INTENT-DRIVEN NETWORK AND SERVICE MANAGEMENT: DEFINITIONS, MODELING AND IMPLEMENTATION

Stephen S. Mwanje[1], Anubhab Banerjee[1,2], Juergen Goerge[1], Abdelrahman Abdelkader[1], Gabor Hannak[3], Péter Szilágyi[3], Tejas Subramanya[1], Julian Goser[1,4], Tobias Foth [1,4]
[1]Nokia Standards, Munich, Germany, [2]Dept. of Informatics, Technical University of Munich, Germany, [3]Nokia Bell Labs, Budapest, Hungary, [4]Dept. of Software Methodologies for Distributed Systems, University of Augsburg, Germany,
Emails: {stephen.mwanje, juergen.goerge, abdelrahman.abdelkader, gabor.hannak, peter.1.szilagyi, tejas.subramanya}@nokia-bell-labs.com, {julian.goser, tobias.foth}.ext@nokia.com, anubhab.banerjee@tum.de
NOTE: Corresponding author: Stephen S. Mwanje, stephen.mwanje@nokia-bell-labs.com

Abstract – *Cognitive autonomous networks imply that networks will independently derive and execute intelligent decisions, and thereby elevating the human operator's role to a higher level of abstraction. At that level, the operator only specifies the desired outcomes, called intents, which must then 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 between intents.*

Keywords – CAN, IBN, IDM, intent

1. INTRODUCTION

Network management, the processes and tools with which network operators seamlessly and efficiently operate and maintain their network, has continuously been adapted to support the higher demand and new service types introduced through the evolution of wireless networks. The first softwarized management tools developed for 2G replaced much of the manual administrative processes through scripted operations. The use of software agents [1], open signaling [2], and active networks [3] were then introduced to reduce the human load of network management. However, as the managed systems became larger and more complex in the newer architectures in 3G and 4G, Policy-Based Network Management (PBNM) [4] came to the fore, promising the enforcement of business processes and rules through policies that configure and control the network and its services. Nevertheless, since PBNM requires an enumeration of the candidate actions in different contexts, it is not adequately scalable to serve the new 5G, 5G advanced and 6G networks with significant explosions in context space resulting from serving orders of the magnitude of more devices and service types. For example, support for IIoT implies new contexts different from eMBB-only networks like in 4G. Instead, operators need 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, among others: i) detect and predict network faults in a timely manner, then prevent or repair it autonomously, ii) be able to process and execute high-level network-operator commands or intents, and iii) have the ability of easy operation and maintenance. So, 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. Among the anticipated benefits are that: Intent-Based Networking (IBN) can be managed in a seamless and efficient way [5]; intents configure and optimize business applications and vertical services separately; and AI and big data technology in IBN will improve network robustness, supporting context-aware, adaptive, and dynamic operation. [6].

There is no globally agreed definition for intent, but one may consider an intent as a statement of the desired characteristics, behavior or outcomes from a system. From the human operator's perspective, intents express the operator's expectations from the network. Ideally, to decouple an intent from its implementation steps, intents are preferably expressed declaratively, i.e., highlighting what shall be achieved and not how to achieve it. It may, for example, 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 to dynamically adapt them towards supporting that goal. Unlike traditional software systems where requirements are analyzed offline to detect and resolve conflicts prior to implementation, intents can potentially result in conflicts since they are added to the system during runtime.

Existing research on IDM has explored different partially related areas, without a comprehensive end-to-end view of the challenges related to realizing such an Intent-Driven Management Systems (IDMSs). 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 the related limitations,
- An intent definition based on the analysis of network management and operability requirements,
- An end-to-end architecture for IDMS highlighting the appropriate intent information models and the critical modules for specifying and fulfilling intents,
- Example solutions for 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 2, we discuss existing research, open source technologies and standards related to IDM. We introduce intent with its definition, dimension and content in Section 3 and the proposal for a generic end-to-end architecture for IDM in Section 4, while adding example solutions for intent fulfillment in RAN in Section 5. We discuss the advanced features for intent fulfillment in Section 6 before we end with concluding remarks in Section 7. For reference, Table 1 lists the critical acronyms used in this article.

Table 1 – List of acronyms

IDM	Intent-Driven Management
IDMS	Intent-Driven Management System
IBN	Intent-Based Networking
RAN	Radio Access Network
NFV	Network Function Virtualization
SDN	Software-Defined Networking
ETSI	European Telecommunications Standards Institute
ZSM	Zero touch network & Service Management
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
RLF	Radio Link Failure

2. RELATED WORK

Interest in IDM comes from multiple institutions with concepts published in multiple pieces of work. The subsections below review the most relevant technical surveys, standards activities and open-source projects on or related to IDM.

2.1 Surveys

Although the earliest work on IDM is focused on fixed networks, the most recent ones [7, 8, 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 systems where the proposed system is expected to be utilized in the context of business support systems. The authors intended to find: 1) existing methods/techniques supporting intent-driven systems and 2) proposals for enabling realizations of intent-driven systems. Their observation was that intention 1) has been covered extensively but that there is inadequate work or techniques that can be combined to realize an intent-driven system.

