Intent-Driven Network and Service Management: Definitions, Modeling and Implementation

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY-NC-SA 4.0

SUBMISSION DATE / POSTED DATE

24-11-2021 / 06-12-2021

CITATION

Mwanje, Stephen S; Banerjee, Anubhab; Goerge, Juergen; Abdelkader, Abdelrahman; Hannak, Gabor; Szilagyi, Peter; et al. (2021): Intent-Driven Network and Service Management: Definitions, Modeling and Implementation. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.17075450.v1>

DOI

[10.36227/techrxiv.17075450.v1](https://doi.org/10.36227/techrxiv.17075450.v1)

Intent-Driven Network and Service Management: Definitions, Modeling and Implementation

Stephen S. Mwanje¹, Anubhab Banerjee^{1,2}, Juergen Goerge¹, Abdelrahman Abdelkader¹, Gabor Hannak³, Péter Szilágyi³, Tejas Subramanya¹, Julian Goser^{1,4}, and Tobias Foth^{1,4}

¹Nokia Standards, Munich, Germany

²Dept. of Informatics, Technical University of Munich, Germany

³Nokia Bell Labs, Budapest, Hungary

⁴Dept. of Software Methodologies for Distributed Systems, University of Augsburg, Germany

Email: {stephen.mwanje, juergen.goerge, abdelrahman.abdelkader, gabor.hannak, peter.l.szilagy, tejas.subramanya, julian.goser.ext, tobias.foth.ext}@nokia-bell-labs.com, anubhab.banerjee@tum.de

Abstract—Cognitive Autonomous Networks require the network to be able to derive and execute intelligent decisions, and thereby elevating the human operator’s role to a higher level of abstraction where the operator can only specify the desired outcomes from the network. These abstract inputs, called intents, must be supported by corresponding intent-driven capabilities in the network or its management functions. Although, Intent-Driven Management (IDM) has been published in multiple works, there is still no globally agreed end-to-end view of such IDM systems, let alone a globally agreed definition of intents. This paper provides a comprehensive discussion on the core aspects of IDM systems and combines them into an end-to-end system view with the related example solutions. Contrasting against a short review of related scientific and standards literature, the paper introduces a flexible, generic definition of intents and an End-to-End IDM System Architecture as well as the related modeling of intents to support their standardization. The paper also introduces implementation examples fitting the architecture and discusses advanced IDM features that need to be provided, including the ability to detect and resolve conflicts among intents.

Index Terms—Intent, IBN, IDM, CAN

I. INTRODUCTION

Network management refers to those processes and tools which enable the network operator to operate and maintain its network seamlessly and efficiently. Alongside the evolution of wireless telecommunication networks, from 2G to 5G, network management has also continuously been adapted in order to serve the higher demand and new kinds of services. The first softwarized and automated management tools were developed for 2G. These replaced much of the manual administrative processes, but still operated independently on different domains of the network. The use of software agents [1], Open Signalling (OpenSig) [2], and Active Networks [3] were subsequently introduced to reduce the human load of network management. With increasing demand and the newer network architectures of 3G and 4G, the managed systems became larger and more complex. Policy-based network management [4] came to the fore, enabling the enforcement of business procedures and rules by translating them into policies that configure and control the network and its services.

Given the explosion in the context space, e.g., since the emerging 5G, 5G Advanced and 6G networks will serve orders of magnitude more devices and service types, Policy-based management cannot scale and will soon become inefficient for managing networks. At this scale it is of paramount importance for the operator to have simple yet efficient means for network management and orchestration, e.g., without diving into the deep technical details of a certain policy or action. Conceptual research [5] agrees that, in order to adapt to the increasing demand and ever more complex infrastructure, the advanced intelligent network needs to: i. predict or detect network faults in a timely manner, then prevent or repair it autonomously, ii. be able to ingest and execute high-level commands or intents of network operator, and iii. have the ability of easy operation and maintenance. Therefore, Intent-Driven Management (IDM) has become a key topic in network management research as it promises to offer the operators a simple, powerful, and efficient way to manage their infrastructure, resources, and services. Although there is no globally agreed definition for intent, one may consider the intent as a statement of the desired characteristics, behavior or outcomes from a system. From the perspective of the human operator, an intent expresses the MNO’s expectations from the network. The anticipated benefits of IDM are many, e.g., that an Intent-Based Networking (IBN) can be managed in a more seamless and efficient way [5] while intents can configure and optimize business applications and vertical services separately; and that the combination of AI and big data technology in IBNs will improve the robustness of the network and a more context-aware, adaptive, and dynamic operation [6].

Ideally, to decouple the intent from its implementation steps, it is better to express the intent declaratively, i.e., highlighting what shall be achieved and not how to achieve it. For example, it may be stated as a utility level goal that describes the properties of a satisfactory outcome rather than prescribing specific ways to achieve that goal. This presents the system with the challenge to explore and evaluate the possible configuration options and then dynamically adapting it in order

to support that goal. Unlike traditional software systems where requirements are analyzed offline to detect and resolve conflicts prior to implementation, intents potentially result in conflicts since they are added to an autonomous system during runtime.

Existing research on IDM has explored different partially related or disjoint areas, without a comprehensive end-to-end view of the challenges related to realizing such Intent-Driven Management System (IDMS). This work attempts to bring this end-to-end view by tying different aspects together and how they combine to either enable or hinder the realization of IDMS. In particular, we present the following:

- A concise characterization of the existing literature and open questions that need to be investigated,
- A derivation of an intent definition based on an analysis of the different dimensions of a network and its management requirements,
- An end-to-end architecture of the expected typical IDMS highlighting the appropriate way to model intents as managed objects and the critical modules for specifying and fulfilling intent,
- Example solutions for realizing intent fulfillment, and
- A list of advanced features that need to be considered in real-life implementation of IDMS including the continuous contextualization of intents, their coordination and related conflict restitution.

The rest of the paper is organized as follows: in Section II, we discuss existing research works, open source technologies and standards related to IDM. We introduce intent with its definition, dimension and content in Section III and the proposal for a generic end-to-end architecture for IDM in Section IV, while adding example solutions for intent fulfillment in RAN in Section V. We discuss the advanced features for intent fulfillment in Section VI before we end with concluding remarks in Section VII. For reference, Table I lists the critical acronyms used in this article.

II. RELATED WORK

Interest IDM has come from multiple directions with ideas and concepts published in multiple works. The subsections below review the most relevant technical surveys, standards activities and open-source projects on IDM.

