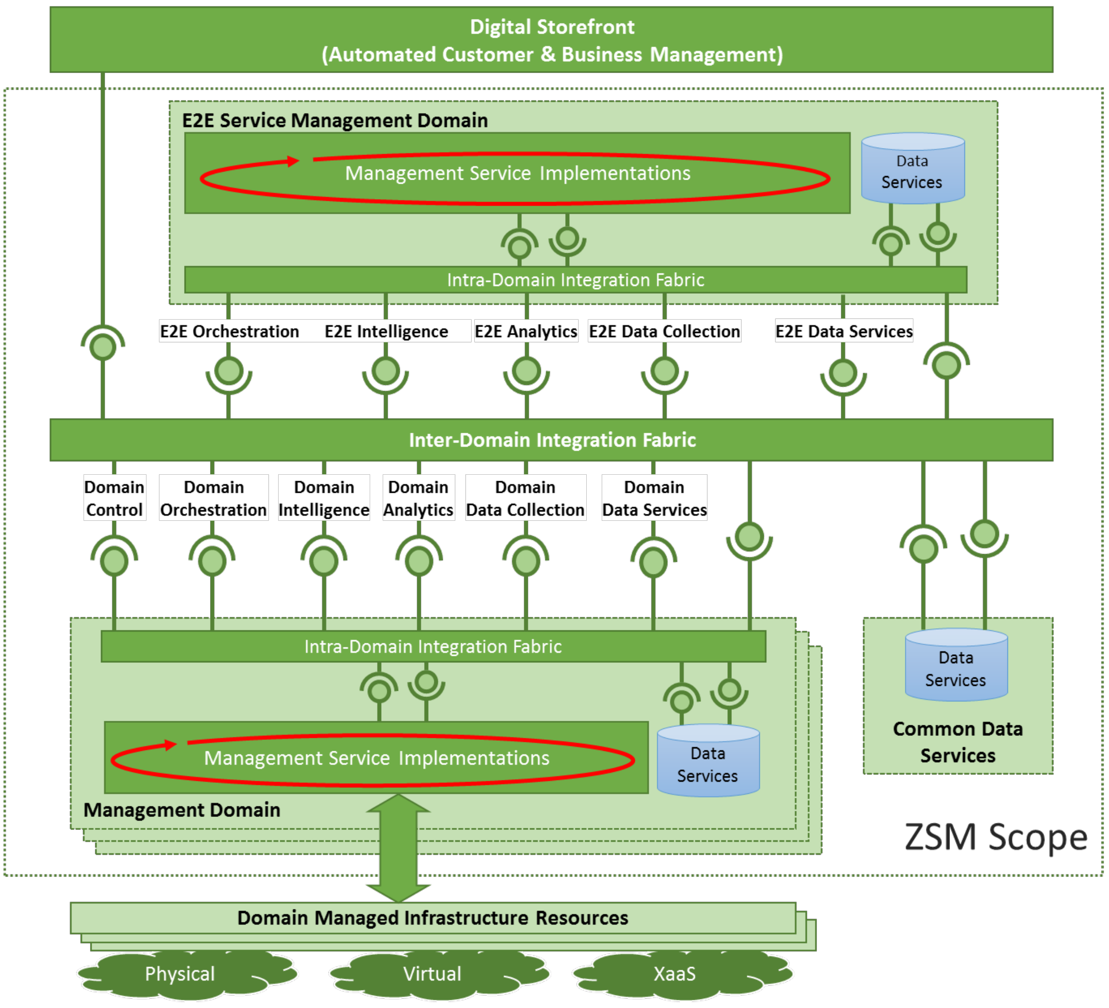


FIGURE 2.7: ZSM Reference Architecture[6]

The ZSM framework presented in Figure 2.7 is composed of distributed management and data services which are separated in management domains and integrated through the integration fabric. The integration fabric allows the management services to communicate with other domains or other management systems. The integration fabric also supports the sharing of data models across domains, allowing higher level management services to troubleshoot issues that cannot managed within the domain. All management services expose their capabilities through the integration fabric.

The ZSM framework reference architecture supports the building and composing of loosely coupled management functions that offer management services and when distributed across domains deliver end-to-end and domain-specific capabilities for zero touch management. These management services expose end points for invocation and communication.

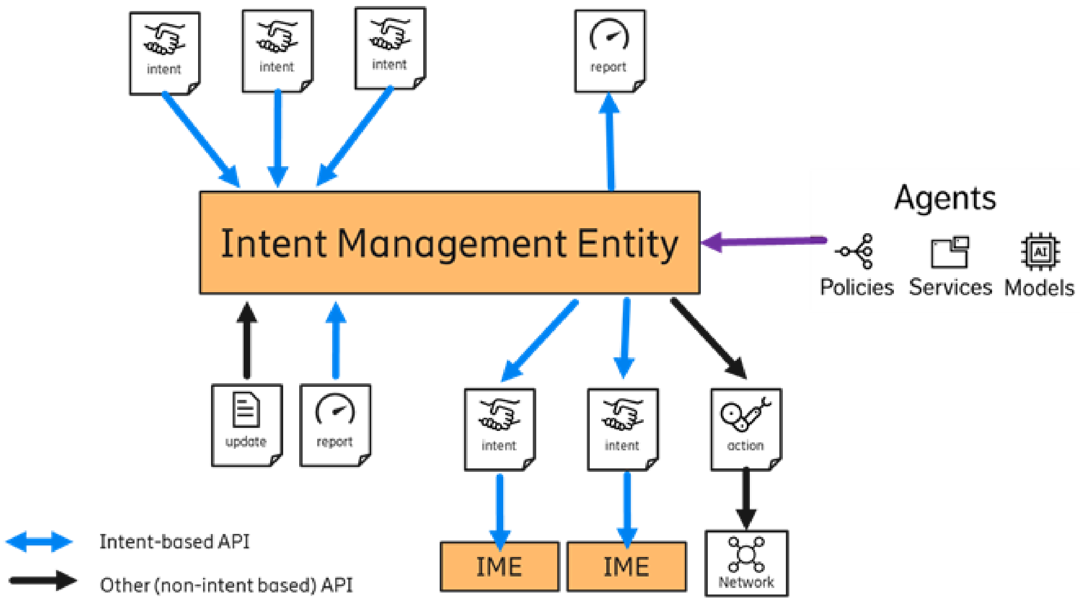


FIGURE 2.8: Intent Management Entity[7]

ZSM detailed the role of intent in previously described framework, Figure 2.8 presents the Intent Management Entity (IME). The IME is described as an autonomous entity in a domain that can play the role of intent owner and/or intent handler and has the ability to make and actuate decisions in order to fulfil intents. The IME contains a knowledge base containing the intent ontology, along with machine reasoning capabilities to support knowledge driven decision making. The interfaces of the IME are generic and domain-independent. The objective of the IME is the manage the life-cycle of the intents and report to higher level management systems.

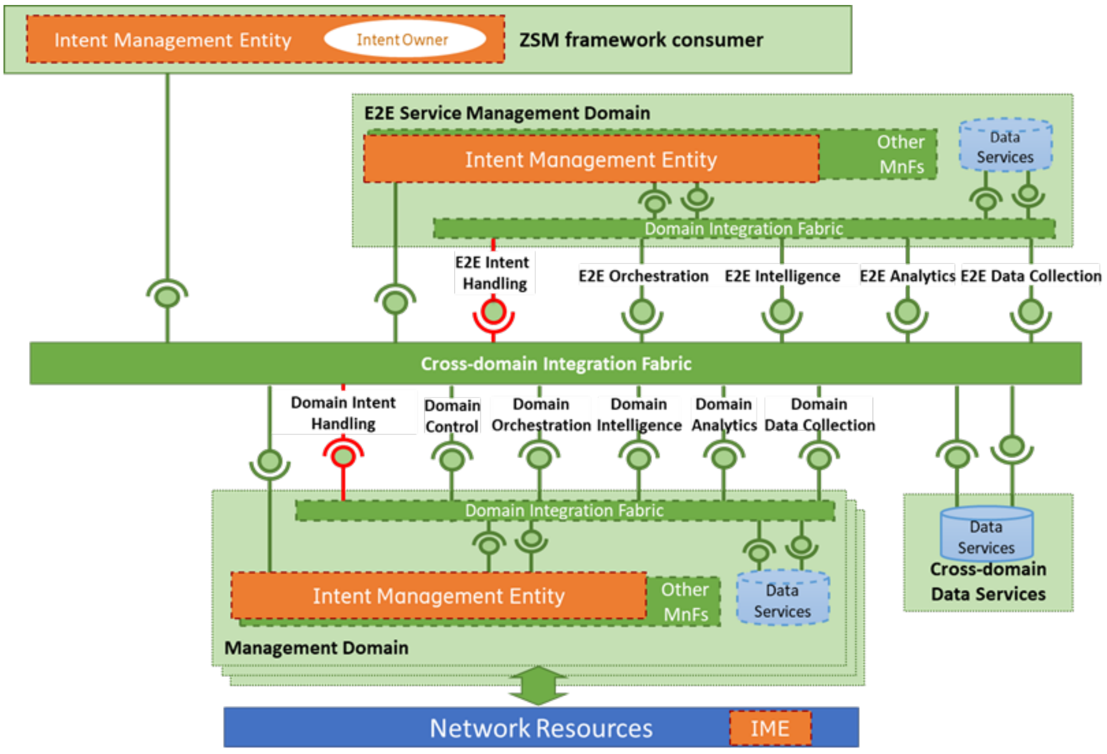


FIGURE 2.9: Intent Management Entity in ZSM Reference Architecture[7]

ZSM described how the IME would fit into the existing reference architecture. The end-to-end service management domain can transform and communicate towards other management domains or interpret the intent into a set of services offered by the management domain. This approach also applies to the communication of intents to network resources through the translation into resource requests. The management domain may also reshape the offered services to match those provided through intent. [7]

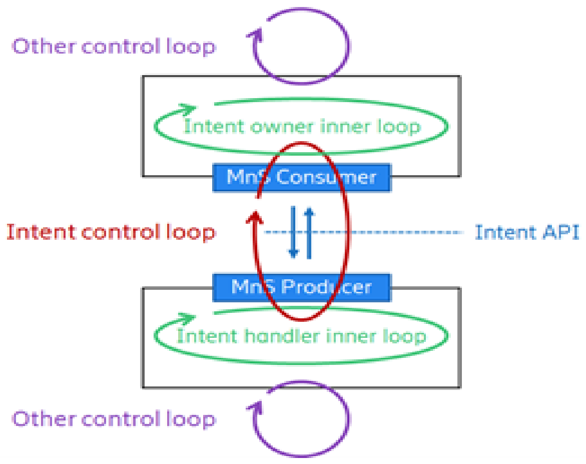


FIGURE 2.10: ZSM Intent Control Loop[7]

In Figure 2.10 the IME operates a closed control loop to produce and maintain the state described through the intent expression. The intents are typically generated and communicated through the system using the Intent API. Most IME instances behave as an intent handler, producing and maintaining the delivery of a service state or outcome. However, some IME instances operate as an intent owner. In this scenario an IME (owner) will communicate with another IME (handler) in a different domain to deliver, maintain and report on a provided service or state. This intent control loop is best understood as a the linking or coordination of the intent owner and intent handler control loops. These interactions and communications between closed control loops are detailed in [216].

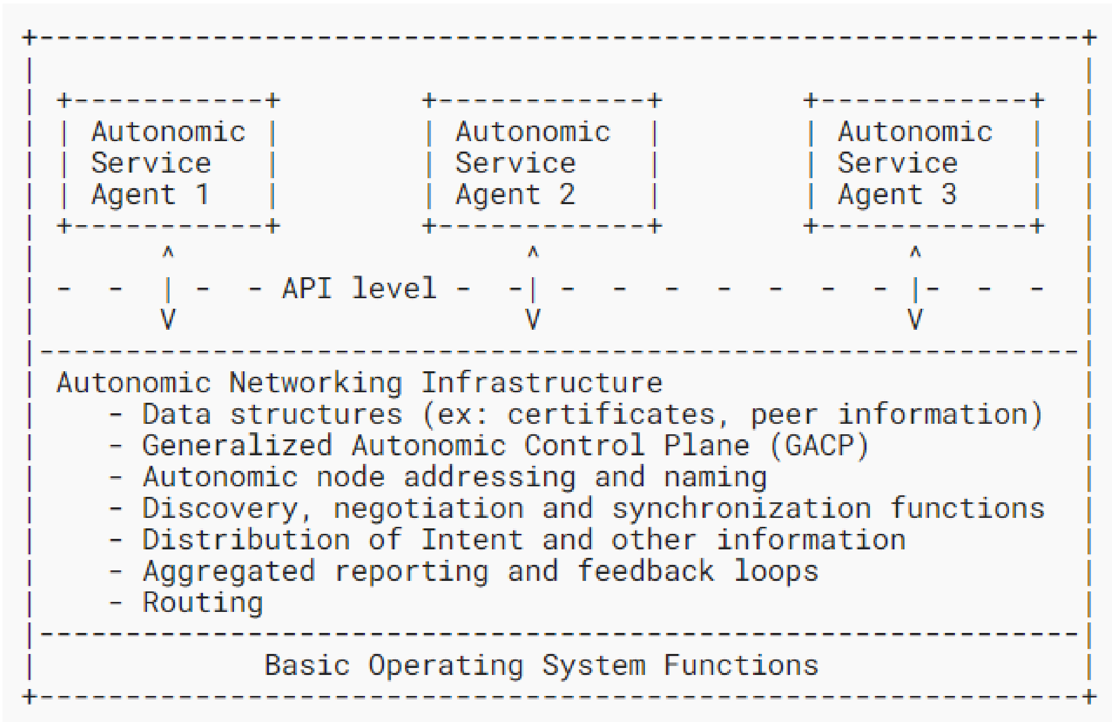
2.5.3.3 NMRG - Intent-Based Networking

The NMRG have provided descriptions of various network elements and autonomic functions with explanations of how these elements coordinate at a high level. More detailed descriptions of the internal working of these autonomic network elements are also provided along with the interfaces used to communicate between them.

```
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:            :       Autonomic Function 1        :                 :
: ASA 1      :      ASA 1      :      ASA 1      :          ASA 1  :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
             :                 :                 :
             :   +- - - - - - - - - - - - - - +  :
             :   :   Autonomic Function 2     :  :
             :   :  ASA 2      :      ASA 2   :  :
             :   +- - - - - - - - - - - - - - +  :
             :                 :                 :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:                Autonomic Networking Infrastructure               :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
+--------+   :    +--------+   :    +--------+   :        +--------+
| Node 1 |--------| Node 2 |--------| Node 3 |----...-----| Node n |
+--------+   :    +--------+   :    +--------+   :        +--------+
```

FIGURE 2.11: High-Level View of an Autonomic Network[8]

Figure 2.11 presents a high level view of an Autonomic Network. The network consists of a collection of autonomic nodes which are visible to each other and communicate over defined interfaces. Each node provides a common set of capabilities across the network referred to as the Autonomic Networking Infrastructure (ANI). The ANI provides functionality such as negotiation, synchronisation and discovery to name a few. Autonomic functions are found throughout the network. The autonomic entities of an autonomic function are referred to as Autonomic Service Agents (ASAs), which are instantiated on the nodes in the network. Autonomic functions are expected to span across the network horizontally while nodes implement the ANI and have access to one or more ASAs. [8]



```
+------------------------------------------------------------+
|                                                            |
| +-----------+        +------------+        +------------+  |
| | Autonomic |        | Autonomic  |        | Autonomic  |  |
| | Service   |        | Service    |        | Service    |  |
| | Agent 1   |        | Agent 2    |        | Agent 3    |  |
| +-----------+        +------------+        +------------+  |
|       ^                    ^                     ^         |
| -  -  | -  - API level -  -| -  -  -  -  -  -  - |-  -  -  |
|       V                    V                     V         |
|------------------------------------------------------------|
| Autonomic Networking Infrastructure                        |
|    - Data structures (ex: certificates, peer information)  |
|    - Generalized Autonomic Control Plane (GACP)            |
|    - Autonomic node addressing and naming                  |
|    - Discovery, negotiation and synchronization functions  |
|    - Distribution of Intent and other information          |
|    - Aggregated reporting and feedback loops               |
|    - Routing                                               |
|------------------------------------------------------------|
|             Basic Operating System Functions               |
+------------------------------------------------------------+
```

FIGURE 2.12: Model of an Autonomic Node[8]

The ASAs can leverage a variety of sources of information such as self-knowledge, network-knowledge through discovery, intents and feedback loops. The autonomic node is shown in Figure 2.12 and is presented in two levels. The ASAs utilise the services of the ANI as part of their execution. The ANI provides node-specific data structures and a collection of generic functions. [217]

2.5.4 Industrial Hosting

This work involved a close working relationship with the Network Management (NM) Lab based in an Ericsson Product Development Unit (PDU). As a result a heavy focus was placed on experimental work which advanced the existing technology and approaches adopted by industry. This was primarily achieved through engagement with European research projects under the umbrella of Horizon 2020[1].

Horizon 2020 aims to increase innovation within Europe, encouraging economic growth through investment in a variety of research areas. Horizon 2020 funds collaborative projects between industry leaders, academic institutions and governing bodies. Collaborative innovation is viewed as a mechanism to drive economic growth ensuring Europe's global competitiveness in the future.[218]

As part of H2020 projects, Adaptive Policy has been utilised in a number of different roles and scenarios. The work carried out as part of these projects has showcased Adaptive Policy's flexibility in making real-time context-aware decisions and was the primary reason for inclusion in intent interpretation. The 5GENESIS and 5G-CLARITY H2020 projects have significantly contributed to the expanded role of APEX providing industrial grade testbeds and experiments to showcase adaptive decision making and the utilisation of the system for intent driven management. In subsubsection 2.5.4.1 the role of adaptive policy as a advanced rules based management approach in a real 5G platform. In subsubsection 2.5.4.2 the role of intelligence in network management is highlights through the creation of an Intelligence Stratum for 5G networks

2.5.4.1 5GENESIS

The 5GENESIS H2020 project started in 2018 looked towards the validation of 5G network KPIs and the verification of 5G technologies with an end-to-end approach. To reach this objective the project aimed to integrate a range of technologies from different R&D projects to produce a full-stack end-to-end 5G facility to meet targeted KPIs. The facility is composed of several platforms, each tasked with addressing a societal challenge and existing technical challenges of 5G networks. Each platform forms a validation setup and the combination of all platforms build and open, flexible and distributed experimentation facility. [219]

APEX was selected as the policy engine for the 5GENESIS Slice Manager. APEX was integrated into the slice manager software stack as a docker container. The use case adopted by the Athens Platform utilised APEX as the decision making system for

[1]`https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en`

the slice manager. The role of APEX is to receive alarms from the Prometheus Alert Manager, which runs as part of the slice monitoring module. When the monitoring framework detects that a slice is reporting abnormal behavior, Prometheus generates an alert message and sends it to the APEX engine via the internal Kafka message bus. APEX processes the alert and recommends a series of slice manager operations, based on the alert and underlying network conditions. The adaptive decision making capabilities of APEX allows for historic monitoring data to be utilised in future decisions. This implementation of APEX shows the utilisation of the context-aware attributes of APEX. However APEX was integrated into the platform slice manager. With a flexible intent based approach, APEX could leverage its context aware and adaptive qualities to drive a variety of different tools and components.

2.5.4.2 5G-CLARITY

The 5G-CLARITY project started in November 2019. The project investigates how the concept of private 5G networks should evolve beyond the 3GPP Release 16[62]. The project aims to bring innovation in two main pillars. The project will produce novel user and control plane components developed to deliver a private 5G network that will integrate 5GNR, Wi-Fi and LiFi. This will enhance the capabilities of 5GNR regarding peak data rates, area capacity, low delay and precise localisation. The project will also investigate management enablers for network slicing, private and MNO network integration, intent driven network operations and the incorporation of network functions implemented through machine learning models. [220]

The project details the Intelligence Stratum, described as an almalgam of AI and intent-based services. The Intelligence Stratum is made up of two components, the AI and Intent Engine. The AI Engine contains and executes advanced analytics models while the Intent Engine provides a flexible mechanism for the execution of models and network operations. [221]

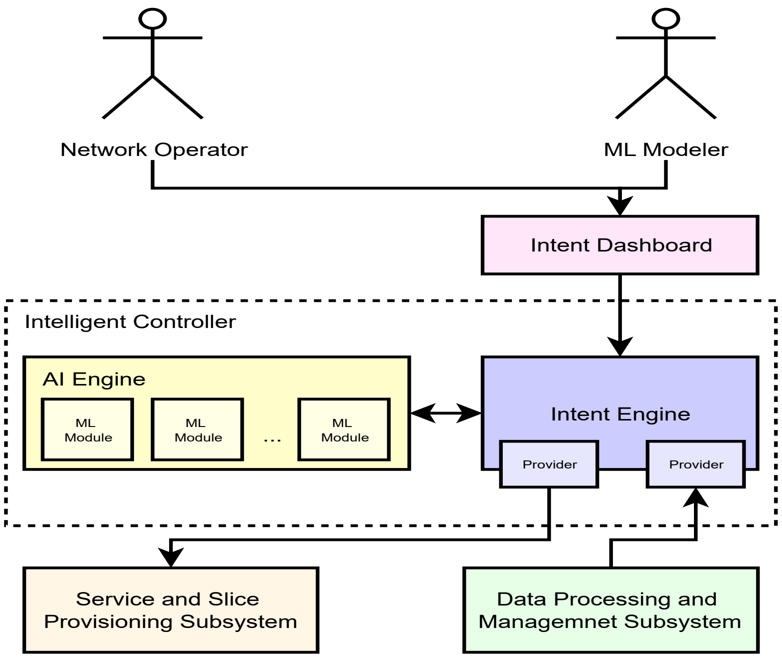


FIGURE 2.13: 5G-CLARITY Architecture[9]

In Figure 2.13 the Intelligence Stratum is presented with an alternative title, Intelligent Controller. The Intelligence Stratum is functionally similar to an intent-driven, AI supported controller with monitoring capabilities. Intent messages are specified by network operators while support AI models are designed by ML modellers. Intents are accepted through the Intent Engine which acts as the interface for the AI Engine and provisioning subsystems. Models contained in the AI Engine can be triggered through intent messages to provide intelligent services and functionality to the network. The actions requested by these models can then be realised through the intent engine by mapping the requests onto the executable actions of the Service and Slice Provisioning Subsystem. This component in Figure 2.13 represents any target management function responsible for realising the request detailed in the intent message.

Chapter 3

Architectural Design

In this chapter we describe the implementation architectures of our work followed by the evaluation methods applied to our adaptive closed-loop and intent-driven systems. The chapter concludes with a detailed description of the architecture over several revisions presenting the final architecture. In section 3.1 we present the implementation architectures and detail our interpretation of adaptive context-aware policy driven system, coordinated as part of a closed control loop. This implementation architecture is expanded, presenting our intent interpretation framework. In section 3.2 we present the evaluation methods applied to our approach detailing the mechanisms used to generate the results of our testing. In section 3.3 we present the evolution of the architecture and approach as the research progressed over the duration of the work.

3.1 Implementation Architecture

This section describes the role of policy in the system along with an outline of the planned implementation. Building off previous work a hybrid of machine learning and rules based decision making will be implemented through the APEX policy engine.

3.1.1 Adaptive Policy Execution

In this section we detail our implementation of adaptive policy in a COMPA control loop. Our work with adaptive policy has produced a working testing environment for adaptive policy execution for business, service optimisation and advanced decision-making policies. Our work to date has introduced control logic for APEX policies, implementing a feed forward neural network for decision making in regard to network path selection for multimedia streaming applications. Additional control logic performs an evaluation of service QoS metrics generating a Mean Opinion Score (MOS). This MOS value is utilised in the neural network to adjust corresponding QoS weights, completing the optimization loop. Our solution applies the COMPA automation pattern to design a closed loop system using components of the Open Network Automation Platform (ONAP). We can use COMPA to break down the roles of the core components in the system, providing an overview of each component and their application.

(C)ontrol: The network controller will be responsible for the reconfiguration of network parameters set by the optimization framework.

(O)rchestration: The deployment and chaining of our core components will be done in ONAP ensuring the accessibility and communication between components through standardised interfaces.

(M)anagement: ONAP Optimisation Framework will deploy specific optimization services directed by context aware adaptive policy. These optimisation services will change network behavior to reflect changes of context in the network.

(P)olicy: APEX will provide adaptive decision making bolstered by a single layer neural network. This provides a machine learning enabled adaptive policy capable of utilising real time context and weighted algorithms in the decision making process.

(A)nalytics: The Data Collection, Analytics and Events subsystem, in conjunction with ONAP components will gather relevant network information used in adaptive policy to build network context. This information can be used to trigger adaptive policy based on real time network monitoring, large changes in network context and the realization of new business goals.

3.1.2 Intent Engine

In this section we present the implementation architecture of the Intent Engine. The initial concept of Intent driven execution as part of the 5G-CLARITY project. Intent execution is described in as the triggering of an intent process resulting in the generation of a context in the form of a report. In a scenario where an intent is requesting a slice reconfiguration, the target management function would be the Slice Manager. This would produce a communication between the Intent Engine and Slice Manager realising the request detailed in in the intent message. Once realised, the underlying managed system can be monitored and the impacts of the reconfiguration is recorded in the Data Processing and Management subsystem.

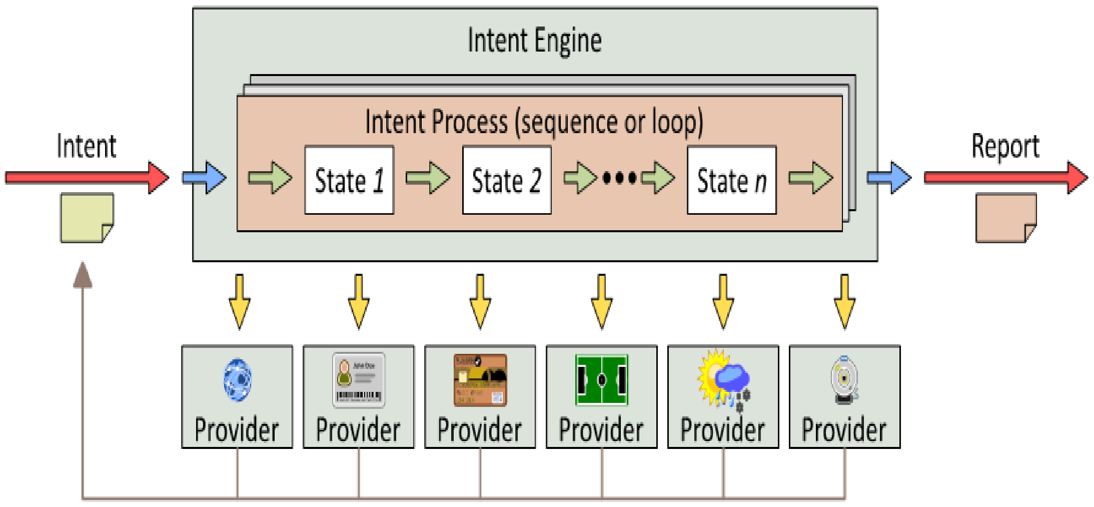


FIGURE 3.1: Intent Engine Concept[9]

The realisation of intent in the 5G-CLARITY system can be broken down into four stages. The first part covers the injection of an intent into the system, typically by the network operator. The intent interface allows for the straightforward expression of the intent through natural language with additional fields for specific parameters.

In the first stage, the intent triggers the execution of the Intent Engine. A high level representation of the intent engine is shown in Figure 3.1. The intent engine stores the intent in the run-time store and validates the request against the ML models running on the AI Engine of the Intelligence Stratum. On the successful identification of the ML model, a new ML model execution intent is generated and sent to the AI Engine.

The second stage of intent realisation covers the execution of the ML model. The pure run-time system requires a fully trained ML model. The result of this model should provide detailed instructions for the Intent Engine. These can be described as intents or a configuration template. If the Intent Engine cannot identify an appropriate ML Model for execution, the Intent Engine will query internal information to generate a response. If the ML model requires additional information, the Intent Engine may query the Data Processing and Management Subsystem and retrieve the required information for the model.

In the third stage the Intent Engine receives detailed instructions and processes them for the appropriate management function. Instructions may be forwarded directly to the selected management systems is possible. Alternatively the instructions may require translation. This approach is expected as it allows for the decoupling of the ML models from the management function. Monitoring and reporting systems are then engaged. Depending on the intent request this may be resolved through the checking of status codes on actions sent to management functions. In the situation of a prolonged intent, a monitoring function may be required to assure the intents fulfillment.

Finally the fourth stage presents the retrieval of monitoring data through the Intent Engine. Different from the process described in stage two, this stage describes the scenario where monitoring data is the objective of the intent. After the data has been requested and retrieved, the Intent Engine may compare the data to existing intents logged on the system. The Intent Engine can check if data associated with an intent has resulted from a configuration. If so, a negotiation can take place between existing intents and new intents introduced to the system. In response the intent issuer may be informed of the potential conflict and recommended an alternative action.

3.1.3 Adaptive Intent Realisation (AIR)

In this section we present the implementation of our Adaptive Intent Realisation (AIR) architecture. The architecture went through three distinct iterations in terms of internal features and how Adaptive Intent Realisation (AIR) communicates with external components.