In [11] the authors analyzed the IBN methods proposed in [7, 8, 9], identifying a number of shortcomings in those methods. They observed that the existing work missed at least one of the four core research areas: (i) processing and life cycle, (ii) orchestration and management, (iii) use-cases analysis and (iv) an architecture framework for intent-driven networks. The authors then introduced an architectural framework that relied on closed loop feedback for intent processing and interpretation. However, they still didn't answer all the questions, so we take a step beyond the architecture to present other framework aspects that are crucial to realizing IDM systems.

The common outcomes from these surveys is a set of enablers and architectural features along with a list of open issues. Among the noted enablers [8] are SDN to allow for flexible configuration or enforcement of network-wide functionality or policies, and NFV to enhance the scalability, adaptability and abstraction. Others include the need for ZSM as the overarching requirement and the use of machine learning as a critical component in intent processing and automation. The proposed architecture is expected to be separated into application, network/service management, orchestration and resource layers. The open issues noted as requiring further work include intent description and translation, the description of services, processes etc. via data modeling languages and the integration of machine learning and NFV/SDN functionality.

2.2 Open source and standards development

Several open-source projects such as Open Network Foundation [12] have specified standardized intent-

based northbound interfaces for 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. [15] describes the intent-driven management architecture, its key elements and their interfaces.

Experiential Networked Intelligence (ENI)[16] examined design options for integrating intent operations into its system architecture. This includes accepting, translating and validating intent statements; determining how intents affects the system's goals and operations as well as their use by business users, application developers and network administrators. Another background study is the 3GPP Technical Report 28.812 [17] that describes the concepts of intent-based network management for service-based architecture (e.g., intent expression, intent translation, intent life cycle) covering various scenarios and 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].

TM Forum has proposed intent-based operation in IG1253 [19], introducing the intent management function as an architectural building block for intent-based operation which will receive an intent, make decisions about suitable actions towards fulfilling the intent, control the execution of these actions and report on the progress. Then, IG1253A [20] introduces the modeling concepts and artifacts that determine how an intent must be expressed, while IG1253C [21] defines the details of the intent life cycle including the related roles and tasks of the intent management function. Furthermore, [21] defines the interface through which the intent objects are exchanged, negotiated and managed as part of the life cycle management process. It however focuses on only the usage and management of intents within a single organization and so the proposed models and interfaces are not applicable for multi-vendor interactions.

Currently, ETSI ZSM011 [22] is investigating intent-based network and service management within the ZSM framework evaluating different ways to model intents and intent-based interfaces as well as their use ZSM management domains and ZSM end-to-end service management domains. The research topics and potential future directions related to ZSM011 are outlined by the original work [23] on Intelligent Intent Based Networks

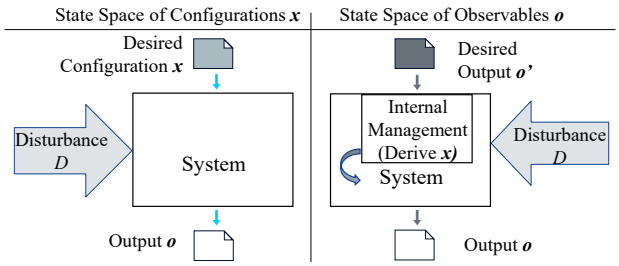

Fig. 1 – A system can be controlled by input on configurations or observations. In the latter, the system must internally translate the desired observations into concrete configurations [24].

(I^2BN). The I^2BN concept argues that intents are easier 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.

3. 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" [24]. On the 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 bound to the theoretical constraints on the observability and controllabilty of networks, which are many in mobile networks including among others the lack of a closed form description of how particular control inputs translate into the observed outcomes.

3.1 Intent as controlling input

Intent-based management means the capability to control a system based on specific controlling input, i.e., 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 of the two, as showed in Fig. 1 [24].

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 ongoing traffic. All these are input parameters that influence the observable output of the system.

The state space of observables comprises all the system's measurable properties. In a mobile network these include e.g. signal strength, throughput, or latency measured by UEs, all Performance Management (PM) data like

counters on network events, and all traces of the signaling interfaces. However, any system that takes desired observable outcomes as controlling input must be able to translate this input into concrete configuration parameters (see Fig. 1), because eventually at the 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.

Both kinds of controlling input apply at any level of abstraction as e.g. defined by the layers of the telecommunication management network: element, network, service, and business management [25]. At any level of abstraction raw measurements might be aggregated to Key Performance Indicators (KPIs) 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 [24]. This shows that intents can be submitted at any of the four management layers, from business to element management as well as on of the four controlling inputs, i.e configurations, measurables, KPIs and weighted metrics. For example, one may state a weighted-metrics, business intent to "only non-mobile support IIoT services". This is similar to a KPI, service-management intent to "not support mobility for IIoT network slices" or a KPI, network management intent to "ensure no Handover success".