A. Surveys

Although the earliest works on IDM are focused on fixed networks, the most recent ones [7]–[9] have focused on IBN for mobile networks as well. Very detailed and comprehensive structural reviews on this topic are covered in [10], [11]. In [10] the authors went beyond the scope of mobile networks to focus on generic intent-driven system where the proposed system is expected to be utilized in the context of business support systems (BSS). With the primary goal of finding existing methods, which can be used to build intent-driven systems, the authors concentrated on two research points which inherently have driven similar publications in the field:

- 1) Existing methods/techniques supporting intent-driven systems which have been presented in the literature.

TABLE I: List of acronyms

IDM	Intent-Driven Management
IDMS	Intent-Driven Management System
IBN	Intent-Based Networking
RAN	Radio Access Network
MNO	Mobile Network Operator
BSS	Business Support Systems
ZSM	Zero-touch Service Management
NFV	Network Function Virtualization
SDN	Software-Defined Networking
ETSI	European Telecommunications Standards Institute
ZSM	Zero touch network & Service Management
NMRG	Network Management Research Group
ONF	Open Network Foundation
ONOS	Open Network Operating System
ODL	OpenDayLight
ENI	Experiential Networked Intelligence
PM	Performance Management
KPI	Key Performance Indicator
ILL	Intent Logic Library
ILU	Intent Logic Unit
ISP	Intent Specification Platform
IFS	Intent Fulfilment System
ILEP	Intent Logic Execution Platform
CAN	Cognitive Autonomous Networks
CF	Cognitive Function

- 2) Proposals to enable realizations of intent-driven systems which have been presented in the literature.

The authors showed that (1) has been covered extensively but there is no existing work to identify techniques that can be combined to realize an intent-driven system.

In [11] the authors analyzed the state-of-the-art IBN methods as proposed in [7]–[9] and mentioned the shortcomings in those methods. The authors pointed out that the existing works do not cover at least one of the four research areas encompassing: (i) intent processing and life cycle, (ii) intent orchestration and management, (iii) analysis of intent use-cases and (iv) an architecture framework for intent-driven networks. In the same paper, the authors also introduced a generalized architectural framework that relied on closed loop feedback for intent processing and interpretation. However, they still didn't answer all the other open questions, like, intent use case analysis, and, so we take a step beyond the architecture to present other framework aspects that are crucial to realizing IDM systems.

The common outcomes from all these surveys is a set of enablers and architectural features while a list of issues have also been consistently noted. The most important enablers are SDN to allow for flexible configuration of network wide functionality and enforcing policies and NFV to enhance the scalability, adaptability and abstraction. Other notable mentions are the need for ZSM as the overarching requirement and the use of machine learning as a critical component in intent processing and automation. On the otherhand, the proposed architecture is expected to be separated into application, network/service management, orchestration and resource layers. Multiple open issues are identified as requiring further work including Intent description and translation, the description of services, processes etc. via data modelling languages and the integration of machine learning and existing NFV/SDN

functionality.

B. open-source projects and Standards Development

Several open-source projects such as ONF [12] have specified standardized intent-based northbound interfaces for Software-defined Networks (SDN) but have rarely published the conceptual ideas and evaluations of these specifications or implementations. On the other hand IDM has only been recently introduced as a topic for discussion in most Standards Development Organizations, only the network management research group (NMRG) of IRTF has previously released internet-drafts on intent-driven network management in [13] and [14]. The NMRG clarifies the concept of network intents, highlights stakeholder perspectives of intents, describes methods to classify intents, defines relevant intent terminologies and provides an overview of functionalities that serves as the foundation for further intent-based network management research. The [15] describes the intent-driven management architecture, its key elements and their interfaces.

The ETSI Experiential Networked Intelligence (ENI) 008 [16] examines various design options for integrating intent operations within the ENI system architecture. This includes accepting, translating and validating intent statements, determining how intent affects the goals and operations of the ENI system, and how it is used by business users, application developers and network administrators.

The 3GPP Technical Report 28.812 [17] describes the concepts of intent-based network management for service-based architecture (e.g., intent expression, intent translation, intent lifecycle) covering various scenarios and their key stakeholders. The potential use cases, their requirements, needed management services, operations and notifications to support intent-driven management for network and service management are described in 3GPP Technical Specification 28.312 [18].

The IG1253 [19] describes the proposal of the TM Forum Autonomous network project for intent-based operation. It introduces the intent management function as an architectural building block for the intent-based operation which will receive an intent, make decisions about suitable actions to improve the intent fulfillment, control the execution of these actions and report on the progress. The IG1253A [20] introduces the models that determine how an intent must be expressed by defining the modelling concepts and artifacts to be used when formulating an intent. The IG1253C [21] defines the details of the intent lifecycle including the related roles and tasks of the intent management function. Furthermore, this document defines the interface through which the intent objects are exchanged, negotiated and managed as part of the life cycle management process.

Currently, ETSI ZSM011 (yet to be published) is investigating the potential of using intent-based network and service management within the ZSM framework by studying the state-of-the-art intent models and intent-based interfaces. It aims to provide guidelines on how to use intent-driven management interfaces between ZSM framework consumers, ZSM management domains and ZSM end-to-end service management

domains. It further investigates the problem of intent conflicts in the ZSM framework and provide coordination solutions for mitigating such issues. Research topics and potential future directions related to ZSM011 are outlined by the original work [22] defining Intelligent Intent Based Networks (I²BN). The concept of I²BN argues that intents may be automatically fulfilled if they are aligned with the capabilities of the network management micro-services, rather than having a broad intent language with free open-ended formalisms that are not supported by the network's implementation. This approach enables the network management to autonomously find and assemble those micro-services that together fulfill an intent in a closed loop manner.

III. INTENT DEFINITIONS

The expectations for intent-based network management raised in many contexts might be summarized to: "Intent-based management will allow human-level abstraction and reduce the need for a very high level of expertise in operating networks" [23]. On one hand there is no commonly agreed definition of the term *intent* in literature yet. On the other hand despite the high expectations, intent-based network management remains to be bound to the theoretical constraints that govern the observability and controllability of networks, which especially in case of mobile networks is limited (for several reasons).

A. Intent as controlling input

Intent-based management means to control a system based on specific controlling input -the *intent*. This *intent* must be mapped to the available strategies for describing and controlling the system - viewed either from the state space of configurations or from the state space of observations or a combination thereof as showed in Fig. 1 [23].

