SLA Management in Intent-Driven Service Management Systems: A Taxonomy and Future Directions

YOGESH SHARMA[*], Thompson Rivers University, Canada
DEVAL BHAMARE, University of Surrey, United Kingdom
NISHANTH SASTRY, University of Surrey, United Kingdom
BAHMAN JAVADI, Western Sydney University, Australia
RAJKUMAR BUYYA, The University of Melbourne, Australia

Traditionally, network and system administrators are responsible for designing, configuring, and resolving the Internet service requests. Human-driven system configuration and management are proving unsatisfactory due to the recent interest in time-sensitive applications with stringent quality of service (QoS). Aiming to transition from the traditional human-driven to zero-touch service management in the field of networks and computing, intent-driven service management (IDSM) has been proposed as a response to stringent quality of service requirements. In IDSM, users express their service requirements in a declarative manner as *intents*. IDSM, with the help of closed control-loop operations, perform configurations and deployments, autonomously to meet service request requirements. The result is a faster deployment of Internet services and reduction in configuration errors caused by manual operations, which in turn reduces the service-level agreement (SLA) violations. In the early stages of development, IDSM systems require attention from industry as well as academia. In an attempt to fill the gaps in current research, we conducted a systematic literature review of SLA management in IDSM systems. As an outcome, we have identified four IDSM intent management activities and proposed a taxonomy for each activity. Analysis of all studies and future research directions, are presented in the conclusions.

Additional Key Words and Phrases: Intent-driven service management, Intent processing, Service level agreements, Cloud computing, Networks, Zero-touch service management

1 INTRODUCTION

Emergence of 5G from nascency to a new global wireless standard is making significant improvements in the current Internet services, such as, mobile broadband. It is also empowering the development, deployment and delivery of new services, for example, smart factories, logistics, remote surgery, precision agriculture and many other applications with low latency requirements. By supporting wide range of applications across various verticals, such as academia, medicine, industry and agriculture; 5G will be driving the global growth and has been predicted to have $13.1 Trillion of global economic output by 2035 [12]. 5G led rapid growth, development and innovation in the networking world will increase the data traffic with many folds. According to Ericsson, the global data traffic alone from mobile devices will reach to 226EB (exabyte) per month by 2026. 5G will lead this growth by accounting for 54% of the total data traffic [39]. To capitalize on such a demand, communication service providers and network service providers must offer services that can cope with such an increase in data generation and consumption. This compels them to expand and

[*]This is the corresponding author

Authors' addresses: Yogesh Sharma, ysharma@tru.ca, Thompson Rivers University, British Columbia, Canada; Deval Bhamare, d.bhamare@surrey.ac.uk, University of Surrey, United Kingdom; Nishanth Sastry, n.sastry@surrey.ac.uk, University of Surrey, United Kingdom; Bahman Javadi, b.javadi@westernsydney.edu.au, Western Sydney University, Australia; Rajkumar Buyya, rbuyya@unimelb.edu.au, The University of Melbourne, Australia.

modernize their methods to deploy and operate networks and services by adopting multi-domain, elastic and scalable solutions characterizing clouds, such as, network function virtualization (NFV) [71] and software defined networks (SDN) [49]. SDN and NFV have had many benefits to simplify network services and management, but all innovation is happening at the deployment level. Consequently, service design and implementation are still human-driven, with system/network architects or engineers interpreting service requirements and implementing them. This is termed as a *"person+process"* approach, which is imperative or prescriptive in nature where the system is required to be told *'how'* to realize the service request [94]. However, the increasing demand of applications with stringent quality of service (QoS) requirements (high availability, throughput, security and low latency) call for human-free service deployment to achieve desired results. It is, therefore, imperative that human intervention be replaced with an autonomous approach to managing service life-cycle.

Driven by such requirements and challenges, Intent-driven service management (IDSM) has been proposed with a goal of transition from traditional policy-based *"person+process"* operations model to zero-touch autonomous model [94]. With intent-driven interactions, users/service-providers express their service expectations and business objectives in a declarative manner without expressing *'how'* they should be achieved. Hence, an intent is defined as a *declarative expression* describing *what a user desires to achieve* instead of *how it should be achieved.* Once an intent is specified, closed control-loop operations of the IDSM system will work in an autonomous manner to meet the requirements of a service request. IDSM makes the deployment of services faster and reduces the errors and misconfigurations caused by manual operations significantly. According to Gartner, IDSM can reduce the network service delivery time by 50% to 90% and can reduce the number and duration of outages by at least 50% [52]. However, the enablement of IDSM systems need complex and multi-layered arrangement including intent-handlers (IH), service orchestrators and controllers running on multiple autonomous domains ranging from the edge-cloud providers, communication service providers (CSP) and hyper-scale cloud providers (HSP). All these components need to interact, coordinate and work together in a closed loop manner towards the fulfillment of intents (Section 2.2). Since IDSM systems are in their infancy, there is limited knowledge about their operations and activities, raising concerns about their reliability and performance variability, which could compromise service level agreements (SLAs) [1]. Therefore, it is imperative to have a deep understanding of the activities an IDSM system performs in order to meet the SLA requirements and to fulfill the intents.

This study is an attempt to provide a systematic landscape of SLA-based research in IDSM systems to understand the state of the art and open challenges. It provides an insight for devising solutions that address the fundamental problems in SLA management in IDSM systems. This survey is carried out while following the guidelines for performing systematic literature reviews proposed by Kitchenham et al. [47]. The main contributions of the paper are as follows:

- Categorization of activities the IDSM system performs to fulfill the intents.
- A comprehensive taxonomy for SLA management in IDSM systems.
- A broad review to explore various existing methods and techniques for SLA management in IDSM systems.
- Comparison and categorization of the existing techniques.
- Identification of research gaps and open challenges in the domain of SLA management in IDSM systems based on the key observations derived from the taxonomy and survey results.

The rest of the article is organized as follows. Section 2 presents the background covering the evolution, architecture and activities of IDSM systems. Section 3 describes the motivation

[1]SLA is an agreement between service provider and consumers regarding QoS expectations and associated reward, if met.

Table 1. List of abbreviations used in the study

Abbreviation	Full-form
IDSM	Intent-driven service management
CSP	Communication service provider
NSP	Network service provider
NFV	Network function virtualization
SDN	Software defined network
QoS	Quality of service
AI	Artificial intelligence
O&M	Operation and management
ECP	Edge cloud provider
HCP	Hyper-scale cloud provider
IoT	Internet of things
VR	Virtual reality
TCO	Total cost of ownership
CPEX	Capital expenditure
OPEX	Operating expenditure
SHV	Standard high volume
$I - NBI$	Intent-northbound interface
IBNS	Intent based networking systems
RMSO	Resource managers and service orchestrators
IDN	Intent-driven network
KPI	Key performance indicator
ACL	Access control list
PNF	Physical network function
ML	Machine learning
VM	Virtual machine
QoE	Quality of experience

behind the review and provides the comparison with existing reviews on IDSM. In Section 4, we discuss the research methodology followed to conduct the review and quantitative outcomes of the methodology. Section 5 presents the results of the review covering taxonomies and analysis of the research articles. Section 6, provides the critical analysis, key observations and future research directions in the area of interest. Finally, conclusions are drawn in Section 7.

2 BACKGROUND

With service requirements or By defining service level agreements (SLAs) as intents, intent-driven service management (IDSM) systems meet these requirements autonomously. This is accomplished by taking decisions about service design, configuration, optimization, and remediation with little or no human involvement. Because of such self-driving and self-organizing properties, IDSM systems have garnered attention recently in the fields of networking [48] and cloud computing [80]. However, research and development (R&D) for IDSM systems is in its preliminary stages and faces many challenges. This makes the IDSM systems a lucrative area for both academic and industrial researchers to work on. To facilitate the R&D efforts in the topic of interest, this section provides the information about the background of IDSM systems covering their evolution, architecture and main activities performed by the IDSM systems for intent management.

2.1 Evolution of Intent-Driven Service Management Systems

Figure 1 shows the evolution summary of IDSM systems. The steady increase in the adoption of cloud computing [20], has increased the operational and administrative complexity of computing and

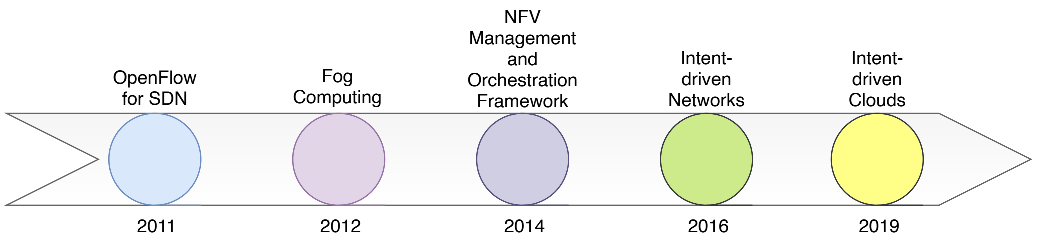


Fig. 1. Evolution of Intent-Driven Service Management systems representing the technologies and architectures that led the way to intent-driven networks followed by recent intent-driven clouds.

networking infrastructure hosting cloud services. For computing infrastructure, the complexity was dealt with significant advancements done in the field of virtualization. However, the advancement of network infrastructure (routers and switches) connecting thousands of servers hosting cloud services lags far behind. This has motivated the researchers and engineers to innovate towards the softwarization of networks. With Stanford's Ethane project, efforts began in 2007 to decouple the data plane and control plane [14]. Using a centralized controller, Ethane enabled the configuration of switches and defined routing flows which led to Software Defined Networks (SDN) [49]. In 2011, OpenFlow was developed, which is a widely accepted protocol for SDNs, thereby simplifying computer networks even further [57].