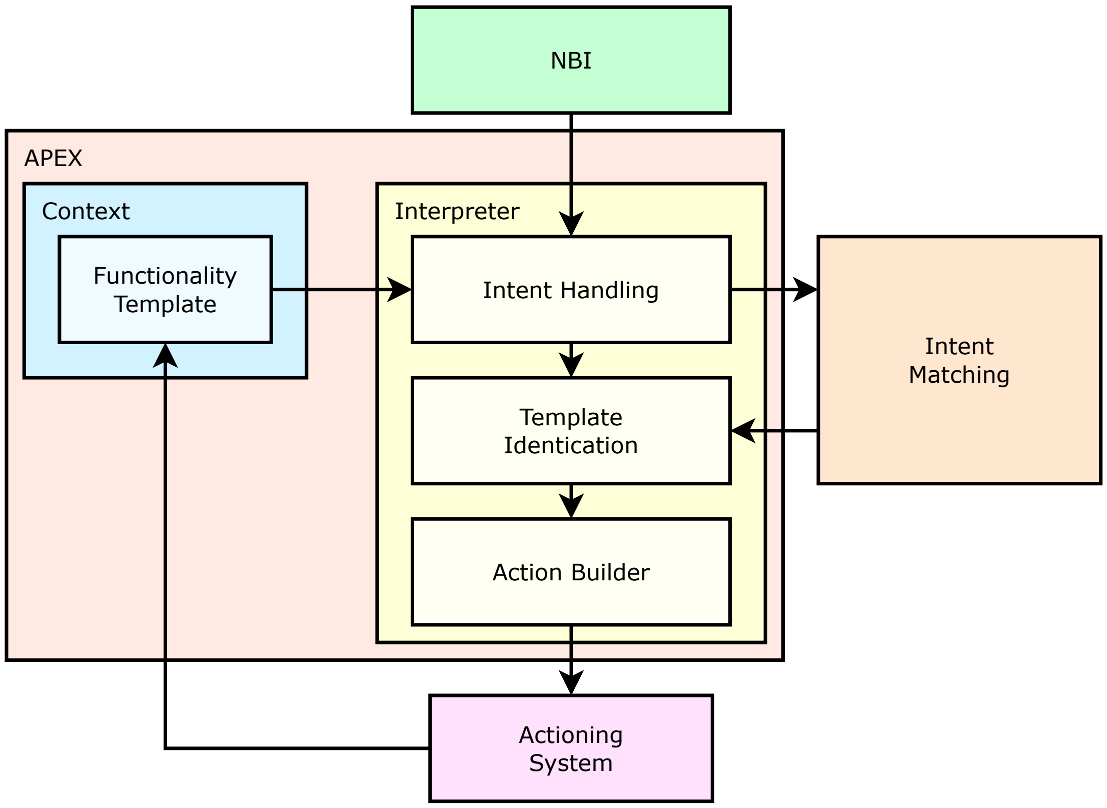


FIGURE 3.2: Intent Interpreter V1

In Figure 3.2 the intent interpreter shows the core features. These are Context, Interpreter and Intent Matching. Context stores the Functionality Templates which describe the capabilities of Actioning Systems available in the network. These are passed to the Interpreter when an intent is received. The Interpreter is responsible for passing the intent and Functionality Templates to Intent Matching and processing the output to identify the relevant Actioning System and appropriate operation. Intent Matching applies mathematical models to text representations of requests and operations to generate a comparative scores based on similarity.

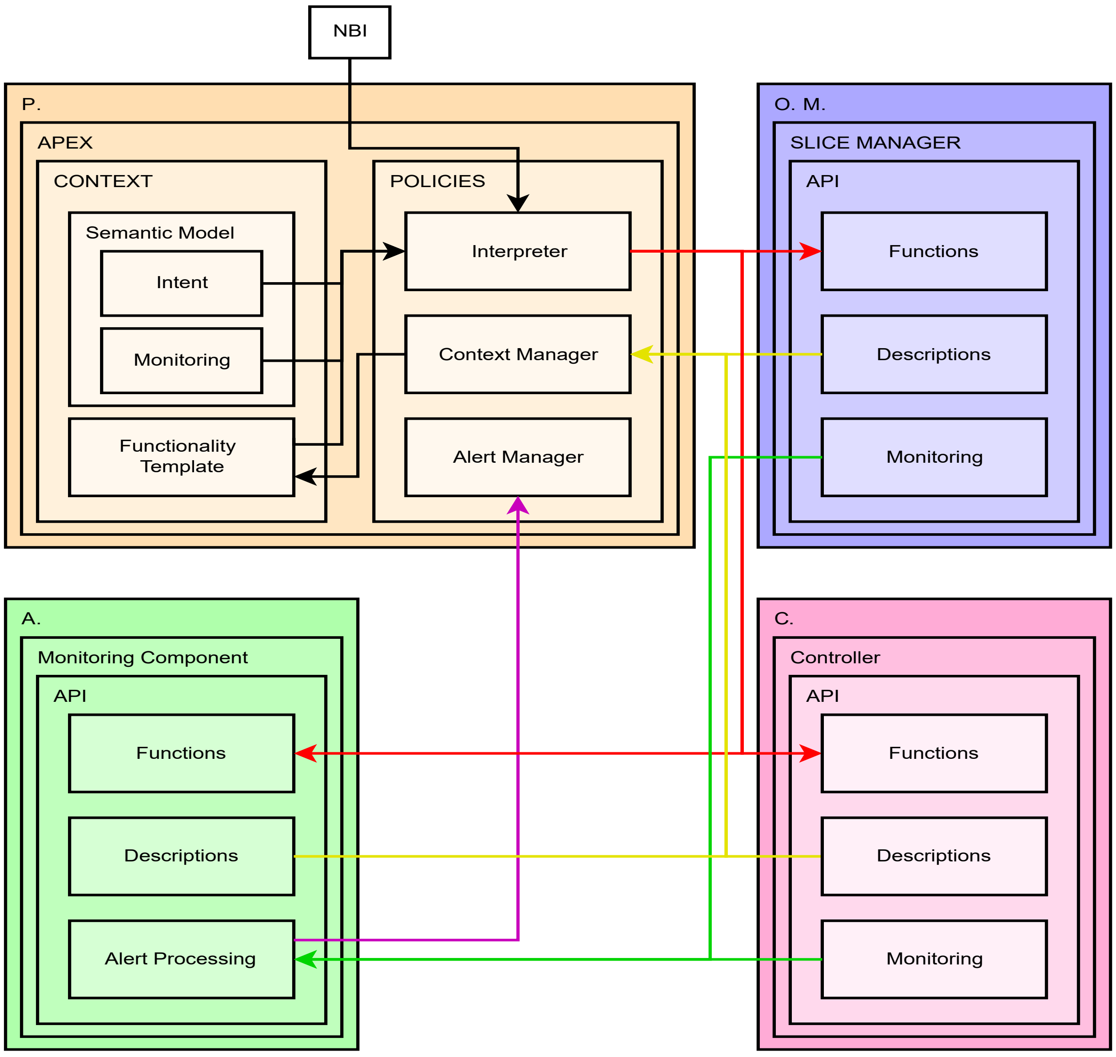


FIGURE 3.3: Intent Interpreter V2

In Figure 3.3 the Intent Interpreter has been extended with a number of important features. In previous versions monitoring was not explicitly addressed as an actioning system. The Intent Interpreter maintains its common interface with components regardless of role (Control, Orchestration, Management, Policy or Analytics) allowing it to easily fit into the closed-control loops of a system. As intent provides an abstract interface to simplify component communication it makes sense to also leverage this when it comes to monitoring. In this case analytical components are treated like every other component in the system, providing a functionality template describing its capabilities through accepted API specification. With this approach the interpreter interacts with every component in the same manner regardless of its role or functional capacity. With the addition of monitoring functionality the Semantic Model was added to Context.

The Semantic Model is used to evaluate incoming intent messages in two perspectives. The first is from an intent reasoning perspective. This perspective looks for keywords in the intent which can be analysed for ambiguities and based on the location of the interpreter within the larger system, intents may be modified before the mapping process is attempted. This will allow from more flexibility in the intent descriptions as location information may provide additional context when comparing the similarity scores from different actioning systems registered with the Intent Interpreter. The second perspective is in regard to monitoring and reporting. This perspective will provide intent fulfillment information to intents received by the Intent Interpreter. Many intents only require a reporting feature through the verification of the status code of intent driven interactions. However, some intents may have implicit prolonged monitoring requirements and these requirements are known by the intent issuer. Therefore a Semantic Model which enables the linking of provisioning actions with complementary monitoring actions, allows for automated intent assurance for a subset of implicit intent requests.

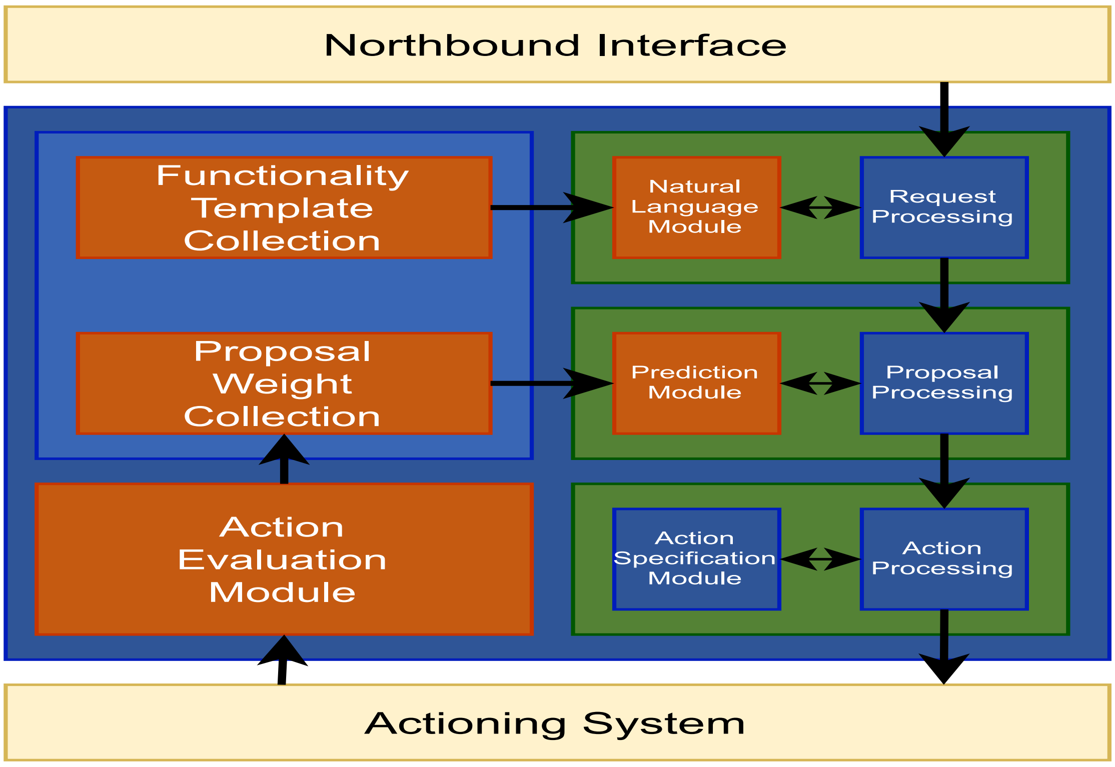


FIGURE 3.4: Intent Interpreter V3

In Figure 3.4 the Intent Interpreter goes through refinement. The Semantic model and Monitoring information has been redesigned to an Action Evaluation Module and Proposal Weight Collection. This module utilises the weights assigned to previous operations to impact the decision making on new intent interpretations. This approach maintains the monitoring aspect of previous intents while improving the future decisions of Intent Interpretation. The Context Manager and Alert Manager have been integrated into the Interpretation process allowing the system to engage with these features through the abstracted interface of the Intent Interpreter. This maintains a single point for entry for users through an abstracted interface for intent-driven management

3.2 Evaluation

This section describes how results will be read, quantified and compared. Each subsection will provide an insight into the system both in terms of the mechanisms established to bring about impact and the nature of this impact on the quality of the system as a whole. In subsection 3.2.1 the monitoring of the underlying network and the available features of OpenDaylight are described with a focus on those which can be influenced through policy decision. In subsection 3.2.2 the Mininet Emulation environment is leveraged for the realisation of policy driven network impact. Additional monitoring features such as probes are described to provide a complete monitored system. In subsection 3.2.3 the checks and balances within policy are described as a mechanism to increase the number of positive impacts produced through policy influence. In subsection 3.2.4 the Slice Manager is described for the execution of intent driven slice provisioning. In subsection 3.2.5 the Positioning Server is described for the execution of intent-driven device localisation as a Service. In subsection 3.2.6 the NFV Orchestrator is described for the execution of intent-driven NFV Instance as a Service.

3.2.1 OpenDaylight

OpenDaylight provides a modular open source platform for network automation and customization. The SDN controller exposes a number of monitoring and configuration functions, such as a modular, plug-in southbound interface approach with extensive support for standard network management interfaces such as OpenFlow. Built in monitoring applications will provide real time information of current network state accompanied by mechanisms for policy enforcement. Through numerous APIs OpenDaylight provides a flexible and transparent look at the network.

3.2.2 Mininet

Mininet provides a fully emulated network test bed for testing. Unlike simulators Mininet network traffic is real, that is traffic running on the network is generated and process much like traffic in todays networks. A straightforward and extensible Python API enables quick and easy network creation and configuration. This allows for the quick generation and configurations of many different networks for testing. Additionally hosts running on the network run standard Linux network software, allows for many different probing softwares to run on the network in parallel with the SDN controller based monitoring.

3.2.3 Policy Framework

Policies running within the system will implement instances of self-monitoring. This will decrease the number of instances of unexplained drift. Alternating supervised and unsupervised learning cycles can be an effective approach to mitigating this issue. However when accompanied by states of self-monitoring, previous network decisions will be taken into consideration in the decision making process to help provide an insight into the current network state.

3.2.4 Slice Manager

The Intent Interpreter was utilised in an intent-driven slice creation experiment. The creation of a 5G-CLARITY slice is a complex process requiring multiple interactions with the Slice Manager function. These interactions must also be informed, requiring expert knowledge of the system and the impact of parameters and attributes. To combat the complexity, several AI model have been generated to automate and simplify this process. The primary model is the Slice Creation Workflow (SCW) model. The role of the SCW model is to encapsulate all the required interactions with the Slice Manager in a single AI model. This allows for the Intent Interpreter to have a single point of contact for slice instantiation. Complementary models include the Radio Node Selection (RNS) model, the Compute Requirements (CR) model and the Slice QoS (SQS) model. Each of these models provide additional parameters to the SCW through the Intent Interpreter. The RNS model determines what radio access nodes are required for the provisioning of the slice. The CR model determines the compute resources required in the Edge cluster to service the slice. The SQS model allows for the configuration of QoS parameters for the underlying system and determines required parameters for the provisioned slice. The Intent Interpreter enables the straightforward execution of these models in a flexible way while allowing for new models to be integrated on the fly.

3.2.5 Positioning Server

The Intent Interpreter was utilised in an intent-driven line of sight experiment. In Release 16, 3GPP specified extensions to include Positioning Reference Signals (PRS) in the physical layer as well as a Location Managmenet Function (LMF) in the 5G Core. PRS provides methods to measure device signals and sends them to the LMF which processes the data using localisation algorithms to estimate device location. The accuracy of localisation algorithms is heavily impacted by obstacles between the device and the gNB. To address this challenge a use case was designed in which an intent message would be generated by the LMF to the 5G-CLARITY Intelligence Stratum to verify link condition of a UE and gNB pair.

A machine learning model referred to as the Non-Line of Sight (NLoS) model is instantiated on the AI Engine and registered with the Intent Interpreter through a Functionality Template. The NLoS model contains a pretrained prediction algorithm which predicts line of sight based on Channel Impluse Response (CIR) data. The experiment begins when the LMF generates an intent for the Intent Interpreter to verify the line of sight condition between a gNB and UE. The intent is processed by the Intent Interpreter and an API call to the NLoS model is generated. When communication between the Intent Interpreter and the NLoS model is established, the NLoS model generates a second intent requesting the CIR data for the UE and gNB defined in the initial intent. This intent is processed by the Intent Interpreter and an API call to the Data Lake is generated. The CIR data is retrieved and forwarded to the NLoS model fulfilling the second intent. The results of the prediction is retrieved from the NLoS model and forwarded to the LMF, fulfilling the initial intent.

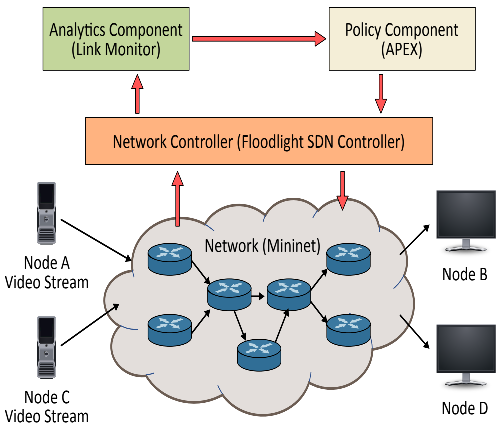
3.2.6 NFV Orchestrator

The Intent Interpreter was utilised in an intent-driven NFVI as a service experiment. Private 5G operators need to connect their enterprise services to deployed 5G networks slices, similar to those described in subsection 3.2.4. Virtualised enterprise services are represented by network service descriptors (NSDs) which can be on boarded to the NFVO. However, direct operation of the NFVO is challenging for private network operators not formally trained in 5G systems. To address this challenge a use case was designed in which an intent message would deploy NFV services on an NFVO.

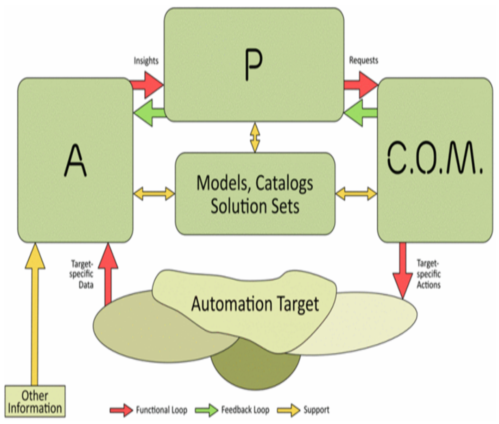
Similar to the approach described in subsection 3.2.4, workflow models are used to sequence the generation of follow up intents to fulfill the initial intent request. The NFVi workflow model is triggered by the Intent Interpreter. Each model in this process requests information from the NFVO through the generation on intents passed through the Intent Interpreter. The intent parameters are used to identify the appropriate NSD which is then passed to the NS selection model. The NS selection model resolves the NSD identifier generating parameter based on those provided in the initial intent. The VIM model then selects appropriate compute resources for the for the NFV service. The NFVi workflow model now containing all the information required to instantiate the network service requests this through the Intent Interpreter providing the required parameters. A record of intents is provided back to the intent issuer detailing the process automated by the models.

3.3 Architecture Maturity

This section details the transition of the work as it matured during the lifespan of the research. The contributions from the publications are highlighted with reference to the architectural adaptations.



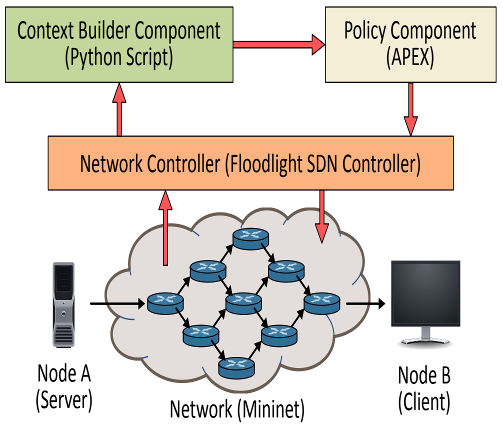
(a) The Closed Loop System Testbed Architecture



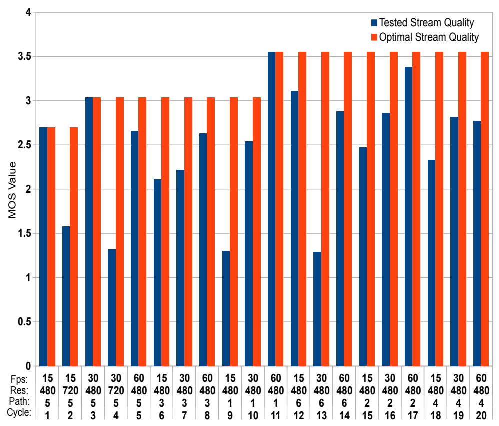
(b) The COMPA Reference Arcitecture

Figure 3.5: Initial Architectures Adopted by the Adaptive Policy testbed

In Figure 3.5 the testbed for policy driven closed loop network management is presented. This work validated Adaptive Policy in a realistic network environment for business goals and high-level network management. This work produced the first open-source network emulation environment for the testing of Adaptive Policy.


Context Builder Component (Python Script)
Policy Component (APEX)
Network Controller (Floodlight SDN Controller)
Node A (Server)
Network (Mininet)
Node B (Client)

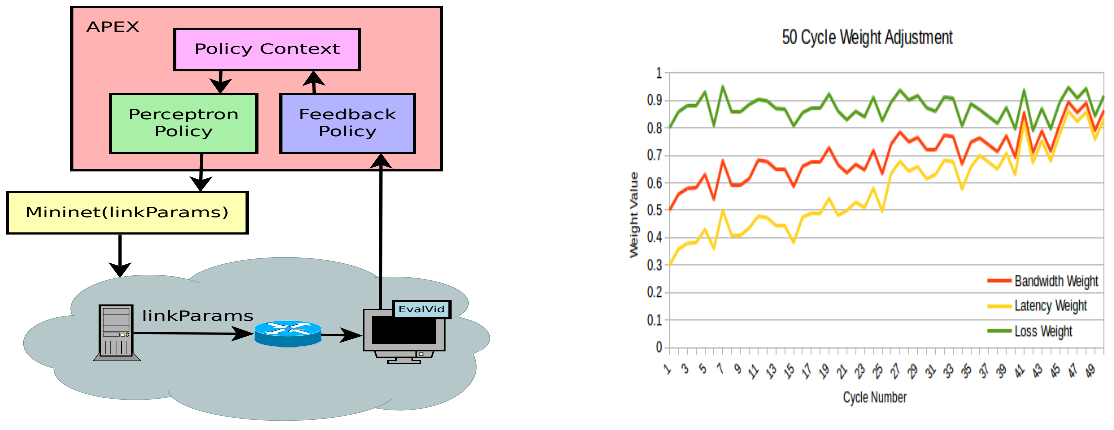
(a) Adapted System Architecture for low level execution



(b) Video Stream Monitoring

Figure 3.6: Architecture and Results for Video Quality Assurance

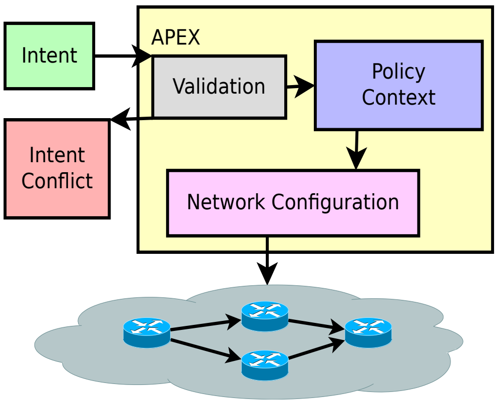
In Figure 3.6 the adaptive policy Approach to video quality assurance is presented. This work validated the use of Adaptive Policy for lightweight, low-level decision-making. Leveraging context to adapt network routing decisions based on QoE metrics. This work with the previous testbed shows Adaptive Policy as a viable mechanism at any level of the network.



(a) Embedded ML for Policy Architecture (b) Experiment Weight Results

FIGURE 3.7: Architecture and results for ML/Policy approach

In Figure 3.7 the hybrid machine learning/policy approach to optimise video path selection is presented. This work shows machine learning paradigms executed within an adaptive policy decision making system. This leverages powerful mathematical models to adaptive policy decisions before they are realized.

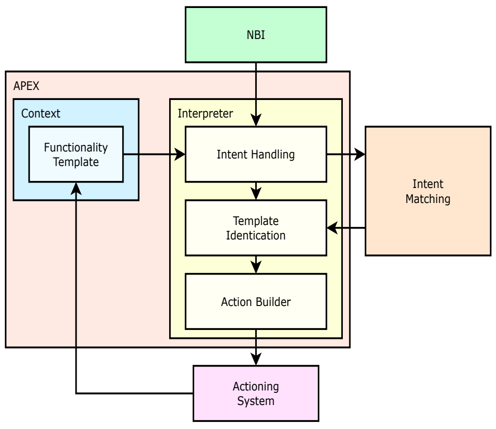


```
{
    "Who": "NetworkAdministrator",
    "What": "KPIEnforcement",
    "When": [{
            "What": "Start Time",
            "When": "08:00"
        },
        {
            "What": "End Time",
            "When": "12:00"
        }
    ],
    "Where": {
        "What": "Bandwidth",
        "How": "<1MBs"
    },
    "How": "null"
}
```

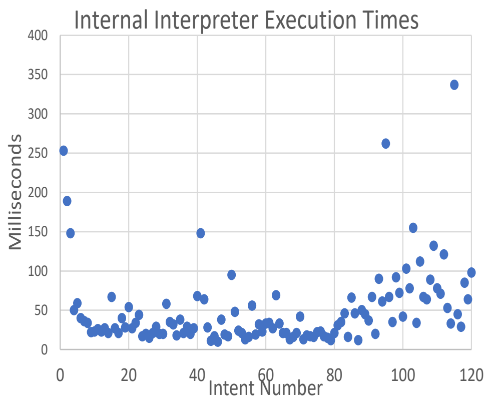
(a) Intent driven adaptive policy architecture (b) Intent example for experiment

FIGURE 3.8: Architecture and input for intent mechanism

In Figure 3.8 the mechanism for intent driven adaptive policy decision making is presented. This work presented the use of recursive model structures. Highlighting their machine readability and LoD when utilized for intent definition. This work also validated the use of a dictionary for intent context mapping.



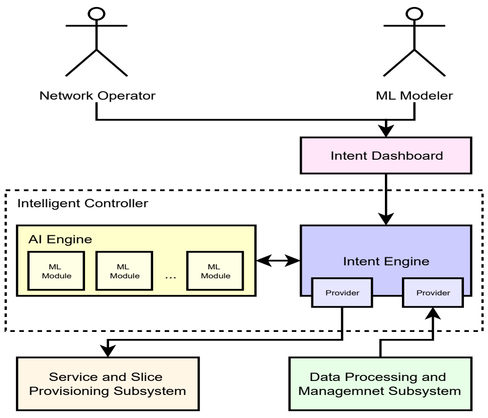
(a) Intent interpreter architecture



(b) Intent realisation times

Figure 3.9: Architecture and execution times for the intent interpreter

In Figure 3.9 the flexible interpreter for intent realisation is presented. This work presented an embedded NLP based component for fast evaluation of intent requests This work also validated the functionality template (an extension of the dictionary from previous work) for the inductive generation of intent-based actions.



(a) 5G-CLARITY architecture

	Slice provisioning	NLoS identification	NS deployment
Average intent execution time	≈ 3 minutes	≈ 1 second	≈ 4 minutes

(b) Intent execution times for UCs

Figure 3.10: Architecture and intent driven UC execution times

In Figure 3.10 NLP powered intent-based network managmeent for private 5G networks is presented. This work validates our approach through 3 varied use cases in real industrial private networks. This work also demonstrates Adaptive Intent Realization (AIR) concepts in Management, Orchestration and near RT RIC applications.

Chapter 4

Detailed Design and Evaluation

This chapter presents the publications as part of this work in chronological order. The core concepts, results and evaluation of each publication is described to illustrate the evolution of the work from the initial adoption of reference based architectures to the adaption and incorporation new concepts resulting in the Adaptive Intent Realisation (AIR) architecture. In section 4.1 Adaptive Policy is validated in a realistic network environment for business goals and high-level network management. This work produced the first open-source network emulation environment for the testing of Adaptive Policy. In section 4.2 Adaptive Policy is validated for lightweight, low-level decision-making. Leveraging context to adapt network routing decisions based on QoE metrics. This work with the previous testbed shows Adaptive Policy as a viable mechanism at any level of the network. In section 4.3 machine learning paradigms are executed within an adaptive policy decision making system. This leverages powerful mathematical models to adaptive policy decisions before they are realized. In section 4.4 recursive model structures are presented for intent-based communication. Highlighting their machine readability and LoD when utilized for intent definition. This work also validated the use of a dictionary for intent context mapping. In section 4.5 an embedded NLP based component is presented for fast evaluation of intent requests. This work also validated the functionality template (an extension based on dictionary from previous work) for the inductive generation of intent-based actions In section 4.6 our approach is validated through 3 varied use cases in real industrial private networks. This work demonstrated Adaptive Intent Realization (AIR) concepts in Management, Orchestration and near RT RIC applications.