The state space of configurations defines the setup of the system. In the case of a mobile network, this may be the number and location of network elements, the values of its (multitude of) configuration parameters, but also the context of the network elements like mountains, buildings, or the ongoing traffic. All these are input parameters that influence the observable output of the system.

The state space of observables comprises all properties of the system that can be measured. In case of a mobile network these are e.g. signal strength, throughput, or latency measured by UEs, all Performance Management (PM) data like counters on handover failures, and all traces of the signaling interfaces. As illustrated by Fig. 1, any system that is able to receive a desired value of an observable as controlling input must be able to translate this desired value into concrete configuration parameters, because eventually at lowest level the system is controlled by concrete parameters. Thereby, the system may apply several tools in a sequence, each generating a policy at an ever more finer granular level to a final point where the last policy is a concrete configuration of the attributes and parameters of the system.

Fig. 1: A system can be controlled by input from the state space of configurations or the state space of observables. Internally the system has to translate desired observations into concrete configurations [23].

		Controlling input			
		Configurations	Measurables (Gauges, Counters, ...)	KPIs	Weighted Metrics
Telecom Management layers	Business Management				
	Service Management				
	Network management				
	Element Management				

Fig. 2: The two-dimensional continuum of control for telecommunication system [23].

Both kinds of controlling input are applicable at any level of abstraction as e.g. defined by the layers of the telecommunication management network: element, network, service, and business management [24]. Moreover, at any level of abstraction raw measurements might be aggregated to key performance indicators (KPI) or can be combined with weighting factors to represent weighted quality metrics to mirror the priorities of an operator, e.g. to balance between cost and quality. This, as illustrated by Fig. 2, results in a two-dimensional continuum of control [23].

B. Definition of intent

Based on the above the term *intent* might be defined as the *desired state of the system*, where the desired state of the system might be described by a combination of components from the state space of configurations and components from the state space of observables. By following this definition, an intent specifies *what* the resulting state of the system shall be, but it does not give any advice on *how* the system has to reach this state.

A state per se is static in time. As a consequence, an intent does not define the transient steps to reach the desired state, it does not define a process flow like a script with commands. Therefore an intent-driven system has to get itself into the desired state, has to observe its current state and react if the observed state deviates from the desired state. Inherently,

an intent-driven system has to employ closed-loop feedback mechanisms.

C. Context to define the scope of an intent

To fully describe the overall state of a system it requires not only the parameter values of the elements configuration and the observable output of the system but also the description of the environment and its impacts on the system, both the partially known or even completely unknown environment. In practice, it is impossible to fully specify the desired state of the system, so instead an intent may only specify a combination of few, selected parameters and observations. To restrict the intent to specific network elements, areas, or time frames the intent might be augmented by additional components from the configuration or observations state spaces to define the *context* of the intent. Due to their sheer amount, the vast majority of parameters and observations cannot be part of an intent or its context and must be left unspecified. One has to *assume* that an intent still is useful although it only covers few dimensions (i.e. parameters and observations) of the overall state space, while most of the dimensions remain undefined.

To illustrate this under-determination challenge, consider the following very simple example: One may desire that "cells shall have a range of 1.5 km". In a heterogeneous RAN (Het Net) composed of macro and small cells, this intent is only useful for the macro cells, but not for the small cells. To make that explicit and clearly restrict the intent to macro cells, the intent must be augmented by the context information "if the cells are macro cells". Most probably even this is insufficient, since the cell range of macro cells at a city center is smaller than in rural areas, therefore the intent has to be augmented with additional components to account for the different areas of a network. In a real environment probably many other factors impacting the optimal size of a cell have to be considered, like the traffic pattern and the number of UEs. Thus it must be acknowledged that no universal rule will define at which level of detail the context of a given intent is sufficiently described.

D. Context to define bordering conditions

Many observables in networks are not independent from each other, so using a specific value for a measurement or KPI as controlling input might result in degradation of other observables. Due to the complexity of mobile networks (especially in RAN) such dependencies and side effects are hard to predict. As consequence the selection of observables and their values that can be used as conflict free controlling input is not a trivial task.

For a system to use an observable as controlling input means a multi-dimensional optimization problem (see [25]), whose basic requirement is to enforce the intent's targets, but that also has to avoid unwanted side effects that represent the bordering conditions of the optimization problem. The IDM system that uses observables as controlling input might as such consider also other important observables as context, that constrains the stated intent.

E. Consequences from the above definition

First, the term intent as "the desired state of the system" does not prescribe any level of abstraction and so support its flexible in different contexts. It allows for intents that require a very specific configuration detail of a system to be exactly as requested, e.g. an operator might want a configuration parameter to be set to the value 5, which represents a very concrete intent. On the other extreme an intent might describe an observable property of the system on a very abstract level like "the mobile network shall cover 90% of a country with a data rate of 5 Mbit/s". Both intents describe a status, both do not impose any restriction or guidance how the intent gets implemented.

Secondly, the definition does not prescribe the way of achieving the outcomes although it does not guarantee them either. In the first example for instance, the operator might use the local maintenance terminal on site, he might use the network management system, might use as script of CLI commands, or might replace the complete configuration database of the network element. Relatedly, although this intent requests a specific setting of the system, it does not request any specific observable output, i.e. a priori it is not clear whether the implementation of the intent will change the system for the worse, to the good, or will not impact any observable output at all. Similarly, in the second example the operator might implement the intent by buying or leasing equipment to build a network by its own or the operator may become a virtual network operator and buy bandwidth from another existing operator. This intent requests a specific observable output of the system, but does not define any specific configuration or system setup. However, just stating the intent does not guarantee that the system is able to fulfill the request.

Thirdly, the definition does not guarantee interpretability or implementability. For systems with an unknown relation between the desired outcome and the corresponding configuration, it is tempting to use intents on observable outcomes (often called "goals") and to delegate the implementation to the internal management (Fig. 1), hoping that the system knows how to reach the goal. However, this still requires the system to translate between the desired goal and a very concrete setup that can fulfill the goal, a translation that might be impossible. The goal might be unrealistic that even from theoretical point of view it is impossible to setup the system in any way that can reach the targeted observable outcome given the limited resources of the system. And even if the goal is (in theory) reachable, it might be impossible for the system to derive the proper setup to get into the desired state. One reason might be that the system transfer function is unknown, i.e. the relation between a given observable outcome and the corresponding required configuration of the system is not known. This is often the case for mobile networks, because for any realistic environments the properties of the RF transmission on the interface between base station and UE can only be approximated.