3.2 Definition of intent

Based on the above the term *intent* is 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 observations. By following this definition, an intent specifies *what* the resulting state of the system shall be, but it does not give any hint 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. Instead, the intent-driven system to which the intent is set has to get itself into the desired state, by continuously observing its current state and accordingly reacting if the observed state deviates from the desired state. Inherently, an intent-driven system has to internally employ closed-loop feedback mechanisms which may not be known to the consumer of the IDM service.

3.3 Context to define the scope of an intent

To fully describe the state of a system requires not only its configuration parameter values and its observable output but also the description of the environment and its impacts on the system, both the partially and the fully unknown environment. To restrict the intent to

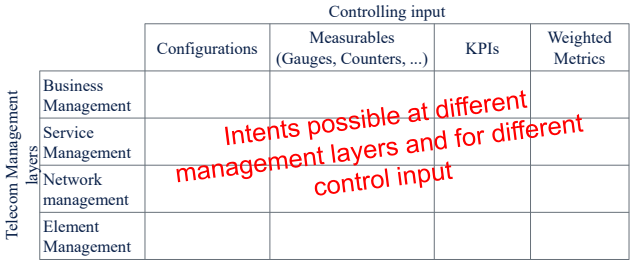

Fig. 2 – The two-dimensional continuum of control for networks [24]

specific network elements, areas, or time frames the intent may be augmented by additional configuration or observations state space information to define the *context* of the intent. In practice, it is impossible to fully specify the desired state of the system, instead an intent can only specify a combination of few, selected parameters and observations in its targets or context. One has to *assume* that the intent still is useful even with only a few dimensions (i.e. parameters and observations) of the overall state space, as most dimensions remain undefined.

Consider this very simple illustration of the underdetermination challenge: One may desire that "cells shall have a range of 1.5 km". In a heterogeneous RAN 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.

3.4 Context to define bordering conditions

Many network observables are not independent from each other, so achieving a specific value for a measurement or KPI 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 such, the selection of observables and their values that can be used as conflict free controlling input is not a trivial task. Instead, the capability of using an observable as controlling input implies a multi-dimensional optimization problem (see [26]). Therefore, the system has a basic requirement of enforcing the intent's targets, and a need to avoid unwanted side effects that are 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.

3.5 Consequences from the above definition

First, as "the desired state of the system", the term intent does not prescribe any particular level of abstraction in terms of a management layer or a control input, but is flexible to support intents in different contexts. It allows for intents that request very specific configuration details of a system to be exactly as stated, e.g. an intent A that a configuration parameter should be set to the value 5, which represents a very concrete intent. On the other extreme an intent may describe an observable property of the system e.g., like an intent B that "the mobile network shall cover 90% of a country with a datarate of 5 Mbps". Both intents describe a status, both do not impose any restriction or guidance how the intent gets implemented.

Secondly, the definition does not prescribe means of achieving the outcomes and does not guarantee them either. For intent A, 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, for intent B the operator may buy or lease equipment to build its own network or 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. In case of an unknown relation between a desired outcome and its 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 the desired goal into a very concrete setup that can fulfill the goal, a translation that might be impossible. The goal may even be unrealistic that it is even theoretically 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 may be impossible for the system to derive the right setup for the desired state. One reason could be that the system trans- fer function is unknown, i.e. there is no known relation between an observable outcome and the corresponding configuration of the system. 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.

Moreover, the definition does not negate that the different components of the intent may be in conflict with 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 requires different handling as discussed in Section 6.4. However, defining an intent as desired state of the system provides the advantage that it excludes dynamically conflicting intents: The intent is either one state or another state.

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.

4. MODELS AND SYSTEM ARCHITECTURE

Intents may be specified in different forms and degrees of complexity. This section presents an end-to-end IDM system architecture fit for supporting simple and complex intents in different specification forms. We discuss an intent's information elements and the related simple intent model, which we then evolve into a fully declarative model with multiple different information elements as well as an imperative model supporting the use of verbs. We use the two models to show the required components of the end-end system, including, the Intent Specification Platform (ISP) that exposes the IDM interface and the Intent Fulfillment System (IFS) to map the specified intent to appropriate network/resources and processes.

4.1 Intent as sentence of components

From the operator's perspective there can be very many kinds of intents, from simple intents achievable via one command on one object to complex intents that include several commands on several network objects and nodes. Example intents may include:

- Restrict handovers of fast-moving users to small cells.
- Allow load balancing to a cell Y 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. Then 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 cell border interference from neighbors is below -70dB.