SDNs and the evolution of cloud computing systems into multi-cloud/inter-cloud environments with mature interoperability [95] enabled efforts to bring computing power closer to end users. This also supported new breed of applications with low latency, real-time processing and high mobility requirements. In 2012, Cisco introduced the fog computing paradigm [10]. Fog computing components act as an intermediate layer providing compute, storage and networking services between the end user and cloud computing infrastructure. Such hierarchical arrangement facilitates the real-time interaction, mobility support, interoperability and scalability between end user applications and back-end cloud infrastructure [54]. These paradigms (others are edge computing [13, 83] and serverless computing [6]) are therefore appropriate for applications that require data intensive operations as well as different processing requirements, such as, Internet of Things (IoT) [44] and Virtual Reality (VR) [101].

Networks are required to expand frequently by adding multi-specialized proprietary networking equipment in order to support high data volume and perform data-intensive operations. Consequently, total cost of ownership (TCO) increases in terms of capital and operating expenditures (CPEX and OPEX). In 2013, European Telecommunications Standards Institute (ETSI) started experimenting with the concept of virtualizing the network equipment as a way of taking softwarization of networks to a whole new level and reducing or eliminating the need for expensive devices [31]. In response, the concept of Virtualized Network Functions (VNFs) was introduced with networking software (control plane and data plane) hosted in VMs or containers running on Standard High Volume (SHV) servers. ETSI released its NFV Management and Orchestration framework in 2014 to provide guidelines for the deployment of VNFs to improve interoperability [71]. In the end, a decade of innovation, advances in virtualization and softwarization of computing and networking components and paradigms leading hierarchical deployment made it a lucrative arrangement for telecommunications. The Open Networking Foundation defined an Intent-Northbound Interface (I-NBI) and initiated the emergence of intent-based networking systems enabling the autonomous deployment and management of telco-grade applications without human intervention [38]. Following the networks, the concept of intents is adopted in the field of cloud computing system when Ericsson published an article on intent-aware cloud computing systems in 2019 [80].

2.2 Intent-driven Service Management System Architecture

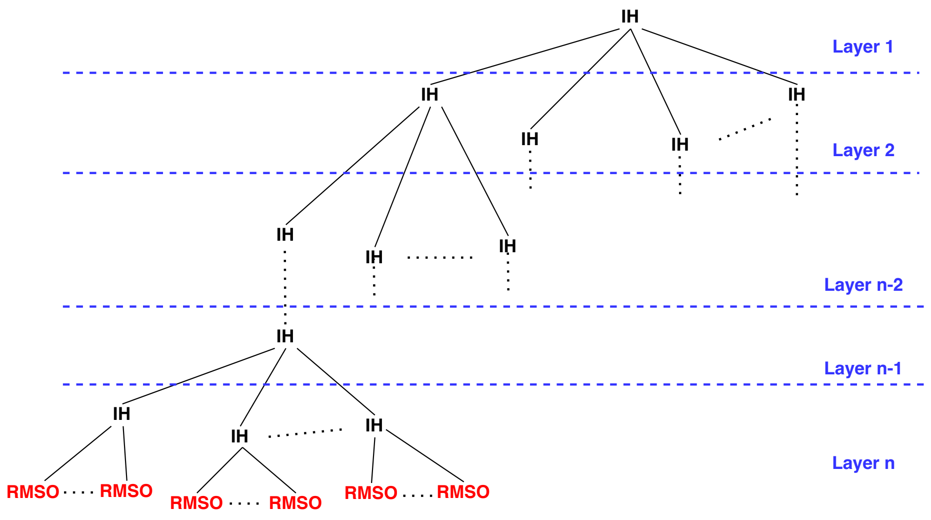


Fig. 2. Layered assembly of Intent-driven Service Management System representing hierarchical arrangement of intent-handlers (IH) and resource managers and service orchestrators (RMSO)

Figure 2 represents an abstract assembly of an intent-driven service management (IDSM) system. IH stands for intent-handler and RMSO stands for resource manager and service orchestrator. IH is an important component of IDSM system. It is defined as *"a function which receives the intent, takes decision if and how to act, dispatches operational actions and report progress back to the source of the intent."* The IDSM system is built by assembling the IHs in a tree-like hierarchical structure sharing parent-child relationship with each other. IHs at different levels are divided into operational layers to represent the diversity of user types and roles. There can be *n* number of operational layers and each layer can have one or more IHs. RMSO represents the domain/sub-system responsible for providing virtual and physical resources to fulfill the intents. An IH can have IHs and/or RMSO as children.

Based on the arrangement shown in Figure 2, a reference architecture of multi-layered IDSM system is shown in Figure 3. The architecture consists of 3 operational layers i.e. business, service and infrastructure. Infrastructure layer consists of three self-governing domains of edge, communication service provider (CSP) and cloud. Each layer and domain has an IH.

(1) Business layer IH handles the business-intents representing the functional requirements of a business user for example, delivery of an application with customized features as defined in SLA.
(2) Service layer IH handles the intents representing the objectives of service user or provider to support business intents. The service layer intents can have more specific non-functional requirements such as latency, bandwidth and availability.
(3) Infrastructure layer IH (domain specific IHs) handles the intents of resource users or providers. They interact with RMSO to provision and allocate resources to the service request specified as intents.

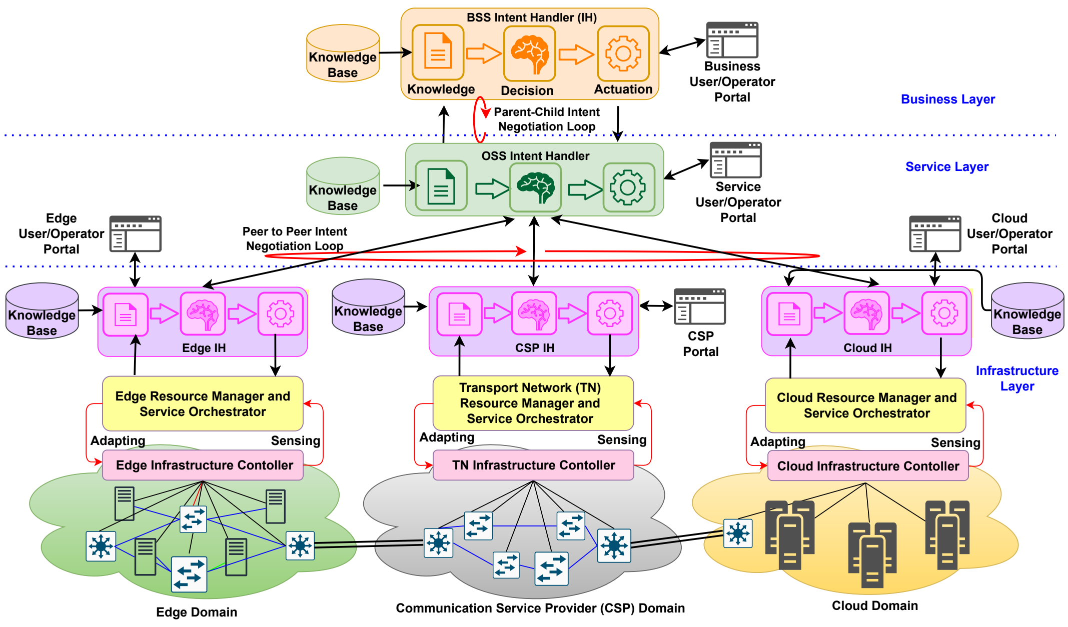

BSS Intent Handler (IH)
Knowledge Base
Knowledge
Decision
Actuation
Business User/Operator Portal
Business Layer
Parent-Child Intent Negotiation Loop
OSS Intent Handler
Knowledge Base
Service User/Operator Portal
Service Layer
Edge User/Operator Portal
Peer to Peer Intent Negotiation Loop
Cloud User/Operator Portal
Knowledge Base
Edge IH
Knowledge Base
CSP IH
CSP Portal
Cloud IH
Knowledge Base
Infrastructure Layer
Edge Resource Manager and Service Orchestrator
Adapting
Sensing
Edge Infrastructure Contoller
Transport Network (TN) Resource Manager and Service Orchestrator
Adapting
Sensing
TN Infrastructure Contoller
Cloud Resource Manager and Service Orchestrator
Adapting
Sensing
Cloud Infrastructure Contoller
Edge Domain
Communication Service Provider (CSP) Domain
Cloud Domain

Fig. 3. Multi-layered IDSM system architecture consisting layered arrangement of intent-handlers, control loops and autonomous domains of edge, CSP and cloud.

IH drives the knowledge about the intent processing operations (Section 5.1.3) from the associated knowledge-base and/or other IHs and users. Knowledge-base stores the data representing human's experience and judgment skills. ML and AI enabled IHs use the information from the knowledge-base to drive their intelligence to perform complex decision-making required to design, deploy and maintain the service management operations to fulfill the intents. IHs of different layers interact with each other and with RMSOs of various domains/sub-systems by using intent-driven interfaces i.e. intent APIs in a closed loop manner. Like operational layers, there can be more or less than 3 domains and each domain can have multiple sub-systems owned by single or multiple service providers. Each sub-system will have an associated RMSO and infrastructure controller. Intents can be originated either directly from the user input through portals or from other IHs in the hierarchy.

Upon receiving an intent, IH performs a preliminary assessment by checking its ability to fulfill the intent by using its knowledge base. If not, intent is rejected and intent-negotiation (Section 5.1.3) is started by proposing alternative intents to the intent specification entity. If yes, IH defines the goals for its child IHs by decomposing the received intent into sub-intents. With each decomposition by IHs in the hierarchy, an intent gets enriched with the service design and resource configuration parameters required for the service deployment. The cycle of intent-decomposition keeps repeating in a top-down manner until decomposed intents reach IHs local to RMSOs of the required domains/sub-systems (IHs at the layer n in Figure 2). Upon receiving the request, the respective RMSO checks the availability of the required resources by probing the corresponding infrastructure controller. If the required resources are available, resource configuration parameters are forwarded to the infrastructure controller for service deployment (Section 5.2). The fulfillment of the intent is ensured throughout its lifetime in a closed-loop manner by performing continuous monitoring (Section 5.3) and remediation (Section 5.4).