4.1 A Testbed For Policy Driven Closed Loop Network Management

Testing network management use cases is extremely challenging. Presenting the output configurations from policy is straightforward, however testing and validating the impact from policy decisions in a network management scenario is difficult. This experience of policy authors motivated the need for a flexible testing architecture utilising emulated network scenarios to allow real data to be analysed in real time. The test bed was divided into four parts (Control, Analytics, Policy and Orchestration and Management) and testers could use components provided in the test bed or exchange components as long as they fulfilled a similar role.

4.1.1 Previous Work

As an introduction to Network Management Systems, policy has played a key role in decision making for these systems. Identifying the traditional role of policy versus the responsibilities of policy today shows that not only has policy expanded to influence larger areas of the network, but these networks have become exponentially more complex. There are a number of mechanisms and systems proposed to enable policy to adapt to this new level of complexity, one of which is APEX.

4.1.1.1 The APEX System

Adaptive Policy EXecution (APEX) is a policy engine for the execution of Adaptive Policy. The adaptive nature of this policy approach related to the policies ability to change its decision making process based on criteria outside of the execution cycle. The characteristics of Adaptive Policy is listed as:

- Decisions are made at run-time, rather than selection of predefined actions,
- Context information is used in the decision making, rather than information confined to the trigger event.
- Self-adaption to decision making is possible at run-time.

The Context feature is utilised to expose external information, such as business goals to adapt policy decisions. This information is shared between all running policies in the engine allowing for conflict detection capabilities. Context can also be exposed by the engine, allowing multiple engines in the same domain to share a common resource. The distribution, locking and persistence of context information is handled by the engine. The component which initialises the policy is referred to as a Trigger System. The component which receives policy output is referred to as an Actioning System. APEX supports a range of communication protocols and interfaces resulting in plug and play integration.

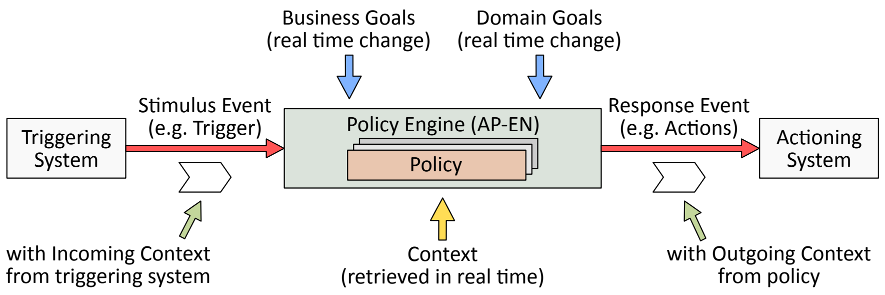


FIGURE 4.1: APEX: Linking Policy to goals and context

APEX provides a lightweight engine with flexible policy authoring features. Policy authors design policies with collections of states and tasks, then the steps and conditions used to navigate the policy is defined. This allows policies to adapt in real-time in response to changes in the domain as shown in Figure 4.1.

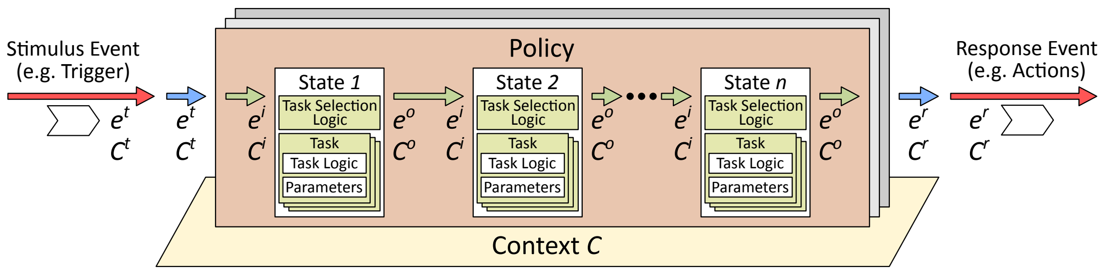


FIGURE 4.2: Policy Model

In Figure 4.2 policies are presented as a collection of state which are sequenced through context and event driven conditions. The number of states is determined by the policy author, however decision making patterns such as ECA (Event Condition Action) can be adopted.

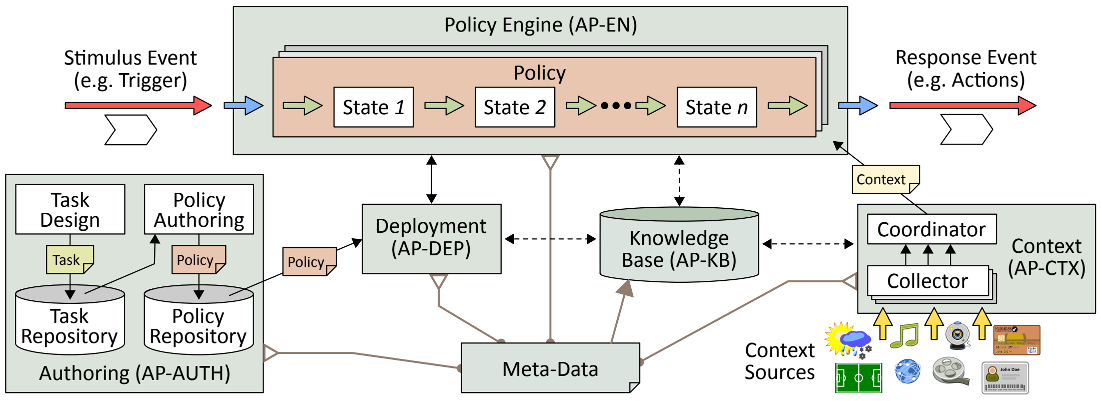


FIGURE 4.3: APEX Architecture

In Figure 4.3 the subsystems of the adaptive policy architecture are presented. These subsystems are compatmentaised into five distinct roles:

- AP-AUTH contains the tooling policy authors with repositories for policy and task specificaiton,
- AP-DEP contains the functionality for engine deployment,
- AP-EN contains the policy engine,
- AP-CTXT contains the connectors for distributed context between running engines,
- AP-KB contains knowledge based for policy metadata.

As part of the demonstration four APEX features are showcased. These are authoring, deployment, execution and monitoring. Three use cases were presented to illustrate the capabilities of adaptive policy. These use cases included video closed loop control, context-aware policy and international sales policy realisation. All use cases were made public available on Github [222].

4.1.2 Testbed Architecture

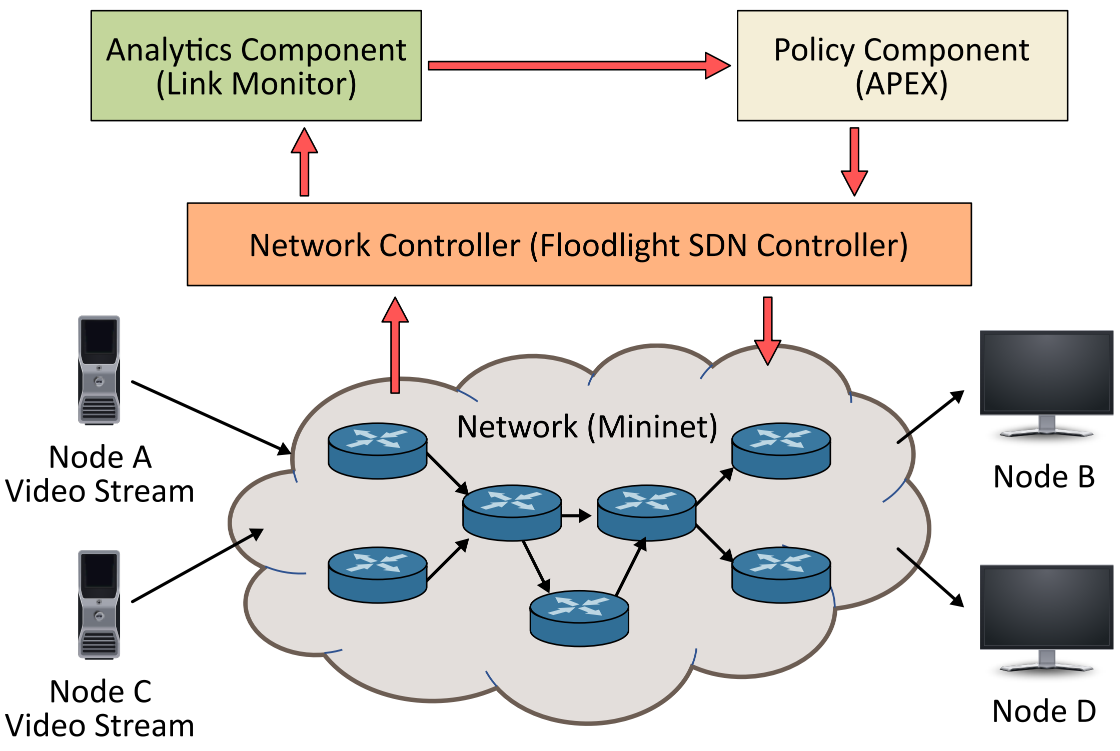


FIGURE 4.4: The Closed Loop System Testbed Architecture

This section describes the architecture of our closed loop testbed. The architecture, shown in Figure 4.4, is designed to be as open and repeatable as possible, in line with the test bed requirements described in section 1.1. The testbed is designed as an autonomic closed loop system. Inspired by the COMPA [19] reference architecture, it has the four main components shown in Figure 4.4. Communication in the architecture is realised using a distributed component approach [223].

The testbed provides a closed loop system which enables an adaptive policy to continuously manage the virtual network in an autonomic manner. When the virtual network experiences a change, information gathered in the SDN controller is packaged as an event. This event is analysed by the Analytics component to assess its significance. Significant events trigger adaptive policies in the Policy component. The response of the policies is sent to the SDN controller of the virtual network for deployment as a reconfiguration on the network.

The Managed Network: The Managed Network is a virtual network executing in the Mininet[1] virtual network emulator. We selected Mininet because it provides a well documented and extensible Python API (Application Programming Interface) and supports a wide range of features such as hosts capable of running basic networking applications and virtual switches that support the OpenFlow standard [224] for SDN. Mininet's lightweight nature avoids the need for installing, configuring and administering multiple orchestration systems [225] and the ease of building new virtual networks through the Python API allows for the quick and easy testing of different network scenarios. Another important capability for the testbed was Mininet's ability to scale to emulate very large networks.

The Network Controller: We selected the Floodlight[2] SDN controller as our network controller. The Network Controller is used to read the network configuration and status from the network and to configure modifications on the network. Floodlight supports OpenFlow on its southbound side through a well-defined *forwarding instruction set.* Floodlight exposes a RESTful (REpresentational State Transfer) northbound API, which is used by clients to monitor and configure the underlying network..

The Analytics Component: This component implements the domain specific analytic logic for the domain in our closed loop system. It monitors the network and if it arrives at an insight for its domain that may require intervention, it forwards that insight to the Policy component. As we used relatively straightforward scenarios in our testbed to date, we developed our own domain specific *LinkMonitor* analytics component as a Python program. There are many network analytic toolkits and platforms that can be used in more complex closed loop domains such as the *DCAE* component from ONAP [20].

The Policy Component: This component makes decisions on whether interventions should be made on the network. It uses the insights produced by the Analytics component to check if the operational goals set up for the network are being breached. If the goals are breached, the Policy component decides if and how to intervene in the network to mitigate those breaches. In our testbed, we used the APEX [15] adaptive policy engine running a set of adaptive policies designed for our scenario. We used a 4 stage Match, Establish, Decide and Act (MEDA) pattern based on the well-known

[1]`http://mininet.org`
[2]`http://www.projectfloodlight.org/floodlight/`

OODA loop [226] for our adaptive policy. The *Match* stage links an external trigger to a task, *Establish* reads the processed trigger and relevant context information, *Decide* identifies a response for the situation and finally *Act* takes the decision and generates an actionable response. During each of these stages/states, context information is continually being updated resulting in policies making informed decisions throughout each stage of the decision making process.

In addition to the main components, we used the Kafka[3] stream processing platform as a message bus for reliable communication between our distributed components. As Floodlight uses a RESTful interface, we developed a Kafka/REST interface in Python for interaction between Floodlight and the Analytics and Policy components.

4.1.3 Scenario

We have used a VPN networking domain to illustrate how our testbed can be used to build and actively manage a network in a closed loop manner using SDN, Analytics, and Policy. We have identified two scenarios, one where a VPN link failure threatens to breach customer SLAs (Service Level Agreements) and another for when SLAs become breached. We have developed and deployed solutions that mitigate against the failure in an adaptive and autonomic manner for those two scenarios.

4.1.3.1 Temporary Failure Scenario

The temporary failure scenario examines how an adaptive policy should handle the failure of a link which is shared between two customers. For the purpose of the scenario both customers are configured with a sample SLA that allows for 90 seconds of downtime per customer per year. One of the customers (Customer B) has already experienced 30 seconds of downtime in the year to date (YTD). Using the context information gathered from the virtual network the adaptive policy tries to mitigate the effect of the network loss on the SLA of Customer B by reducing the level of service to other customers who have experienced less down time in the year to date.

[3]`https://kafka.apache.org/`

4.1.3.2 Prolonged Failure Scenario

The prolonged failure scenario examines how should adaptive policy handle the prolonged failure of a link which is shared between the a set of customers. In the case of a prolonged failure, the SLAs of all customers will eventually be breached. Using context information gathered from the virtual network, the adaptive policy prioritises a specific subset of customers as a form of damage control.

4.1.4 Implementation

In this section, we describe the algorithms we have developed and implemented on our testbed for the scenarios described in subsection 4.1.3.

Algorithm 1 Network Initialisation and Control

```
Create a Mininet Object;
Based on topology information create and add all hosts(4),
  switches(7) and links(11) to the Mininet Object;
Create and add the Floodlight controller to the
  Mininet Object;
Start the Mininet Object;
while Pingall not successful do
    Ping all hosts;
end while
Create Kafka Consumer Object and subscribe
  to APEX output;
while Message in Kafka broker do
    if Action in message then
        Read action from message;
        Post REST request containing action
        to SDN controller;
    end if
end while
Stop Mininet Object;
```

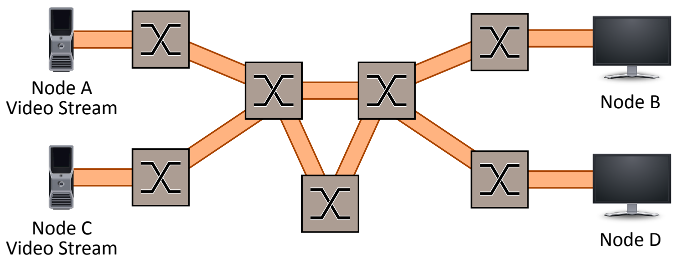


FIGURE 4.5: VPN Domain Network Topology

Algorithm 1 initialises and controls the Mininet and Floodlight components in the testbed. In Algorithm 1, a Mininet virtual network is initialised using the predefined topology and configuration shown in Figure 4.5. When the virtual network has been built, the algorithm starts the Floodlight controller. It then starts a Kafka Consumer which listens for configuration events sent from the APEX engine. When an event requesting a configuration change is received, it is parsed into a REST request and is forwarded to the Floodlight SDN controller. The controller then executes the configuration change on the network in Mininet.

Algorithm 2 Monitoring and Analysis

```
Create Floodlight Static Flow Pusher Object;
Create REST call to retrieve links on virtual network
  from SDN controller;
Create Kafka Producer Object;
while Virtual network exists do
    Sleep for 30 seconds;
    Retrieve links from network;
    if First iteration then
        Send virtual link information to APEX;
        Send virtual Customer information to APEX;
    else
        while List of links has next do
            if Link is not healthy then
                Send message to APEX containing
                context information;
            end if
        end while
    end if
end while
```

Algorithm 2 monitors and analyses the networking domain to check if any insights are observed. It is the implementation of the *Analysis* component of Figure 4.4 for the testbed. Algorithm 2 sends a REST request to the Floodlight SDN controller every 30 seconds (configurable), returning a summary of all the links in the virtual network. On the first iteration of the loop the information retrieved is set as the current known steady state of the system. In further iterations, the links are checked for irregularities by determining if the system has deviated from its steady state. If an irregularity is found, then this irregularity is forwarded to the *Policy* component as an event.

Algorithm 3 The Adaptive Policy For VPN SLA Mitigation

```
MATCH:
Read incoming information;
Check fields against Context album;
Update Context album;
Output incoming information with additional status
  information;
ESTABLISH:
Read information from the Match state;
Use context information to find if the issue is new
  or reoccurring;
Output link context information along with the specific
  problem information;
DECIDE:
Read information from the Establish state;
Read SLA information
if Unbroken SLA exits then
    Read customer context information and extract
      affected customers;
    Select customer who is furthest away from breaking
      their SLA;
else
    Read customer priority information and extract
      customer priorities;
    Select customer with the lowest priority value;
end if
Output selected customer context;
ACT:
Read information from the Decide state;
Extract necessary customer context information for
  the generation of the Floodlight REST request.
Package the information into an event along with
  any additional information for the transportation
  of the response;
Output the response;
```

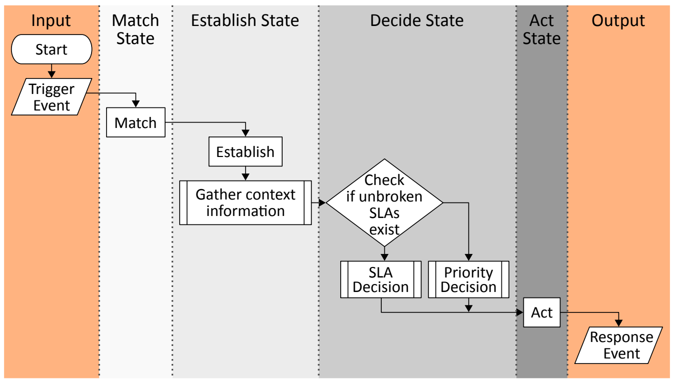


FIGURE 4.6: The Adaptive Policy as a Flowchart

Algorithm 3 performs SLA mitigation in the event of link failures on the VPN network. The algorithm, shown as a flowchart in Figure 4.6, is composed of four MEDA states.

In the *Match* state, the content of the incoming event is analysed and an update is made to the current state of the network if required.

The *Establish* state compares the state of the link reported on the incoming event with the current steady state of the link in its network model. It checks if the update is reporting that a link that was up has failed or if a link that had failed has recovered. It forwards this result to the next state.

The *Decide* state examines the SLA values for the customers on the link that has failed or has come up and determines in what manner traffic should be steered to give the best possible overall SLA mitigation outcome for the given link failure and customer SLA values.

The *Act* state packages the appropriate response and sends it to the SDN controller over Kafka.

4.1.5 Results & Experiences

In this section we will present the results from our experiment, highlighting the impact of adaptive policy through monitoring data of video stream during the scenario. A screenshot showing the user perceived quality of the video stream is also presented to better convey the user experience.

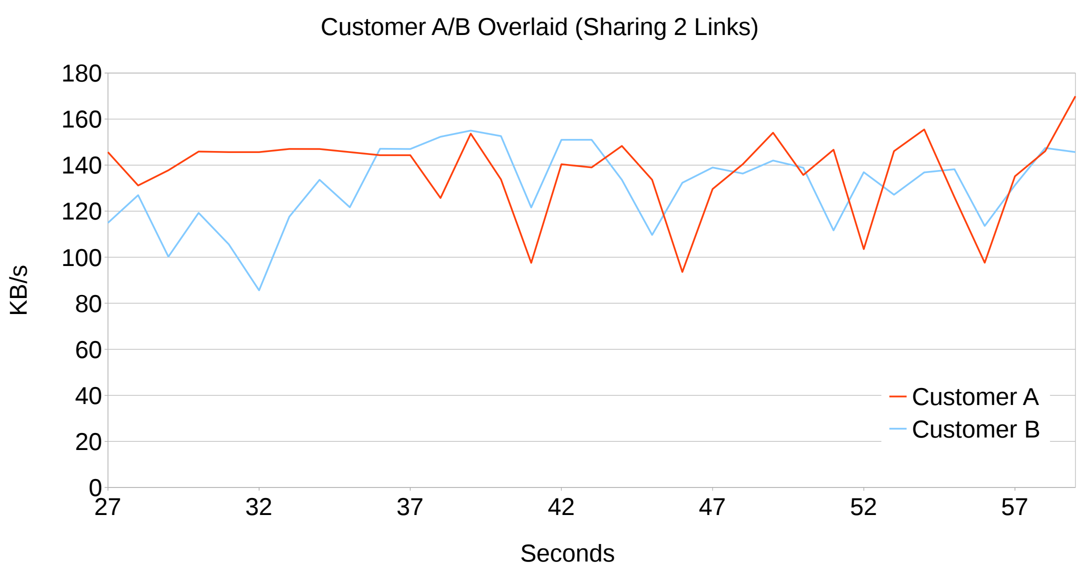
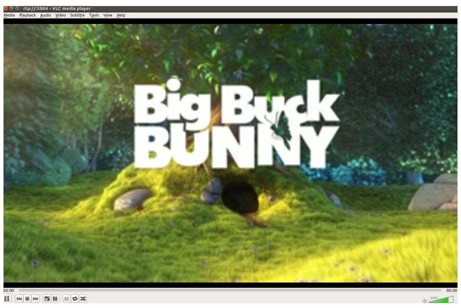
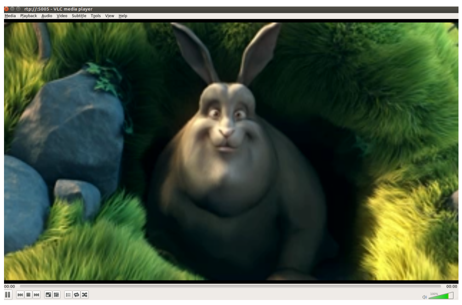


FIGURE 4.7: Customer A and B Sharing two Links



(a) Customer A



(b) Customer B

FIGURE 4.8: Customer A and B Sharing two Links: Stream Quality

At the beginning of the scenario, Customers *A* and *B* share the bandwidth of two links. In Figure 4.7 we see both customers experiencing between 100-150KB/s. This is sufficient bandwidth for both streams to be viewed without a reduction in image quality as seen in Figure 4.8.

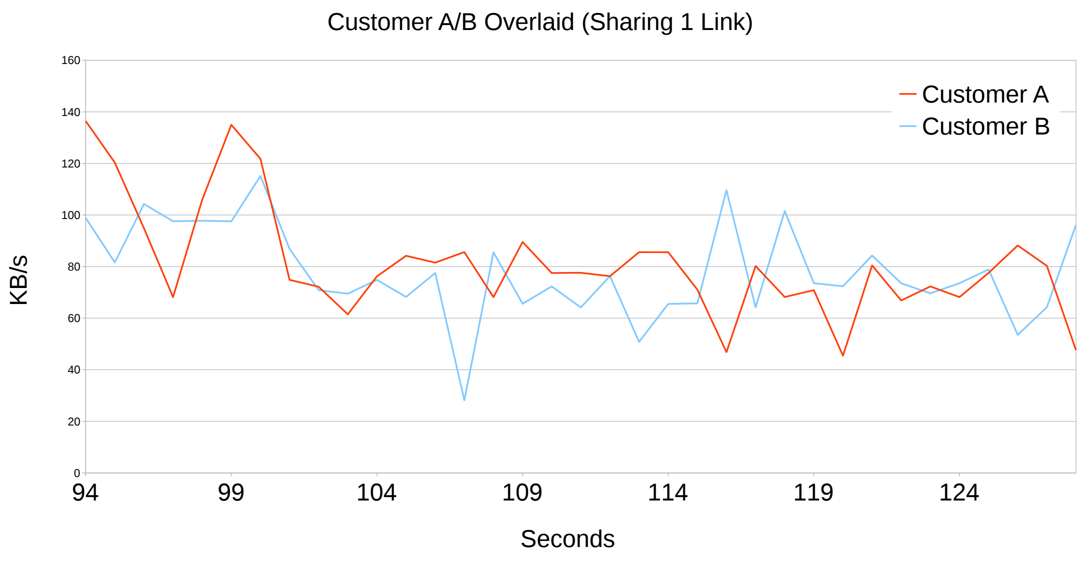
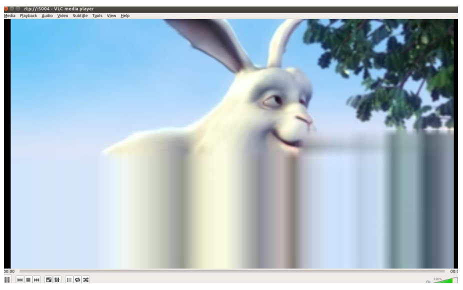
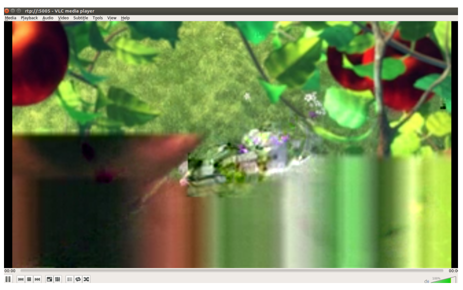


FIGURE 4.9: Customer A and B Sharing one Link



(a) Customer A



(b) Customer B

FIGURE 4.10: Customer A and B Sharing one Link: Stream Quality

As the scenario progresses, one of the shared links is brought down. The Floodlight controller quickly reconfigures the flow tables allowing for the stream to continue. However, as both streams are now using a single link, a bottleneck situation arises. In Figure 4.9 we see the bandwidth received by both customers drops to 60-80KB/s. This bandwidth drop causes a reduction in the image quality of both customer streams, as we can see in Figure 4.10. While the two streams are carried over the remaining link, neither of the streams provide a watchable experience.

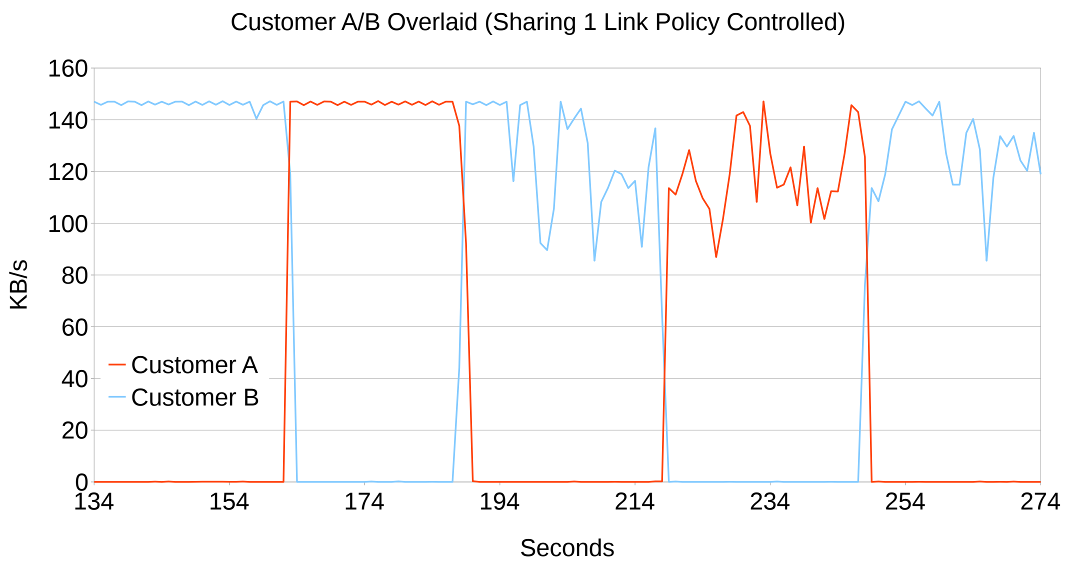
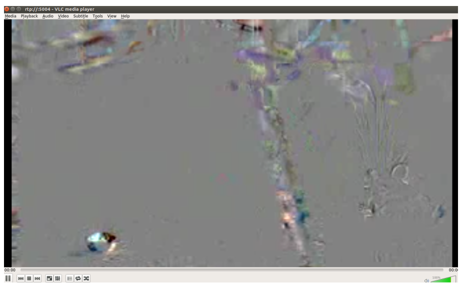
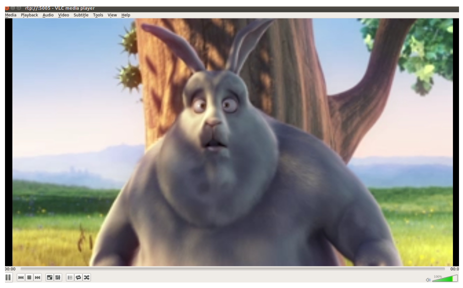


FIGURE 4.11: Policy cuts Customer B Stream



(a) Customer A

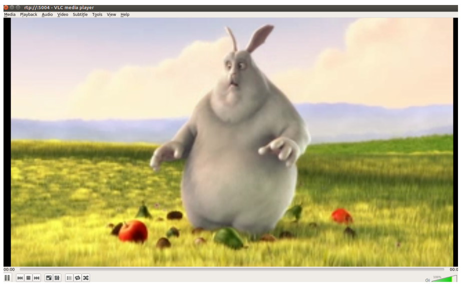


(b) Customer B

FIGURE 4.12: Policy cuts Customer A Stream: Stream Quality

To mitigate this situation, one of the streams must be restricted in order to release bandwidth to allow the quality of the other stream to recover. Our adaptive policy uses context information gathered from the Floodlight SDN Controller to decide which stream to restrict. In Figure 4.11 we can see the effect of the policy on the network. The policy ensures that at least one customer has an acceptable viewing experience during the fault situation.