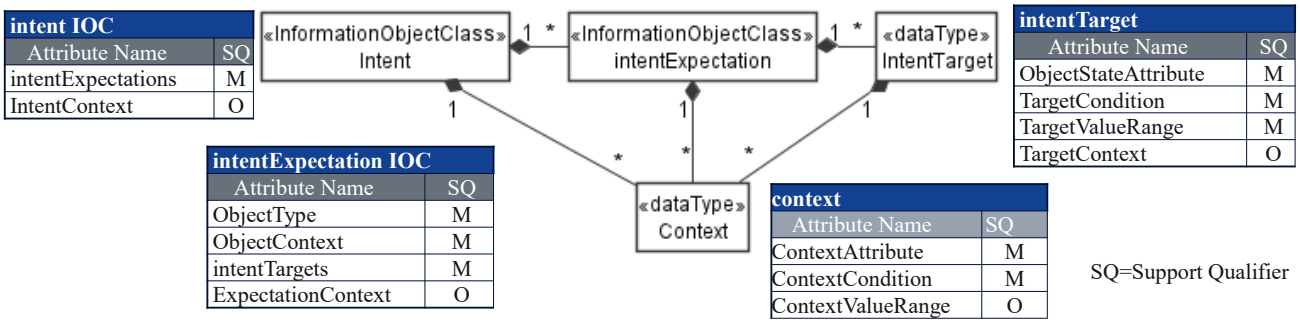

Fig. 3 – The declarative intent model - The intentExpectation is a list of IntentTargets on a type of object with context that scopes the objects and targets.

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, 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:

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

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. For example, for the intent to "rehome a BS", the state dimensions and values could be scope=BS, homeNB=B. Given this observation, one might consider the intent to be modeled as a simple list of these components, i.e., as:

$$intent := [\ scope(s), target1, target2, \dots \\ constraint1, constraint2, \dots] \qquad (2)$$

This in fact matches the idea that an intent defines a subspace (a 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.

4.2 Declarative intent model

The intent information model needs to include the information elements that identify its scope, requirements or goals and its constraints. The declarative model in Fig. 3 explicitly defines each intent as such using the intentExpectations, intentTargets and Contexts. The intentExpectation is the set of requirements, goals and constraints for one type of object and an intent may have one or more such sets e.g. one for cells and another for network slices. A desired outcome in an intentExpectation is the list of one or more intentTargets to be enforced by the IDM system. Each intentTarget is a triplet of the attribute on which the target is set, the condition constraining the outcome (e.g. "=", "<",">") and the desired valueRange. In this case, the attribute is any aspect of the control or observation state spaces of the system under control.

Each information element may have context that defines its scope and/or constraints. Context may be set for the IntentTarget and intentExpectation as IntentContext and ExpectationContext respectively. Also, the scope of the expectation can be made explicit using the ObjectContext that together with the object type define the specific applicable objects. Relatedly, a global intentContext may be added to apply to all intentExpectations within an intent.

4.3 Imperative intents: using verbs in intents

The declarative model described above (and 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 2, the declarative statement can be very complex to compile and requires a lot of prior information. Consider for example, that an operator wishes to request that no energy saving should be executed for a given cell or for cells in Central Business District (CBD) as in the example intents in Table 2. Stating this declaratively 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 affected by energy saving to formulate the desired state or values.

On the other hand, the imperative statement (as stated in the imperative column in Table 2) 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 consumer to state this intent using the verb, as opposed to determining the parameter "ES_On" and its possible values 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

Table 2 – Example specification of network management intents

	Goal (Imperative Intent)	Declarative Intent (Ensure that ..)	Imperative Intent Challenge
1.	Restrict Handovers of high mobility users to small cells	for Object [cell, Size=small] [HO_Allowed=True] Object [cell, Size≠ small], [HO_Allowed=False] if cellMobilitiy=high;	Interpret "Restrict" and "small cells" as "True" for small cells; "False" otherwise
2.	Restrict ES to Rural cells on week days	for Object [cell, Location =Rural] [ES_On=False; Time=Weekdays;]	Interpret "Rural" as filter to "create cell"; and "week days" as context for ES_On=False
3.	Avoid ES for cell Y / small cells / cells in CBD	for Object cell, Location=CBD), ES_On=False]	Interpret "CBD" as a filter on Object cellList"
4.	Rehome a nodeB from RNC A to RNC B.	for Object [cell, id= x]; [RNC is B]	Interpret "Rehome" into "cell attribute, RNC" "is equal to"
5.	Create an IIoT network slice	for Object [Slice, id="new"] [SliceProfile=IIoT]	Interpret "Create" into "id=new"