On the contrary, if enough resources are not available, RMSO shares the information about the available resources with the local IH. By using the information, IH composes the alternate

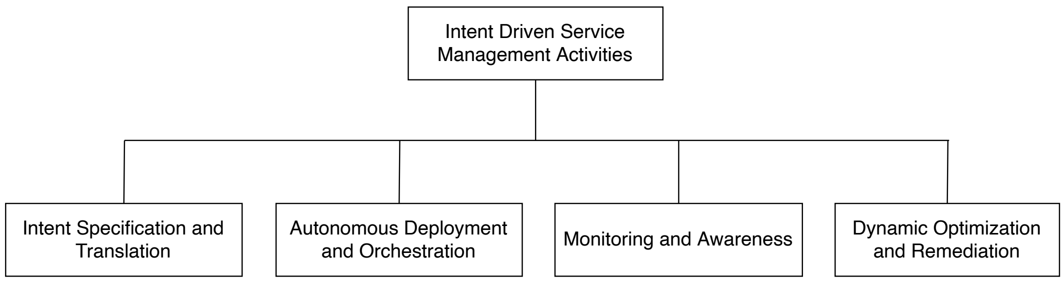

Intent Driven Service Management Activities
Intent Specification and Translation
Autonomous Deployment and Orchestration
Monitoring and Awareness
Dynamic Optimization and Remediation

Fig. 4. Activities the intent-driven service management system performs to fulfill an intent.

intents with changed or degraded service requirements. The alternate intents are used to initiate the intent-negotiation either with the parent IH or with peer IHs. If the negotiation is successful and an alternate intent is accepted then service is deployed. Alternatively, the current IH pushes the alternate intents to its parent IH in the hierarchy. The parent IH again performs the intent-composition and negotiation with its parent and peer IHs to decide about the acceptance or rejection of alternate intents. This process of intent-composition and negotiation keeps repeating in bottom-up manner until either an alternate intent is accepted or the IH where the intent was specified at fist place is reached. This is where the final decision on intent rejection or acceptance takes place and user is notified and/or asked to re-specify the intent. Together all these inter-connected components of multiple layers provide an autonomous, optimal and reliable service delivery and management at a scale and velocity which in not achievable in traditional human-driven service management systems.

2.3 Activities for Intent Management

In intent-driven service management (IDSM) systems, a user/tenant specifies the intents and system adapts and changes by itself to achieve the desired results without human intervention. The journey from defining an intent to its fulfillment involves four activities that IDSM systems perform to satisfy the intent owner's service requirements (Figure 4) [103]. In this section, we are defining these activities in brief. However, all these activities are explored in depth in Section 5.

(1) *Intent Specification and Translation:* The IDSM system accepts service requirements from users specified with high-level of abstraction as intents and convert them into system design and configuration instructions with the help of an intent handler.
(2) *Autonomous Deployment and Orchestration:* Resource managers and service orchestrators (RMSO) accept the service design and configuration instructions generated by the intent handlers. The required changes are performed autonomically across the software/hardware resources of multiple domains/sub-systems to fulfill the intents.
(3) *Monitoring and Awareness:* The goal of monitoring and awareness activity is to measure the satisfaction level of intents. During this activity, the telemetry data is collected to evaluate the current state of the system and correlate it with the desired state of the system to identify any performance deviation and anomaly that can impact the fulfillment of an intent.
(4) *Dynamic Optimization and Remediation:* In case of a performance deviation or anomaly identified during monitoring and awareness activity, IDSM system takes the corrective actions by performing internal service and resource optimizations and re-configurations to safeguard the fulfillment of intents or by notifying the end-users about its inability to fulfill the intents.

Ideally, IDSM systems are required to be able to perform all these four activities. However, during this survey, specific solutions are seen addressing fewer activities and still be the part of an IDSM solution (Section 5). In the next section we discuss the motivation behind this review work.

3 MOTIVATION BEHIND THE REVIEW

It has been observed that there are very few detailed surveys of intent-driven service management (IDSM) systems available in the literature. Table 2, summarizes the existing important survey works on the related topic and compares them with our survey.

Table 2. Comparison of Available Surveys in Intent-driven Service Management with this Survey

Authors	Systematic Review	Evolution & Origin of IDSM	Activity Distribution of IDSM	Taxonomy for IDSM	Comparative Analysis of IDSM Solutions	Key Observations & Challenges
Zeydan et al.[108]		★				★
Pang et al.[70]		✓				★
Wei et al.[103]			✓			★
Mehmood et al.[59]	✓	✓			✓	★
This Survey	✓	✓	✓	✓	✓	✓

Note: ✓denotes the broad discussion on the respective issue.
Note: ★ denotes the partial discussion on the respective issue.

The existing surveys are not systematic reviews except Mehmood et al. [59] and are performed in an ad-hoc manner. Existing intent based surveys are limited to the networking field, i.e., intent-driven networks (IDN). These surveys do not discuss the activities that must be performed during the lifetime of an intent and do not provide a taxonomy classifying the methods and solutions for IDSM. Moreover, none of the existing surveys present a critical analysis of the existing IDSM solutions and highlight their limitations. The need of addressing these shortcomings motivated us to conduct a systematic review presented in this article. Besides constructing the taxonomies and comparing the existing IDSM solutions, we performed a critical analysis of the existing literature and made a few key observations. This results in the identification of research gaps and provides the future directions to the researchers working to improve the IDSM systems.

The following section presents the details of the research methodology used to carry out this systematic review. The research methodology is based on the guidelines provided by Kitchenham et al. [47] to conduct systematic surveys.

4 RESEARCH METHODOLOGY

This systematic study is performed following a multi-stage research methodology, including the selection of search keywords to retrieve information from various online venues, formation of review methodology and analysis; and management of retrieved information by using review methodology. This section gives the information about all the components of the multi-stage research methodology and its outcomes.

4.1 Research Questions

The main goal of this systematic review is to understand the current research and development trends focusing on SLA management in IDSM systems and to identify the open challenges and research gaps in the existing research. Table 3 presents a list of IDSM activity wise (Figure 4) research questions drafted to drive this review.

4.2 Sources of Information

In order to identify the articles on the topic of interest, electronic database search using different search keywords (Table 4) is performed. Various research articles and reports are retrieved from the

Table 3. Activity wise research questions answered in this systematic review.

Activity	Research Questions
Intent Specification and Translation	1. What are the different types of intents? 2. What are different languages to express or define the intents? 3. What are different intent stakeholders? 4. What are various attributes an intent can have? 5. What are various steps and methods/techniques to process an intent into a system adaptable form?
Autonomous Deployment and Orchestration	1. What are the service level agreement (SLA) parameters of interest to intent stakeholders? 2. What are various SLA-based network and resource provisioning and allocation techniques used to realize the translated intents?
Monitoring and Awareness	1. What are various performance challenges or bottlenecks that can breach the constraints of intents? 2. What are the available methods to monitor the compliance of intents? 3. What are various Key Performance Indicators (KPIs) used by performance monitoring methods? 4. What are the available methods to predict the dynamics of performance changes across the multiple layers of intent-driven service management (IDSM) systems?
Dynamic Optimization and Remediation	1. What are various intention guarantee management methods? 2. What are the available system optimization and refinement methods require to safeguard the fulfillment of intents against any anomaly detected or predicted during monitoring and awareness activity?

different venues, such as, conferences, journals, master and PhD thesis, magazines and white papers (technical reports as well as industry research work). Following is the list of searched electronic databases.

- IEEE Xplore - https://ieeexplore.ieee.org/Xplore/home.jsp
- ACM Digital Library - https://dl.acm.org/
- ScienceDirect - https://www.sciencedirect.com/
- Wiley Online Library - https://onlinelibrary.wiley.com/
- Springer - https://link.springer.com/
- Taylor & Francis Online - https://www.tandfonline.com/
- Google Scholar - https://scholar.google.com/
- Tmforum - https://www.tmforum.org/

4.3 Search Criteria

Table 4 describes the search keywords used to retrieve the research articles from different e-resources as discussed above. The keyword "intent" is included in almost all the searches and found in the abstract of every searched article. We performed a careful database search to ensure the completeness of our study. Even so we couldn't get some of the research works during the predefined search method. This is due to the non-availability of search keywords in the abstract because of the synonyms being used. We retrieved some of those missed research articles by using the references of the identified papers (snowball technique). Articles published from 2016 to 2021 are considered in this review.

4.4 Inclusion and Exclusion Criteria

Figure 5 shows the multi-stage review methodology representing inclusion and exclusion criteria used in this systematic review. By using the search keywords, in total we had obtained 4900 research

Table 4. Various search keywords, period and venue types used to retrieve research articles for the review.

Search Keywords	**Period**	**Venue Type**
Intent based systems		
Intent driven/based networks (IDN)		
Intent driven/based clouds/cloud computing		
Intent Specification		Conferences
Intent Decomposition		Journals
NorthBound Interface (NBI)	2016-2021	Technical and Industrial Reports
Intent North Bound Interface (I-NBI)		White Papers
Intent Deployment in Networks/Clouds		Master and Ph.D. Thesis
Intent Orchestration in Networks/Clouds		
Intent Monitoring in Networks/Clouds		
Intent Optimisation in Networks/Clouds		

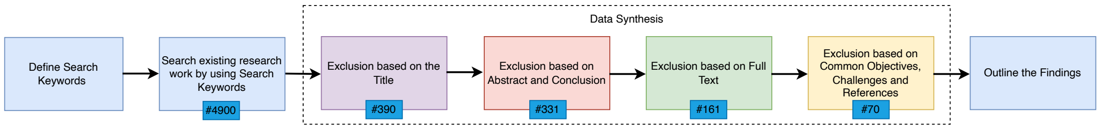

Data Synthesis
Define Search Keywords
Search existing research work by using Search Keywords
#4900
Exclusion based on the Title
#390
Exclusion based on Abstract and Conclusion
#331
Exclusion based on Full Text
#161
Exclusion based on Common Objectives, Challenges and References
#70
Outline the Findings

Fig. 5. Review methodology representing different stages to carry out the systematic review

articles from digital libraries. In the first stage of data synthesis, the irrelevant articles were excluded if word 'intent' was not present in the titles. As a results, 390 research articles were obtained on which the second stage exclusion process was performed by using their abstracts and conclusions. In the second stage, the articles were marked as eligible for inclusion in the review only if their focus of study was intent-driven service management (IDSM) systems. In the literature, voice command systems such as Apple's Siri, Amazon's Alexa and Google are also termed as intent-driven systems. Articles related to such systems were excluded and the exclusion results in 331 articles. After performing a thorough study and analysis of the full text of the 331 remaining articles w.r.t the research questions (Table 3), in the third stage, we obtained 161 articles of interest. These articles are further filtered to 70 in the fourth exclusion stage on the basis of their overlaps and common objectives (found in the papers from the same research group). Following the rigorous analysis of 70 articles, findings are summarized as taxonomies and tables; and presented in Section 5 and Section 6 of this paper.