In Figure 4.12 we observe the viewing experience of Customer A and B while A has been restricted on the network. Customer B returns to an optimal viewing experience similar to what we saw in Figure 4.8.


(a) Customer A

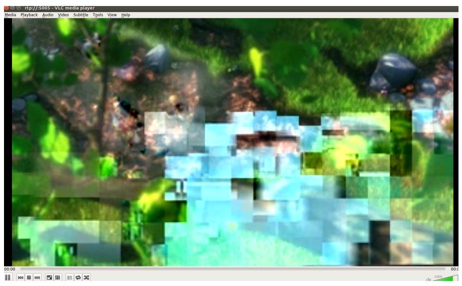

(b) Customer B

Figure 4.13: Policy cuts Customer B Stream: Stream Quality

As the scenario progresses, the policy realises that Customer A's SLA is closer to being violated compared to Customer B. It now changes its restricting decision and restricts the stream of Customer B so that Customer A's session is restored. This check is performed at 30 second intervals which is reflected in Figure 4.11. The variation in time of each cycle can be contributed to the time it takes for the execution of the control loop. In our testbed this execution time averaged 2.8 seconds. In Figure 4.13 we see the result of this decision, with Customer B being restricted and Customer A receiving a clear image.

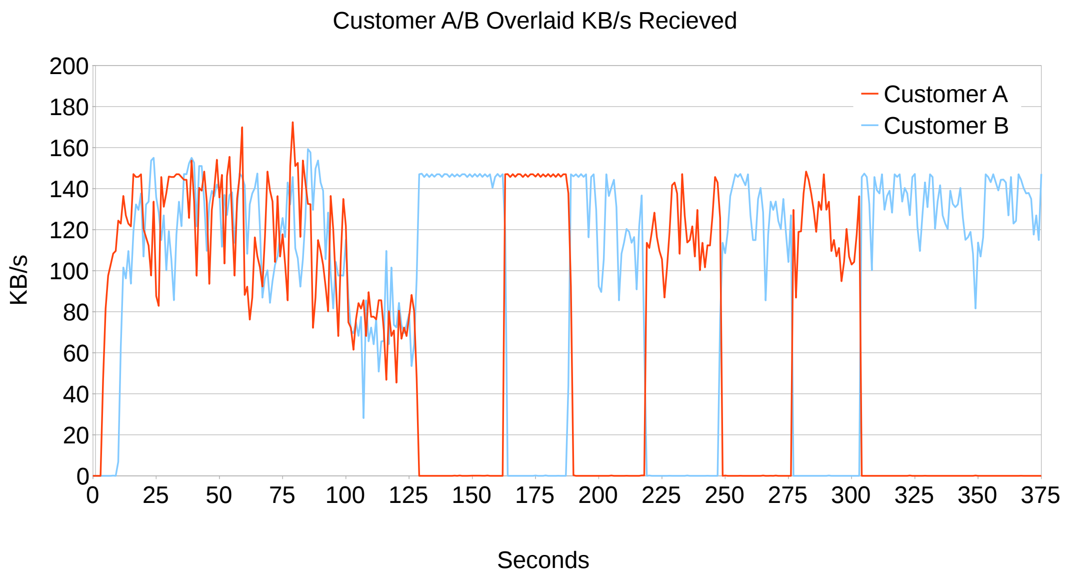


Figure 4.14: Customer Stream Overview

After a prolonged period of time both customers will violate their SLAs. At this point the policy's decision making strategy changes, instead focusing on the customer with the higher priority. The prioritised customer's stream will return to its optimal image quality at the expense of other. Figure 4.14 shows the prioritisation of Customer B near the end of the test when the policy shifts its decision making criteria from being SLA based to being priority based.

This work validated Adaptive Policy in a realistic network environment for business goals and high-level network management. This work also produced the first open-source network emulation environment for the testing of Adaptive Policy. The next step in our work was to investigate adaptive policy as a viable low level execution approach which require high performance, such as low latency decision making.

4.2 An Adaptive Policy Approach to Video Quality Assurance

There are a number of roles adaptive policy can have in networks today, one of which is a decision making mechanism for network or service optimisation. Given the prevalence of video in modern networks, adaptive policy could provide dynamic decision making to video service optimisation strategies while also acting as a layer of abstraction between different strategies. Policy has the capabilities to interact with many different network features and optimisation strategies, this work describes the initial steps into probing a unified approach to video quality assurance. Two requirements for this system are access and interaction to a network evaluation mechanism and an actioning system for decision implementation. This work presents a video quality assurance usage scenario through network path evaluation and selection.

4.2.1 System Architecture

This section presents the extension of the testbed architecture described in our previous work [227].

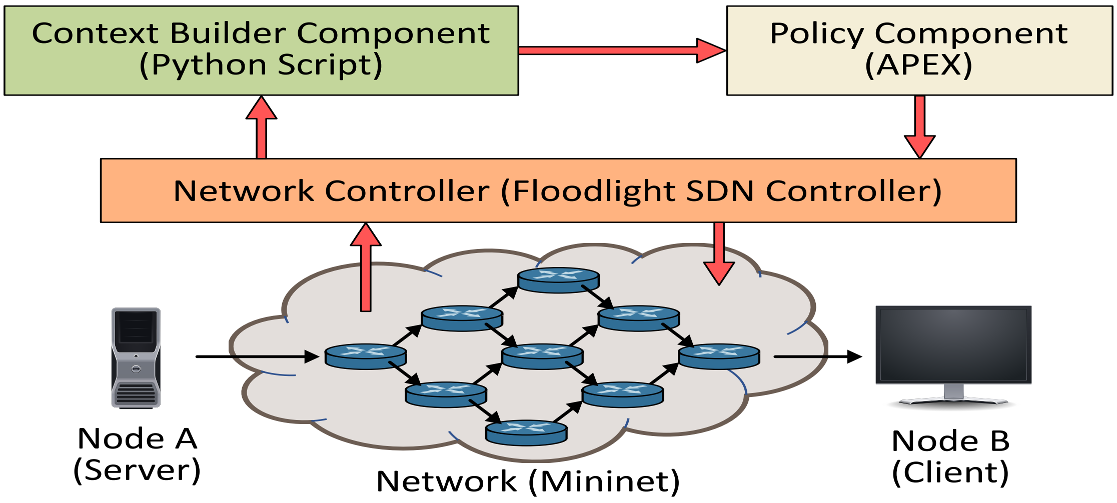


Figure 4.15: System Architecture

The architecture of our closed loop system shown in Figure 4.15 is expanded through the utilization of component capabilities and the addition of a video evaluation framework. Full description of the core components of the system architecture can be found at [228]. Modifications to the previous architecture include the *Context Builder* and the *EvalVid tool-set*.

Context Builder realizes a mediator between the deployed SDN controller and APEX. The main output of the context builder is default MOS and predefined paths, which become a basis for decision making. The component is realized as a script to facilitate fast changes and adaptation.

EvalVid tool-set[4] is used to evaluate the quality of the video stream. It allows for the generation of a MOS [229], which is a QoE metric. In this initial experimentation, we focus on packet loss, mainly to keep the dynamicity of the testbed low. A full QoS metric (jitter, delay, throughput) can be added later.

Policy Component is the implementation of our APEX engine [15] to deploy, trigger, and execute policies. For this paper we have defined a single policy, which uses a video quality evaluation metric to influence network configuration to optimize video streams. Policy decisions are implemented through Floodlight to reroute network streams.

All components in the architecture are Free and Open Source Software (FOSS). This means that our experimentation can be easily repeated using the instructions from [228].

[4]`http://www2.tkn.tu-berlin.de/research/evalvid/fw.html`

4.2.2 Testbed and Policy

This section details the implementation of our approach in four algorithms, presented in pseudocode.

Algorithm 4 Configure and Run Mininet

```
1: procedure MININET                                   ▷ Mininet, Floodlight, and Kafka
2:     FLC ← new Floodlight Controller
3:     MO ← Mininet Object (TCLink)
4:     MO ← Node A, Node B & 9 switches
5:     MO ← 14 (link × packetLossPercentage)
6:     MO ← FLC
7:     start MO                                        ▷ topology & controller
8:     while ¬Pingall do                               ▷ wait for nodes
9:         pinghosts()
10:    end while
11:    FLC ← enableFirewall()
12:    FLC ← cfgDefPath()                              ▷ configure default path
13:    while ¬Pingall do                               ▷ wait for configurations
14:        pinghosts()
15:    end while
16:    MOCL ← Mininet Object Command Line
17:    MOCL ← Kafka Consumer Object
18:    MOCL ← APEX output subscription
19:    while message do                                ▷ process Kafka messages
20:        if activePath ∈ message then
21:            ap ← activePath(message)
22:            FLC ← cfgFirewall(ap)
23:        end if
24:    end while
25:    stop MO                                         ▷ cleanup
26: end procedure
```

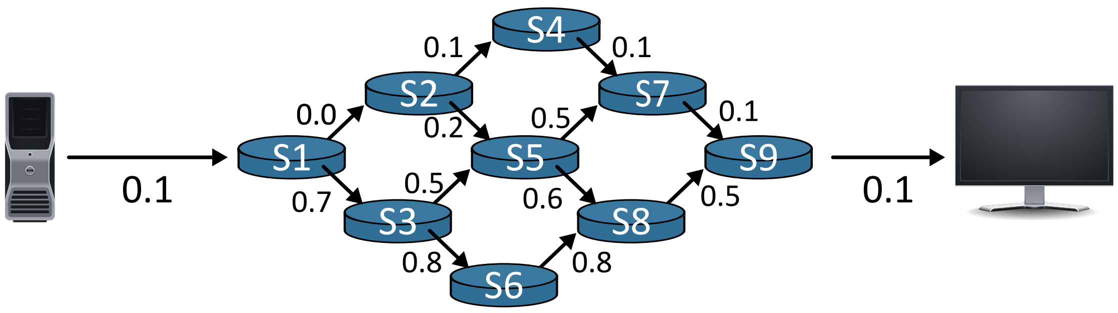


FIGURE 4.16: Network Diagram containing packet loss information for links

Algorithm 4 shows the creation of the Mininet emulated network topology displayed in Figure 4.16, connection of the Floodlight SDN controller and the enabling of the firewall used to enforce video stream paths. Hosts are then pinged to ensure the successful configuration of the network (lines 1-15). Access is then given to the Mininet Object Command Line for the execution of the VLC stream and client laid out in Algorithm 5. The script concludes with a messaging loop responsible for reconfiguring the firewall to adopt a new path specified by our policy running in APEX (lines 16-24). When the experiment is finished, we simply stop the MO (line 25).

Algorithm 5 Video Stream

```
1: procedure VIDEO STREAM                              ▷ Node B ← Node A
2:     MOCL ← Mininet Object Command Line
3:     XTB ← MOCL.xterm(NodeB)
4:     XTB ← vlcwrapper(url).record()
5:     XTA ← MOCL.xterm(NodeA)
6:     XTA ← vlcwrapper(stream).start(XTB.IP)
7:     XTA                          ▷ use RTP/MPEG, deactivate transcoding
8: end procedure
```

Algorithm 5 shows the procedure to generate a video stream from the server (Node A) to the client (Node B) where it is recorded for analyses. VLC is executed on Node B where it is configured to receive and record the video stream (lines 2-4). VLC is also executed on Node A where it is configured to stream video to our client using the RTP/MPEG protocol (lines 5-7).

Algorithm 6 Generating MOS

```
1: procedure REFERENCE                                        ▷ reference PSNR
2:     yuv ← decodeYuv(file)
3:     craw ← compRawVideo(yuv, fps, false)
4:     refmp4 ← mp4(craw)                                     ▷ hint RTP transport track
5:     yuvMp4 ← decodeYuv(refmp4)
6:     refPsnr ← psrn(yuvMp4)
        STORE(refPsnr)
7: end procedure
8: procedure MOS                                              ▷ streamed vs. reference PSNR
9:     yuvStream ← decodeYuv(streamedMp4)
10:    streamPsnr ← psrn(yuvStream)
        STORE(streamPsnr)
        GENERATEMOS(refPsnr, streamPsnr)
11: end procedure
```

Algorithm 6 shows the procedure to generate each MOS. The original video is used to generate a Peak Signal to Noise Ratio (PSNR) (lines 1-7) and this is stored as a reference PSNR (refPsnr). A PSNR is also generated for the streamed video (streamPsnr). Both PSNRs are used to generate a MOS through the EvalVid Tool-set (lines 8-11).

Algorithm 7 MOS/Active Path Policy

```
1:  procedure MATCH                                          ▷ Match state
2:      $e^o_m \leftarrow e^i_m.mos$
3:      $e^o_m \leftarrow e^i_m.vidParam$
4:  end procedure
5:  procedure ESTABLISH                                      ▷ Establish state
6:      $a_p \leftarrow ctxt(path)$
7:      $p_r \leftarrow ctxt(prevRes)$
8:      $p_f \leftarrow ctxt(prevFps)$
9:      if $e^i_e.mos > ctxt(maxMos)$ then
10:         $ctxt(maxMos) \leftarrow e^i_e.mos$
11:         $ctxt(optRes)$
12:         $ctxt(optFps)$
13:     end if
14: end procedure
15: procedure DECIDE                                         ▷ Decide state
16:     if $p_r = e^i_d.vidParam.res$ then
17:         if $p_f = e^i_d.vidParam.fps$ then
18:             $a_p \leftarrow rand(path)$
19:             $e^o_d \leftarrow a_p$
20:         end if
21:     end if
22:     if $e^i_d.mos >= ctxt(maxMos)$ then
23:         if $e^i_d.vidParam.res \mathrel{!}= maxRes$ then
24:             $e^o_d \leftarrow increaseRes$
25:         end if
26:     else if $e^i_d.vidParam.fps \mathrel{!}= maxFps$ then
27:         $e^o_d \leftarrow increaseFps$
28:     end if
29: end procedure
30: procedure ACT                                            ▷ Act state
31:     if $e^o_d.a_p$ then
32:         $e^o_a \leftarrow genCmd(e^o_d.a_p)$
33:     else if $e^o_d.increaseRes$ then
34:         $e^o_a \leftarrow genCmd(e^o_d.increaseRes)$
35:     else if $e^o_d.increaseFps$ then
36:         $e^o_a \leftarrow genCmd(e^o_d.increaseFps)$
37:     end if
38: end procedure
```

Algorithm 7 consumes the MOS of a streamed video and updates the recorded MOS for the path in the policy's context. The policy then determines whether this new MOS should cause the path for video streaming in the network to be amended or not. *Match*: the MOS of the incoming event is verified to ensure an expected value is received (i.e. a value between 1 and 5). After verification the MOS is passed on to the next state. *Establish*: compares the stored max MOS against the incoming MOS and records the greater in policy context. *Decide*: examine the possible paths stored in the policy context and the path with the best MOS is selected as the active path. If a new active path is selected the path identifier is passed onto the next state. *Act*: take the path identifier for the new active path and packages it into a response. This response is used to configure the new path for the video stream through the Floodlight SDN controller. The defined policy realizes a context-aware MEDA (Match, Establish, Decide, Act) policy [14].

4.2.3 Preliminary Evaluation

Using the network configuration outline in Figure 4.16 we created a brute force analysis of media streaming characteristics using MOS for six network paths with the packet loss rates; Path 1=0.5%, Path 2=1.0%, Path 3=1.5%, Path 4=3.0%, Path 5=2.5% and Path 6=2.0%. The brute force analysis involved the streaming of the Akiyo video sample at 15, 30 and 60 frames per second at resolutions of 480p, 720p and 1080p for each of the six paths. This created a baseline brute force analysis of 54 separate tests illustrated in Figure 4.17.

Our results found that the overall maximum MOS was achieved for path 1 which had the lowest packet loss rate at 0.5%. Path 4 generated the lowest maximum MOS, 2.82, and experienced the largest packet loss rate, 3.0%. While the highest and lowest generated path MOSs align with their packet loss rates the distribution of MOSs in Figure 4.17 show the association is not tight, making the path selection decision more challenging.

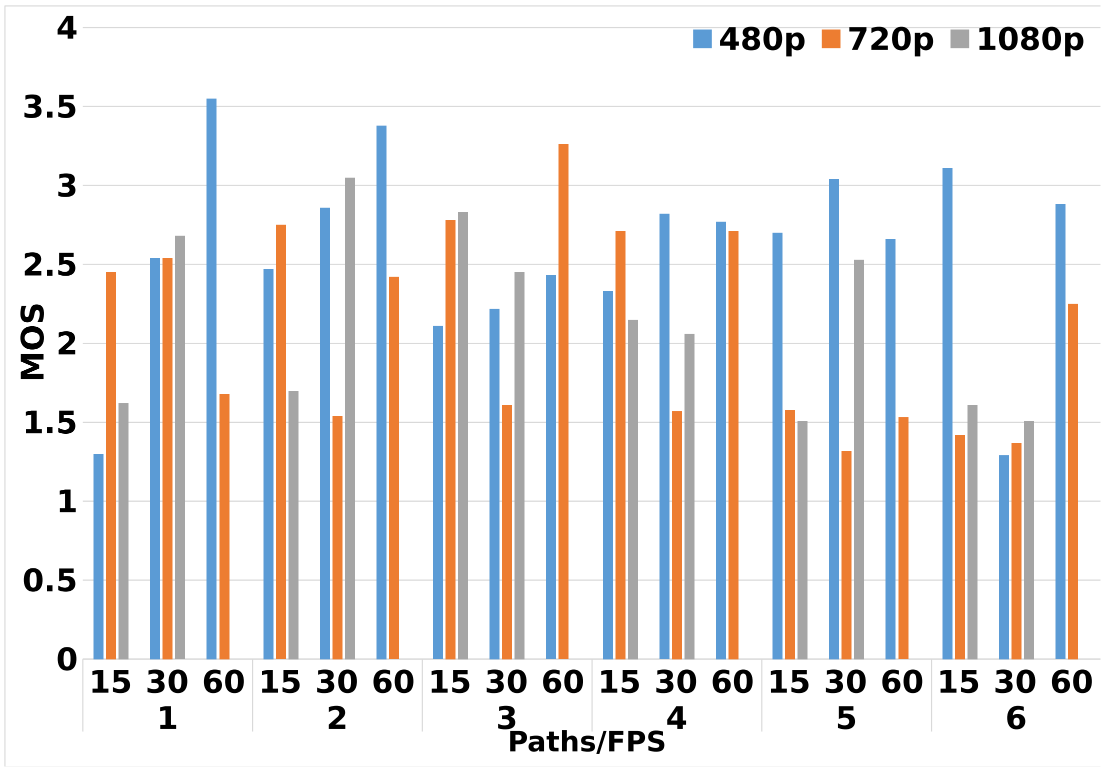


FIGURE 4.17: Brute force analysis results

The policy described in Algorithm 7 attempts to use a path climbing approach to optimize the video stream. In this experiment, Path 5 is arbitrarily selected and the MOS for a resolution of 480p at 15fps is calculated as 2.7. Path 5 is implemented by the adaptive path selection policy as optimal with the corresponding frame rate and resolution. In the 2nd cycle the Akiyo video sample is streamed with a resolution of 720p at 15fps. As the MOS attained is 1.58 and less than the previous optimal no path re-selection occurs and the stream configuration of 480p at 15fps remains optimal. Given that a resolution increase degraded the MOS, policy decides to increase the frame rate to 30fps and return the resolution to 480p. This configuration results in a MOS of 3.04. As this value exceeds the current maximum MOS the optimal stream configuration is updated to 480p at 30fps. No path re-selection is required, Path 5 is still considered optimal. Paths 3 and 1 are then considered. Path 1 with a resolution of 480p and a frame rate of 60fps results in a MOS of 3.55. This value exceeds the current optimal MOS. The optimal path is reconfigured to 1 and the optimal stream configuration is set to 480p at 60fps. The final remaining paths do not alter the optimal path selection.

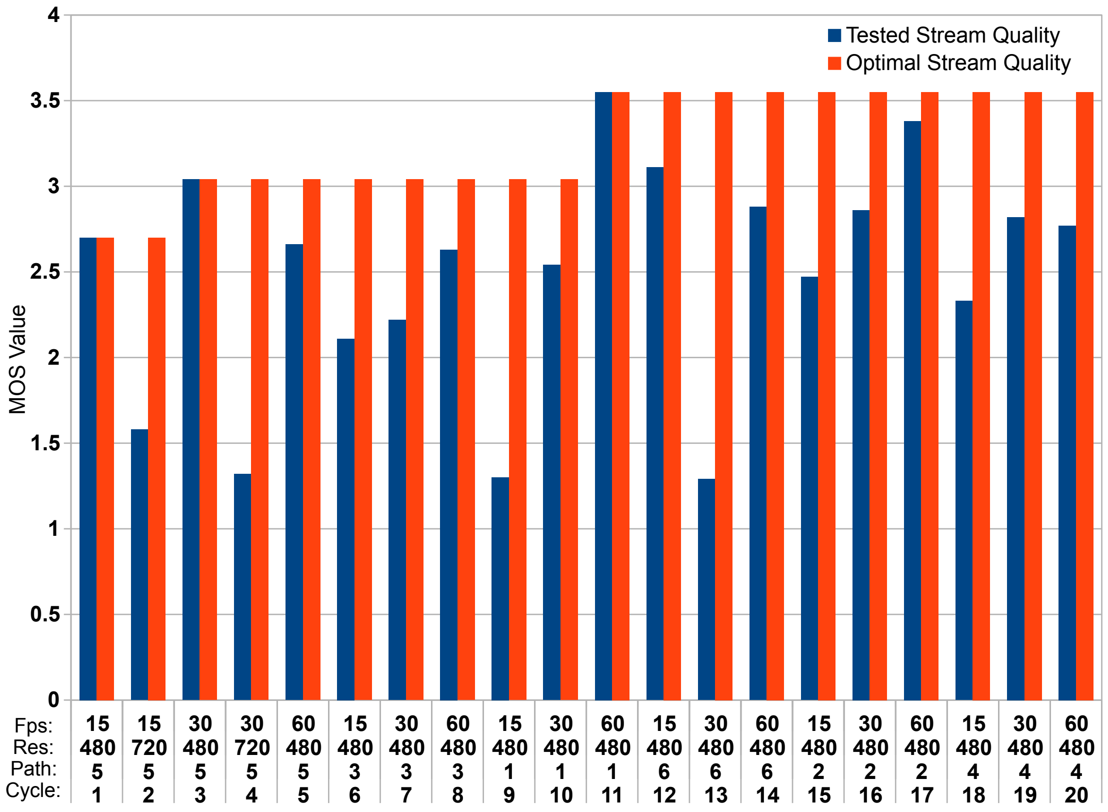


FIGURE 4.18: Current and Optimal Stream Comparison

Figure 4.18 shows the MOS for each tested path along side the MOS generated from the optimal path.

This work validated the use of Adaptive Policy for lightweight, low-level decision-making. Leveraging context to adapt network routing decisions based on QoE metrics. In co-operation with a collection of probes, network operators may adapt this approach to optimise video streams by leveraging low-level network context. This work with the previous testbed shows Adaptive Policy as a viable mechanism at any level of the network. The next step in our work was to explore the decision making capabilities of adaptive policy context awareness with state based execution. This lead to an investigation into machine learning paradigms and their application in adaptive policies.

4.3 A Hybrid Machine Learning/Policy Approach to Optimise Video Path Selection

Policy was introduced to network management as a mechanism for simplifying the complex task of managing a network. As networks are evolving becoming more complex, policy based management systems using predefined rules are struggling to manage these new dynamic networks. Context aware decisions are now required to manage these networks, as oppose to the predefined rule selection of traditional policy. A machine learning and policy hybrid could produce the level of adaptability needed to accurately and efficiently manage these networks in a context aware manner. This approach was applied to a real time network path selection use case.

4.3.1 System Architecture

This section presents the system architecture for the hybrid machine learning / policy approach. The approach is compartmentalised into three roles, these are policy context, network emulation and video evaluation.

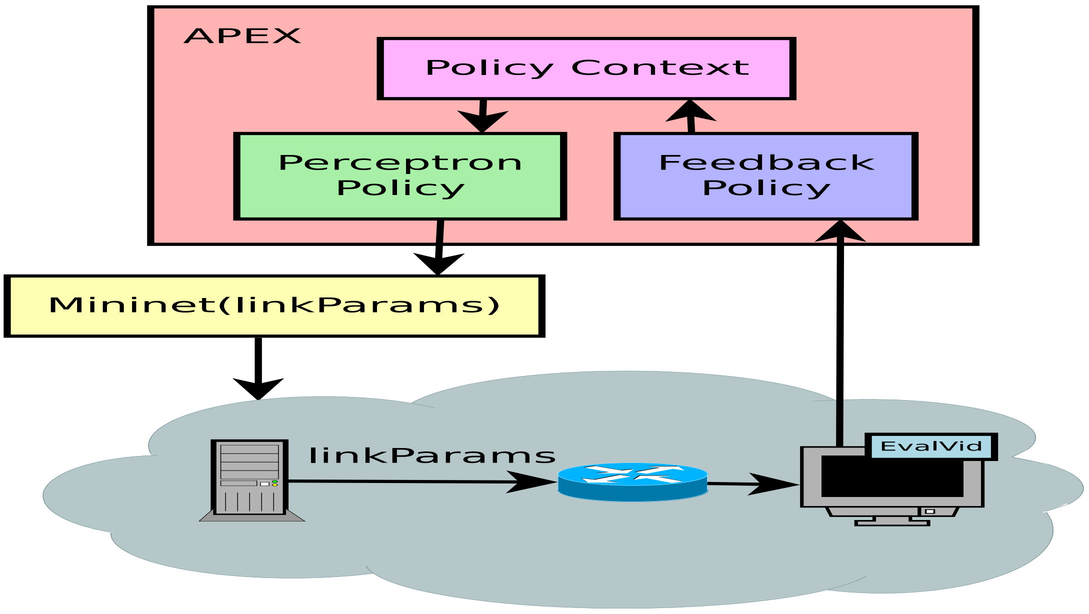


FIGURE 4.19: System Architecture

The diagram in Figure 4.19 shows our system architecture. The system runs a closed loop supervised learning cycle, with each cycle starting with initialisation of a predefined Mininet emulated network and closing with the update of the network metric weights stored in the Adaptive Policy EXecution engine. In this section we describe the primary components of system architecture and their role within the system.

Policy Context is handled by the APEX (Adaptive Policy Execution) engine. In this work we have defined two policies. The Perceptron Policy applies weights for specific Quality of Service metrics to predefined network configurations, both of which are stored as context information on the policy engine. The policy outputs the optimum network configuration for a video, based on weights stored in context. This configuration event is sent to Mininet which builds the appropriate network for the cycle and the video is stream and evaluated. The returning event triggers the Feedback Policy. The Feedback Policy stores the video evaluation metric in context and compares this value with the metric received from the previous cycle. The degree of change from the video evaluation metrics is used to adjust the weights stored in the policy context.

The Mininet Framework creates an emulated network. Mininet supports rapid configuration and emulation of a virutal network running real kernel, switch and application code[5]. Transmission of real video streams over this network is the foundation of our testing environment. As network characteristics and scenario configuration can be automated without the need for simulation, emulation maintains the integrity of obtaining real data in a flexible networking environment.

Video Evaluation is carried out by the Eval-Vid toolset[6]. This application runs on the client side of the network, generating Peak Signal to Noise Ratio, the ratio of signal power and corrupting noise of both the original and streamed videos, The toolset functionality enables automated generation of a Mean Opinion Score. This Mean Opinion Score is generated by comparing the PSNR of each frame in the original reference video with the streamed video, counting the number of frames with a MOS worse than the original in a given interval. The MOS generated by Eval-Vid provides an insight to whether changes made to network characteristics have contributed to a positive or negative impact on video stream quality.

[5]`http://www.mininet.org`

[6]`http://www2.tkn.tu-berlin.de/research/evalvid/fw.html`

4.3.2 Implementation

This section describes the implementation of the policies, network emulation and the video evaluation process for our approach, detailed as four algorithms in pseudocode.