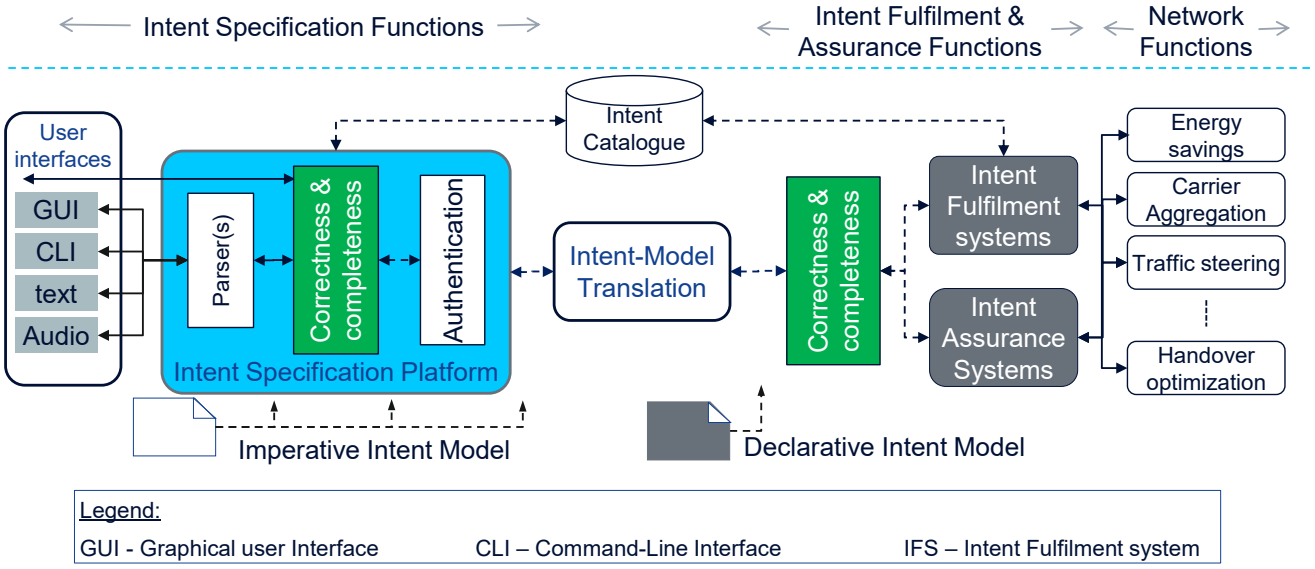

Fig. 4 – The end-to-end architecture of an intent-driven management system

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 2.

4.4 End-to-end architecture

The intent models as discussed above characterize the interface(s) over which the consumers and the producers of IDM service can interact to express and 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 proposed architecture, shown in Fig. 4, provides the means to capture intents, format them into realizable outcomes and execute their fulfillment. The intents may be submitted in a 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 speci ication platform to which he specifies the intent to be executed. The ISP may take in the operator's 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 2. 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 shown 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 depend on the nature ad contents of the intents. The next section dis- cusses some of the existing ideas on intent fulfillment.

5. 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 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.

5.1 Intent Logic Units and Library (ILU-ILL)

The ILU-ILL 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 Fig. 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 fulfill 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 Fig. 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 x specification

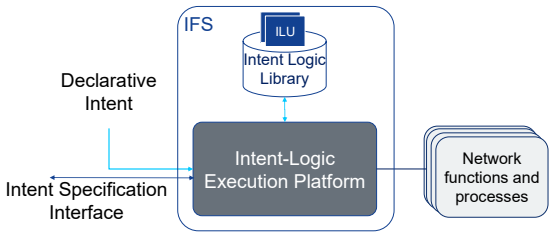

Fig. 5 – Intent fulfillment using Intent Logic Units (ILU) stored in an Intent Logic Library (ILL)

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, the 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 relearn the ILU for that intent (12).

5.2 Intent-driven orchestration of Cognitive Autonomous Network (CAN) functions

Another way of implementing the fulfilment of intents (especially operator intents) is by using an orchestrator, here called the Intent-Driven Network Automation Function Orchestrator (IDNAFO) [27], which orchestrates the execution of automation functions in a Cognitive Autonomous Network (CAN) [24]. In principle, as implementation for the intent fulfillment system block in Fig. 7, the IDNAFO takes an intent specified according to the intent model in Section 4 and generates configurations and controls for network automation functions. We summarize here the operation of its two core functionalities.

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 [28] and the objective of determining the optimal value of the control parameters and respectively the KPI.