4.5 Data Extraction

Table 5 displays the guidelines for data extraction from all the 70 research articles included in this systematic review. The data extraction guidelines were designed when we had started the information gathering and analysis which was sufficient to address the drafted research questions.

While conducting systematic review, problems were faced regarding the extraction of suitable data, in case, information is missing or not clearly available in the article. In order to get the clarification about the missing information, we contacted the authors of the respective research article. While extracting the data, all the authors of this study communicated and held meetings regularly and performed in-depth analysis of the research works as described below

- After in-depth review, first author extracted and analyzed the data from 70 research papers.
- Other authors cross-checked the review results to check the consistency of the extracted data.
- Conflicts occurred during cross-checking were resolved during the meetings.

Table 5. Data extraction guidelines representing data items extracted from all research articles.

Data Item	Description
Bibliographic Information	Author, year, title, source of the article
Type of the article	Journal, conference, thesis, symposium, technical report
Study Classification	Type of article research article or survey paper, targeted domain, publication institution
Study Context	What are research focus and aims of the work?
What are intent-driven service management systems?	It explicitly refers to activities of intent-management systems and their attributes.
Critical Analysis	This refers to the identification of strengths and weaknesses of each research work.
Study Findings	Major finds or conclusions drawn from the primary study.
Evaluation Method	How does one evaluated the proposed methods, i.e., simulator study and/or proof-of-concept?

4.6 Quantitative Analysis of Research Methodology

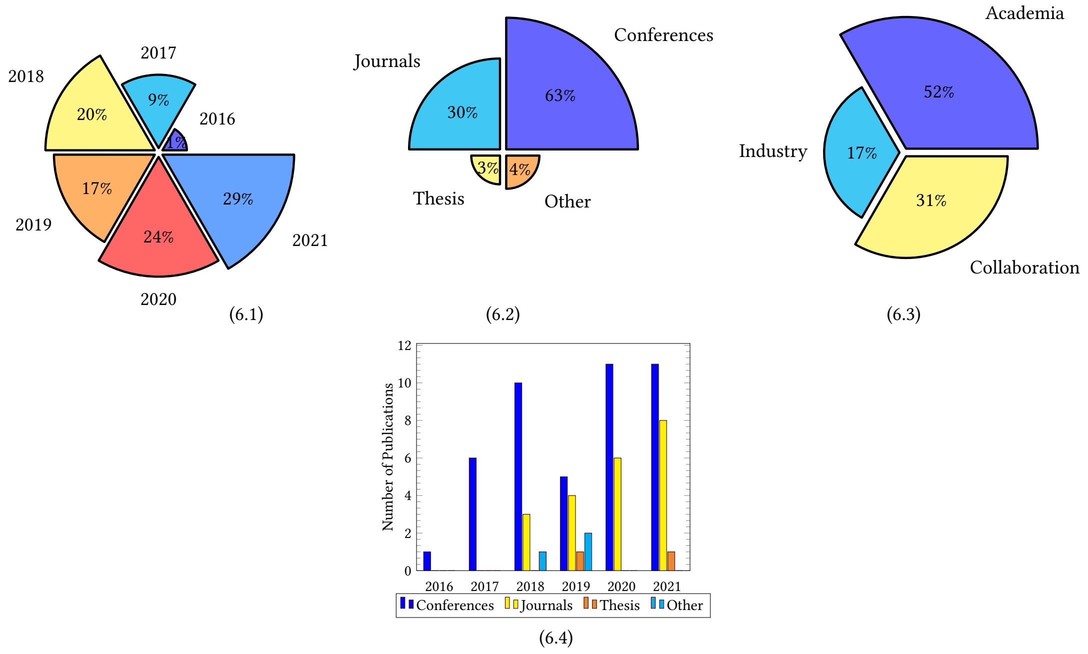


Fig. 6. Quantitative analysis of research methodology showing distribution of research articles according to (i) year of publication (ii) venue of publication (iii) publishing institution (iv) Comparison of number of publications vs venue types vs year of publication.

Figure 6 depicts the distributions of 70 research articles considered in this study. In Figure 6.1, it has been observed that 53% of total research articles are published during the time period of 2020-2021 with 2021 having the biggest share of 29%. This shows the increasing interest of researchers in intent-driven service management (IDSM) systems. Figures 6.2 and 6.3 represent the publication venue and institution wise distribution of research articles. As depicted, most of the research is published in conferences (63%) followed by journal (30%). Whereas, publications coming out of academic institutions are the major contributors (52%) followed by the articles published in collaboration of academic institutions and their industrial partners. Figure 6.4 is an aggregated representation of number of publications vs venue type and year of publication. It can be seen that number of publications in journals are increasing consistently since 2018. This represents that the research in IDSM systems in progressing and quality of solutions is improving and maturing, which is analyzed and explained in the following section.

5 A TAXONOMY

In Figure 4, we identified four activities the intent-driven service management (IDSM) systems perform to fulfill the service level agreement (SLA) requirements of the intents. In this section, a thorough study of each activity is performed and corresponding taxonomies and formal definitions are provided. This section also compares the solutions for IDSM systems from the literature.

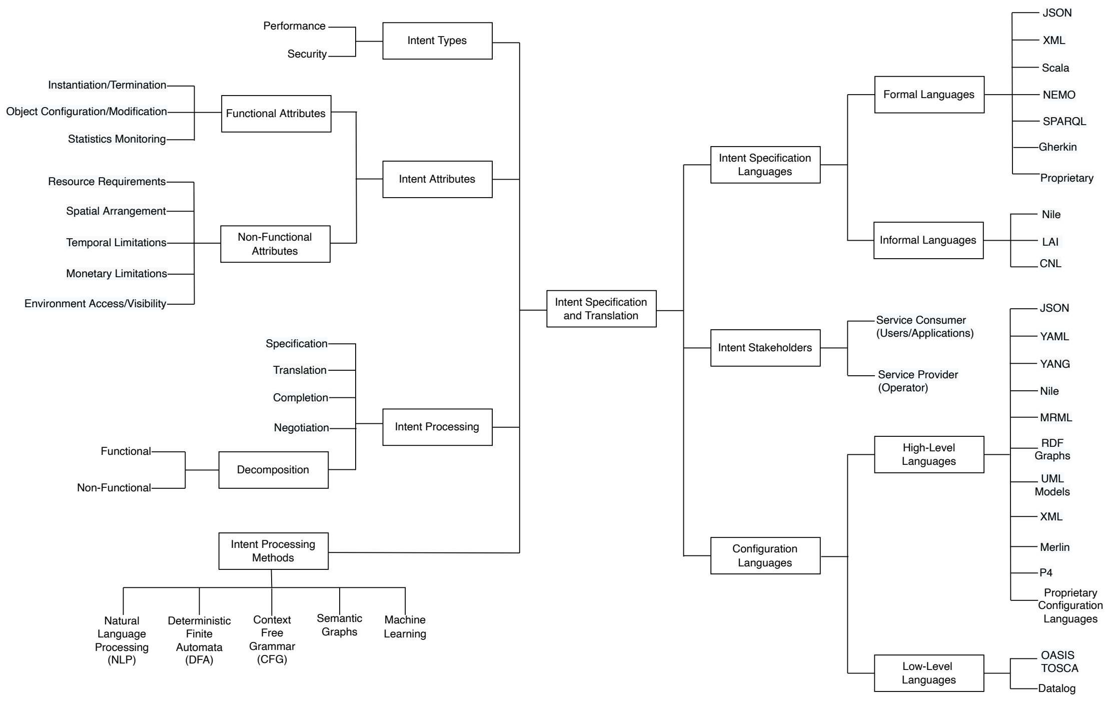


Fig. 7. Taxonomy for Intent Specification and Translation Activity

5.1 Intent Specification and Translation

In this activity, intent handler (IH) captures the high-level intents and converts them to required system design and policies. Figure 7 shows the taxonomy for intent specification and translation representing various components of an intent, such as, intent types, attributes, specification languages, intent processing methods and languages of configuration output after processing an intent. Each component is discussed in the following sections along with their sub-components and suitable examples. The analysis of various methods and solutions addressing intent specification and translation is presented in Table 6.

5.1.1 Intent Specification: It refers to an act of stating/describing an intent representing expected outcomes/results in the form of high-level service request. An intent can have multiple stakeholders, such as, service users and service providers and can be specified by using any formal or informal language.