Algorithm 8 Perceptron Policy

```
1: procedure MATCH                                                ▷ Match state
2:     while $e^i_m.pathList.hasNext()$ do                       ▷ process pathList
3:         $e^o_m \leftarrow e^i_m.pathList.next()$
4:     end while
5: end procedure
6: procedure ESTABLISH                                            ▷ Establish state
7:     $e^o_e \leftarrow e^i_e.paths$
8:     $e^o_e \leftarrow ctxt(metricWeights.bwWeight)$
9:     $e^o_e \leftarrow ctxt(metricWeights.latencyWeight)$
10:    $e^o_e \leftarrow ctxt(metricWeights.lossWeight)$
11: end procedure
12: procedure DECIDE                                              ▷ Decide state
13:    for $p$ in $e^i_d.paths$ do                                ▷ Loop paths
14:        $weightedBw \leftarrow normalise(p.bw) * bwWeight$
15:        $weightedLatency \leftarrow normalise(p.latency) * latencyWeight$
16:        $weightedLoss \leftarrow normalise(p.loss) * lossWeight$
17:        $weightedSum \leftarrow weightedBw + weightedLatency + weightedLoss$
18:        if $weightedSum > largestWeightedSum$ then
19:            $largestWeightedSum \leftarrow weightedSum$
20:            $largestPathId \leftarrow p.id$
21:        end if
22:    end for
23:    $e^o_d \leftarrow largestWeightedSum$
24:    $e^o_d \leftarrow largestPathId$
25: end procedure
26: procedure ACT                                                 ▷ Act state
27:    $e^o_a \leftarrow parse(e^i_a.largestPathId, e^i_a.largestWeightedSum)$
28: end procedure
```

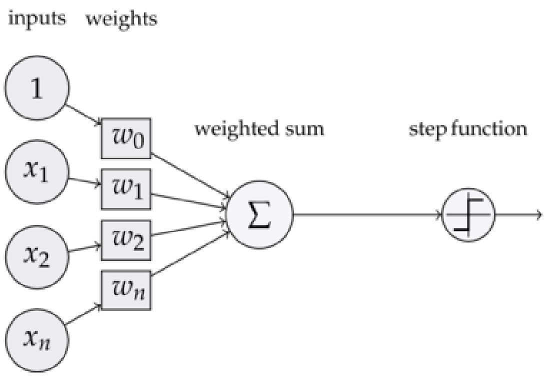


FIGURE 4.20: Graphical Representation of a Perceptron

The *Perceptron Policy* (Algorithm 8) presents the four states used in the policy's application of a single layer neural network approach. *Match*: Processes a list of optional paths, one of which will be recommended for video streaming. *Establish*: Queries policy engine context information for weights corresponding to predefined link characteristics/attributes. *Decide*: Applies a perceptron single layer neural network approach through normalisation of path attributes, application of respective weights and extraction of the path identifier with the largest weighted sum of attributes. This processes in shown in Figure 4.20. *Act*: Parses the output of the *Decide* state, prepares and outputs an event from the policy.

Algorithm 9 Mininet Topology & Video Stream

```
 1: procedure MININET                                ▷ Mininet, Floodlight, and Kafka
 2:     MO ← Mininet Object (TCLink)
 3:     bwMetric ← (float) sys.arg[1]
 4:     ltMetric ← (float) sys.arg[2]
 5:     lsMetric ← (float) sys.arg[3]
 6:     linkParam ← [bwMetric, ltMetric, lsMetric]
 7:     MO ← c1                                      ▷ Controller C1
 8:     MO ← n[a1, a2]                               ▷ Node A1, A2
 9:     MO ← s1                                      ▷ Switch S1
10:     l1 ← l(linkParam)
11:     MO ← [l1, l2]                                ▷ Link L1 L2
12:     start MO                                     ▷ topology & controller
13:     while ¬Pingall do                            ▷ wait for nodes
14:         pinghosts()
15:     end while
16:     MOCL ← Mininet Object Command Line
17:     XTB ← MOCL.xterm(NodeA2)
18:     XTB ← vlcwrapper(url).record()
19:     XTA ← MOCL.xterm(NodeA1)
20:     XTA ← vlcwrapper(stream).start(XTB.IP)
21:     XTA                                          ▷ use RTP/MPEG, deactivate transcoding
22:     stop MO                                      ▷ cleanup
23: end procedure
```

The *Mininet Topology & Video Stream* (Algorithm 9) shows initialisation of the Mininet emulated network topology and recording of the video streamed across the network. The network is configured with parameters for bandwidth (*bwMetric*), latency (*ltMetric*), and packet loss (*lsMetric*). A *Pingall* is executed to ensure successful initialisation of the network. *VLC* is used as the video streaming tool, executed on Node A1 and A2, configured to record the streamed mp4.

Algorithm 10 Video Evaluation

```
1: procedure REFERENCE                                        ▷ reference PSNR
2:     yuv ← decodeYuv(file)
3:     craw ← compRawVideo(yuv, fps, false)
4:     refmp4 ← mp4(craw)                                     ▷ hint RTP transport track
5:     yuvMp4 ← decodeYuv(refmp4)
6:     refPsnr ← psrn(yuvMp4)
        STORE(refPsnr)
7: end procedure
8: procedure MOS                                              ▷ streamed vs. reference PSNR
9:     if $newStreamedMp4$ then
10:        yuvStream ← decodeYuv(streamedMp4)
11:        streamPsnr ← psrn(yuvStream)
        STORE(streamPsnr)
12:        mos ← generateMOS(refPsnr, streamPsnr)
13:        output ← policyEvent(mos)
14:    end if
15: end procedure
```

The *Video Evaluation* (Algorithm 10) shows how the video MOS (Mean Opinion Score) is generated using the Eval-Vid toolset. Firstly, a PSNR (Peak Signal to Noise Ratio) is generated for the original video file, described in the reference procedure. Next a PSNR is generated for the recorded streamed video. The comparison of the PSNR files results in a MOS value, a QoS metric with a range 1 to 5 with 1 representing exceptionally poor quality and 5 representing no degradation in quality.

Algorithm 11 Feedback Policy

```
 1: procedure MATCH                                                ▷ Match state
 2:     $e_m^o \leftarrow parse(e_m^i.mos)$
 3: end procedure
 4: procedure ESTABLISH                                            ▷ Establish state
 5:     $e_e^o \leftarrow e_e^i.mos$
 6:     $e_e^o \leftarrow ctxt(mosValues.size())$
 7:     if $ctxt(mosValues.size()) - 1 < 0$ then
 8:         $isFirstMos \leftarrow true$
 9:         $previousMos \leftarrow 0$
10:     else
11:         $isFirstMos \leftarrow false$
12:         $previousMos \leftarrow mosValues[size - 1]$
13:     end if
14:     $e_e^o \leftarrow (isFirstMos, previousMos)$
15: end procedure
16: procedure DECIDE                                               ▷ Decide state
17:     if $e_d^i.firstMos! = true$ then
18:         $slope \leftarrow \frac{e_d^i.mos - e_d^i.previousMos}{2-1}$
19:         $learningConst \leftarrow 0.1$
20:         $weightChange \leftarrow slope * learningConst$
21:         if $mosValues.size()\%5! = 0$ then
22:             $randomBw = 0$
23:             $randomLt = 0$
24:             $randomLs = 0$
25:         else
26:             $randomBw = (random() * 10) - 0.05$
27:             $randomLt = (random() * 10) - 0.05$
28:             $randomLs = (random() * 10) - 0.05$
29:         end if
          STORE()$ctxt(metricWeights.bwWeight) + weightChange + randomBw$
          STORE()$ctxt(metricWeights.ltWeight) + weightChange + randomLt$
          STORE()$ctxt(metricWeights.lsWeight) + weightChange + randomLs$
30:         $metricWeightsUpdated \leftarrow true$
31:         $e_d^o \leftarrow metricWeightsUpdated$
32:     else
33:         $metricWeightsUpdated \leftarrow false$
34:         $e_d^o \leftarrow metricWeightsUpdated$
35:     end if
36: end procedure
37: procedure ACT                                                  ▷ Act state
38:     $e_a^o \leftarrow parse(e_a^i.metricWeightsUpdated)$
39: end procedure
```

The *Feedback Policy* (Algorithm 11) presents the four states used to adjust weight values when our video evaluation metric is received by the policy. *Match*: Processes the MOS into a workable format for the policy. *Establish*: Queries policy engine context information for last received MOS value. *Decide*: Using context information accompanied by the MOS value, a slope is generated to represent the improvement/degradation in video quality. A learning constant is then applied to create our weight change variable. A random adjustment is implemented every 5 cycles of the policy ranging from -0.05 to 0.05. Our new weights are then stored in our policy engine context information and a report is prepared for the next state. *Act*: Prepares the report from the *Decide* state the policy output.

4.3.3 Preliminary Evaluation

In this section we present results of the preliminary evaluation of our approach. One goal of our preliminary evaluation is to get initial indications on the approaches applicability. Another goal is to assess if the selected learning parameters and weightings are appropriate and to determine the degree and length of time for which learning should be applied.

Cycle:	MOS:	Bandwidth Weight	Latency Weight	Loss Weight
1	1.75	0.5	0.3	0.8
2	2.33	0.55799997	0.358	0.858
3	2.54	0.579	0.379	0.879
4	2.57	0.582	0.382	0.882
5	3.05	0.63	0.43	0.93

Table 4.1: Weight adjustment for cycles 1-5

The first five results of a 50 cycle test are shown in Table 4.1. Bandwidth, Latency and Loss Weights are the configured path metric for the video stream in Mininet for the current test cycle. The initial value of the weights are set manually.

Packet loss of I-frame packets can have a significant impact on video quality[230]. Therefore, we have assigned packet loss the largest weight of the three metrics. Bandwidth is assigned a lower weight and latency is assigned the lowest weight. In the first cycle we see a reported MOS value of 1.75, as this is the first test the weights are not affected and we continue to cycle two. Cycle two reports a MOS value of 2.33, this increase in MOS from 1.75 generates a slope of approx 0.58, after applying the learning constant of 0.1 we are left with a proposed weight change of 0.058. This cycle is repeated for the remainder of the test, with weights adjusted accordingly.

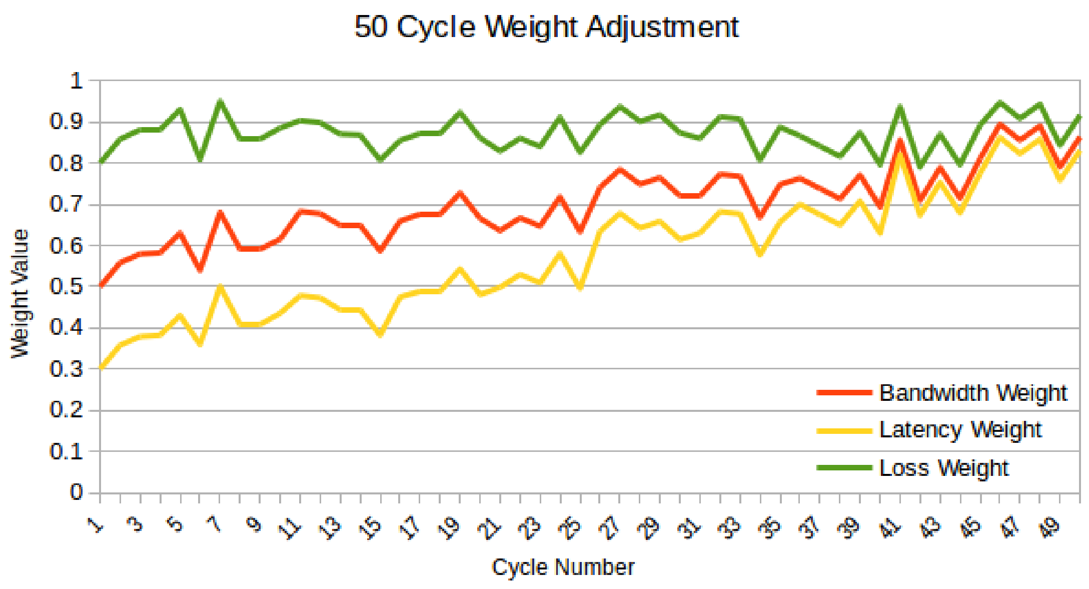


Figure 4.21: Weight Adjustment for 50 Cycle Test

The observed weights for a 50 cycle test are presented in Figure 4.21, detailing the weight adjustment made in each cycle and the overall weight trends. From Figure 4.21 we see the Loss Weight (green) exhibits an overall increase from 0.8 to 0.91, showing that although it had been assigned the largest initial weight, it did not increase significantly after 50 cycles. The Bandwidth weight (red) was assigned the second largest weight and at the end of our 50 cycle test the weight value had increased from 0.5 to 0.86, a notable increase. This increase of 0.36 shows that bandwidth had more of an impact on the MOS generated from video than initially thought, observable in the upward trend depicted in the graph. The Latency weight (yellow) had the most significant increase of the three weights. Increasing from 0.3 initially to end the 50 cycle test at 0.83. This steady upward trend shows that latency had a much larger impact on of video stream quality than our initial weights represent.

The results indicate that 50 cycles are not sufficient to allow weight to plateau at their representative values. Nevertheless, we were encouraged by the results. With adjustments to the our Perceptron policy learning rate in tests with more cycles, we are confident we can achieve a representative weighted path evaluation algorithm for video streams.

This work shows machine learning paradigms executed within an adaptive policy decision making system. This work also leverages powerful mathematical models to adaptive policy decisions before they are realized. However, deploying these models in a network management scenario is challenging. The next step in our work aimed to abstract the policy selection and execution from the user. We investigated intent as a viable abstraction technique for engagement with adaptive policy.

4.4 A Mechanism for Intent Driven Adaptive Policy Decision Making

This demonstration focuses on the comparison and identification of conflicts between independent goals described through intent. Our approach, implemented in the Adaptive Policy EXecution system, is presented in 3 stages:

- Intent generation, describing the structure of an intent,
- Intent comparability, detailing the requirements allowing for the effective comparison of goals,
- Conflict resolution, detailing the process of identifying appropriate responses in accordance with already established intent goals.

Our demonstration adopts intent as a driving mechanism for a network configuration usage scenario. The usage scenario describes a network emulation tool influenced through policy. Intent events received by policy and trigger a validation cycle where active intents are compared and influence new network configurations. The objective of the demonstration is to identify the requirements of an intent driven approach and provide processes to meet these requirements. In subsection 4.4.1 we detail the structuring of the intent information. In subsection 4.4.2 we discuss the granularity required in intent information to produce actionable responses and identify achieved goals. In subsection 4.4.3 we describe the mechanism responsible for identifying and mitigating conflicts between distinct intents. In subsection 4.4.4 the usage scenario is described for this paper.

4.4.1 Intent Generation

The structuring of intent for this system was influenced by two factors. The maintaining of readability for humans and machine while not limiting the level of descriptive detail possible in the intent. From a generic policy perspective, allowing for a high level of detail is important as information stored within the event can be used to provide context to the situation requiring a decision. In this case our policy is less concerned with highly detailed events given the concept of intent based networking, however the intent must provide a condition to be met with the detail necessary to identify the violation of the condition. Our approach builds intent as a recursive structure with the keywords Who, What, When, Where and How. *Who*: Can describe the agent responsible for the intent creation or the module or component to be affected. *What*: Can describe the job to be undertaken or the goal to be achieved. *When*: Can describe a time frame for the intent to be enforced as a once off or as a reoccurring goal. *Where*: Can describe the components to be affected or a condition / situation where the intent is to be implemented. *How*: Can describe conditions on to which the goal is achieved or can define done in regard to the intent goal.

```
{
    "Who": "NetworkAdministrator",
    "What": "KPIEnforcement",
    "When": [{
            "What": "Start Time",
            "When": "08:00"
        },
        {
            "What": "End Time",
            "When": "12:00"
        }
    ],
    "Where": {
        "What": "Bandwidth",
        "How": "<1MBs"
    },
    "How": "null"
}
```

FIGURE 4.22: Usage Scenario Intent

The *Usage Scenario Intent* (Figure 4.22) describes a KPI enforcement goal with a time frame condition. The intent describes who generated it along with a general description of what the intent is. The goals and conditions are then described lower in the intent structure.

4.4.2 Intent Comparison

On receiving an intent the policy engine triggers an intent validation cycle. This cycle is used to ensure that newly received intents are in accordance with already active intents in the policy engine. The first step is the translation of the new intent event into a common internal structure. The use of an internal common structure allows the policy to map the intent to a supported framework without introducing additional dependencies to external systems. As a result formatting issues are avoided and adjusting to new industry formats can be done through a small policy update. A recursive function is used to navigate the intent parsing the data into a collection of path like statements similar to a folder directory. This collection of statements produces a tree structure which, through the keywords, can be navigated to identify the depth of the tree and the associations between values.

```
/who/NetworkAdministrator
/what/KPIEnforcement
/when/what/startTime
/when/when/08:00
/when/what/endTime
/when/when/12:00
/where/what/Bandwidth
/where/how/<1MBs
/how/null
```

FIGURE 4.23: Intent Path Collection

The *Intent Path Collection* (Figure 4.23) shows the parsed intent for our usage scenario shown in Figure 4.22. Navigating the intent path collection is straightforward allowing new intents and current network state information to be evaluated quickly. Building intent as a recursive structure provides a number of important features for the system. When parsing the intent, the structure can be broken down into a collection of paths which are easily navigated and stored by the APEX system. This allows for the direct comparison of different intents within the policy framework.

4.4.3 Intent Resolution

Direct comparison of independent intent path collections can only identify commonality between intents based on the overlapping of keywords. The system can identify that two independent intents are referring to a similar condition, however the system has no context to what the condition is or how it can be impacted. Our solution was to implement a dictionary designed with our policies role in mind. With a dictionary the system can map the values of the intent to a predefined structure. This structure provide attributes to recognized values, allowing them to be compared on a common level given they share comparable features An ontological approach introduces a dependency into the system, the modeling of intent values, however the standardization of these values internally provides the meta data required to enable informed comparisons. Without informed comparisons the policy cannot resolve independent intents impacting common network attributes.

4.4.4 Usage Scenario

In this section we present the system architecture for our experiment and demonstrate the impact of our intent driven approach on an emulated network

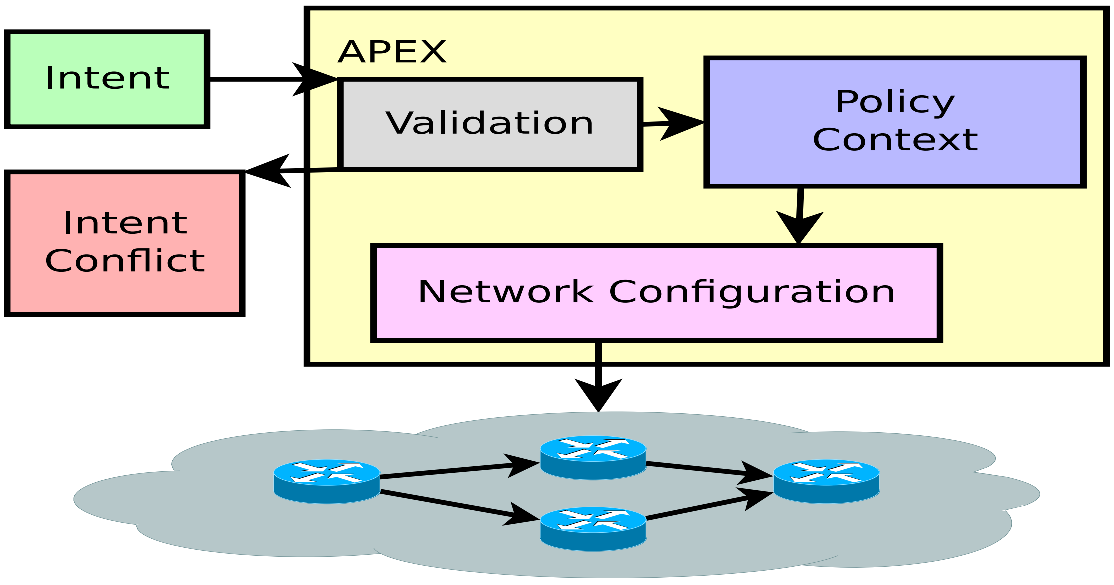


FIGURE 4.24: Architecture

The architecture shown in Figure 4.24 describes the relationships between our components. Intents are generated and sent to the APEX engine. The intent received by the APEX system is shown in Figure 4.22. These intents are processed and compared, as a result a new Mininet configuration is generated. After the generation of the Mininet configuration a new intent is introduced to the system, triggering a new validation attempt. During this validation attempt new intents are processed into the path collection format shown in Figure 4.23. Using the dictionary, value identifiers are mapped to the corresponding structures. Newly mapped values are then compared, generating values that share validity across stored intents. These values are used in the generation of a network configuration file.

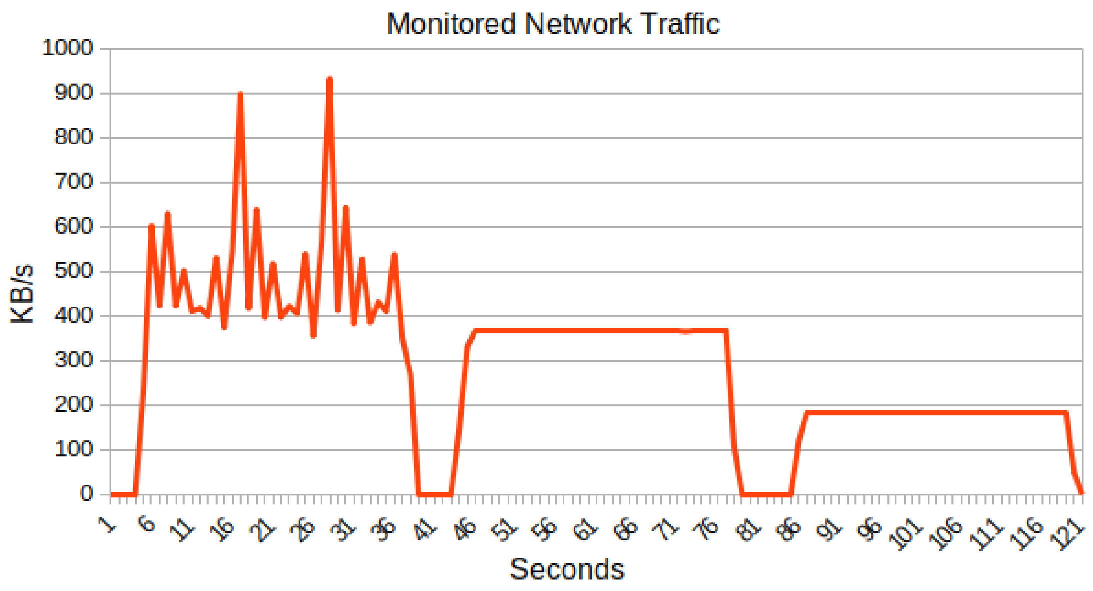


FIGURE 4.25: Network Traffic Data

Figure 4.25 presents the impact of these intent driven network configurations. The monitoring data shows the start up of the default network which is unrestricted in regards to bandwidth. This results in full utilisation of the available resources, hence the fluctuation in performance. The impact of the intent driven approach is then seen with a reconfiguration of the network resulting in an threshold placed on user speed. The system rejects conflicting intents only allowing for reconfiguration for new intents compliant with the existing established intent collection.

This work presented the use of recursive model structures. Highlighting their machine readability and LoD when utilized for intent definition. This work also validated the use of a dictionary for intent context mapping. The next step in our work aimed at addressing intent mapping in adaptive policy. Regardless of the abstraction technique, intent mapping was required at some point in the execution chain to get from an abstract entity to an executable action. Our approach required extensive semantic modeling to expand functionality trigger-able through intent. This lead to an investigation into an inductive approach to intent realisation.

4.5 A Flexible Interpreter For Intent Realisation

This work presents our intent interpretation approach. In subsection 4.5.1 papers published around the topic of intent are discussed, with a focus on translation/mapping and comparison between model and mathematical based approaches. In subsection 4.5.2 our approach to flexible intent interpretation in introduced. In subsection 4.5.3 an in depth description of the implementation is provided accompanied by figures and example input. In subsection 4.5.4 the preliminary results are presented, the execution times for the interpreter are inspected and notable interpreter decisions are discussed.

4.5.1 Representations and Semantics

In this section we describe two high level approaches to intent representation from both a mathematical and model driven perspective. This is followed by an explanation of each approach and how it is adopted for the purposes of Intent representation.

4.5.1.1 Mathematical

A mathematical approach to intent representation can be realised using graph theory as seen in [231] [232]. The authors of [232] propose a graph based abstraction model for user defined intent, which undergoes a compilation process to authorize, check feasibility and map to appropriate controller executions. Intent is described as a directed graph comprised of a collection of network policies between endpoints. The underlying network is abstracted allowing the controller to map these policies how it sees fit at the time of execution.

4.5.1.2 Model-based

Due to the abstract nature of intent based goals and objectives, modeling is a popular approach to Intent representation. Models provide an agreed upon structure for Intent descriptions while providing templates and dictionaries to map meaning to elements of the Intent. The authors of [150] developed an extensive model and meta-model collection allowing for Intents which describe network functions and conditions at a high level to be translated and parsed to lower level actions. This model driven approach is effective as functionality is triggered through modeled interactions, abstracting the complexity by allowing internal component decision makers to actualize based on high level parameters. The authors of [161] introduce an intent definition language called Nile. This language acts as an intermediate between a natural language processed intent and the relevant actioning systems. The Nile language provides a wrapper for identified entities from the initial intent description and output is returned to the network operator for verification. This approach solves two problems. By applying the intent entities to a known template, this allows for the information to be easily navigated. Also by applying the dictionary characteristics of the wrapper, only identifiable entities will be outputted back to the network operator. This compliments the bottom up approach of what the system can do versus what the intent wants. The authors of [233] and [132] extended the Nile language to identify key entities related to their use case, which in turn trigger associated executions to achieve the goals described. The authors of [155] generated Intent descriptor rules, from an analysis on a history of user requirements recorded by cloud consultants. Through this analysis they identified a number of common key categories and then provided models for basic expressions and rules. This generation of expressions based on user requirements reduces the required modeling to produce actionable Intent statements.

Mathematical and model based intent representations provide structuring to intent descriptions through different methods. However, the ability to derive meaning from these statements still require mapping to match identified intent properties to appropriate actions. This mapping is usually achieved in two ways. Through a purpose built mapper before triggering the actioning system, i.e., the system executing the action. Or by the actioning system internally, provided that it supports an intent driven interface that recognizes the incoming intent. Mapping must occur at one or more levels in the execution chain, but as the intent translates towards execution the context around intent properties may change how the intent is to be realized. In this scenario extensive mapping is required to maintain meaning behind the initial intent and the executions required for realization. In our initial work we have an approach more akin to an interpreter with a core focus on flexible interactions which we have broken into three stages.

4.5.2 Flexible Interpretation

In this section we describe our initial work on a flexible interpreter for intent driven systems. The interpreter aims for the flexible handling of intent expressions. The interpreter is broken down into 3 core functions. The first function is the discover-ability of action system capabilities. When an actioning system is added to the network it must provide a functionality template. This functionality template serves the same purpose as API documentation which is to identify the syntax and parameters of executable phrases and the context in which they are useful. The second function is the matching of intent with the descriptions of functions present in the functionality templates available to the Interpreter. The third function is the translation of an intent. This function takes the processed intent message and uses it to inform the building of the appropriate action in response to the original intent message. The action is then performed fulfilling the intent.

4.5.2.1 Functionality Templates

A functionality template describes the capabilities/functions of an actioning system within the system. This is requested by the interpreter when an actioning system is connected. The functionality template contains information similar to what would be included in REST API documentation. In this paper we have prepared a Functionality Template for the OpenFaaS framework [234][235]. Using the OpenFaaS API documented with Swagger[7] as a resource we developed a functionality template to describe the deploy and invoke functions of OpenFaaS. Swagger documentation identifies the required parameters of requests, this is an important attribute for the intent driven building of the API call. The functionality template is expressed in JSON format and is sent to the interpreter at the beginning of the scenario. The interpreter stores this information and it is called when a new intent is received by the interpreter.

4.5.2.2 Intent Matching

One big challenge in intent-based networking is translating the user generated intent (natural language) into a concrete executable network function. The approach used

[7]https://swagger.io/

here is to employ natural language processing (NLP) to match the given intent with available network functions that are registered in the functionality template. The NLP model determines the semantic similarity between the user input and the functional description of each network function. This similarity is evaluated as a score from 0 - 1. The functional description can come from the documentation that describes the various available network functions (e.g. through an API description like Swagger) or have been provided manually during the registration of the function.

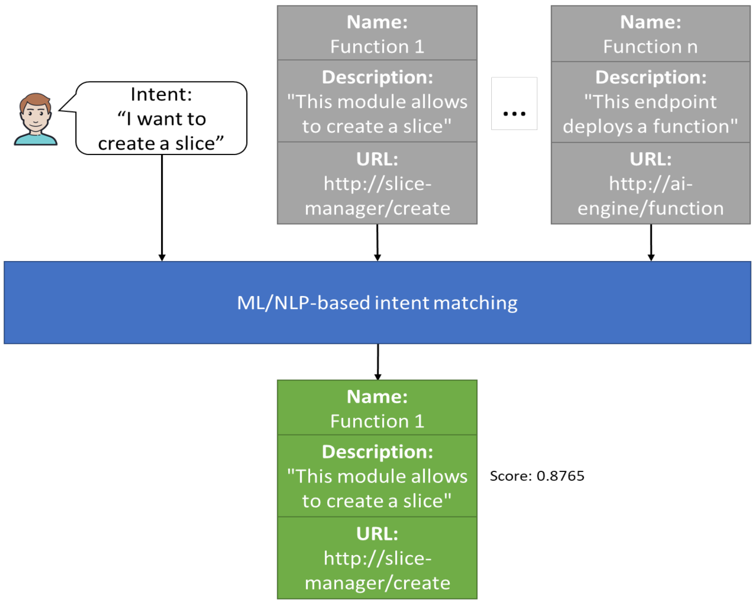

Intent:
“I want to create a slice”
Name:
Function 1
Description:
"This module allows to create a slice"
URL:
http://slice-manager/create
...
Name:
Function n
Description:
"This endpoint deploys a function"
URL:
http://ai-engine/function
ML/NLP-based intent matching
Name:
Function 1
Description:
"This module allows to create a slice"
URL:
http://slice-manager/create
Score: 0.8765

FIGURE 4.26: Intent Matching

Figure 4.26 exemplifies the mapping between user-generated intent and the available network functions that are known to the interpreter. In this example, the user wants to create a new network slice. The Interpreter has a number of functionality templates stored, along with their description and service location (URL). The task of the intent matching module is to find the closest match of function to the user’s intent. In this example, the intent “I want to create a slice” is matched to the function/service that is located at ”http://slice-manager/create”.