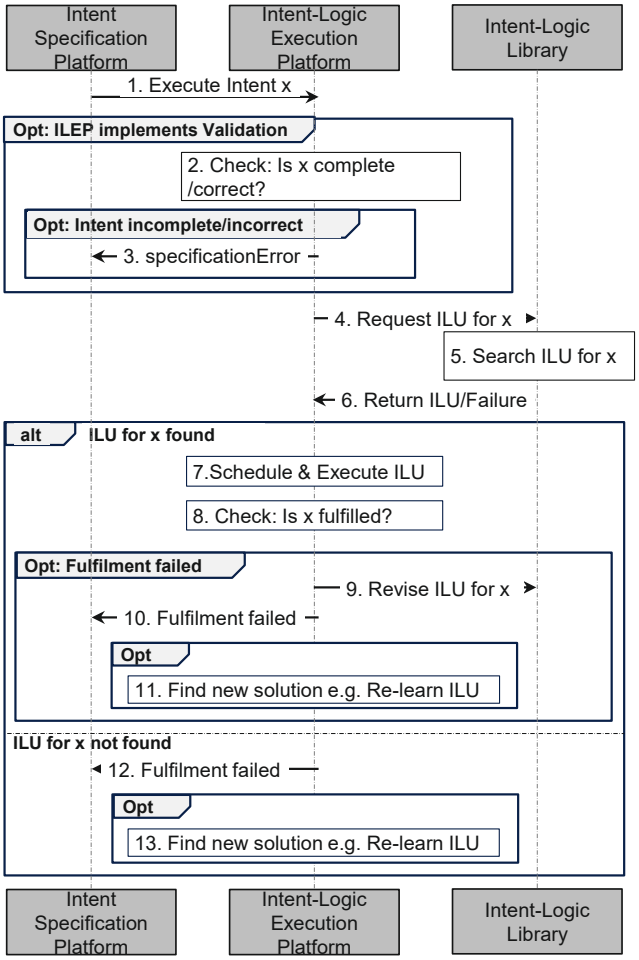

Fig. 6 – Implementing intent fulfillment via ILL and ILUs

Very often, multiple KPIs are influenced by a single control parameter, e.g., both Radio Link Failure (RLF) and downlink throughput change with change in Transmit Power (TXP). With each CF working 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. So, 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 a learning agent [29], learning the configuration (i.e. the control-parameter values) 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 parameter configuration that optimizes 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 all CFs [28, 29, 30].

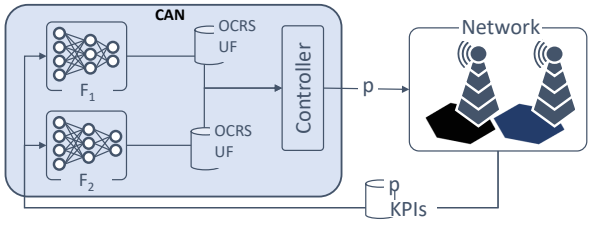

Fig. 7 – CAN abstraction with the CFs and the controller

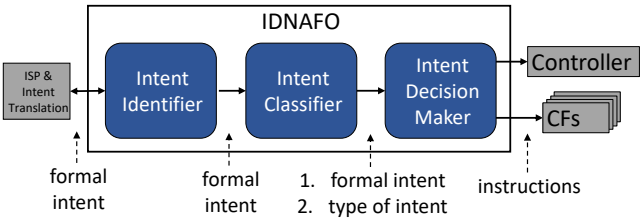

Fig. 8 – IDNAFO functionality blocks

Without any intricate details, we can abstract a CAN as shown in Fig. 7, where the controller lays between the CFs and the network. In this case, F_1 and F_2 are two CFs with a shared control parameter: p. Whenever F_1 or F_2 finds a new individually optimal configuration of p, it sends this value to the controller [28, 29], and requests the Controller to recalculate p as shown in Fig. 7. The Controller requests the preference of the other CF, 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 environment 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.

As shown in Fig. 8, the IDNAFO takes as input a formal intent and generates the actions for the Controller and CFs as its output. The formal intent may be structured according to either of the imperative or declarative models. The IDNAFO consists of three sequential steps related to three functional 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.

6. ADVANCED FEATURES FOR IDM

6.1 Tracking intents

Intents may be defined by human operators with different roles, levels of expertise and responsibility areas, e.g., business, operations or technical support experts 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.

6.2 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-to-end (e2e) service domain level, contextualization includes creation of cross-domain insight for efficient e2e optimization, information that allows accurate enough network and service state discovery modeling 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 for 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.

6.3 Intent feasibility checking

For coherent operation, after intents are syntactically and semantically validated, they need to be examined for their technical feasibility as a syntactically and semantically valid intent (e.g. that causes no conflicts) might still not be easy to fulfill due to missing network-side functionalities. Checking the technical feasibility of an intent requires the collection, discovery and up-to-date storage of network status, capabilities and configuration data for 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.

6.4 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. Therefore, 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 an *event→c ondition→a ction* policy principle one can point out and formalize conditions on the temporal overlaps be- tween *observation/measurement*, *action*, and *effect* time spans of the policies [31, 32, 33, 34, 35]. 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 [36, 37, 38]. 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 have observed the following four (non-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 the 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 4.1) 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.

7. CONCLUSION AND FUTURE DIRECTION

Intent-Based Networking (IBN) or Intent-Driven Management (IDM) is seen as a major step towards autonomous network management systems. 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. The main goal of this paper was to comprehensibly discuss the 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 2, we provided a highly-flexible, generic definition of intents in Section 3. 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, in Section 4, an end-to-end system architecture of IDM systems and a declarative intent model based on that definition. 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 5 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 trough 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 6. First, we argued for the necessity of tracking intents in such systems, where the intents' statuses and their life cycles 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 net- work 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 article's 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 support reception and execution of 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 not introduce new complexities but ensures to minimize them e.g though the use of imperative intents. The remaining open challenge is the 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 which we also expect to focus in our future work.