Regardless of these challenges, the definition provides a practically workable framework for achieving IDM. Goal-based

intents are robust against changes of the environment as long as the intent-based management system is able to adapt the configuration of the system to meet the desired state in response to forces in the changing environment (disturbance) that attempts to push the system away from the desired state. In principle, the simplification by focusing on "what is the target" instead of defining all the details of "how to reach the target" enables the system to dynamically adapt to its conditions. Moreover, defining an intent as desired state of the system provides the advantage that it excludes dynamically conflicting intents: It is either one state or another state. Conflicts might still occur in case the different components of the intent are in conflict to each other, e.g. the component: "cells shall have a large range" might be in conflict with the component: "Interference at the cell borders shall be low". This challenge is further discussed in section VI-D.

IV. INTENT MODELING AND SYSTEM ARCHITECTURE

An intent specified by an operator may take different forms and degrees of complexity. Thus, we present in this work an end-to-end intent based management system architecture that is capable of fulfilling simple as well as complex intents, while supporting different forms of intent specification. First, we discuss the information elements of an intent and how they lead to a first simple intent model, which we then evolve into a fully declarative model characterized by the different information elements. We extend the declarative model to a model supporting imperative intents and then use the two models to show the required components of the end-end system. We describe the needed capabilities of the major components, including the intent specification platform (ISP) that exposes the IDM MnS (to the operator) as well as the Intent Fulfillment System (IFS) and related assurance system that map the specified intent to appropriate network/resources and processes.

A. Intent as sentence of components

From the operator's perspective there can be very many kinds of intents, from simple intents that can be fulfilled with a single command on a specific network object to very complex intents that include multiple network nodes and several commands on several network objects. Example intents may for example include:

- Restrict Handovers of high mobility users to small cells
- Allow load balancing to a cell Y or to small cells or to only urban cells
- Rehome a BS from controller A to controller B.
- Create a network slice of type IIoT

It is desirable that any intent is stated in a declarative form. Accordingly based on the examples, the overall intent might be composed of a set of components, specifically:

- Intent targets, e.g.: Cells shall have a range of 1.5 km.
- Scope of the intent, e.g.: If they are macro cells.
- Bordering conditions and constraints, e.g.: if the interference from neighbors at the cell border is below a -70 +/-5 dB

Fig. 3: The declarative Intent Model specifies the intentExpectation as a list of IntentTargets on a single type of Object with context used to scope the objects and constrain the targets

Note that the intent may include multiples of each component type, e.g. besides interference the constraint may also include "and if the signal strength at the cell border is above -60 +/-5 dB."

The desired outcome, which is a list of measurable (state) values $s_n[t]$ that describe a managed network at discrete time t can be collected into a state vector:

$$\mathbf{s}[t] = (s_1[t], s_2[t], \dots, s_N[t])^T \quad \text{for } t = 0, 1, \dots, \quad (1)$$

where t represents a generalized discrete time index that is derived from the measurement periods of the respective network functions. The state values include, among others, configuration parameters of the network functions, PM counters, QoS or QoE metrics, latency values, failure counters and any other observed or derived metric while their combination refers to the state of the network or the fulfillment/assurance of the intent. Given this observation, one might consider the intent to be modeled as a simple list of these components, i.e., as:

$$\text{intent} := [\text{scope}(s), \text{target1}, \text{target2}, \dots, \text{constraint1}, \dots]. \quad (2)$$

This in fact matches with the expectation that an intent defines a subspace (a point or set of points) in a multi-dimensional space, i.e. the intent specifies the desired/acceptable regions of the subspace. The challenge is that the intent does not only state the boundaries of regions but also includes conditions under which those boundaries shall be applied. As such the intent must explicitly state the objects under consideration and their context as well as the context to be applied as boundary conditions or constraints for a single target or a group of targets. The intent should as such be extensively modeled to distinguish these information elements as discussed in the next section.

B. Declarative Intent Model

To appropriately model the intent, the intent information object should explicitly define all the information elements needed to identify its scope, requirements or goals as well as its constraints. The explicit way of stating this is using a declarative model - represented by Fig. 3 that defines each intent using the intentExpectations, intentTargets and Contexts.

The intentExpectation is the set of requirements, goals and constraints for a given type of object and an intent may have one or more of these e.g. one for cells and another for network slices. A single desired outcome within an intentExpectation is the intentTargets, i.e. the intentTargets may have multiple targets, as the list of outcomes to be enforced by the IDM system. Each target is a triple stating the attribute on which the target is set, the condition constraining the outcome (e.g. " $=$ ", " $<$ ", " $>$ ") and the valueRange in which the attribute should be.

Each information element may be made explicit using a context that defines its scope and/or constraints. Specific constraints may be set for the IntentTarget and intentExpectation as IntentContext and ExpectationContext respectively, while the scope of the expectation can be made explicit using the ObjectContext that together with the object type define the specific objects for which the targets should apply. Relatedly, one could define global intentContext as context that applies to all intentExpectations within an intent.

C. Imperative Intents: Verbs on declarative statements

The declarative model as described above (and summarized by the Tables in Fig. 3) is explicit in defining all the desired components of the intent but is not concise. For human users and as illustrated by Table II, the declarative statement can be very complex to compile and requires a lot of prior information. For example, an operator wishes to request that no energy saving should be executed for a given cell or for cells in CBD as in the example intents in Table II. Stating this declarative is challenging, since: the user has to know either the parameters that manage energy saving in the cell (i.e. ES_ON) or the metrics energy saving contributes to in order to formulate the desired state or desired values interpretable by the system.