4.5.2.3 Translation

Translation only occurs when the intent matching identifies a meaningful request in the intent description and a functionality template exists to realize the request. Translation first looks at the intent description and compartmentalizes the request. The parameters of the intent are extracted and used to populate the associated fields of the identified function. A one to one mapping can occur in this scenario, matching the parameters described in the intent with the parameters described in the functionality template.

For our scenario we are informing our intent Interpreter with the capacity to deploy and invoke functions of the OpenFaaS framework. This framework is used to model common Slice Manager functions which will be driven by intents received by the Interpreter. The scenario can be broken into 3 stages:

- Informing - The process of injecting a Functionality Template into the system.
- Matching - The process of collecting Functionality Template descriptions and providing them to the intent matching component.
- Realisation - The process of dynamically building the identified function and delivery to the appropriate actioning systems.

4.5.3 Implementation

In this section, we present an overview of our implementation and describe the algorithms we have developed.

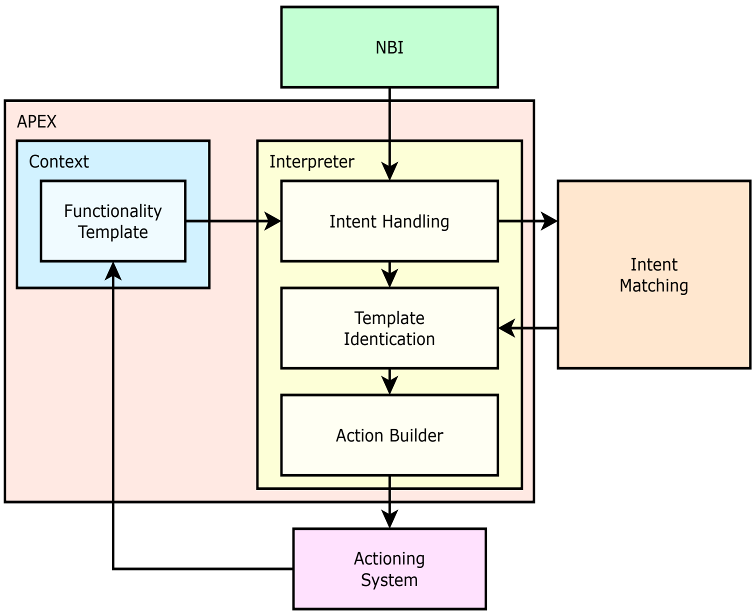
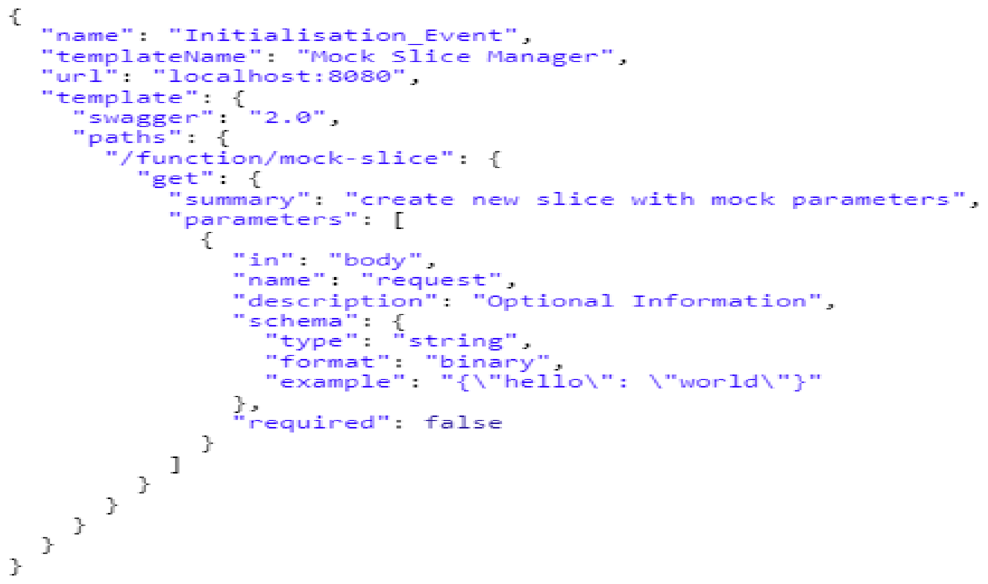


FIGURE 4.27: System Architecture

The system architecture presented in Figure 4.27 provides a high level representation of the system implemented in this paper. A North Bound Interface is used to inject an intent message into our system. This intent message is received by Adaptive Policy EXecution (APEX)[15][227]. APEX is an adaptive policy engine built like a state machine executor which allows for flexible and adaptive executions of policies. The functionality template is stored on the APEX engine through the Context feature. This allows information to be available to policies at the time of execution. The Interpreter receives an intent message from a North Bound Interface containing the request “I want to deploy a network function”. At this point the interpreter checks the content of the request against all functionality templates stored in Context.



```
{
  "name": "Initialisation_Event",
  "templateName": "Mock Slice Manager",
  "url": "localhost:8080",
  "template": {
    "swagger": "2.0",
    "paths": {
      "/function/mock-slice": {
        "get": {
          "summary": "create new slice with mock parameters",
          "parameters": [
            {
              "in": "body",
              "name": "request",
              "description": "Optional Information",
              "schema": {
                "type": "string",
                "format": "binary",
                "example": "{\"hello\": \"world\"}"
              },
              "required": false
            }
          ]
        }
      }
    }
  }
}
```

FIGURE 4.28: Functionality Template Event Message

The event message presented in Figure 4.28 shows a packaged functionality template that contains one function to generate a mock network slice using default parameters. The functionality template is passed into the interpreter while it is running. This allows for more actioning systems to be added dynamically, provided they produce the required information for communication and detail executable commands which can be triggered externally.

Once the request of the intent message has been extracted the interpreter triggers the template collection state. A recursive search is performed on each functionality template stored within the system, extracting each field with the keyword ”summary”. The summary keyword contains a description of what an operation does. This is a fixed field in the Swagger specification. Each description is packaged into a large collection which is sent, along with the original intent request, to the intent matching component. Intent matching is performed using pre-trained language models to compute semantic text similarity. These models were trained on Wikipedia using fastText[8]. These vectors in dimension 300 were obtained using the skip-gram model described in [236] with default parameters.

On reception of intent matching output, the interpreter is provided with the scored descriptions of the most similar functions to the original intent request. At this point the interpreter identifies the functionality template of the highest scoring description and checks the functionality template type. This identifies the appropriate process for navigating and building the function based on the model used for the functionality template. As Swagger API documentation in a JSON format was used to build the OpenFaas functionality template this triggers Swagger specific logic inside the interpreter to build the Swagger defined functions. Parameters provided in the initial intent request are now compared to the parameters of the identified function. If a required parameter is not provided in the intent this would be flagged to the intent issuer. When the mapping of parameters to the function is complete, the location of the actioning system is processed and the function is executed.

[8]https://fasttext.cc/docs/en/pretrained-vectors.html

Algorithm 12 Interpreter

```
 1: procedure INTENT HANDLING                                          ▷ State 1
 2:     intentMessage ← incomingEvent
 3:     context ← intentMessage
 4:     intentRequest ← parse(intentMessage)
 5:     for template in FunctionalityTemplates[] do
 6:         for description in recursiveSearch(template) do
 7:             descriptionCollection[] ← description
 8:         end for
 9:     end for
10:     stateOutput ← intentRequest
11:     interpreterOutput ← descriptionCollection[]
12: end procedure
13: procedure TEMPLATE IDENTIFICATION                                  ▷ State 2
14:     intentRequest ← stateInput
15:     scoredDescriptions[] ← matchingInput
16:     highestDescription ← scoredDescriptions[]
17:     for template in FunctionalityTemplates[] do
18:         if highestDescription in recurciveSearch(template) then
19:             identifiedTemplate ← template
20:             identifiedFunction ← template.function
21:         end if
22:     end for
23:     stateOutput ← identifiedTemplate
24:     stateOutput ← identifiedFunction
25: end procedure
26: procedure ACTION BUILDER                                           ▷ State 3
27:     identifiedTemplate ← stateInput
28:     identifiedFunction ← stateInput
29:     intentMessage ← context
30:     intentParameters ← intentMessage.parameters
31:     if identifiedTemplate.format == swagger then
32:         Trigger swagger navigation logic
33:         functionParameters ← identifiedFunction
34:         populatedFunction ← (functionParameters, intentParameters)
35:     end if
36:     interpreterOutput ← populatedFunction
37: end procedure
```

Algorithm 12 provides a formalized description of the interpreter in a three state execution.

4.5.4 Preliminary Results

This section presents the timing for 120 independent intent executions through the interpreter. At the beginning of the scenario the interpreter is started and is provided with 3 unique functionality templates. The first is a dummy template to provide a variety of function descriptions that will be checked during the intent matching stage. The second is the functionality template for the OpenFaaS Framework. The third is an expanded version of the functionality template shown in Figure 4.28 populated with additional operations. The Slice Manager mock-up is an OpenFaaS function that exposes API endpoints similar to what would be expected from a generic slice manager component.

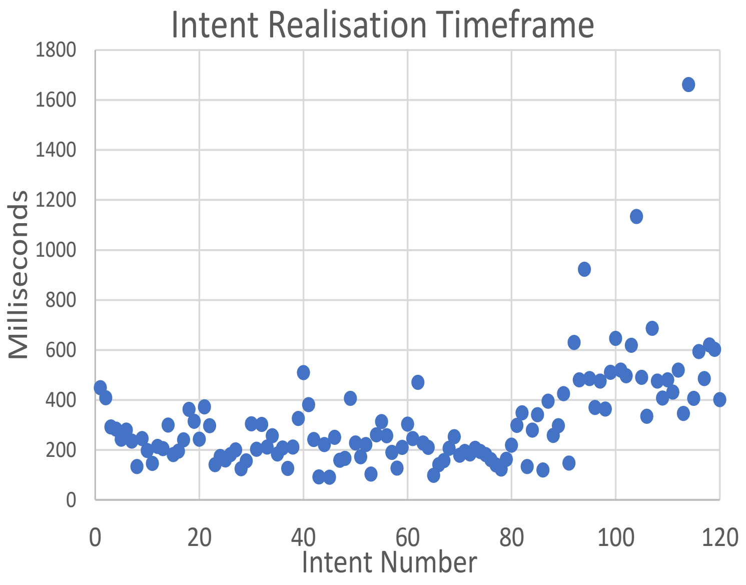


FIGURE 4.29: Intent Realisation Timeframe

The timings shown in Figure 4.29 are the duration from the moment the intent is received by the interpreter to the time the action is outputted by the system. The initial interpreter executions record timings around the 400ms mark. This is primarily due to the absence of cached data on the system. In the following executions, the timings decrease to around the 200ms mark and occasionally going as low as 100ms. The system reaches its shortest consistent executions between the 70th and 80th mark. After which the timings become more volatile averaging around the 500ms mark with a number of large outliers.

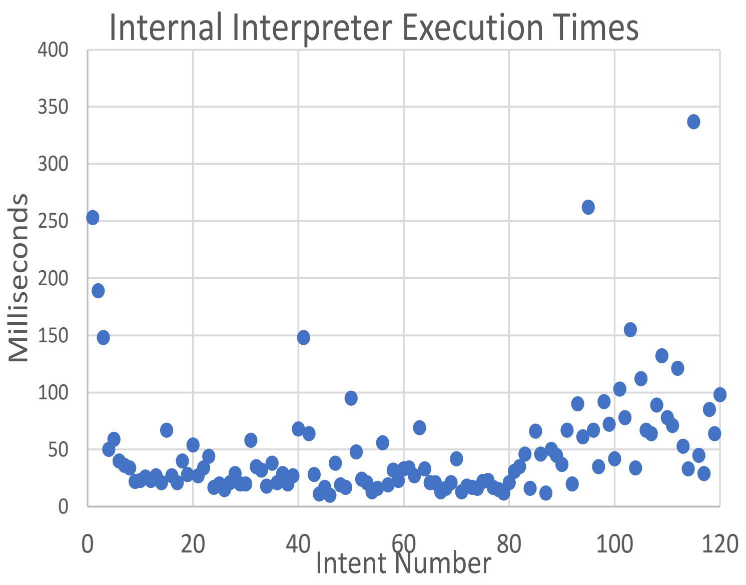


Figure 4.30: Internal Interpreter Execution Times

To get a better insight into what is causing this increase in execution time, Figure 4.30 shows the execution time of logic inside the Interpreter. Similar to the overall timing, executions start high and as more processes are cached by the Interpreter this number decreases as low as 25ms. In the latter part of the scenario the inconsistency is mirrored by the overall timings however many of these executions are below 100ms. This identifies the REST interfaces used in the system as a significant contributor to the overall execution time. In some cases the REST interfaces contribute over 400ms to the overall execution time of the interpreter. As latency is a key performance indicator in many network management scenarios the results show that there is room for improvement in respect to how the Interpreter engages with its REST interface. In future work, overall latency can be improved through integration of the intent matching component with the interpreter. This would decrease the reliance on the REST interface to communicate large volumes of data.

The consistency of the fastText pre-trained model provided the interpreter with a robust mechanism for matching intent requests with concrete executable actions. During the scenario a variety of intent requests were sent to the interpreter. Issues primarily arose due to descriptions of functions in the functionality templates. Descriptions being too short, not providing enough descriptive material for the intent matching to work with, and the absence of defined required data were the main causes of failed intent executions. Different functions that perform similar or augmented versions of the same tasks were identified as potential problem areas if not described adequately.

This work presented an embedded NLP based component for fast evaluation of intent requests. This work also validated the functionality template (an extension based on dictionary from previous work) for the inductive generation of intent-based actions. The next step in our work aimed to validate and demonstrate our approach in real network environments.

4.6 NLP Powered Intent Based Network Management for Private 5G Networks

5G-CLARITY is a novel architecture for 5G private networks integrating 5GNR, Wi-Fi and Li-Fi access networks. The 5G-CLARITY architecture is composed of three strata, namely an infrastructure stratum, a virtualised network and application function stratum and a management and orchestration stratum, and provides support for multi-connectivity, positioning and slice provisioning. Thus, deploying end-to-end network services in 5G-CLARITY requires the provisioning and configuration of both virtual and physical network functions (V/PNFs), imposing a steep learning curve for private network operators that are not familiar with 5G technologies.

4.6.1 Intent Engine

The Intent Engine is built upon Adaptive Policy EXecution (APEX), a state machine approach to policy execution that incorporates a context functionality into the decision making mechanism. APEX was initially published in [15] and was later adopted as a Policy Decision Point (PDP) in the Open Network Automation Platform[9]. The work published in [237] provides the core NLP-based interpretation mechanism for the Intent Engine.

[9]https://docs.onap.org/projects/onap-policy-parent/en/latest/apex/APEX-Introduction.html

The mechanism was then adapted to expand the action and interface generation features to incorporate the AI Engine, ML models and orchestration type components. This allowed the Intent Engine to communicate with slice provisioning systems, active ML models running in the AI Engine and production-quality Network Service Orchestrators (NSO). Next, we describe the key design principles of our Intent Engine, namely:

- *Intent Design*, describing how intents are built,
- *Intent Matching Engine*, describing how intents are matched to specific API end-points,
- *South-bound provider integration*, describing how different management functions can be integrated with the Intent Engine.

4.6.1.1 Intent design

An Intent message has two fields: *request* and *parameters*. The request field contains an intent in the form of an English sentence. This is used in the matching process to compare the users request with operations available to the Intent Engine. The parameters field contains information the user would like to inject into the action building process. This is required information that is not provided through component defaults or through a supplementary ML model in the AI Engine.

```
{
  "intent": {
   "request": "Create a slice",
   "parameters": {
    "name": "nova",
    "user-list": [
     {
      "imsi": "001035432100005"
     }
    ],
    "location": {
     "latitude": 0.0,
     "longitude": 0.0
    },
    "technology":[
     "AMARISOFT_CELL",
     "SUB6_ACCESS"
    ]
   }
  }
}
```

Listing 4.1: Slice provisioning intent example.

The request and parameters fields can be seen in Listing 4.1 in the form of an example, namely a Slice Provisioning Intent. In the example, the request details the creation of a slice through an ML model in the AI Engine. As this is a complex process containing several sequenced executions the ML model populates many of the sequenced operations, however some parameters cannot be queried or generated. These parameters are detailed in the example such as *user-list*, *location* and *technology*.

4.6.1.2 Intent Matching Engine

The Intent Engine adopts an NLP-based matching process to correlate intents with executable operations available in the moment. This is achieved through a text distancing algorithm trained on Wikipedia data, which scores the intent request against a range of operation descriptions. The model used in this scenario is pretained word vector model provided by FastText. The model is trained on 2017 Wikipedia dataset, UMBC webbase corpus and statmt.org news dataset (16B tokens)[238]. The operation descriptions are stored in a model referred to as the Functionality Template. The user provides the

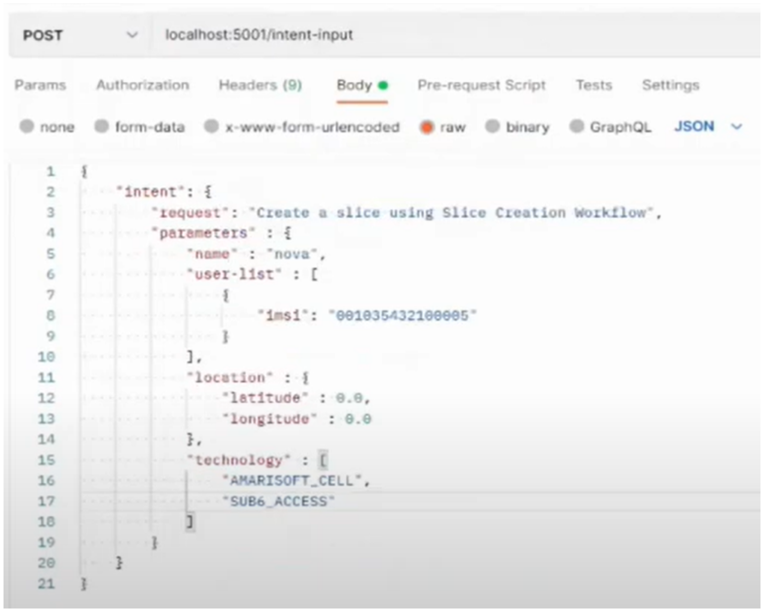
Functionality Template to the Intent Engine through a registration process, informing the Intent Engine how and where to communicate with the component. This template follows OpenAPI Specification allowing it to be processed in the intent matching and action generation stage of the execution. Information from this model also informs the dynamic building of the interface to allow communication between the Intent Engine and the target component as a result of the processed intent.

4.6.1.3 South-bound provider integration

Informed through the Functionality Template, the Intent Engine dynamically builds the interfaces for the south-bound communications. Through the use cases we show the Intent Engine communicating several requests to three components operating in two independent strata. These interfaces are built at execution time to ensure the most up to date information is used during the action building and interface building stage of execution. As a result, changes or re-configurations of components do not require a hard restart of the Intent Engine. Instead, the reissuing of an updated Functionality Template would restore the communication.

4.6.2 Demonstration

This section will describe the demonstration of three use cases in industrial settings. Each demonstration was benchmarked based on intent execution time. This records the amount of time from the communication of the first intent message to the point were the intent is enforced in the network.

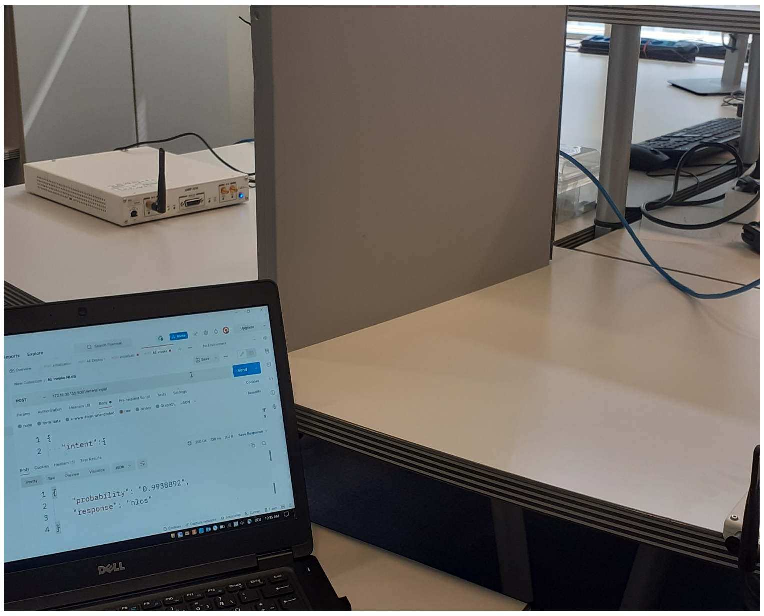


localhost:5001/intent-input

```json
{
    "intent": {
        "request": "Create a slice using Slice Creation Workflow",
        "parameters" : {
            "name" : "nova",
            "user-list" : [
                {
                    "imsi": "001035432100005"
                }
            ],
            "location" : {
                "latitude" : 0.0,
                "longitude" : 0.0
            },
            "technology" : [
                "AMARISOFT_CELL",
                "SUB6_ACCESS"
            ]
        }
    }
}
```

Figure 4.31: Intent based slice provisioning through POSTMAN based interface

Figure 4.31 depicts a snapshot from our demonstration where an intent request is readied to be sent using the Postman API platform. To benchmark the execution time the experiment was repeated 10 times averaging in an execution time of approximately 3 minutes (presented in Table 4.2). This time is considered to be reasonable given the configuration time of 5G cells and access points as well as spanning new core network related VNFs.


New Collection / AE Invoke NLoS
POST
172.16.30.155:5001/intent-input
Send
200 OK 738 ms 202 B

```
{
    "intent":{
```

```
{
    "probability": "0.9938892",
    "response": "nlos"
}
```

FIGURE 4.32: Intent based NLoS identification, with object blocking line of sight

Figure 4.32 depicts a snapshot of the demonstrator where a board is blocking line of sight between the UE and the gNB. Later in the demonstration the board is removed and the experiement is repeated to reflect and change in the line of sight prediction. The time recorded for this experiment was averaged to approximately 1 second (presented in Table 4.2). Given that the model is pretrained this time is primarily due to the communication of CIR data to the model and the return of the prediction result to the user.

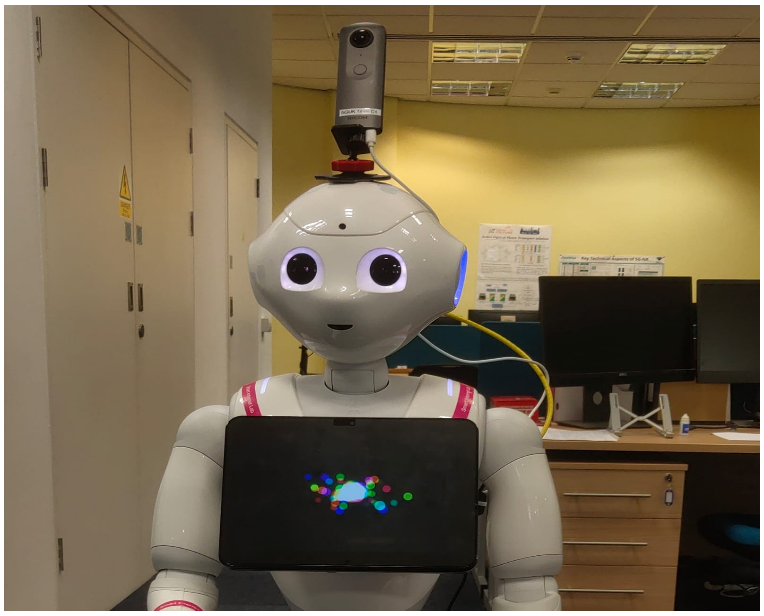


Figure 4.33: Intent based network service deployment, depicting the robot mounted 360° camera

Figure 4.33 depicts a snapshot of the robot mounted 360 degree camera. The demonstration enacted a smart tourism service in which surveillance video from the robot was requested through a mobile device using intents generated by a public safety officer. Using intent, the user provisioned a virtual media forwarding unit at the edge in a smart museum environment. The feed from the robot camera is then forwarded to the user device. The average time to instantiate the video service through intent is 234 second or approximately 4 minutes (presented in Table 4.2). This time is accounted for through the layers of interaction between the OSM, Intent Interpreter and video resolution.

Table 4.2: Average intent execution time per use case.

	Slice provisioning	NLoS identification	NS deployment
Average intent execution time	$\approx$ 3 minutes	$\approx$ 1 second	$\approx$ 4 minutes

Table 4.2 presents the average intent execution times for each of the use cases. These times are acceptable given the complexity of the operations being handled in by the system. All recordings created as part of this project are hosted on Youtube[10] along with the three demonstrations described in this work.

[10]https://www.youtube.com/@5g-clarity458

This work validates our approach through 3 varied use cases in real industrial private networks. Demonstrating Adaptive Intent Realization (AIR) concepts in Management, Orchestration and near RT RIC applications.

Chapter 5

Conclusions and Future Work

This chapter consists of two sections: section 5.1 discusses the conclusions of the work to date and section 5.2 describes future work and the next steps in the research.

5.1 Conclusions

The scenarios and results presented in section 4.1 demonstrate how effective the test bed is at managing a virtual network through adaptive policy execution. The test bed gives policy writers a testing environment where they can push the boundaries of their adaptive policies by applying them to emulated networks. The network configuration can be extended and made more complex by amending and building up the configuration of the virtualised network in Mininet.

A key feature of the test bed was the inclusion of an virtual network emulator rather than a simulator. While both have their advantages, tests carried out on an emulated network provide a level of authenticity to the results of the test. The tester can trust in the integrity of the results as they replicate the hardware and software, and pass real data around rather than simulating it.

The inclusion of an industrial grade SDN controller adds to the legitimacy of the test bed by providing more flexible and dynamic routing capabilities to the emulated network. The Floodlight SDN controller is straightforward and very well documented. However, drawbacks have been identified with Floodlight. In particular, the REST API lacks expressivity and support for performing many fundamental operations. Since the initial selection of Floodlight there has been little in terms of contributions from the Floodlight community to the development of the controller. Without an active community behind it, the controller will quickly stagnate. Other controllers, such as the OpenDaylight controller [239], have an strong community behind them. OpenDaylight in particular supports most of the features Floodlight provided and has additional features that would be useful for the test bed.

Designing and implementing policies with the APEX engine was relatively straightforward and the scenarios used in the early stages of this work did not push the limits of the test bed nor the potential of the APEX engine. The state machine approach to the policy execution resulted in policies which are easy to implement while also allowing for ongoing iterative authoring/validation cycles to handle more complex scenarios.

The importance of current OTT and network-centric video optimization strategies is discussed in section 4.2 while motivating the need for a closed control, policy-driven approach for 5G operators to mitigate between OTT, other service offerings, and the available video optimization techniques. This work describes a starting point for the role of adaptive policy, in service assurance, for video quality control through the evaluation of network resources using MOS.

Policy controlled closed control loop mechanisms that can steer video optimization frameworks are important when considering new technologies such as MEC and VNFs. The network resource evaluation described in section 4.2, although preliminary, is one of many ways adaptive policy can be incorporated into specific optimization strategies and with the inclusion of APEX in the Open Networking Automation Platform (ONAP), adaptive policy may play a bigger role in 5G networks. All components, scripts, and other artifacts for the experiment are available online [222]. The provided instructions allow any interested party to run the experiment.

The paper described in section 4.3 proposes a directed feed forward neural network for network path selection for multimedia streaming applications within the APEX Adaptive Policy Execution Engine, which outputs the optimum network configuration for a video, based on a context established through a weighted score. The selected configuration is sent to Mininet, which builds the appropriate network for the cycle, over which video is streamed and evaluated. The returning event triggers a video quality evaluation policy.

The degree of change in the video evaluation metrics is used to adjust weights for the next cycle. The results generated from initial tests produced some notable insights. A clear trend is apparent in the adjustments of the weights stored in policy context. It is also clear that the policy would benefit from an increased number of cycles in a testing scenario, to allow the changes in weights to plateau. The directed feed forward neural network policy would also benefit from a larger array of network configurations during the testing phase which would increase the learning rate of the policy.

The paper described in section 4.4 introduced a straightforward intent approach where the network traffic displayed in Figure 4.25 showed the realization of three intents. The first intent is described in Figure 4.22, the second intent is the same as the first except reducing bandwidth to 366KB/s and the third reduces bandwidth further to 184KB/s. In the event that a newly received intent directly conflicts with an existing intent a notification event is generated containing information on the owners of the conflicting intent and the values responsible for the conflict.