- *Formal Languages:* Languages with precise syntax and semantics are called formal languages. Intents specified using formal languages need less or no pre-processing before being fed

to the intent handler for further processing (Section 5.1.3). Various formal languages used to specify intents are: JSON [25, 53, 97, 99], XML [18, 28, 45], Scala [33], NEMO [96] and SPARQL [16].
- *Informal Languages:* These languages are either Controlled Natural Languages (CNL) used by the humans in daily routine or a blend of formal and CNL also called 'pseudo code'. Informal languages are more solution/user specific languages with a loosely defined syntax that can change according to the use case. Intents specified using informal languages are tend to have ambiguities. An intermediate processing system is required to resolve such ambiguities before they can be used as input to an intent handler. Apart from CNL [5, 46, 78, 104] and proprietary languages [40, 50, 90], other informal languages used to specify intents are: Language for Access Control List Intents (LAI) [93], Nile [73, 75, 101] and Gerkin [30].

5.1.2 Intent Attributes: Intent Attributes provide the key information about the characteristics of an envisioned service request specified as an intent by a user. Intent attributes are of two types: Functional and Non-Functional attributes.

- *Functional Attributes:* Functional attributes represent what a service or system is expected to do or perform to fulfill the objectives of an intent. In Figure 8, keywords 'features', 'transfer' and 'Topology' represents the functional attributes illustrating the need to connect site X and Y by deploying a link between them. Based on the characteristics identified, an intent can have three classes of functional attributes, such as, (1) Instantiation/termination, (2) Object configuration/modification and (3) Statistics monitoring.
  Instantiation/termination attributes represent the need to start or stop a service instance. Object configuration/modification attributes express the requirement of changing the configuration of an instance of a service. Statistics monitoring attributes represent the demand for the collection of telemetry data. Tsuzaki et al. [96], Riftadi et al. [75], Esposito et al. [30], Chung et al. [18] and Bezahaf et al. [9] are some of the works with intents covering both instantiation/termination and object configuration/modification attributes, whereas Tian et al. [93], Davoli et al. [23] and Dzeparoska et al.[26] have intents with statistics monitoring attributes.
- *Non-functional Attributes:* Non-functional attributes represent the quantitative or qualitative constraint or parameters required to be obliged while fulfilling an intent. In Figure 8, keywords 'Latency', 'Cost', 'Availability', 'Start' and 'Stop' represents the non-functional attributes providing configuration values and corresponding constraints for a link required to be deployed between site X and site Y. Based on the characteristics of non-functional attributes, we have divided them into five categories, i.e., (1) Resource requirements, (2) Spatial arrangement, (3) Temporal limitations, (4) Monetary limitations and (5) Environment access/visibility.
  Resource requirement attributes of an intent represents the essential compute or network resources (CPU, memory, storage, bandwidth) asked by an intent owner. Temporal and Monetary limitations are the attributes for imposing time and cost related constraints on a service request, respectively (start and stop; and cost constraints in Figure 8). Spatial arrangement attributes represent the space related constraints, for example, storage service hosted in a data center of a specific zone/region is requested to fulfill an intent because of the sensitivity and legality of the data. Environment access attributes are related to intents for security services, such as, firewall and intrusion detection systems (IDS). Abhashkumar et al. [2] and Sköldström et al. [88] have discussed all the five types of non-functional attributes.

Table 6. Comparative Analysis of Existing Solutions based on Taxonomy of Intent Specification and Translation Activity

Authors	Intent Type	Intent Attributes	Specification Language	Intent Processing	Intent Processing Method	Configuration Language
Sung et al.[90]	Performance	Object configuration, Instantiation/Termination, Resource requirements	Proprietary	Completion, Decomposition		Proprietary language
Scheid et al.[78]	Security	Instantiation/Termination, Spatial arrangement, Environment access	Controlled natural language	Decomposition	Semantic graphs, Machine learning	Proprietary language
Tsuzaki et al.[96]	Performance	Object configuration, Instantiation/Termination, Resource requirements	NEMO	Decomposition		Proprietary language
Abhashkumar et al.[2]	Performance, Security	Object configuration, Spatial arrangement, Resource requirements, Temporal limitations, Environment access	Proprietary	Decomposition	Semantic graphs	Proprietary language
Kang et al.[40]	Performance	Object configuration, Environment access	Proprietary	Decomposition	Semantic graphs	Datalog
Alsudais et al.[5]	Performance, Security	Object configuration, Instantiation/Termination, Spatial arrangement, Environment access	Controlled natural language	Decomposition	NLP	
Sköldström et al.[88]	Performance, Security	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements, Temporal limitations, Environment access	JSON		DFA, CFG	JSON
Liu et al.[53]	Performance	Object configuration, Spatial arrangement, Resource requirements	JSON	Decomposition	Machine learning	
Comer et al.[19]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements	Proprietary	Negotiation, Decomposition		Proprietary language
Elhabbash et al.[28]	Performance, Security	Object configuration, Instantiation/Termination, Resource requirements	XML	Decomposition		XML
Dzeparoska et al.[25]	Security	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements, Environment access	JSON	Completion, Decomposition	Semantic graphs	JSON
Vilalta et al.[99]	Performance	Instantiation/Termination, Resource requirements	JSON	Decomposition		
Tuncer et al.[97]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements	JSON	Composition, Decomposition		Proprietary language
Esposito et al.[30]	Performance, Security	Object configuration, Instantiation/Termination, Spatial arrangement, Environment access	Gherkin			JSON
Chao et al.[15]	Performance, Security	Resource requirements, Environment access	Controlled natural language	Decomposition	NLP, Machine learning	YAML

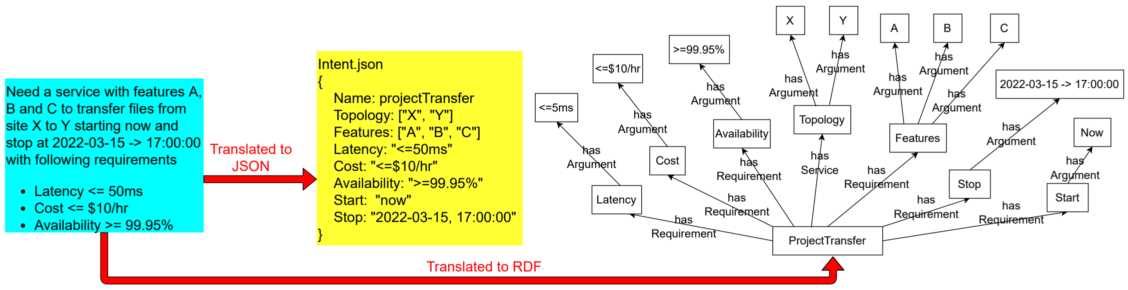
Authors	Intent Type	Intent Attributes	Specification Language	Intent Processing	Intent Processing Method	Configuration Language
Monga et al.[62]	Performance	Instantiation/Termination, Resource requirements, Temporal limitations	JSON	Completion, Negotiation	Semantic graphs	MRML
Yang et al.[107]	Security	Object configuration, Resource requirements, Environment access	XML	Decomposition	DFA, CFG	XML
Kiran et al.[46]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements, Temporal limitations	Controlled natural language	Translation, Completion, Negotiation, Decomposition	NLP	RDF
Davoli et al.[23]	Performance	Object configuration, Statistics monitoring, Spatial arrangement, Resource requirements, Temporal limitations	JSON	Decomposition		JSON
Wang et al.[100]	Security	Spatial arrangement, Environment Access	JSON	Decomposition		YANG
Szyrkowiec et al.[92]	Security	Instantiation/Termination, Resource requirements, Environment access	JSON			YANG
Riftadi et al.[75]	Performance	Object configuration, Resource requirements	Proprietary	Completion	Machine learning	P4
Wu et al.[104]	Performance	Instantiation/Termination, Resource requirements	Controlled natural language	Decomposition	NLP	YAML
Aklamanu et al.[3]	Performance	Object configuration, Instantiation/Termination, Resource requirements, Temporal limitations	JSON	Negotiation, Decomposition	Semantic graphs	JSON
Riftadi et al.[74]	Performance	Object configuration, Instantiation/Termination, Resource requirements	Nile	Decomposition	Semantic graphs	P4
Borsatti et al.[11]	Security	Instantiation/Termination, Spatial arrangement, Environment access	JSON	Decomposition		YAML
Tian et al.[93]	Security	Object configuration, Statistics monitoring, Spatial arrangement, Environment access	LAI	Negotiation, Decomposition	CFG	Proprietary language
Kumar et al.[50]	Security	Object configuration, Instantiation/Termination, Temporal limitations, Environment access	Proprietary language	Decomposition	CFG	JSON
Chen et al.[16]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements	SPARQL	Decomposition	Semantic graphs	RDF
Chung et al.[18]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements, Temporal limitations	Controlled natural language			JSON
Jacobs et al.[36]	Security	Object configuration, Spatial arrangement, Environment access	Controlled natural language		NLP, Machine learning	Nile

Authors	Intent Type	Intent Attributes	Specification Language	Intent Processing	Intent Processing Method	Configuration Language
Scheid et al.[79]	Security	Object configuration, Instantiation/Termination, Resource requirements, Temporal limitations, Environment access	Controlled natural language	Completion, Decomposition	DFA, CFG	JSON
Khan et al.[42]	Performance	Object configuration, Resource requirements	Proprietary language			JSON, YAML, OASIS TOSCA
Chung et al.[17]	Security	Object configuration, Resource requirements, Temporal limitations	JSON, XML		DFA, CFG	XML
Ujcich et al.[98]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement	Proprietary language	Decomposition	Semantic graphs	Proprietary language
Alalmaei et al.[4]	Performance	Object configuration, Resource requirements	Controlled natural language		NLP	JSON
Mahtout et al.[55]	Performance	Instantiation/Termination, Spatial arrangement, Resource requirements, Temporal limitations	Controlled natural language	Negotiation	NLP, Machine learning	RDF
Nagendra et al.[65]	Security	Object configuration, Spatial arrangement, Environment access		Decomposition		
Gao et al.[33]	Performance	Object configuration, Instantiation/Termination, Resource requirements	Scala	Decomposition	Semantic graphs	Proprietary language
Ribeiro et al.[73]	Security	Spatial arrangement, Environment access	Nile			JSON
Nazarzadeoghaz et al.[66]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements		Decomposition		UML models
Kim et al.[45]	Security	Object configuration, Instantiation/Termination, Environment access	XML	Decomposition	DFA, CFG	XML
Wang et al.[102]	Security	Object configuration, Spatial arrangement, Temporal limitations, Environment access	Nile			Proprietary language
Marsico et al.[56]	Performance	Resource requirements	JSON	Negotiation	Semantic graphs	Proprietary language
Rafiq et al.[72]	Performance	Spatial arrangement, Resource requirements	JSON			JSON
Yang et al.[106]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements	Controlled natural language	Negotiation, Decomposition	NLP, Machine learning	JSON
Zhang et al.[109]	Security	Spatial arrangement, Environment access	Proprietary language	Negotiation, Decomposition		Proprietary language
Gritli et al.[35]		Spatial arrangement	Proprietary language	Decomposition	Semantic Graphs	Proprietary language
Mehmood et al.[60]	Performance	Object configuration, Instantiation/Termination	Controlled natural language			YANG
Khan et al. [43]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements	Proprietary language			Proprietary language