On the other hand the imperative statement (as stated in the imperative column in Table II) is very concise and can be stated with the same degree of completeness. The statements "Restrict ES to Rural cells" and "for Object Rural cells [ES'On=False]" are equivalent, but it is easier for the management service consumer to state this intent using the verb, as opposed to determining the parameter "ES'On" and its possible values

TABLE II: Example specification of network management intents

	Requirement (Imperative Intent)	Declarative Intent (Ensure that ..)	Imperative Intent Challenge
1.	Restrict Handovers of high mobility users to small cells	for Object [List(cell), Size=Small] [HO`Allowed=True] Object [List(cell), Size≠small], [HO`Allowed=False] if cellMobility=high;	Interpret “Restrict” and “small cells” as “True” for small cells and “False” otherwise
2.	Restrict ES to Rural cells on week days	for Object [List(cell), Location =Rural] [ES`On=False; Time=Weekdays;]	Interpret “Rural” as filter to create List(cell); and “week days” as context for ES`On=False
3.	Avoid ES for cell Y / small cells / cells in CBD	for Object List(cell), Location=CBD), ES`On=False]	Interpret “CBD” as a filter on Object cellList“
4.	Rehome a base station from controller A to controller B.	for Object [cell x]; [RNC is B]	Interpret “Rehome” into “cell attribute, RNC” “is equal to”
5.	Create a network slice of type IIoT	for Object [Slice, Filter=new] [SliceProfile=IIoT]	Interpret “Create” into “Filter=new“

Fig. 4: The End-to-end Architecture of an Intent Driven Management System

before stating the intent. Such imperative intents can thus be modeled by adding a verb to the declarative model. As a default an imperative intent expectation implies that by using some terms (e.g., by using a verb), not all the fields of the intent expectation model (as expected for declarative intent expectations) need to be fully stated. For realization however, any such imperative intent needs to be interpreted into an declarative intent, e.g. as illustrated by the examples in Table II.

D. End-to-end IDM system architecture

The intent models as discussed above characterize the interface or interfaces over which the consumers and the producers of IDM service can interact to express or specify and/or receive the intents. However, for the fulfillment of the intents, it is necessary to insert the interfaces into the end-end architecture of the IDM system.

The end-end architecture for IDMS needs (as illustrated in Figure 4) to provide means to capture intents, format the into realizable outcomes and provide means to realize those intents. It is expected that the intents will come into the IDMS in form of text e.g. via a text file, legacy commands via a Command-Line Interface (CLI), through interaction with a graphical user interface (GUI) or as speech e.g. via an audio interface that captures the operator’s audio commands. Regardless of any such input interface, the user has access to an intent specification platform to which he specifies the intent to be executed. The ISP may take in the operators request

and parses the input to identify the fields fitting to the defined intent specification syntax, according to the imperative or the declarative intent models. The ISP may also implement other functionality e.g. authentication and validation functions that may be needed to ensure that the right entities instantiate and execute intents and that the right intents are executed on the system.

The system includes an Intent-model Translation Function (ITF) that translates the (possibly ambiguous) imperative intent into and unambiguous declarative intent. The ITF interprets a verb to identify objects of the declarative intent and may interpret other terms to derive filter information to identify the specific managed Objects, or contexts for individual targets and/or groups thereof. The ITF may be implemented as part of either the specification platform or as part of the fulfillment systems.

The outcome of the intent specification platform and the related ITF will be a hierarchy of hash functions according to the declarative intent model that describe the features of the intent to be fulfilled. For example, The intent specifications may be similar to the declarative intent entries in table II. The declarative intent is the input to the Intent Fulfillment and Assurance Systems which then undertake actions to realize the intent. In many implementations, Fulfillment systems may not be separate from intent assurance systems, but they are showed separately here to clarify that they may have distinct requirements which need to be considered. Similarly, there can be multiple methods for fulfilling intents, which may also

Fig. 5: Intent Fulfillment using Intent Logic Units (ILU) stored in an Intent Logic Library (ILL)

depend on the nature and contents of the intents. The next section discusses some of the existing ideas on intent fulfillment.

V. INTENT FULFILLMENT

Although the IBN concept is widely discussed, there has been no discussion on how such a system may be realized. In particular, it remains open how the any specified intent may be fulfilled. This section introduces two methods for fulfilling user or operator specified intents within the automated control of communication networks.

A. Intent Logic Units (ILU) and Intent Logic Library (ILL)

The ILL-ILU concept is a method for fulfilling user or operator intents using Intent Logic Units (ILU) stored in an Intent Logic Library (ILL) as illustrated by Figure 5.

Each ILU may be considered as a wrapper around the logic or command sequences that need to be executed to achieve a specific Intent. ILUs may be manually written by the system manufacturers and operators or may be learned by the system. Different ILUs may be created for different variations in the features of the intents. Accordingly, each ILU is characterized by an identifier and the logic or command sequences to be executed to implement it. It may; however, also be characterized by a human-readable name and a description intended for human users who may either want to revise, reuse or remove the ILU. For example, the human user may want to combine multiple ILUs to form another more aggregate ILU.

The ILL is a catalog of ILUs that can be searched to find the appropriate ILU for a given intent. Intent execution may be fulfilled via an Intent-Logic Execution Platform (ILEP) that identifies the appropriate ILU and schedules its execution on the network or its objects. In that case, the IF is the combination of the ILEP and the ILL, the ILL implemented either as part of or separate from the ILEP.

To fulfil an intent, the ILEP interacts with the ISP, the ILL and the network via message exchanges as illustrated by the end-to-end intent fulfillment flow in Figure 6.

Given an intent x from the ISP (1), the ILEP confirms the completeness of the intent specification, e.g. that for the stated verb, the applicable context and parameters are supplied (2). In case of an inconsistency, the ILEP informs the ISP accordingly (3), otherwise the ILEP requests for the ILU from the intent Logic Library (4). The ILL searches for an ILU that matches

Fig. 6: Implementing intent fulfillment via ILL and ILUs

x specification and if the ILU exists (5), the ILL returns the ILU to the ILEP otherwise it returns a failure event (6).

If the ILU is returned, the ILEP schedules the execution of the ILU, i.e. the ILEP checks if ILU can be immediately executed or if it has to wait and must be scheduled at a different time (7). Otherwise, if the ILU is not returned, ILEP may inform the ISP that the intent cannot be fulfilled possibly with the reason, e.g. that the corresponding ILU does not exist (13). The ILEP may however, take steps to find alternative means of executing the intent, especially to ensure that x can be executed in future (14). For example, the ILEP may initiate a process of learning the appropriate ILU for the intent.

After the ILU or sequence of actions thereof are activated on the network (8), the ILEP evaluates the outcomes of the ILU activation to confirm that the intent is achieved (9). If the intent is fulfilled, the ILEP may inform the ISP accordingly, otherwise If the ILU does not fulfill the intent, the ILEP may decide to delete the ILU or to mark it as ineffective (10). Relatedly, the ILEP may initiate actions to ensure the intent is realizable in future, e.g. to re-learn the ILU for that intent (12).