The paper described in section 4.5 presented the interpreter and the use of the functionality template. Often intent languages package the semantics of statements and functionality together. This is a successful approach in a closed environment. Modern networks exhibit many dynamic traits and with the ability to spin up a variety of virtual functions, it is important for the system to know what capabilities it has at its disposal. By decoupling the semantic and the function and implementing through the functionality template, new virtual functions can express their functionality to the interpreter and become intent driven. Many network components already expose the information required to produce a functionality template, for example Swagger UI is a popular API documentation tool which exposes API resources and is automatically generated from an OpenAPI Specification. In summary we have presented our initial work in the development of a flexible interpreter for intent realization. Its capabilities have been demonstrated through intent driven executions utilising a simulated network management components.

Our approach in section 4.5 requires that the user provides functionality templates for actioning systems they wish to incorporate into the intent driven system. As functionality templates can be accepted during run-time this allows for the new actioning systems to be dynamically incorporated which is a useful feature with the increasing role virtualisation plays in modern networks. This work aims at shifting focus from the rigid mapping of component specific intents to predefined executions. Instead opting for dynamic interpretation of intent messages through the lens of functionality available to the system. Moving away from direct mapping introduces a level of variability in the approach. This requires additional safeguards to be put in place to protect against instances of mismatching, mapping intent requests to undesired functionality.

The work described in section 4.6 highlights a strong interest in designing systems that can simplify the interactions between humans and complex digital systems through the use of natural language based interfaces. Intent based networking has been devised as the architectural framework to achieve this simplification in the management of mobile networks, where advances in this regard are especially needed in the area of private 5G networks owned by verticals that often lack skilled 5G personnel. The Intent Engine allowed users to input intents in the form of English level sentences, and in coordination of components of the Intelligence Stratum provide low-level context missing in the original intent definition. This intent driven management was demonstrated in three private 5G network use cases, namely a slice provisioning use case, an indoor positioning use case, and a network service deployment use case. All use cases were benchmarked in terms of intent provisioning time.

5.2 Future Work

The core objective of the future work is to apply the lessons learned in the development of the test bed, policies interaction with optimisation techniques and the implementation of straightforward learning algorithms in the policy to create a closed control loop mechanism within ONAP which uses machine learning techniques in an adaptive policy environment for dynamic context-aware network optimisation and repair.

The development of the test bed validated the adoption of the COMPA automation pattern motivating the further development of the test bed by integrating more advanced analytics and adding additional (application, network, radio) controllers to extend the scope of the testbed to much more complex scenarios. A DevOps inspired iterative authoring/validation approach to verify policy specifications and analytics tasks is a fundamental requirement for supporting autonomic management. Incorporating elements for network simulation into the test bed can also compliment and augment the approach presented. Simulation can play a vital role in validating the properties of management control loops at authoring/pre-deployment time, but with the caveat that properly configured emulators such as Mininet provide a more accurate result. As a result of the test bed we experienced more users engaging with adaptive policy, this is an important note in regard to Adaptive Intent Realisation (AIR) as a reliable and applicable testing environment is an important feature of new technology.

The work incorporating policy and video quality assurance produced a number important lessons for future work, especially in regard to policy placement. In previous work policy was visualised as low in the network, working closely with the SDN controller. While this in itself is not an issue, it limited its application as without a layer of abstraction between the decision making and the actioning of the decision, policy becomes a static, expressing specialised application specific behaviours. This motivated an alternative approach. Utilising the compartmentalisation of the COMPA pattern along with the separation of concerns design principle, would reduce the responsibilities of policy by focusing on decision making process while Analytics and components associated with Orchestration and Management act as buffers either side of policy. This understanding is important as with the flexibility of Adaptive Intent Realisation (AIR) it may be necessary to assign generalised roles to different instances of AIR in the same domain.

As part of our future work, we plan to continue investigating how recent advances in NLP, e.g. based on the use of Large Language Models (LLMs) [240], can be leveraged to further simplify the operation of private 5G networks. For example, an LLM could learn to map a user intent directly to a complex sequence of API endpoints, thus simplifying the need for the manually handcrafted workflow models that are needed in our current system to support different use cases.

Bibliography

[1] Experiential Networked Intelligence (ENI). Etsi, 2023. URL https://www.etsi.org/technologies/experiential-networked-intelligence?highlight=YToxOntpOjA7czozOiJuZnYiO30=. Online; accessed 25 June 2023.

[2] 3GPP. Telecommunication management; Study on scenarios for Intent driven management services for mobile networks. Technical report (TR) 28.812, 3rd Generation Partnership Project (3GPP), 03 2020. URL https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3553. Version 0.10.0.

[3] Open Network Automation Platform (ONAP). Architecture, 2019. URL https://www.onap.org/architecture. Online; accessed 07 June 2020.

[4] Open Network Automation Platform (ONAP). Optimization service design framework, 2020. URL https://wiki.onap.org/display/DW/Optimization+Service+Design+Framework. Online; accessed 25 June 2023.

[5] TM Forum. Autonomous Networks – Reference Architecture. Introductory Guide IG1251, TM Forum, 07 2022. URL https://www.tmforum.org/resources/how-to-guide/ig1251-autonomous-networks-reference-architecture-v1-0-1/. Version 1.0.1.

[6] Nurit Sprecher. The zsm story: the power to transform, 2018. URL https://www.etsi.org/newsroom/blogs/technologies/entry/the-zsm-story-the-power-to-transform. Online; accessed 25 June 2023.

[7] ETSI. Zerotouch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects. GROUP REPORT (GR) 011, European

Telecommunications Standards Institute, 02 2023. URL `https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/011/01.01.01_60/gr_ZSM011v010101p.pdf`. Version 1.1.1.

[8] M. Behringer, B. Carpenter, T. Eckert, L. Ciavaglia, and J. Nobre. *RFC 8993: A Reference Model for Autonomic Networking.* RFC Editor, USA, 2021.

[9] 5G-CLARITY. Validation of 5G-CLARITY SDN/NFV Platform, Interface Design with 5G Service Platform, and Initial Evaluation of ML Algorithms . Project Deliverable 4.2, Horizon 2020 Research and Innovation, 07 2021. URL `https://www.5gclarity.com/wp-content/uploads/2021/11/5G-CLARITY_D42.pdf`.

[10] HG Hegering, S Abeck, and B Neumair. Integrated management of network systems: Concepts, architectures and their operational application, 1999.

[11] Alexander Clemm. *Network management fundamentals.* Cisco Press, 2006.

[12] S. van der Meer, J. Keeney, and L. Fallon. Taming policy complexity: Model to execution. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–8, April 2018. doi: 10.1109/NOMS.2018.8406172.

[13] S. van der Meer. 5g & autonomic networking - challengesin closing the loop, May 2015. URL `http://www.5gsummit.org/docs/slides/Sven-Meer-5GSummit-Princeton-05262015.pdf`.

[14] S. van der Meer, J. Keeney, and L. Fallon. Dynamically adaptive policies for dynamically adaptive telecommunications networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 182–186, Nov 2015. doi: 10.1109/CNSM.2015.7367357.

[15] L. Fallon, S. van der Meer, and J. Keeney. Apex: An engine for dynamic adaptive policy execution. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 699–702, April 2016. doi: 10.1109/NOMS.2016.7502880.

[16] L. Fallon, J. Keeney, M. McFadden, J. Quilty, and S. van der Meer. Using the compa autonomous architecture for mobile network security. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 747–753, May 2017. doi: 10.23919/INM.2017.7987370.

[17] L. Fallon, J. Keeney, and S. van der Meer. Distributed management information models. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 414–420, May 2017. doi: 10.23919/INM.2017.7987306.

[18] ARCFIRE consortium. H2020 arcfire deliverable d2.2: Converged service provider network design report, December 2016. URL `http://ict-arcfire.eu`.

[19] Göran Rune, Erik Westerberg, Torbjörn Cagenius, Ignacio Mas, Balázs Varga, Henrik Basilier, and Lars Angelin. Architecture Evolution for Automation and Network Programmability. *Ericsson Review*, 11(3):2–10, 2014. URL `http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2014/er-evolved-network-architecture.pdf`.

[20] The Linux Foundation. The open network automation platform (onap), 2017. URL `https://www.onap.org/`.

[21] Ericsson. The Ericsson Mobility Report, June 2017. URL `https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf`.

[22] Ericsson. Mobile traffic analysis by application, November 2017. URL `https://www.ericsson.com/en/mobility-report/reports/november-2017/mobile-traffic-analysis-by-application`.

[23] Ericsson. Enhancing the event experience, November 2017. URL `https://www.ericsson.com/en/mobility-report/reports/november-2017/enhancing-the-event-experience`.

[24] Lin Xiang, Derrick Wing Kwan Ng, Toufiqul Islam, Robert Schober, Vincent W. S. Wong, and Jiaheng Wang. Cross-layer optimization of fast video delivery in cache- and buffer-enabled relaying networks. *IEEE Transactions on Vehicular Technology*, 66(12):11366–11382, December 2017. doi: 10.1109/TVT.2017.2720481. URL `http://ieeexplore.ieee.org/document/7959207/`.

[25] Harish Viswanathan, Danny De Vleeschauwer, Andre Beck, Steven Benno, Raymond B. Miller, Gang Li, Mark M. Clougherty, and David C. Robinson. Mobile video optimization at the base station: Adaptive guaranteed bit rate for http adaptive streaming. *Bell Labs Technical Journal*, 18(2):157–174, September 2013. doi: 10.1002/bltj.2161. URL `http://ieeexplore.ieee.org/document/6772140/`.

[26] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys Tutorials*, 17(1):469–492, Firstquarter 2015. ISSN 1553-877X. doi: 10.1109/COMST.2014.2360940.

[27] N. Bouten, S. Latré, J. Famaey, W. Van Leekwijck, and F. De Turck. In-network quality optimization for adaptive video streaming services. *IEEE Transactions on Multimedia*, 16(8):2281–2293, Dec 2014. ISSN 1520-9210. doi: 10.1109/TMM.2014.2362856.

[28] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. Qoe-driven rate adaptation heuristic for fair adaptive video streaming. *ACM Trans. Multimedia Comput. Commun. Appl.*, 12(2):28:1–28:24, October 2015. ISSN 1551-6857. doi: 10.1145/2818361. URL `http://doi.acm.org/10.1145/2818361`.

[29] ISO/IEC. Information technology – dynamic adaptive streaming over http (dash) – part 5: Server and network assisted dash (sand). Edition 1, February 2017. URL `https://www.iso.org/standard/69079.html`.

[30] TNO. MPEG-DASH SAND, 2017. URL `https://tnomedialab.github.io/sand/`.

[31] Abbas Mehrabi, Matti Siekkinen, and Antti Ylä-Jääski. Joint optimization of qoe and fairness through network assisted adaptive mobile video streaming. In *13th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, October 2017. doi: 10.1109/WiMOB.2017.8115778. URL `http://ieeexplore.ieee.org/document/8115778/`.

[32] D. C. Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26, March 2002. ISSN 0890-8044. doi: 10.1109/65.993219.

[33] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo. Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165, Jan 2018. ISSN 0163-6804. doi: 10.1109/MCOM.2018.1700560.

[34] S. van der Meer, J. Keeney, and L. Fallon. 5g networks must be autonomic! In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, April 2018. doi: 10.1109/NOMS.2018.8406185.

[35] Y. Tsuzaki and Y. Okabe. Reactive configuration updating for intent-based networking. In *2017 International Conference on Information Networking (ICOIN)*, pages 97–102, 2017. doi: 10.1109/ICOIN.2017.7899484.

[36] J. Augé and M. Enguehard. A network protocol for distributed orchestration using intent-based forwarding. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 718–719, 2019.

[37] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020.

[38] E. Zeydan and Y. Turk. Recent advances in intent-based networking: A survey. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5, 2020.

[39] B. E. Ujcich, A. Bates, and W. H. Sanders. Provenance for intent-based networking. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 195–199, 2020. doi: 10.1109/NetSoft48620.2020.9165519.

[40] B. Lewis, L. Fawcett, M. Broadbent, and N. Race. Using p4 to enable scalable intents in software defined networks. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 442–443, 2018.

[41] Y. Han, J. Li, D. Hoang, J. Yoo, and J. W. Hong. An intent-based network virtualization platform for sdn. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 353–358, 2016.

[42] F. Callegati, W. Cerroni, C. Contoli, and F. Foresta. Performance of intent-based virtualized network infrastructure management. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[43] H. Zhang, Y. Wang, X. Qi, W. Xu, T. Peng, and S. Liu. Demo abstract: An intent solver for enabling intent-based sdn. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 968–969, 2017. doi: 10.1109/INFCOMW.2017.8116514.

[44] D. Sanvito, D. Moro, M. Gullì, I. Filippini, A. Capone, and A. Campanella. Onos intent monitor and reroute service: enabling plug play routing logic. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 272–276, 2018. doi: 10.1109/NETSOFT.2018.8460064.

[45] B. E. Ujcich and W. H. Sanders. Data protection intents for software-defined networking. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 271–275, 2019. doi: 10.1109/NETSOFT.2019.8806684.

[46] C. Wu, S. Horiuchi, and K. Tayama. A resource design framework to realize intent-based cloud management. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 37–44, 2019. doi: 10.1109/CloudCom.2019.00018.

[47] J. Kim, E. Kim, J. Yang, J. Jeong, H. Kim, S. Hyun, H. Yang, J. Oh, Y. Kim, S. Hares, and L. Dunbar. Ibcs: Intent-based cloud services for security applications. *IEEE Communications Magazine*, 58(4):45–51, 2020. doi: 10.1109/MCOM.001.1900476.

[48] Chen Li, Olga Havel, Will (Shucheng) LIU, Adriana Olariu, Pedro Martinez-Julia, Jéferson Campos Nobre, and Diego Lopez. Intent Classification. Internet-Draft draft-irtf-nmrg-ibn-intent-classification-03, Internet Engineering Task Force, March 2021. URL `https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-ibn-intent-classification-03`. Work in Progress.

[49] Alexander Clemm, Laurent Ciavaglia, Lisandro Zambenedetti Granville, and Jeff Tantsura. Intent-Based Networking - Concepts and Definitions. Internet-Draft draft-irtf-nmrg-ibn-concepts-definitions-05, Internet Engineering Task Force, September 2021. URL `https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-ibn-concepts-definitions-05`. Work in Progress.

[50] Benoît Claise, Jean Quilbeuf, Diego Lopez, Daniel Voyer, and Thangam Arumugam. Service Assurance for Intent-based Networking Architecture. Internet-Draft draft-ietf-opsawg-service-assurance-architecture-01, Internet Engineering Task Force, July 2021. URL `https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-service-assurance-architecture-01`. Work in Progress.

[51] David C Reeve. *A New Blueprint for Network QoS.* PhD thesis, Computing Laboratory, University of Kent, Canterbury, Kent, UK, August 2003. URL `http://www.cs.kent.ac.uk/pubs/2003/1892`.

[52] T. L. Marzetta. Noncooperative cellular wireless with unlimited numbers of base station antennas. *IEEE Transactions on Wireless Communications*, 9(11):3590–3600, November 2010. ISSN 1558-2248. doi: 10.1109/TWC.2010.092810.091092.

[53] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta. Energy and spectral efficiency of very large multiuser mimo systems. *IEEE Transactions on Communications*, 61 (4):1436–1449, April 2013. ISSN 1558-0857. doi: 10.1109/TCOMM.2013.020413. 110848.

[54] G. Zhang, M. De Leenheer, A. Morea, and B. Mukherjee. A survey on ofdm-based elastic core optical networking. *IEEE Communications Surveys Tutorials*, 15(1): 65–87, First 2013. ISSN 2373-745X. doi: 10.1109/SURV.2012.010912.00123.

[55] B. Li, Q. Qu, Z. Yan, and M. Yang. Survey on ofdma based mac protocols for the next generation wlan. In *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 131–135, March 2015. doi: 10.1109/ WCNCW.2015.7122542.

[56] P. Misra and P. Enge. *Global Positioning System: Signals, Measurements and Performance.* Ganga-Jamuna Press, 2006.

[57] A. Brown and M. Sturza. Vehicle tracking system employing global positioning system (gps) satellites, US Patent 5,225,842, 6 July 1993.

[58] O. J. Woodman. An introduction to inertial navigation. Technical report (tr), University of Cambridge, Computer laboratory, 08 2007.

[59] L. Mitrou A. Mylonas, V. Meletiadis and D. Gritzalis. Smartphone sensor data as digital evidence. In *Computers & Security*, volume 38, pages 51–75, 2013.

[60] Won-Jae Yi, Weidi Jia, and Jafar Saniie. Mobile sensor data collector using android smartphone. In *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 956–959, 2012. doi: 10.1109/MWSCAS.2012.6292180.

[61] Peter Widhalm, Philippe Nitsche, and Norbert Brändie. Transport mode detection with realistic smartphone sensor data. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 573–576, 2012.

[62] 3GPP. Release 16 Description; Summary of Rel-16 Work Items. Technical report (TR) 21.916, 3rd Generation Partnership Project (3GPP), 06 2022. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3493`. Version 16.2.0.

[63] 3GPP. E-UTRA physical channels and modulation. Technical specification (TS) 36.211, 3rd Generation Partnership Project (3GPP), 09 2018. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2425`. Version 15.3.0.

[64] S. Fischer. *Observed time difference of arrival (OTDOA) positioning in 3GPP LTE*. Qualcomm, 2014.

[65] Antonio Manzalini, Cagatay Buyukkoc, Prosper Chemouil, Slawomir Kuklinski, Franco Callegati, Alex Galis, Marie Paule Odini, Chih-Lin I, Noel Crespi, Eileen Healy, and Stuart Sharrock. Towards 5g software-defined ecosystems, 07 2016.

[66] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys Tutorials*, 20(3):2429–2453, thirdquarter 2018. ISSN 2373-745X. doi: 10.1109/COMST.2018.2815638.

[67] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015. ISSN 1558-2256. doi: 10.1109/JPROC.2014.2371999.

[68] Syed Asif Raza Shah, Jin Kim, Sangwook Bae, Amol Jaikar, and Seo-Young Noh. Network softwarization: A study of sdn and nfv integration. In *2016 International Conference on Convergence Technology (ICCT)*, volume 6, June 2016.

[69] Deze Zeng, Lin Gu, Shengli Pan, and Song Guo. Software defined systems: Sensing, communication and computation, 01 2020.

[70] Gregor Schaffrath, Christoph Werle, Panagiotis Papadimitriou, Anja Feldmann, Roland Bless, Adam Greenhalgh, Andreas Wundsam, Mario Kind, Olaf Maennel, and Laurent Mathy. Network virtualization architecture: Proposal and initial prototype, 01 2009.

[71] N. M. Mosharaf Kabir Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications Magazine*, 47(7):20–26, July 2009. ISSN 1558-1896. doi: 10.1109/MCOM.2009.5183468.

[72] Norbert Egi, Adam Greenhalgh, Mark Handley, Mickaël Hoerdt, Felipe Huici, and Laurent Mathy. Towards high performance virtual routers on commodity hardware, 01 2008.

[73] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture, 10 2008.

[74] Adam Greenhalgh, Felipe Huici, Mickaël Hoerdt, Panagiotis Papadimitriou, Mark Handley, and Laurent Mathy. Flow processing and the rise of commodity network hardware. *ACM SIGCOMM Computer Communication Review*, 39:20–26, 03 2009. doi: 10.1145/1517480.1517484.

[75] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software-defined networking. *IEEE Communications Magazine*, 51(2):136–141, February 2013. ISSN 1558-1896. doi: 10.1109/MCOM.2013.6461198.

[76] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie. A survey on software-defined networking. *IEEE Communications Surveys Tutorials*, 17(1):27–51, Firstquarter 2015. ISSN 2373-745X. doi: 10.1109/COMST.2014.2330903.

[77] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634, Third 2014. ISSN 2373-745X. doi: 10.1109/SURV.2014.012214.00180.

[78] W. Wassapon, P. Uthayopas, C. Chantrapornchai, and K. Ichikawa. Real-time monitoring and visualization software for openflow network. In *2017 15th International Conference on ICT and Knowledge Engineering (ICT KE)*, pages 1–5, Nov 2017. doi: 10.1109/ICTKE.2017.8259622.

[79] Wolfgang Braun and Michael Menth. Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, 6:302–336, 05 2014. doi: 10.3390/fi6020302.

[80] 5G PPP Architecture Working Group. View on 5G Architecture, July 2018. URL `https://5g-ppp.eu/wp-content/uploads/2017/07/5G-PPP-5G-Architecture-White-Paper-2-Summer-2017_For-Public-Consultation.pdf`.

[81] 3GPP. Management and orchestration; Concepts, use cases and requirements. Technical specification (TS) 28.530, 3rd Generation Partnership Project (3GPP), 09 2020. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3273`. Version 16.3.0.

[82] ETSI. Zerotouch network and Service Management (ZSM); End-to-end management and orchestration of network slicing. GROUP SPECIFICATION (GS) 003, European Telecommunications Standards Institute, 06 2021. URL `https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/003/01.01.01_60/gs_ZSM003v010101p.pdf`. Version 1.1.0.

[83] 3GPP. 5G System (5GS) Location Services (LCS); Stage 2. Technical specification (TS) 23.273, 3rd Generation Partnership Project (3GPP), 03 2020. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3577`. Version 16.3.0.

[84] José Santa, Pedro Fernandez, Jordi Ortiz, Ramon Sanchez-Iborra, and Antonio Skarmeta. Offloading positioning onto network edge. *Wireless Communications and Mobile Computing*, 2018, 10 2018. doi: 10.1155/2018/7868796.

[85] Janis Werner, Mario Costa, Aki Hakkarainen, Kari Leppanen, and Mikko Valkama. Joint user node positioning and clock offset estimation in 5g ultra-dense networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2015. doi: 10.1109/GLOCOM.2015.7417360.

[86] Z. Pi and F. Khan. An introduction to millimeter-wave mobile broadband systems. *IEEE Communications Magazine*, 49(6):101–107, June 2011. ISSN 1558-1896. doi: 10.1109/MCOM.2011.5783993.

[87] Z. Pi and F. Khan. System design and network architecture for a millimeter-wave mobile broadband (mmb) system. In *34th IEEE Sarnoff Symposium*, pages 1–6, May 2011. doi: 10.1109/SARNOF.2011.5876444.

[88] W. Roh, J. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun, and F. Aryanfar. Millimeter-wave beamforming as an enabling technology for 5g cellular communications: theoretical feasibility and prototype results. *IEEE Communications Magazine*, 52(2):106–113, February 2014. ISSN 1558-1896. doi: 10.1109/MCOM.2014.6736750.

[89] Patrick Marsch, Ömer Bulakci, Olav Queseth, and Mauro Boldi. *Network Management and Orchestration*, pages 227–261. Wiley Telecommunications, 2018. doi: 10.1002/9781119425144.ch10.

[90] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. *RFC 6241: Network Configuration Protocol (NETCONF)*. RFC Editor, USA, 2011.

[91] OASIS. Topology and Orchestration Specification for Cloud Applications. Oasis standard, OASIS, 11 2013. URL `http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html`. Version 1.0.

[92] Muhammad Waseem Akhtar, Syed Ali Hassan, Rizwan Ghaffar, Haejoon Jung, Sahil Garg, and M. Shamim Hossain. The shift to 6g communications: vision and requirements. *Human-centric Computing and Information Sciences*, 10(1): 53, Dec 2020. ISSN 2192-1962. doi: 10.1186/s13673-020-00258-2. URL `https://doi.org/10.1186/s13673-020-00258-2`.

[93] Chamitha De Alwis, Anshuman Kalla, Quoc-Viet Pham, Pardeep Kumar, Kapal Dev, Won-Joo Hwang, and Madhusanka Liyanage. Survey on 6g frontiers: Trends, applications, requirements, technologies and future research. *IEEE Open Journal of the Communications Society*, 2:836–886, 2021. doi: 10.1109/OJCOMS.2021.3071496.

[94] Gerhard P. Fettweis and Holger Buche. On 6g and trustworthiness. *Commun. ACM*, 65(4):48–49, mar 2022. ISSN 0001-0782. doi: 10.1145/3512996. URL `https://doi.org/10.1145/3512996`.

[95] Alagan Anpalagan, Waleed Ejaz, Shree Krishna Sharma, Daniel Benevides Da Costa, Minho Jo, and Jaeho Kim. Guest editorial special issue on "green communication and computing technologies for 6g networks" in ieee transactions on green communications and networking. *IEEE Transactions on Green Communications and Networking*, 5(4):1653–1656, 2021. doi: 10.1109/TGCN.2021.3125233.

[96] Syed Usman Jamil, M. Arif Khan, and Sabih ur Rehman. Intelligent task offloading and resource allocation for 6g smart city environment. In *2020 IEEE 45th Conference on Local Computer Networks (LCN)*, pages 441–444, 2020. doi: 10.1109/LCN48667.2020.9314819.

[97] Sabuzima Nayak and Ripon Patgiri. *6G Communication Technology: A Vision on Intelligent Healthcare*, pages 1–18. Springer Singapore, Singapore, 2021. ISBN 978-981-15-9735-0. doi: 10.1007/978-981-15-9735-0_1. URL `https://doi.org/10.1007/978-981-15-9735-0_1`.

[98] Fengxiao Tang, Yuichi Kawamoto, Nei Kato, and Jiajia Liu. Future intelligent and secure vehicular network toward 6g: Machine-learning approaches. *Proceedings of the IEEE*, 108(2):292–307, 2020. doi: 10.1109/JPROC.2019.2954595.

[99] 5G Americas. Mobile Communications Beyond 2020 - The Evolution of 5G Towards the Next G. White paper, 5G Americas, 12 2020. URL `https://www.5gamericas.org/wp-content/uploads/2020/12/Future-Networks-2020-InDesign-PDF.pdf`.

[100] Gustav Wikström, Patrik Persson, Stefan Parkvall, Gunnar Mildh, Erik Dahlman, Bipin Balakrishnan, Peter Öhlén, Elmar Trojer, Göran Rune, Jari Arkko, Zoltán Turányi, Dinand Roeland, Bengt Sahlin, Wolfgang John, Joacim Halén, and Håkan Björkegren. *6G – Connecting a cyber-physical world.* Ericsson, February 2022.

[101] N. Kaviani, D. Gaševic, M. Milanovic, M. Hatala, and B. Mohabbati. Model-driven engineering of a general policy modeling language. In *2008 IEEE Workshop on Policies for Distributed Systems and Networks*, pages 101–104, June 2008. doi: 10.1109/POLICY.2008.29.

[102] Distributed Management Task Force. CIM Simplified Policy Language (CIM-SPL), July 2009. URL `https://www.dmtf.org/sites/default/files/standards/documents/DSP0231_1.0.0.pdf`.

[103] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. *RFC3060: Policy Core Information Model – Version 1 Specification.* RFC Editor, USA, 2001.

[104] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. *RFC3198: Terminology for Policy-Based Management.* RFC Editor, USA, 2001.

[105] R. Romeikat and B. Bauer. Formal specification of domain-specific eca policy models. In *2011 Fifth International Conference on Theoretical Aspects of Software Engineering*, pages 209–212, Aug 2011. doi: 10.1109/TASE.2011.29.

[106] ETSI. Experiential Networked Intelligence (ENI); Context-Aware Policy Management Gap Analysis. GROUP REPORT (GR) 003, European Telecommunications Standards Institute, 05 2018. URL `https://www.etsi.org/deliver/etsi_gr/ENI/001_099/003/01.01.01_60/gr_eni003v010101p.pdf`. Version 1.1.1.

[107] W. Liu, C. Xie, J. Strassner, G. Karagiannis, M. Klyus, J. Bi, Y. Cheng, and D. Zhang. *RFC 8328: Policy-Based Management Framework for the Simplified Use of Policy Abstractions (SUPA).* RFC Editor, USA, 2018.

[108] ETSI. Experiential Networked Intelligence (ENI); Overview of Prominent Control Loop Architectures. GROUP REPORT (GR) 017, European Telecommunications Standards Institute, 08 2021. URL `https://www.etsi.org/deliver/etsi_gr/ENI/001_099/017/02.01.01_60/gr_ENI017v020101p.pdf`. Version 2.1.1.

[109] ETSI. Experiential Networked Intelligence (ENI) System Architecture. GROUP SPECIFICATION (GS) 005, European Telecommunications Standards Institute, 12 2021. URL `https://www.etsi.org/deliver/etsi_gs/ENI/001_099/005/02.01.01_60/gs_ENI005v020101p.pdf`. Version 2.1.1.

[110] ETSI. Experiential Networked Intelligence (ENI); Terminology for Main Concepts in ENI. GROUP REPORT (GR) 004, European Telecommunications Standards Institute, 12 2021. URL `https://www.etsi.org/deliver/etsi_gr/ENI/001_099/004/02.02.01_60/gr_ENI004v020201p.pdf`. Version 2.2.1.

[111] IBM. An architectural blueprint for autonomic computing., June 2005. URL `https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf`.

[112] B. Jennings, S. V. Der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, and J. Strassner. Towards autonomic management of communications networks. *IEEE Communications Magazine*, 45(10):112–121, October 2007. ISSN 0163-6804. doi: 10.1109/MCOM.2007.4342833.

[113] Rune G., Westerberg E., Cagenius T., Mas I., Varga B., Basilier H., and Angelin L. Architecture evolution for automation and network programmability, November 2014. URL `https://www.ericsson.com/assets/local/publications/ericsson-technology-review/docs/2014/er-evolved-network-architecture.pdf`.