In all, 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 to- wards advanced intelligent networks.

REFERENCES

[1] Vu Anh Pham and Ahmed Karmouch. "Mobile Software Agents: an Overview". In: *IEEE Communications magazine* 36.7 (1998), pp. 26–37.

[2] Andrew T Campbell, Irene Katzela, Kazuho Miki, and John Vicente. "Open Signaling for ATM, Internet and Mobile Networks". In: *ACM SIGCOMM Computer Communication Review* 29.1 (1999), pp. 97–108.

[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". In: *Journal of Network and Systems Management 15.4 (2007), pp. 447–480.*

[5] Yiming Wei, Mugen Peng, and Yaqiong Liu. "Intent-based networks for 6G: Insights and challenges". In: *Digital Communications and Networks* 6.3 (2020), pp. 270–280.

[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". In: *IEEE Vehicular Technology Magazine* 14.3 (2019), pp. 28–41.

[7] Engin Zeydan and Yekta Turk. "Recent advances in intent-based networking: A survey". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020- Spring)*. IEEE. 2020, pp. 1–5.

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

[9] Yiming Wei, Mugen Peng, and Yaqiong Liu. "Intent-based networks for 6G: Insights and challenges". In: *Digital Communications and Networks* 6.3 (Aug. 2020), pp. 270–280. DOI: 10.1016/j.dcan.2020.07.001. URL: https://doi.org/10.1016/j.dcan.2020.07.001.

[10] Johan Silvander, Krzysztof Wnuk, and Mikael Svahnberg. "Systematic literature review on intent-driven systems". In: *IET Software* 14.4 (2020), pp. 345–357. DOI: https://doi.org/10.1049/iet-sen.2018.5338. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-sen.2018.5338. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-sen.2018.5338.

[11] Kashif Mehmood, Katina Kralevska, and David Palma. *Intent-driven Autonomous Network and Service Management in Future Networks: A Structured Literature Review*. 2021. arXiv: 2108.04560 [cs.NI].

[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". In: *ONF TR-523* (2016).

[13] Alexander Clemm, Laurent Ciavaglia, Lisandro Granville, and Jeff Tantsura. "Intent-based networking-concepts and definitions". In: *IRTF draft* (2020).

[14] Chen Li, O Havel, W Liu, A Olariu, Pedro Martinez- Julia, JC Nobre, and Lopez DR. "Intent Classification". In: *IETF draft* (2019).

[15] B Claise, J Quilbeuf, Y El Fathi, D Lopez, and D Voyer. "Service Assurance for Intent-based Networking Architecture". In: *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] ETSI. *Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects*. Not yet published. URL: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=61992.

[23] Szilágyi Péter. "I2BN: Intelligent Intent Based Networks". In: *Journal of ICT Standardization* 9.2 (2021), pp. 159–200.

[24] S. S. Mwanje and C. Mannweiler. *Towards Cognitive Autonomous Networks: Network Management Automation for 5G and Beyond*. John Wiley & Sons, 2020.

[25] ITU-T. "Principles for a telecommunications management network". In: *M.3010* (2000).

[26] J Strassner and J Halpern. "Intent-based Policy Management". In: *IETF 95, 2016. Proceedings (https://www.ietf.org/proceedings/95/).* IETF. 2016.

[27] 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.

[28] 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)*. IEEE. 2020, pp. 1–8.

[29] 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)*. IEEE. 2021, pp. 494–500.

[30] A. Banerjee, S. S. Mwanje, and G. Carle. *On the Necessity and Design of Coordination Mechanism for Cognitive Autonomous Networks*. arXiv:2001.07031. 2020. arXiv: 2001.07031 [cs.NI].

[31] T. Bandh. "Coordination of autonomic function execution in Self-Organizing Networks". PhD thesis. Technische Universität M nchen, 2013.

[32] T. Bandh and L. C. Schmelz. "Impact-time concept for SON-Function coordination". In: *2012 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE. 2012, pp. 16–20.

[33] 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*. IEEE. 2011, pp. 827–840.

[34] 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.* IEEE. 2003, pp. 98–109.

[35] A. Banerjee, S. S. Mwanje, and G. Carle. "Contradiction Management in Intent-driven Cognitive Autonomous RAN". In: *2022 IFIP Networking Conference (IFIP Networking)*. IEEE. 2022.

[36] 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)*. IEEE. 2014, pp. 1–6.

[37] J. Chomicki, J. Lobo, and S. Naqvi. "Conflict resolution using logic programming". In: *IEEE Transactions on Knowledge and Data Engineering* 15.1 (2003), pp. 244–249.

[38] E. Bertino, A. Mileo, and A. Provetti. "PDL with preferences". In: *Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POL- ICY'05)*. IEEE. 2005, pp. 213–222.