B. Intent Fulfillment using CAN

An alternative way of implementing the fulfillment of intents (especially operator intents) is by using an Orhestartor, here called the Intent-Driven Network Automation Function Orchestrator (IDNAFO) [26], which orchestrates the execution of automation functions in a Cognitive Autonomous Networks

Fig. 7: CAN abstraction with the CFs and the Controller

(CAN) [23]. Here below we summarize the operation of the two core functionalities.

1) **CAN**: The CAN is the network environment where learning-based Network Automation Functions, called Cognitive Functions (CFs), are deployed to automate the control of network resources (like cell parameters) and optimize network metrics (like cell KPIs). For each KPI, there is a closed-loop CF with the responsibility of learning how the KPI varies with changes in the control parameters [27] and the objective of determining the optimal value of the control parameters and respectively the KPI.

Very often it is found that multiple KPIs are influenced by a single control parameter. For example, both RLF and downlink throughput change when TXP is changed. With the assumption that each CF works independently without communicating with others (which is typically the case), if each CF changes a certain control parameter solely according to its own objective, the stability of the whole system may be compromised. accordingly, a control and coordination mechanism (here referred to as the Controller) is added to the set of CFs to resolve conflicts among the CFs relating to any control parameter. In CAN, the CF is always learning [28], to be able to determine the configuration (i.e. values of the control parameters) for which its objective is optimal in a certain network state. However, only the Controller can adjust the control parameters in the network. If a CF determines a new configuration of a specific parameter to optimize its objective, it conveys this configuration to the Controller. The Controller then takes into account all configurations collected from different CFs and determines the optimal value of this parameter for the combined interest of the system [27]–[29].

Without any intricate details, we can abstract a CAN in where the Controller lays between the CFs and the network as shown in Figure 7. In this example, F_1 and F_2 are two CFs with a shared control parameter: p . Whenever F_1 or F_2 finds a new optimal configuration of p individually, it send its preference to the controller [27], [28], and requests the Controller to recalculate p as shown in Figure 7. The Controller asks the other CF to send its preference too, and calculates a value of p which is optimal for the combined interest of F_1 and F_2 and sets the new value in the network.

From the overview of CAN, we see that CAN works in a self-sustainable closed-loop way, i.e., in a changing envi-

Fig. 8: IDNAFO functionality blocks

ronment CAN always adjusts the network parameters to reach an equilibrium. As such given an external interface, the CAN could be used to execute configurations or deliver desired outcomes from the operator. The proposed IDNAFO provides such an interface.

2) **Intent-Driven Network Automation Function Orchestrator (IDNAFO)**: As the entry to an intent fulfillment system, the IDNAFO takes a formal intent as its input and generates appropriate actions for the Controller and CFs as its output. The formal intent may be according to either of the imperative or declarative models. As shown in Figure 8, end-to-end workflow of IDNAFO consists of three sequential steps, based on which we introduce three different functionality blocks:

- **Intent Identifier**: as soon as IDNAFO receives a formal intent from ISP, the first task is to check if the intent is valid for CAN, i.e., if the intent can be executed by the Controller and CFs.
- **Intent Classifier**: after the intent identifier identifies that an intent is executable by CAN, the next task is to classify the intent based on its content.
- **Intent Decision Maker**: after the classification is done, based on the type of intent, IDNAFO takes subsequent actions like sending specific commands to the Controller or to the CFs or to both.

VI. ADVANCED FEATURES FOR INTENT FULFILLMENT

A. Tracking Intents

Intents may be defined by human operators with different roles, levels of expertise and responsibility areas, e.g., BSS experts, OSS experts or technical engineers for different network domains (e.g., end-to-end, domain and sub-domain). Different domain-specific IFSs may be responsible for different intents depending on the intents' scope. The IFS is responsible of the validation, acceptance or rejection, fulfillment (contextualization, conflict resolution, deletion, adaptation, etc.) and assurance of the intents through their full life-cycle (creation/definition, operation and termination by the operator or when the life-time of an intent expires). In all these tasks the IFS needs to inform the operator accordingly. The Intent Interface should provide the appropriate insight on the network status, configuration, performance, and on the operation of the IBN framework itself including information about the status of the intents, their consistency and potential conflicts between intents.

B. Contextualizing Intents

Intents will typically be high-level objectives, without any specifics on how they should be converted, contextualized and executed/enforced in a coherent, robust and consistent manner. To ensure that intents can be fulfilled without jeopardizing the consistency and robustness of the network, the intents need to be contextualized, in a continuous closed loop mechanism that follows and adapts to relevant changes in the intent's networking context. Thereby, the IFS needs to compile relevant insight that enables its operation, be it end-end service domain level or in the specific (sub)domains as described below.

At the end-end service domain level, contextualization includes creation of cross-domain insight for efficient e2e optimization, information that allows accurate enough network and service state discovery modelling and self-learning and adaptation capabilities that enable qualifying the efficiency and impact of its own actions. After processing and contextualizing the Intents, the e2e IFS performs domain specific adaptation, which is the process of converting the e2e Intent to the syntactical and semantical framework of the Domain Specific IFSs. The e2e IFS then transfers this specialized information through the Domain's own Intent Interface, which can also be used directly by the human operator for the same purposes.

The (sub) Domain Specific IFSs implement the same functionality as the e2e IFS, including validation, contextualization, optimization, self-learning by the Intent fulfillment Logic and the maintenance of Intent Repository for the accepted intents. Additionally, the domains implement southbound IDM interfaces towards the network by acting as consumers of the services provided by the corresponding (sub)domain controllers or network functions, e.g., the Domain Specific IFSs are responsible of converting the intents to commands or configurations according to the services provided by the northbound interface of the (sub)domain controllers or network functions. During this process, Domain Specific IFS uses the feedback and measurements collected from the of the (sub)domain controllers and network functions for continuous adaptation and contextualization via this southbound API.

C. Intent feasibility checking

In order to provide coherent operation, intents are first validated both syntactically and semantically. Syntactically and semantically valid intents should then be examined for their technical feasibility as a valid intent that causes no conflicts whatsoever might still not be fulfillable due to missing network-side functionalities. Checking the technical feasibility of an intent requires the collection, discovery and up-to-date storage of the network status, capabilities and configuration data the level of abstraction where the e2e or domain (subdomain) specific IFS operates. Invalid intents are rejected and the cause of rejection (e.g. a reason code) is provided as a feedback to allow resubmission with improved specification. Note that a syntactically and semantically valid intent can still be infeasible due to inconsistency or conflicts with existing intents, as such consistency checks are required for each received intent. Those

intents for which the conflict resolution is not possible will typically be rejected but the determination and resolution of conflicts is a major challenge as discussed in the next section.