[114] ETSI. Network Functions Virtualisation (NFV) Release 4 Management and Orchestration; Report on policy information and data models for NFV-MANO. GROUP REPORT (GR) 042, European Telecommunications Standards Institute, 11 2021. URL `https://www.etsi.org/deliver/etsi_gr/NFV-IFA/001_099/042/04.01.01_60/gr_NFV-IFA042v040101p.pdf`. Version 4.1.1.

[115] Engin Zeydan and Yekta Turk. Recent advances in intent-based networking: A survey. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5, 2020. doi: 10.1109/VTC2020-Spring48590.2020.9128422.

[116] H.J. Wang, R.H. Katz, and J. Giese. Policy-enabled handoffs across heterogeneous wireless networks. In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, pages 51–60, 1999. doi: 10.1109/MCSA.1999.749277.

[117] Miguel A. Labrador and Sujata Banerjee. Packet dropping policies for atm and ip networks. *IEEE Communications Surveys*, 2(3):2–14, 1999. doi: 10.1109/COMST.1999.5340708.

[118] P. Steenekamp and J. Roos. Implementation of distributed systems management policies: a framework for the application of intelligent agent technology. In *Proceedings of IEEE International Workshop on System Management*, pages 127–135, 1996. doi: 10.1109/IWSM.1996.534155.

[119] M. Post, Chien-Chung Shen, and J. Wei. The manager/agency paradigm for distributed network management. In *Proceedings of NOMS '96 - IEEE Network*

Operations and Management Symposium, pages 44–53 vol.1, 1996. doi: 10.1109/NOMS.1996.539391.

[120] S. Hinrichs. Policy-based management: bridging the gap. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, pages 209–218, 1999. doi: 10.1109/CSAC.1999.816030.

[121] TM Forum. Intent Management API Profile. Specification 921A, TM Forum, 04 2022. URL `https://www.tmforum.org/resources/standard/tmf921a-intent-management-api-profile-v1-1-0/`. Version 1.1.0.

[122] Minh Pham and Doan B Hoang. Sdn applications - the intent-based northbound interface realisation for extended applications. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 372–377, 2016. doi: 10.1109/NETSOFT.2016.7502469.

[123] Ying Ouyang, Chungang Yang, Yanbo Song, Xinru Mi, and Mohsen Guizani. A brief survey and implementation on refinement for intent-driven networking. *IEEE Network*, 35(6):75–83, 2021. doi: 10.1109/MNET.001.2100194.

[124] TM Forum. Intent Modeling. Introductory Guide IG1253A, TM Forum, 07 2021. URL `https://www.tmforum.org/resources/standard/ig1253a-intent-modeling-v1-0-0/`. Version 1.0.0.

[125] TM Forum. Intent Extension Models. Model TR291, TM Forum, 06 2022. URL `https://www.tmforum.org/resources/technical-report/tr291-intent-extension-models-v1-1-0/`. Version 1.1.0.

[126] TM Forum. Intent Life Cycle Management and Interface. Introductory Guide IG1253C, TM Forum, 07 2021. URL `https://www.tmforum.org/resources/standard/ig1253c-intent-life-cycle-management-and-interface-v1-0-0/`. Version 1.0.0.

[127] Dana A Cooperson. *Intent in Autonomous Networks.* TM Forum, 2022.

[128] Aris Leivadeas and Matthias Falkner. Vnf placement problem: A multi-tenant intent-based networking approach. In *2021 24th Conference on Innovation in*

Clouds, Internet and Networks and Workshops (ICIN), pages 143–150, 2021. doi: 10.1109/ICIN51074.2021.9385553.

[129] Roberto Riggio, Imen Grida Ben Yahia, Steven Latré, and Tinku Rasheed. Scylla: A language for virtual network functions orchestration in enterprise wlans. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 401–409, 2016. doi: 10.1109/NOMS.2016.7502837.

[130] Fred AKLAMANU, Sabine RANDRIAMASY, and Eric RENAULT. Demo: Intent-based 5g iot application network slice deployment. In *2019 10th International Conference on Networks of the Future (NoF)*, pages 141–143, 2019. doi: 10.1109/NoF47743.2019.9015048.

[131] Fred Aklamanu, Sabine Randriamasy, Eric Renault, Imran Latif, and Abdelkrim Hebbar. Intent-based real-time 5g cloud service provisioning. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2018. doi: 10.1109/GLOCOMW.2018.8644457.

[132] M. Riftadi and F. Kuipers. P4i/o: Intent-based networking with p4. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 438–443, 2019. doi: 10.1109/NETSOFT.2019.8806662.

[133] Daphne Tuncer, Marinos Charalambides, Gioacchino Tangari, and George Pavlou. A northbound interface for software-based networks. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 99–107, 2018.

[134] Du Chen, Deyun Gao, Wei Quan, Qianpeng Wang, Gang Liu, and Hongke Zhang. Promoting network automation for heterogeneous networks collaboration. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–5, 2020. doi: 10.1109/VTC2020-Fall49728.2020.9348586.

[135] Dirk Schulz. Intent-based automation networks: Toward a common reference model for the self-orchestration of industrial intranets. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 4657–4664, 2016. doi: 10.1109/IECON.2016.7792959.

[136] Joon-Myung Kang, Jeongkeun Lee, Vasudevan Nagendra, and Sujata Banerjee. Lms: Label management service for intent-driven cloud management. In *2017*

IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pages 177–185, 2017. doi: 10.23919/INM.2017.7987278.

[137] Talha Ahmed Khan, Afaq Muhammad, Khizar Abbas, and Wang-Cheol Song. Intent-based networking platform: An automated approach for policy and configuration of next-generation networks. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, SAC '21, page 1921–1930, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450381048. doi: 10.1145/3412841.3442064. URL `https://doi.org/10.1145/3412841.3442064`.

[138] Khizar Abbas, Muhammad Afaq, Talha Ahmed Khan, Asif Mehmood, and Wang-Cheol Song. Ibnslicing: Intent-based network slicing framework for 5g networks using deep learning. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 19–24, 2020. doi: 10.23919/APNOMS50412.2020.9237008.

[139] Walter Cerroni, Chiara Buratti, Simone Cerboni, Gianluca Davoli, Chiara Contoli, Francesco Foresta, Franco Callegati, and Roberto Verdone. Intent-based management and orchestration of heterogeneous openflow/iot sdn domains. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9, 2017. doi: 10.1109/NETSOFT.2017.8004109.

[140] C. Janz et al. Intent NBI—Definition and Principles. Technical report (TR) 523, Open Networking Foundation (ONF), 10 2016. URL `https://opennetworking.org/wp-content/uploads/2014/10/TR-523_Intent_Definition_Principles.pdf`.

[141] Qiong Sun, Will (Shucheng) LIU, and Kun Xie. An Intent-driven Management Framework. Internet-Draft draft-sun-nmrg-intent-framework-00, Internet Engineering Task Force, July 2019. URL `https://datatracker.ietf.org/doc/draft-sun-nmrg-intent-framework/00/`. Work in Progress.

[142] Joseph McNamara, Liam Fallon, and Enda Fallon. A mechanism for intent driven adaptive policy decision making. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–3, 2020. doi: 10.23919/CNSM50824.2020.9269073.

[143] Leonid Ryzhyk, Nikolaj Bjørner, Marco Canini, Jean-Baptiste Jeannin, Cole Schlesinger, Douglas B. Terry, and George Varghese. Correct by construction networks using stepwise refinement. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 683–698, Boston, MA, March 2017. USENIX Association. ISBN 978-1-931971-37-9. URL `https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/ryzhyk`.

[144] Alexander Clemm, Laurent Ciavaglia, Lisandro Zambenedetti Granville, and Jeff Tantsura. Intent-Based Networking - Concepts and Definitions. RFC 9315, October 2022. URL `https://www.rfc-editor.org/info/rfc9315`.

[145] ETSI. Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Network Service Templates Specification. GROUP SPECIFICATION (GS) 014, European Telecommunications Standards Institute, 05 2021. URL `https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV-IFA%20014v4.2.1%20-%20GS%20-%20Network%20Service%20Templates%20Spec.pdf`. ETSI GS NFV-IFA 014 V4.2.1.

[146] Davide Borsatti, Walter Cerroni, Gianluca Davoli, and Franco Callegati. Intent-based service function chaining on etsi nfv platforms. In *2019 10th International Conference on Networks of the Future (NoF)*, pages 144–146, 2019. doi: 10.1109/NoF47743.2019.9015069.

[147] Adeel Rafiq, Asif Mehmood, Talha Ahmed Khan, Khizar Abbas, Muhammad Afaq, and Wang-Cheol Song. Intent-based end-to-end network service orchestration system for multi-platforms. *Sustainability*, 12(7), 2020. ISSN 2071-1050. doi: 10.3390/su12072782. URL `https://www.mdpi.com/2071-1050/12/7/2782`.

[148] Eder J. Scheid, Cristian C. Machado, Muriel F. Franco, Ricardo L. dos Santos, Ricardo P. Pfitscher, Alberto E. Schaeffer-Filho, and Lisandro Z. Granville. Inspire: Integrated nfv-based intent refinement environment. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 186–194, 2017. doi: 10.23919/INM.2017.7987279.

[149] Federica Paganelli, Francesca Paradiso, Monica Gherardelli, and Giulia Galletti. Network service description model for vnf orchestration leveraging intent-based

sdn interfaces. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–5, 2017. doi: 10.1109/NETSOFT.2017.8004210.

[150] N. Nazarzadeoghaz, F. Khendek, and M. Toeroe. Automated design of network services from network service requirements. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 63–70, 2020. doi: 10.1109/ICIN48450.2020.9059324.

[151] Thomas Szyrkowiec, Michele Santuari, Mohit Chamania, Domenico Siracusa, Achim Autenrieth, Víctor López, Joo Yeon Cho, and Wolfgang Kellerer. Automatic intent-based secure service creation through a multilayer SDN network orchestration. *CoRR*, abs/1803.03106, 2018. URL `http://arxiv.org/abs/1803.03106`.

[152] Kai Gao, Luis M. Contreras, and Sabine Randriamasy. Bi-directional network and application interaction: Application intents upon abstracted network information (invited paper). In *Proceedings of the Workshop on Network Application Integration/CoDesign*, NAI '20, page 43–50, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380447. doi: 10.1145/3405672.3409491. URL `https://doi.org/10.1145/3405672.3409491`.

[153] Tejas Subramanya, Roberto Riggio, and Tinku Rasheed. Intent-based mobile backhauling for 5g networks. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 348–352, 2016. doi: 10.1109/CNSM.2016.7818445.

[154] Mehmet Toy. Intent-based networking for connectivity and cloud services. *Advances in Networks*, 9:19, 01 2021. doi: 10.11648/j.net.20210901.12.

[155] W. Chao and S. Horiuchi. Intent-based cloud service management. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–5, 2018. doi: 10.1109/ICIN.2018.8401600.

[156] Azzam Alsudais and Eric Keller. Hey network, can you understand me? In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 193–198, 2017. doi: 10.1109/INFCOMW.2017.8116375.

[157] Chen Li, Olga Havel, Adriana Olariu, Pedro Martinez-Julia, Jéferson Campos Nobre, and Diego Lopez. Intent Classification. RFC 9316, October 2022. URL `https://www.rfc-editor.org/info/rfc9316`.

[158] Hui Yang, Kaixuan Zhan, Qiuyan Yao, Xudong Zhao, Jie Zhang, and Young Lee. Intent defined optical network with artificial intelligence-based automated operation and maintenance. *Science China Information Sciences*, 63(6):160304, May 2020. ISSN 1869-1919. doi: 10.1007/s11432-020-2838-6. URL `https://doi.org/10.1007/s11432-020-2838-6`.

[159] ETSI. Experiential Networked Intelligence (ENI); InTent Aware Network Autonomicity (ITANA) . GROUP REPORT (GR) 008, European Telecommunications Standards Institute, 03 2021. URL `https://www.etsi.org/deliver/etsi_gr/ENI/001_099/008/02.01.01_60/gr_ENI008v020101p.pdf`. ETSI GR ENI 008 V2.1.1.

[160] Haipeng Yao, Chunxiao Jiang, and Yi Qian. *Developing Networks using Artificial Intelligence*. Springer, 04 2019. ISBN 978-3-030-15028-0. doi: 10.1007/978-3-030-15028-0.

[161] Arthur Jacobs, Ricardo Pfitscher, Ronaldo Ferreira, and Lisandro Granville. Refining network intents for self-driving networks. *ACM SIGCOMM Computer Communication Review*, 48:55–63, 01 2019. doi: 10.1145/3310165.3310173.

[162] Talha Ahmed Khan, Khizar Abbas, Afaq Muhammad, Adeel Rafiq, and Wang-Cheol Song. Gan and drl based intent translation and deep fake configuration generation for optimization. In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 347–352, 2020. doi: 10.1109/ICTC49870.2020.9289564.

[163] Mariam Kiran, Eric Pouyoul, Anu Mercian, Brian Tierney, Chin Guok, and Inder Monga. Enabling intent to configure scientific networks for high performance demands. *Future Generation Computer Systems*, 79:205–214, 2018. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2017.04.020. URL `https://www.sciencedirect.com/science/article/pii/S0167739X1730626X`.

[164] Hocine Mahtout, Mariam Kiran, Anu Mercian, and Bashir Mohammed. Using machine learning for intent-based provisioning in high-speed science networks. In

Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics, SNTA '20, page 27–30, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379809. doi: 10.1145/3391812.3396269. URL `https://doi.org/10.1145/3391812.3396269`.

[165] Inder Monga, Chin Guok, John MacAuley, Alex Sim, Harvey B. Newman, Justas Balcas, Phil DeMar, Linda Winkler, Tom Lehman, and Xi Yang. Software-defined network for end-to-end networked science at the exascale. *CoRR*, abs/2004.05953, 2020. URL `https://arxiv.org/abs/2004.05953`.

[166] Eder J. Scheid, Patrick Widmer, Bruno B. Rodrigues, Muriel F. Franco, and Burkhard Stiller. A controlled natural language to support intent-based blockchain selection. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2020. doi: 10.1109/ICBC48266.2020.9169473.

[167] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[168] P. Louridas and C. Ebert. Machine learning. *IEEE Software*, 33(5):110–115, Sep. 2016. ISSN 1937-4194. doi: 10.1109/MS.2016.114.

[169] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[170] V. S. Timofeev, V. Y. Shchekoldin, and A. Y. Timofeeva. Identification methods for linear regression based on data of household budget surveys. In *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, pages 311–314, Oct 2018. doi: 10.1109/APEIE.2018.8545726.

[171] X. Feng, Y. Zhou, T. Hua, Y. Zou, and J. Xiao. Contact temperature prediction of high voltage switchgear based on multiple linear regression model. In *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 277–280, May 2017. doi: 10.1109/YAC.2017.7967419.

[172] S. Patil and U. Kulkarni. Accuracy prediction for distributed decision tree using machine learning approach. In *2019 3rd International Conference on Trends in*

Electronics and Informatics (ICOEI), pages 1365–1371, April 2019. doi: 10.1109/ICOEI.2019.8862580.

[173] R. K. Amin, Indwiarti, and Y. Sibaroni. Implementation of decision tree using c4.5 algorithm in decision making of loan application by debtor (case study: Bank pasar of yogyakarta special region). In *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, pages 75–80, May 2015. doi: 10.1109/ICoICT.2015.7231400.

[174] S. S. Gavankar and S. D. Sawarkar. Eager decision tree. In *2017 2nd International Conference for Convergence in Technology (I2CT)*, pages 837–840, April 2017. doi: 10.1109/I2CT.2017.8226246.

[175] H. Echoukairi, A. Kada, K. Bouragba, and M. Ouzzif. A novel centralized clustering approach based on k-means algorithm for wireless sensor network. In *2017 Computing Conference*, pages 1259–1262, July 2017. doi: 10.1109/SAI.2017.8252252.

[176] V. S. Chandrawanshi, R. K. Tripathi, and N. U. Khan. A comprehensive study on k-means algorithms initialization techniques for wireless sensor network. In *2016 International Conference on Signal Processing and Communication (ICSC)*, pages 154–159, Dec 2016. doi: 10.1109/ICSPCom.2016.7980567.

[177] S. Wang, J. Cai, Q. Lin, and W. Guo. An overview of unsupervised deep feature representation for text categorization. *IEEE Transactions on Computational Social Systems*, 6(3):504–517, June 2019. ISSN 2373-7476. doi: 10.1109/TCSS.2019.2910599.

[178] Z. Li and X. Li. Fault detection in the closed-loop system using one-class support vector machine. In *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 251–255, May 2018. doi: 10.1109/DDCLS.2018.8515960.

[179] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *The Foundations of Machine Learning*. MIT Press Ltd, 2012.

[180] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779.

[181] Christopher Watkins. *Learning From Delayed Rewards.* PhD thesis, University of Cambridge, Cambridge, UK, 1989.

[182] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 2nd edition, 2018.

[183] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. A survey of text representation and embedding techniques in nlp. *IEEE Access*, 11: 36120–36146, 2023. doi: 10.1109/ACCESS.2023.3266377.

[184] Muhittin Isik and Hasan Dag. The impact of text preprocessing on the prediction of review ratings. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, 28:1405–1421, 05 2020. doi: 10.3906/elk-1907-46.

[185] Ayisha Tabassum and Dr. Rajendra R. Patil. A survey on text pre-processing & feature extraction techniques in natural language processing. *International Research Journal of Engineering and Technology (IRJET)*, 07, 06 2020.

[186] Iskander Akhmetov, Alexandr Pak, Irina Ualiyeva, and Alexander Gelbukh. Highly language-independent word lemmatization using a machine-learning classifier. *Computacion y Sistemas*, 24:1353–1364, 10 2020. doi: 10.13053/CyS-24-3-3775.

[187] Safaa I. Hajeer, Rasha M. Ismail, Nagwa L. Badr, and Mohamed Fahmy Tolba. *A New Stemming Algorithm for Efficient Information Retrieval Systems and Web Search Engines*, pages 117–135. Springer International Publishing, Cham, 2017. ISBN 978-3-319-44270-9. doi: 10.1007/978-3-319-44270-9_6. URL `https://doi.org/10.1007/978-3-319-44270-9_6`.

[188] Sheikh Abujar, Mahmudul Hasan, and Syed Akhter Hossain. Sentence similarity estimation for text summarization using deep learning. In Anand J. Kulkarni, Suresh Chandra Satapathy, Tai Kang, and Ali Husseinzadeh Kashan, editors, *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, pages 155–164, Singapore, 2019. Springer Singapore. ISBN 978-981-13-1610-4.

[189] Muhammad Haroon. Comparative analysis of stemming algorithms for web text mining. *International Journal of Modern Education and Computer Science*, 10: 20–25, 09 2018. doi: 10.5815/ijmecs.2018.09.03.

[190] Afian Rizki, Aris Tjahyanto, and Rahmat Trialih. Comparison of stemming algorithms on indonesian text processing. *Telkomnika (Telecommunication Computing Electronics and Control)*, 17:95–102, 02 2019. doi: 10.12928/TELKOMNIKA.v17i1.10183.

[191] Sundar Singh and R Pateriya. A survey on various stemming algorithms. *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS*, 351:2349–7084, 01 2015.

[192] Garrett Nicolai and Grzegorz Kondrak. Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1138–1147, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1108. URL `https://aclanthology.org/P16-1108`.

[193] Deepa Gupta, R.c Kumar Yadav, and Nidhi Sajan. Improving unsupervised stemming by using partial lemmatization coupled with data-based heuristics for hindi. *International Journal of Computer Applications*, 38:1–8, 2012.

[194] I Purnajiwa and Ngurah Agus Sanjaya ER. Lemmatization in balinese language. *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, 8, 02 2020. doi: 10.24843/JLK.2020.v08.i03.p04.

[195] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar 2003. ISSN 1532-4435.

[196] Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 443–452, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450310154. doi: 10.1145/2207676.2207738. URL `https://doi.org/10.1145/2207676.2207738`.

[197] Xing Wei and W. Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, page 178–185, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933697. doi: 10.1145/1148170.1148204. URL `https://doi.org/10.1145/1148170.1148204`.

[198] Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1024–1033, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://aclanthology.org/D07-1109`.

[199] Michael Evans. A computational approach to qualitative analysis in large textual datasets. *PloS one*, 9:e87908, 02 2014. doi: 10.1371/journal.pone.0087908.

[200] Gabor Berend. Opinion expression mining by exploiting keyphrase extraction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1162–1170, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing. URL `https://aclanthology.org/I11-1130`.

[201] Yongzheng Zhang, Nur Zincir-Heywood, and Evangelos Milios. World wide web site summarization. *Web Intelli. and Agent Sys.*, 2(1):39–53, jan 2004. ISSN 1570-1263.

[202] Anette Hulth and Beáta B. Megyesi. A study on automatically extracted keywords in text categorization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 537–544, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220243. URL `https://aclanthology.org/P06-1068`.

[203] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. *Automatic Keyword Extraction from Individual Documents*, pages 1–20. Wiley, 03 2010. ISBN 9780470689646. doi: 10.1002/9780470689646.ch1.

[204] Suppawong Tuarob, Line Pouchard, and Lee Giles. Automatic tag recommendation for metadata annotation using probabilistic topic modeling. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, pages 239–248, 07 2013. doi: 10.1145/2467696.2467706.

[205] Amit Singhal. Introducing the knowledge graph: things, not strings, 2012. URL `https://www.blog.google/products/search/introducing-knowledge-graph-things-not/`. 2020-11-13.

[206] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *International Conference on Semantic Systems*, 2016.

[207] Jihong Yan, Chengyu Wang, Wenliang Cheng, Ming Gao, and Aoying Zhou. A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12(1):55–74, Feb 2018. ISSN 2095-2236. doi: 10.1007/s11704-016-5228-9. URL `https://doi.org/10.1007/s11704-016-5228-9`.

[208] R. V. Guha, Dan Brickley, and Steve Macbeth. Schema.org: Evolution of structured data on the web. *Commun. ACM*, 59(2):44–51, jan 2016. ISSN 0001-0782. doi: 10.1145/2844544. URL `https://doi.org/10.1145/2844544`.

[209] Dean Allemang, Jim Hendler, and Fabien Gandon. *Semantic Web for the Working Ontologist: Effective Modeling for Linked Data, RDFS, and OWL*, volume 33. Association for Computing Machinery, New York, NY, USA, 3 edition, 2020. ISBN 9781450376174.

[210] Tim Berners-Lee, James Hendler, and Olli Lassila. The semantic web" in scientific american. *Scientific American Magazine*, 284:28–37, 05 2001.

[211] ETSI. Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; NFV descriptors based on YANG Specification. GROUP SPECIFICATION (GS) 006, European Telecommunications Standards Institute, 07 2021. URL `https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/006/03.05.01_60/gs_NFV-SOL006v030501p.pdf`. Version 3.5.1.

[212] ETSI. Network Functions Virtualisation (NFV) Release 3; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point. GROUP SPECIFICATION (GS) 005, European Telecommunications Standards Institute, 10 2021. URL `https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/03.05.01_60/gs_nfv-sol005v030501p.pdf`. Version 3.5.1.

[213] ITU-T. Itu focus group on autonomous networks (fg-an), 2020. URL `https://www.itu.int/en/ITU-T/focusgroups/an/Pages/default.aspx`.

[214] TM Forum. ODA Technical Architecture Release Notes. Reference RN369, TM Forum, 11 2020. URL `https://www.tmforum.org/resources/reference/rn369-oda-technical-architecture-release-notes-v1-0-0/`. Version 1.0.0.

[215] 3GPP. Management and orchestration; Architecture framework. Technical specification (TS) 28.533, 3rd Generation Partnership Project (3GPP), 12 2021. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3416`. Version 16.8.0.

[216] ETSI. Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers. GROUP SPECIFICATION (GS) 009, European Telecommunications Standards Institute, 06 2021. URL `https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf`. Version 1.1.1.

[217] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia. *RFC 7575: Autonomic Networking: Definitions and Design Goals.* RFC Editor, USA, 2015.

[218] Infrastructure European Climate and Environment Executive Agency. H2020 programme, 2023. URL `https://cinea.ec.europa.eu/programmes/horizon-europe/h2020-programme_en`. Online; accessed 25 June 2023.

[219] Harilaos Koumaras, Dimitris Tsolkas, Georgios Gardikis, Pedro Merino Gomez, Valerio Frascolla, Dionysia Triantafyllopoulou, Marc Emmelmann, Vaios Koumaras, Maria L. Garcia Osma, Daniele Munaretto, Eneko Atxutegi, Jara Suárez de Puga, Ozgu Alay, Anna Brunstrom, and Anne Marie Cristina Bosneag. 5genesis: The genesis of a flexible 5g facility. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2018. doi: 10.1109/CAMAD.2018.8514956.

[220] 5G-CLARITY. Use Case Specifications and Requirements. Project Deliverable 2.1, Horizon 2020 Research and Innovation, 03 2020. URL `https://www.5gclarity.com/wp-content/uploads/2020/06/5G-CLARITY_D2.1.pdf`.

[221] 5G-CLARITY. Primary System Architecture. Project Deliverable 2.2, Horizon 2020 Research and Innovation, 10 2020. URL `https://www.5gclarity.com/wp-content/uploads/2020/12/5G-CLARITY_D22.pdf`.

[222] Ericsson. Apex code repo on github, 2018. URL `https://github.com/Ericsson/apex`.

[223] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley, 5th edition, 2012. ISBN 978-0-13-214301-1. URL `file:///Users/liam/References/books/Coulouris2012_Distributed_Systems_Concepts_and_Design_5th_Edition.pdf`.

[224] Open Networking Foundation. Openflow switch specification. Technical Report ONF TS-025, Open Networking Foundation, March 2015. URL `https://www.opennetworking.org/software-defined-standards/specifications/`.

[225] Bob Lantz and Brian O'Connor. A Mininet-based Virtual Testbed for Distributed SDN Development, 2015.

[226] John R Boyd. The Essence of Winning and Losing. *Unpublished lecture notes*, June 1996. URL `http://tobeortodo.com/wp-content/uploads/2011/11/essence_of_winning_losing.pdf`.

[227] J. McNamara, J. Keeney, L. Fallon, S. van der Meer, and E. Fallon. A testbed for policy driven closed loop network management. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, April 2018. doi: 10.1109/NOMS.2018.8406144.

[228] Ericsson. Apex documentation, 2018. URL `https://ericsson.github.io/apex-docs/`.

[229] R. Schatz, S. Egger, and A. Platzer. Poor, good enough or even better? bridging the gap between acceptability and qoe of mobile broadband data services. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6, June 2011. doi: 10.1109/icc.2011.5963220.

[230] Q. Dai and R. Lehnert. Impact of packet loss on the perceived video quality. In *2010 2nd International Conference on Evolving Internet*, pages 206–209, Sep. 2010. doi: 10.1109/INTERNET.2010.51.

[231] Chaithan Prakash, Jeongkeun Lee, Yoshio Turner, Joon-Myung Kang, Aditya Akella, Sujata Banerjee, Charles Clark, Yadi Ma, Puneet Sharma, and Ying Zhang. Pga: Using graphs to express and automatically reconcile network policies. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 29–42, New York, NY, USA, 2015. Association

for Computing Machinery. ISBN 9781450335423. doi: 10.1145/2785956.2787506. URL `https://doi.org/10.1145/2785956.2787506`.

[232] S. Arezoumand, K. Dzeparoska, H. Bannazadeh, and A. Leon-Garcia. Md-idn: Multi-domain intent-driven networking in software-defined infrastructures. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–7, 2017. doi: 10.23919/CNSM.2017.8256016.

[233] Rafael Hengen Ribeiro, Arthur Selle Jacobs, Ricardo Parizotto, Luciano Zembruzki, Alberto Egon Schaeffer-Filho, and Lisandro Zambenedetti Granville. A bottom-up approach for extracting network intents. In Leonard Barolli, Flora Amato, Francesco Moscato, Tomoya Enokido, and Makoto Takizawa, editors, *Advanced Information Networking and Applications*, pages 858–870, Cham, 2020. Springer International Publishing. ISBN 978-3-030-44041-1.

[234] Andrei Palade, Aqeel Kazmi, and Siobhán Clarke. An evaluation of open source serverless computing frameworks support at the edge. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642-939X, pages 206–211, 2019. doi: 10.1109/SERVICES.2019.00057.

[235] David Balla, Markosz Maliosz, and Csaba Simon. Open source faas performance aspects. In *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pages 358–364, 2020. doi: 10.1109/TSP49548.2020.9163456.

[236] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 06 2017. ISSN 2307-387X. doi: 10.1162/tacl_a_00051. URL `https://doi.org/10.1162/tacl_a_00051`.

[237] Joseph McNamara, Erik Aumayr, Liam Fallon, and Enda Fallon. A flexible interpreter for intent realisation. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2022. doi: 10.1109/NOMS54207.2022.9789910.

[238] Facebook Open Source. English word vectors, 2022. URL `https://fasttext.cc/docs/en/english-vectors.html`. Online; accessed 25 June 2023.

[239] The Linux Foundation. The open daylight open source sdn platform, 2017. URL `https://www.opendaylight.org/`.

[240] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10, 2022.