5.1.3 Intent Processing: An intent expression is required to be processed by the intent-handlers in order to reach a well-formed and valid expression that can be used by resource managers and service orchestrators (RMSO) to realize the service request. Processing of an intent consists of four

Authors	Intent Type	Intent Attributes	Specification Language	Intent Processing	Intent Processing Method	Configuration Language
Mercian et al.[61]	Security	Instantiation/Termination, Spatial arrangement, Environment access	Controlled natural language	Decomposition	NLP	Proprietary language
Bensalem et al.[8]	Security	Object configuration, Statistics monitoring, Temporal limitations, Environment access	Controlled natural language	Negotiation	CFG	JSON
Bezahaf et al.[9]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Temporal limitations	Controlled natural language		NLP	JSON
Ouyang et al.[69]	Performance	Instantiation/Termination, Resource requirements, Temporal limitations	Controlled natural language		NLP, DFA	JSON
Dzeparoska et al.[26]	Performance	Object configuration, Statistics monitoring, Spatial arrangement, Resource requirements	Controlled natural language	Decomposition	NLP	JSON
Abbas et al.[1]	Performance	Object configuration, Instantiation/Termination, Resource requirements, Temporal limitations	Proprietary language			JSON, OASIS TOSCA
Karrakchou et al.[41]	Performance	Object configuration, Instantiation/Termination, Spatial arrangement, Resource requirements, Environment access	Controlled natural language			P4
el houda Nouar et al.[27]	Performance	Object configuration, Resource requirements	Controlled natural language	Completion		Proprietary language
Kuwahara et al.[51]	Performance	Object configuration, Resource requirements, Monetary limitations	Proprietary language	Completion	Semantic Graphs	OASIS TOSCA
Banerjee et al.[7]	Performance	Object configuration, Resource requirements	Controlled natural language	Negotiation		Proprietary language
de Sousa et al.[24]	Performance	Instantiation/Termination, Spatial arrangement, Temporal limitations	Controlled natural language			RDF
Jacobs et al.[37]	Performance, Security	Object configuration, Spatial arrangement, Resource requirements, Environment Access	Controlled natural language	Completion	NLP, Machine learning	Nile, Merlin
Gomes et al.[34]	Performance	Object configuration, Instantiation/Termination, Resource requirements		Decomposition		RDF
Curtis-Black et al.[22]	Performance, Security	Object configuration, Spatial arrangement, Resource requirements, Environment Access	Controlled natural language	Completion	NLP, Machine learning	JSON

stages, which are: (1) Intent translation, (2) Intent completion, (3) Intent negotiation and (4) Intent decomposition. With the execution of each stage, an intent expression becomes richer and moves closer to RMSO usable form.

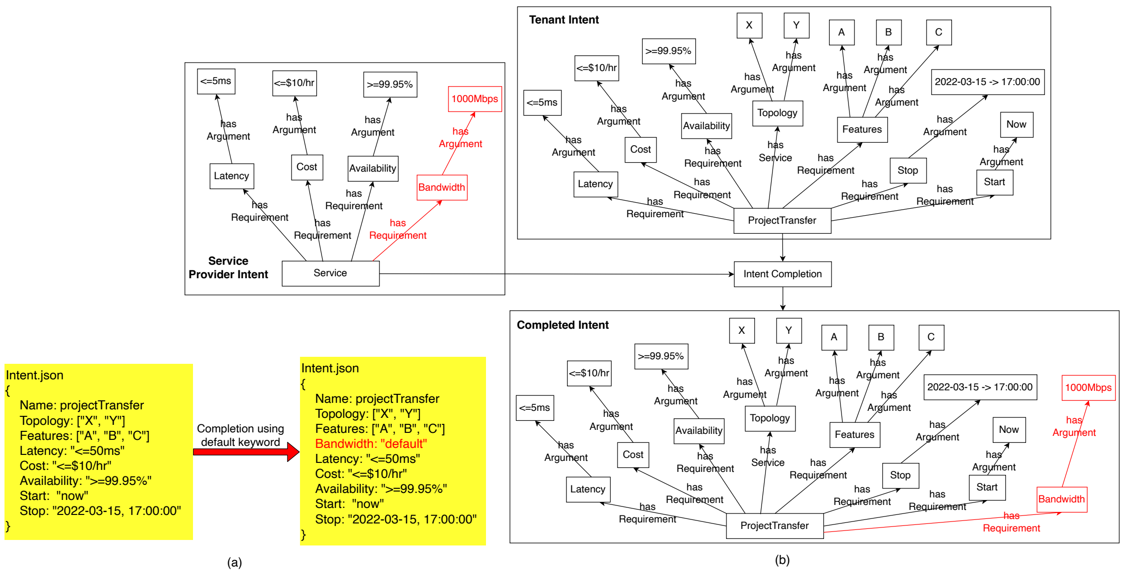

Need a service with features A, B and C to transfer files from site X to Y starting now and stop at 2022-03-15 -> 17:00:00 with following requirements

- Latency <= 50ms
- Cost <= $10/hr
- Availability >= 99.95%

```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Latency: "<=50ms"
    Cost: "<=$10/hr"
    Availability: ">=99.95%"
    Start:  "now"
    Stop: "2022-03-15, 17:00:00"
}
```

Fig. 8. Intent translation from controlled natural language to JSON and RDF format

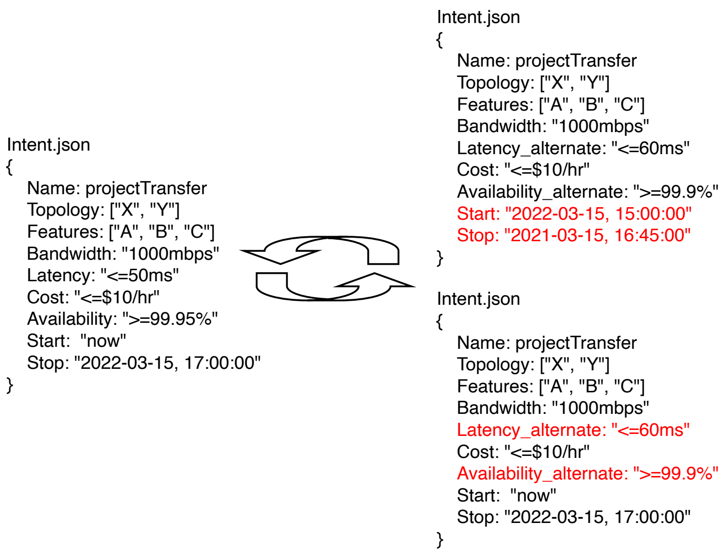
- *Intent Translation:* It refers to changing the notation of an intent specified by using any formal or informal language and predefined template or without template to make it interpretable by the IDSM system. Translation keeps the level of abstraction of an intent same as of specification and does not add or remove any information or details. As shown in Figure 8, an intent specified in a controlled natural language (highlighted in blue) is translated to a template defined in JSON (highlighted in yellow) and RDF format, respectively without adding or removing any information.


```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Latency: "<=50ms"
    Cost: "<=$10/hr"
    Availability: ">=99.95%"
    Start:  "now"
    Stop: "2022-03-15, 17:00:00"
}
```

```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Bandwidth: "default"
    Latency: "<=50ms"
    Cost: "<=$10/hr"
    Availability: ">=99.95%"
    Start:  "now"
    Stop: "2022-03-15, 17:00:00"
}
```

Fig. 9. (a) Intent completion is performed by adding Bandwidth parameter with "default" keyword. (b) Intent completion is performed by obtaining Bandwidth parameter by integrating service provider's intent.

- *Intent Completion:* It is a process to determine the imprecise or unknown parameters an intent expression may contain which are required to be present in the IDSM system acceptable intent format. Such unknown parameters can be added by the intent handlers implicitly or explicitly. While using implicit methods, one way to introduce the parameters by using the default keywords (Figure 9(a)). Such keywords get replaced with the quantitative values during the process of parameter estimation while decomposing the intent [97]. The other way is to obtain such unknown parameters by integrating the service provider's intent with the user's intent (Figure 9(b)). In explicit method, the intent handler uses a combination of iterative steps involving the intent user to ask for clarifications about the unknown parameters. This method is used by Monga et al. [62] and Kiran et al. [46] where they employed a chat-box to ask for clarification from the users about the missing parameters. Intents obtained after the completion process has the minimal information about the service design and are farthest from RMSO usable form.



```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Bandwidth: "1000mbps"
    Latency: "<=50ms"
    Cost: "<=$10/hr"
    Availability: ">=99.95%"
    Start:  "now"
    Stop: "2022-03-15, 17:00:00"
}
```

```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Bandwidth: "1000mbps"
    Latency_alternate: "<=60ms"
    Cost: "<=$10/hr"
    Availability_alternate: ">=99.9%"
    Start: "2022-03-15, 15:00:00"
    Stop: "2021-03-15, 16:45:00"
}
```

```
Intent.json
{
    Name: projectTransfer
    Topology: ["X", "Y"]
    Features: ["A", "B", "C"]
    Bandwidth: "1000mbps"
    Latency_alternate: "<=60ms"
    Cost: "<=$10/hr"
    Availability_alternate: ">=99.9%"
    Start:  "now"
    Stop: "2022-03-15, 17:00:00"
}
```

Fig. 10. Intent negotiation by providing alternative solutions with relaxed temporal and resource requirements to select from during system's inability to satisfy an intent because of resource scarcity.