D. Intent Coordination and conflict resolution

Naturally, we expect an IDMS to efficiently handle and coordinate multiple simultaneously active intents, each of which may be decomposed into multiple control actions and control loops which act in a dynamic, context-aware manner, on a shared set of resources guided by a shared set of input data. Interactions and interference between intents is thus inherently unavoidable, and intents active at the same time in the same IDMS may conflict, so their coordination is absolutely necessary. Thereby, the coordination in IDMS must support conflict prevention, detection, and resolution.

Prior research on conflicts in network control and management focuses on policy-based control and explores conflicts from simple characteristics such as temporal conflicts, outcome conflicts, conditional conflicts etc. with correspondingly simple forms of coordination, e.g., prioritization, partial fulfillment, or queuing. For example, by exploring the temporal aspects within a *event*→*condition*→*action* policy principle one can point out and formalize conditions on the temporal overlaps between *observation/measurement*, *action*, and *effect* time spans of the policies [30]–[33]. Another, abstract approach employed predominantly in IT services is to associate events, conditions, and actions with abstract mathematical objects and then to employ the tools of formal languages and first-order logic to discover conflicts [34]–[36]. Our goal is to focus on a conceptual view of IDMS while pointing out some key properties of conflicts and conflict detection.

Without claiming to be exhaustive, we can point out at least the following four (not independent) conflict patterns:

- *Direct target conflict*: where at least one network parameter is targeted by (at least) two different intents in an incompatible way, e.g., where one intent desires to decrease the value of parameter while the assurance of a second intent requires to increase it.
- *Dependency conflict*: where the outputs and inputs of the control actions and loops that serve two different intents may end up being in a circular or otherwise inconsistent dependency relationship.
- *Contextual conflict*: which refer to any conflict whose existence or non-existence depends on the environment state. As the IBN operates in a context-aware manner, changes in the environment (e.g., user/traffic demand) will trigger changes in control actions and loops. Thus, conflicts may emerge and disappear depending on the usage of the network.
- *Latent conflict*: where there is not any explicit direct or dependency conflict, but the effect of the fulfillment/assurance of the two intents on the network and its users is ambiguous, counteractive, or in some other way dissonant and undesired as judged by the operator or customers.

To more formally define conflicts, we suggest (as discussed in section IV-A) to consider the network-state vector representation of the eventual outcomes of an intent or its fulfillment status. Note that the network state can change even when the network is not being controlled by intents, e.g. due to its interaction with its dynamic environment, like the changing number of users and traffic demand. Clearly, the consequence of the fulfillment or assurance of an intent is that the network will be steered towards a (possibly dynamically changing) desired state vector, or kept in a certain confined state region/subspace that is associated with the desired outcomes stated in the intent. Typically, the dimensionality of both the state vector and the control inputs is very high that manual administration and orchestration of the intents' control loops is infeasible and automated methods are necessary to coordinate intents. As such automated coordination solutions that evaluate the overlap of the state vectors are needed.

VII. CONCLUSION AND FUTURE DIRECTION

Intent-Based Networking (IBN) or Intent-Driven Management (IDM) is seen as one of the next steps towards autonomous network management system. As a vast number of organizations contribute to realize this paradigm, a common and clear understanding of the underlying concepts is crucial for future research. Main goal of this paper was to comprehensively describe and discuss core aspects of such IDM systems, combine them into a complete end-to-end view and show potential solutions via examples.

After an extensive review on existing survey papers, standardization efforts and open source approaches in the context of IBN and IDM in section II, we provided a highly-flexible, generic definition of intents in section III. The key aspects of our definition are that: 1) it does not restrict intents to any single technical degree of abstraction, 2) it does not dictate the manner in which intents are handled in the system, and 3) it does not include interpretability or implementability of the intent itself.

Given the definition, we then proposed an end-to-end system architecture of IDM system using a declarative intent model, based on the above definition, in section IV. Declarative formats naturally describe a desired or intended state rather than dictating imperative actions to be performed. This not only fits the conceptual idea of Intent-Based and Intent-Driven Networks but enables intent status controls to conveniently build upon the deviation of the network state from the desired state described directly by the formal intent. On the other hand, imperative formats may be more concise at times and well-known to operators. In order to ensure that the system uses the advantages from both models, we suggested an Intent-model Translation Function which is able to translate imperative statements into model-conforming declarative statements. However, this translation is based on the assumption that both models are able to describe the same statements, differing only syntactically. As this assumption may not always hold, further investigation is required as part of future work.

We presented two Intent Fulfillment Systems in section V as examples which fit well with the proposed generic architecture of IBN. The first one is a modular approach that uses formal Intent Logic Units (ILUs) implementing specific Intent Fulfillment logic and stored in an Intent Logic Library. Then, for an incoming intent, the relevant ILU(s) are picked from the library and their actions are executed via an Intent Logic Execution Platform. The second extends a Cognitive Automation Network (CAN) by adding a so-called Intent-Driven Network Automation Function Orchestrator, which is responsible for identifying and classifying intents to decide on those that can be fulfilled through the CAN and its functions. Both systems allow the fulfillment of intents in their respective contexts but need to be extended in the future to support more features of intent fulfillment.

Following these examples, we rounded off the paper describing advanced features of Intent Fulfillment Systems in section VI. First, we argued for the necessity of tracking intents in such systems, where the intents' statuses and their lifecycles have to be made comprehensibly trackable to various consumers in their respective roles. Secondly, we discussed the need and the requirements of contextualizing intents such as the necessity to create cross-domain insight on an end-to-end service domain level. Third, we mentioned that for intent feasibility checking, it is necessary to discover, collect and store network state data to support the syntactical and semantic validity checking of intents and for the appropriate feedback to be given. Finally, owing to the possibility of conflicts among active intents, we discussed conflict resolution and intent coordination including a categorization of intent conflicts.