AUTHORS

Stephen S Mwanje is a research project manager in the "Network Automation" research team at Nokia Standards. He currently leads the team's research and standardization activities on the enablers for network automation. He received his Ph.D. (Dr. Ing.) degree with summa cum laude at the Ilmenau University of Technology, in Germany. His thesis on multi agent coordination of cognitive self-organizing network functions proposed the first approach towards the operation of multiple cognitive functions in cellular networks. He holds a BSc. Electrical Engineering from Makerere University in Uganda and MS Electrical Engineering from the University of Rochester in Rochester, NY, USA. He has more than 8 years of experience in mobile network operations, where he managed various projects on GSM/GPRS network planning, deployment, and optimization; microwave radio engineering

and spectrum/interference management as well as fiber optic network planning, deployment and operations. He is an inventor or co-inventor on more than 50 patent applications, a co authors on more than 30 peer reviewed publications and the main author of the book "Towards Cognitive Autonomous Networks: Network Management Automation for 5G and Beyond".

Anubhab Banerjee received the Bachelor of Engineering degree from the Department of Electronics and Telecommunication Engineering, Jadavpur University, India, in 2015, and the Master of Science degree from the Department of Electrical and Computer Engineering, Technical University of Munich (TUM). He is currently pursuing the Doctoral degree with the Department of Informatics, TUM, funded by Nokia Bell Labs, Munich. He is an inventor or co-inventor on more than 10 patent applications, and he published several peer-reviewed articles in several prestigious conferences and journals, including IEEE TNSM, IEEE IM, IEEE CNSM, and Computer Networks (Elsevier). His primary research interests include game theory, machine learning and network and service management.

Juergen Goerge is a research project manager in the "Network Automation" research team at Nokia Standards. He received his Ph.D. Physics at the Research Center Jü lich of the University Bonn. He has more than 20 year experience in network management Systems Engineering in Nokia designing CM tools for RAN management of all RAN features, from northbound interfaces down to the network elements, and participating in design of all SON LTE features (plug-n-play, ANR, cell outage compensation, load balancing, energy saving). Prior to that, he had worked in research on radar tracking at the Research Center for Applied Science, Wachtberg and on modeling highway traffic at the Gesellschaft fü r Verkehrsdaten in Dü sseldorf Germany. He is a holder of several patents and patent applications related to SON and has published several peer-reviewed articles in several prestigious con- ferences and journals. He was also a major contributor to the two books on network management automation "LTE Self Organising Networks (SON): Network Management Automation for Operational Efficiency" published in 2012 and "Towards Cognitive Autonomous Networks: Network Management Automation for 5G and Beyond" published in 2020.

Abdelrahman Abdelkader is a senior research and standardization engineer at Nokia in Munich, Germany. Before that, he worked as a research associate at the Technical University of Dresden (TUD) and as an early stage researcher at the 5Gwire-less ITN project, a Marie Curie project funded by the European Commission. His research focuses on traditional and ML-based optimization in wireless networks including channel modelling, system design, resource allocation problems and network coding, as well as Device to Device communications and Internet of things applications. He has extensive experience with programming languages such as Matlab and C++/C#, as well as functionality in python and assembly. Apart from the research experience, he also has intensive knowledge of the telecommunications market, standardization ecosystem and patenting.

Gabor Hannak received his M.Sc. and Ph.D. degrees in electrical engineering and information technology in 2013, SCL and 2017, SCL respectively from the Vienna University of Technology, Austria, with major in telecommunications. In 2017, he was a visiting research scholar at the University of Edinburgh, UK. Since 2019, he has been a research engineer with Nokia Bell Labs. His activities include include 5G RAN standardization and research in machine learning driven network management features such as massive MIMO and energy saving techniques.

Péter Szilágyi received an M.Sc. degree in Software Engineering from the Budapest University of Technology and Economics (BME), Hungary, in 2009. He is a senior research engineer and DMTS at Nokia Bell Labs. His research history includes telecommunications and software, artificial intelligence, data analytics, 5G and verticals, connected vehicles, intent based management and service automation, end-to-end system engineering and rapid prototyping. He has more than 40 patents and 25 publications in book chapters, journals and conferences.

Tejas Subramanya is a senior research engineer at Nokia Standards in Munich, Germany. His current research interests include cognitive management of future mobile networks and applying machine learning to network automation use cases. He has over 8 years of experience in mobile networks related research and development in different roles in India, Finland, Italy, and Germany. He is a (co-)author of several articles and papers on network management automation for next-generation mobile networks. Tejas received his MS degree in radio communications from the Aalto University in Finland and his PhD on cognitive network management and orchestration from the University of Trento in Italy.

Julian Goser received his M.Sc. and a BSc. both in computer science from the University of Augsburg, in Augsburg Germany.

Tobias Foth is doctoral researcher with dual assignment at Nokia and at the University of Augsburg. He received his M.Sc. and a BSc. both in computer science from the University of Augsburg, in Augsburg Germany.