- *Intent Negotiation:* It is an iterative bi-directional process of reaching an agreement between the intent user and service provider by offering alternative intents (with changed or degraded service requirements) to a given intent. This happens when the current state of the service provider cannot meet the requirements of an intent submitted by a user. Figure 10 represents the process of intent negotiation where the intent handler offers the two alternate solutions to the user to select from. One with changed temporal constraints (start and stop time) and other with relaxed performance constraints (availability and latency). Marsico et al. [56] proposed an intent negotiation framework equipped with alternative solution selection algorithm which provides alternative solutions during resource scarcity with relaxed bandwidth, latency and availability requirements. Tian et al. [93] proposed an intent-driven access control list (ACL) updating system 'JinJing' for Alibaba's global wide area network (WAN). The system is able to detect any policy conflicts while updating the ACL configurations and provides alternate solutions to choose from to avoid such conflicts. Comer et al. [19], Aklamanu et al. [3], Mahtout et al. [55] and Banerjee et al. [7] have also employed intent negotiation methods while processing an intent.
- *Intent Decomposition:* Intent decomposition breaks down a higher-level intent into sub-intents for its dissemination across different intent-handlers or sub-systems required for its fulfillment. During intent decomposition, an intent gets enriched with the information required for the service deployment, such as, service design and configuration parameters. Intent decomposition is of two types: Functional decomposition and Non-functional decomposition
  - *Functional Decomposition:* In order to satisfy the functional attributes of an intent, functional decomposition unit obtains the information about the appropriate functional components

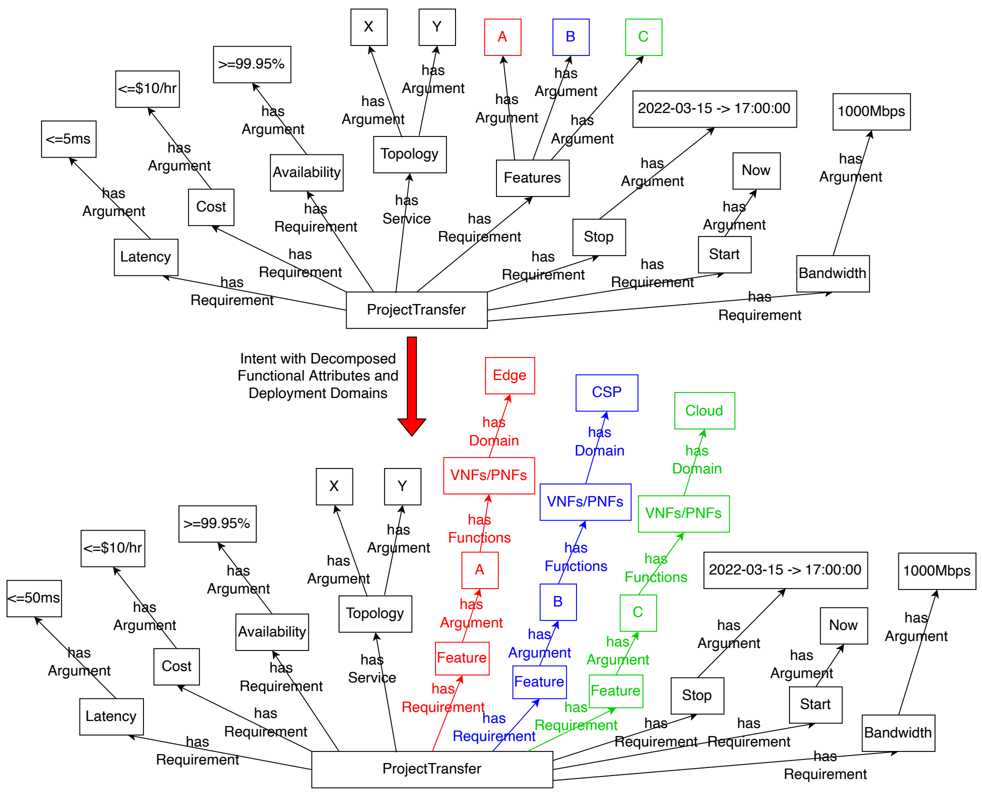


Fig. 11. Functional Decomposition of an intent with selected VNFs/PNFs and deployment domains/sub-systems.

required to deploy a service. This includes the selection of virtual/physical functions, their order of deployment, such as, service chains representing interconnections between the selected virtual/physical functions and corresponding deployment domains/sub-systems. Figure 11 represents the functional decomposition of intent shown in Figure 9(b) to a more precise service request. In the given figure, required virtual and physical network functions (VNFs/PNFs) are decided to host a connectivity service between X and Y with features A, B and C. Features in the present context stands for quality of service (QoS) functions similar to encryption, error detection and correction, firewall, traffic forwarding and intrusion detection system. Apart from deciding about VNFs and PNFs, domains/sub-systems such as, edge, communication service provider (CSP) and cloud, where these functions will be hosted has also been decided. Nazarzadeoghaz et al . [66] proposed an intent decomposition framework for both functional and non-functional attributes of the intents specified specifically for provisioning and deployment of network slices. The proposed framework uses a UML based ontology (knowledge base) to get the information about required network functions and their order of deployment and corresponding configuration parameters for a network slice. Sung et al. [90], Davoli et al. [23], Chen et al. [16], Ujcich et al. [98] and Gritli et al. [35] are some of the other works addressing the challenge of functional decomposition of intents.

– *Non-Functional Decomposition:* It refers to breaking down of the performance constraints specified in an intent to sub-intents and estimation of configuration parameters for the

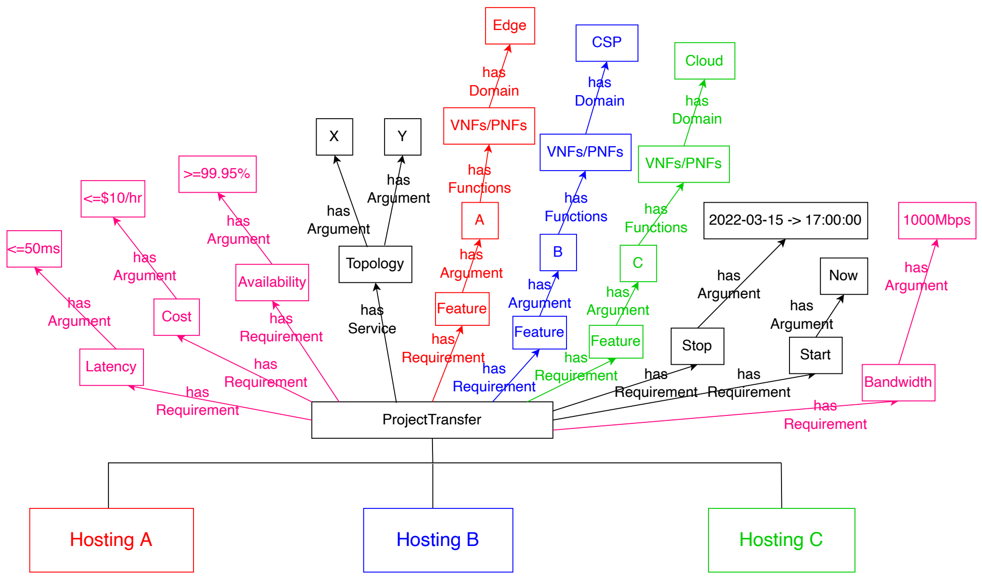


Fig. 12. Decomposition of non-functional attributes of the intent obtained after functional decomposition. It results in the break-down of the original intent into sub-intents corresponding to each deployment domain/sub-system obtained during functional decomposition.

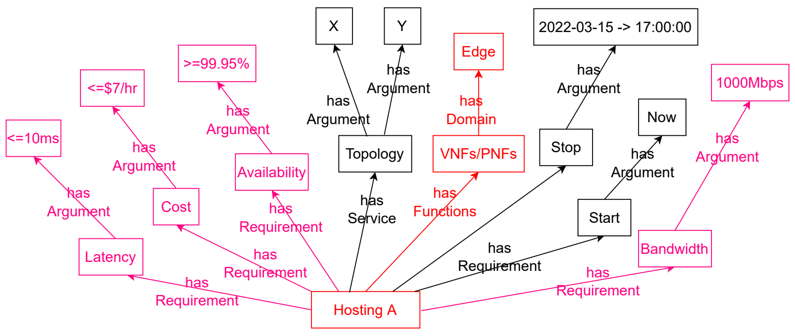


Fig. 13. Sub-intent obtained after Non-Functional Decomposition of intent in Figure 10 to host feature A on Edge sub-system.

selected physical/virtual functions during functional decomposition. Figure 12 shows the non-functional decomposition of an intent obtained after functional decomposition in Figure 11. The intent is decomposed into three sub-intents corresponding to each domain/sub-system i.e., edge (Figure 13), CSP (Figure 14) and cloud (Figure 15) hosting feature A, B and C, respectively. The non-functional attributes, such as, latency, cost and availability are decomposed according to the characteristics of the domain/sub-system the VNFs/PNFs going to get hosted on where as the bandwidth remains same for all the sub-systems.