As stated in the paper introduction, the advanced intelligent network has three critical features two of which can be met through Intent-Based/Intent-Driven Management Systems. The intelligent network needs to be able to ingest and execute high-level commands or intents of network operators (requirement ii) and the ability of easy operation and maintenance (requirement iii). The proposed capabilities and mechanisms for Intent-Based/Intent-Driven Management shall ensure achievement of both requirements first because they allow the operators to control the network through intents and because the use of intents significantly simplifies network management. Moreover the proposed system does introduce no complexities but ensures to minimize them e.g. though the use of imperative intents. The remaining open challenge is ability to detect or predict network faults in a timely manner and prevent or repair them autonomously. This is a required internal capability for any such autonomous networks with no clear impact on the intent driven management system or its interfaces. It is however, an ongoing research task in Nokia for we also expect to focus in our future research.

Nevertheless, the concepts presented here are core functionalities providing an end-to-end view for Intent-Based and Intent-Driven Management Systems and will thus help to promote further efforts in realizing such network management systems and to close the still open challenges towards advanced

intelligent networks.

REFERENCES

- [1] Vu Anh Pham and Ahmed Karmouch. Mobile Software Agents: an Overview. *IEEE Communications magazine*, 36(7):26–37, 1998.
- [2] Andrew T Campbell, Irene Katzela, Kazuho Miki, and John Vicente. Open Signaling for ATM, Internet and Mobile Networks. *ACM SIGCOMM Computer Communication Review*, 29(1):97–108, 1999.
- [3] David L Tennenhouse and David J Wetherall. Towards an Active Network Architecture. In *Proc. of DANCE*, San Francisco, CA, USA, 2002.
- [4] Raouf Boutaba and Issam Aib. Policy-based management: A Historical Perspective. *Journal of Network and Systems Management*, 15(4):447–480, 2007.
- [5] Yiming Wei, Mugen Peng, and Yaqiong Liu. Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280, 2020.
- [6] Zhengquan Zhang, Yue Xiao, Zheng Ma, Ming Xiao, Zhiguo Ding, Xianfu Lei, George K. Karagiannidis, and Pingzhi Fan. 6g wireless networks: Vision, requirements, architecture, and key technologies. *IEEE Vehicular Technology Magazine*, 14(3):28–41, 2019.
- [7] 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. IEEE, 2020.
- [8] Lei Pang, Chungang Yang, Danyang Chen, Yanbo Song, and Mohsen Guizani. A survey on intent-driven networks. *IEEE Access*, 8:22862–22873, 2020.
- [9] Yiming Wei, Mugen Peng, and Yaqiong Liu. Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280, August 2020.
- [10] Johan Silvander, Krzysztof Wnuk, and Mikael Svahnberg. Systematic literature review on intent-driven systems. *IET Software*, 14(4):345–357, 2020.
- [11] Kashif Mehmood, Katina Kravlevska, and David Palma. Intent-driven autonomous network and service management in future networks: A structured literature review, 2021.
- [12] Christopher Janz, Nigel Davis, David Hood, Mathieu Lemay, David Lenrow, Li Fenkai, Fabian Schneider, John Strassner, and Andrew Veitch. Intent nbi – definition and principles. *ONF TR-523*, 2016.
- [13] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. Intent-based networking-concepts and definitions. *IRTF draft*, 2020.
- [14] Chen Li, O Havel, W Liu, A Olariu, Pedro Martinez-Julia, JC Nobre, and Lopez DR. Intent classification. *IETF draft*, 2019.
- [15] B Claise, J Quilbeuf, Y El Fathi, D Lopez, and D Voyer. Service assurance for intent-based networking architecture. *IRTF draft*, 2021.
- [16] InTent Aware Network Autonomicity (ITANA). Standard, ETSI GR ENI, 2021.
- [17] Study on scenarios for Intent driven management services for mobile networks. Standard, 3GPP, 2020.
- [18] Intent driven management services for mobile networks. Standard, 3GPP, 2021.
- [19] Intent in Autonomous Networks. Standard, TM Forum, May 2021.
- [20] Intent Modeling. Standard, TM Forum, May 2021.
- [21] Intent Life Cycle Management and Interface. Standard, TM Forum, May 2021.
- [22] Szilágyi Péter. I2bn: Intelligent intent based networks. *Journal of ICT Standardization*, 9(2):159–200, 2021.
- [23] S. S. Mwanje and C. Mannweiler. *Towards Cognitive Autonomous Networks: Network Management Automation for 5G and Beyond*. John Wiley & Sons, 2020.
- [24] ITU-T. Principles for a telecommunications management network. *M.3010*, 2000.
- [25] J Strassner and J Halpern. Intent-based policy management. In *IETF 95, 2016. Proceedings* (<https://www.ietf.org/proceedings/95/>). IETF, 2016.
- [26] A. Banerjee, S. S. Mwanje, and G. Carle. An intent-driven orchestration of cognitive autonomous networks for ran management. In *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021.
- [27] A. Banerjee, S. S. Mwanje, and G. Carle. Game theoretic conflict resolution mechanism in cognitive autonomous networks. In *2020 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE, 2020.
- [28] A. Banerjee, S. S. Mwanje, and G. Carle. Optimal configuration determination in cognitive autonomous networks. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 494–500. IEEE, 2021.
- [29] A. Banerjee, S. S. Mwanje, and G. Carle. On the necessity and design of coordination mechanism for cognitive autonomous networks. *arXiv:2001.07031*, 2020.
- [30] T. Bandh. *Coordination of autonomic function execution in Self-Organizing Networks*. PhD thesis, Technische Universität München, 2013.
- [31] T. Bandh and L. C. Schmelz. Impact-time concept for SON-function coordination. In *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pages 16–20. IEEE, 2012.
- [32] T. Bandh, R. Romeikat, H. Sanneck, and H. Tang. Policy-based coordination and management of SON functions. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 827–840. IEEE, 2011.
- [33] N. Dunlop, J. Indulska, and K. Raymond. Methods for conflict resolution in policy-based management systems. In *Seventh IEEE International Enterprise Distributed Object Computing Conference, 2003. Proceedings.*, pages 98–109. IEEE, 2003.
- [34] B. L. A. Batista, G. A. L. de Campos, and M. P. Fernandez. Flow-based conflict detection in OpenFlow networks using first-order logic. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2014.
- [35] J. Chomicki, J. Lobo, and S. Naqvi. Conflict resolution using logic programming. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):244–249, 2003.
- [36] E. Bertino, A. Mileo, and A. Provetti. PDL with preferences. In *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY’05)*, pages 213–222. IEEE, 2005.