5.1.4 Intent Processing Methods: In order to process an intent from its specified form to a well defined RMSO interpretable format, four main intent processing methods are: (1) Natural Language Processing (NLP), (2) Deterministic Finite Automata (DFA), (3) Context Free Grammars (CFG) and (4) Semantic Graphs. All these methods are found to be used independently as well as in conjunction

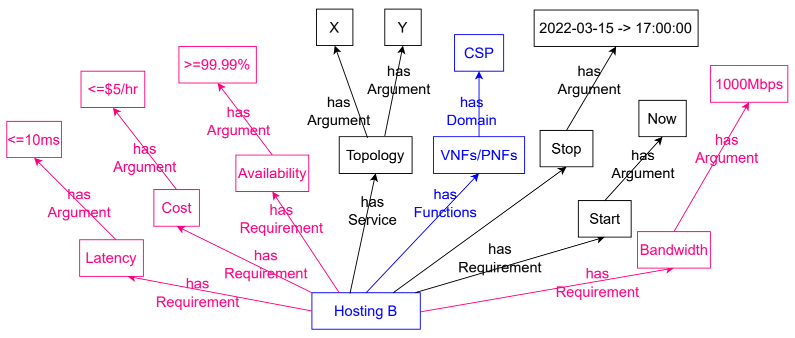


Fig. 14. Sub-intent obtained after Non-Functional Decomposition of intent in Figure 10 to host feature B on CSP sub-system.

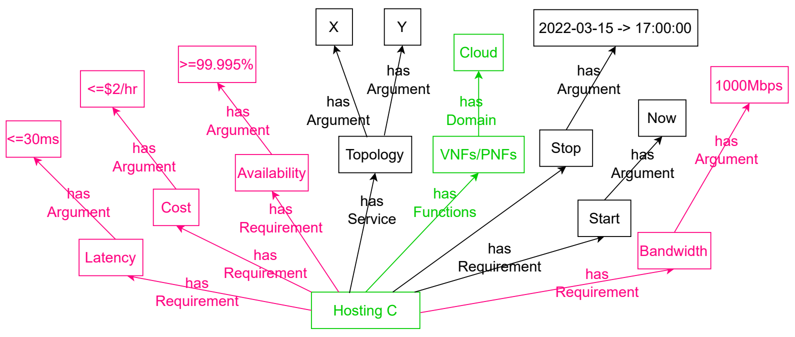


Fig. 15. Sub-intent obtained after Non-Functional Decomposition of intent in Figure 10 to host feature C on Cloud sub-system.

with each other. NLP (also known as computational linguistics [91]) is a method to interpret and manipulate human language to a computer native language and is used to process intents specified in CNL or other informal languages [4, 46, 61, 104]. Modern practitioners and researchers have started to use ML methods to improve the efficiency and effectiveness of classic NLP methods as a result of advancements in big data methods for managing and analyzing large amounts of data. Chao et al. [15], Jacobs et al. [36], Mahtout et al. [55], Yang et al. [106] and Jacobs et al. [37] used NLP in conjunction with ML to process the intents. DFA is a finite state machine which takes strings as input and perform actions and produces an output for each state transition. DFA machines are used to extract the strings/keywords of interest from a high-level intent and convert them to machine compatible values. Scheid et al. [78], Yang et al. [107] and Kim et al. [45] used DFA to extract strings, for example, names of the users from high-level intents and replace them with corresponding IP addresses by using database maintaining IP addresses of all users. CFG is a formal grammar with certain types of production rules required to process an intent to get details about system design and configuration parameters [21]. Most of the solutions use CFG in alliance with DFA to process an intent [18, 78, 88]. Semantic graph is a network with labeled edges and nodes used to represent semantic relationships between concepts [89]. These are very useful to maintain the knowledge bases required to process an intent and can be used either independently [2, 3, 5, 25, 40, 51] or in association with other methods, i.e., ML [79].

5.1.5 Configuration Languages: After processing an intent, output is generated in languages called as 'Configuration Languages'. Based on the abstraction level of the configuration language, we have divided them into two categories: Low-Level and High-Level configuration languages.

- *Low-Level Languages:* An intent processed into a low level language has no abstraction from the language acceptable by RMSO responsible of service deployment. No intermediate process is required to convert the output generated in a low-level language to RMSO acceptable language and can be directly accepted as input by the underlying system. OASIS TOSCA [42, 51] and Datalog [40] are the low-level languages in which the service design solutions are generated and applied directly to the underlying RMSO.
- *High-Level Languages:* An intent processed to a high level language has a high-level of abstraction from the languages acceptable by RMSOs. The intent processing solutions generated in high-level languages need an intermediate processing unit (some kind of compiler or interpreter) to make them acceptable by RMSO for service deployment. JSON [18, 42, 50, 79], YAML [11, 15, 42, 104], Yet Another Next Generation (YANG) [92], Nile [36], Multi Resource Markup Language (MRML)[62], RDF graphs [16, 34, 46, 55], Unified Modeling Language (UML) models [66], P4 [41, 74, 75] and proprietary configuration languages [2, 78, 90, 96] are the high-level languages to represent output of an intent handler.

5.2 Autonomous Deployment and Orchestration

This activity deals with hosting the decomposed intents (system design and configuration parameters) on the underlying virtual and physical infrastructure. An intelligent resource manager and service orchestrator (RMSO) accepts the generated service design and configuration information. It performs the required changes in the underlying infrastructure by provisioning and allocating the best virtual/physical resources across multiple domains/sub-systems to fulfill an intent by meeting its service level agreement (SLA) parameters. Figure 16 provides the taxonomy for autonomous deployment and orchestration representing SLA parameters the intent stakeholders (users and service providers) target and various resource provisioning and management methods to host and fulfill the intents. Table 7 summarizes the existing research covering autonomous deployment and orchestration of intents.

5.2.1 SLA Parameters: SLA is a contractual agreement between two parties, i.e., service provider and its consumer written in a legal format which both parties are abide to follow during the specified period of the contract. Specification of an SLA is usually done in the measurable terms representing what a service provider will furnish in terms of QoS parameters, also know as SLA parameters. It also cover the penalties the service provider will pay for example, monetary compensation when the promised service is not maintained or delivered. It could be possible that two or more parties come together to provide a service, which is a case in IDSM systems where edge cloud provider (ECP), CSP and hyper-scale cloud provider (HCP) as domains/sub-systems create an ecosystem for providing a service (Figure 3). In such cases, an SLA will be a multi-party SLA with domain/sub-system specific SLA parameters. Based on the characteristics of SLA parameters targeted by the intents, we have divided them in two categories: Networking and Computing SLA parameters.

- *Networking SLA parameters:* SLA parameters for networks are the performance parameters within which a network service is required to be provided to fulfill an intent. Various networking SLA parameters that are targeted by intent stakeholders include delay, latency, hop count, bandwidth utilization, data rate and availability (Figure 16). Delay (refers to transmission delay) describes the time required to transmit/transport a data packet from one end (source) of the network to the other (destination) [87, 106]. Latency is also a measure of delay representing a round-trip time taken by a data packet to reach its destination and back again

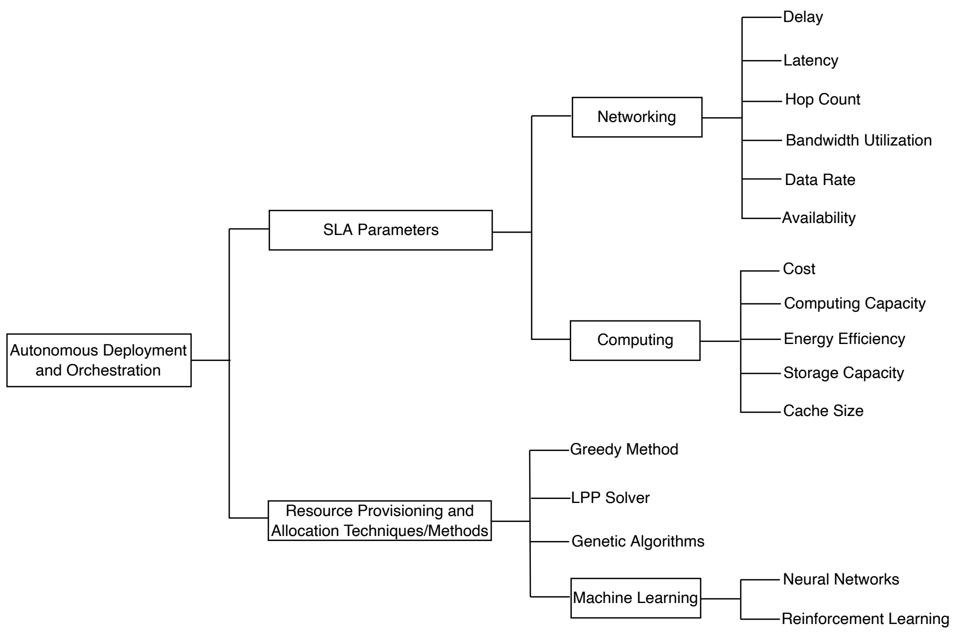


Fig. 16. Taxonomy for Autonomous Deployment and Orchestration Activity

Table 7. Comparative Analysis of Existing Solutions based on Taxonomy of Autonomous Deployment and Orchestration

Authors	SLA Parameters	Intent Realizing Resource Allocation Techniques/Methods
Abhashkumar et al.[2]	Bandwidth utilization	Greedy method, LPP solver
Elhabbash et al.[28]	Cost, Storage capacity, Cache size	Greedy method, Genetic algorithm
Vilalta et al.[99]	Bandwidth utilization	Greedy method
Kumar et al.[50]	Hop count	LPP solver
Nagendra et al.[65]	Hop count	Greedy method
Shi et al.[87]	Delay, Cost	Reinforcement Learning
Marsico et al.[56]	Bandwidth utilization, Latency, Availability	Greedy method
Rafiq et al.[72]	Throughput, Hop count, Bandwidth utilization	Greedy method
Alalmaei et al. [4]	Data rate	
Gao et al. [33]	Hop count, Bandwidth utilization	
Mehmood et al.[58]	Energy efficiency, computing capacity	
Yang et al.[106]	Delay, Bandwidth utilization, Latency	Neural networks, Reinforcement Learning
Mehmood et al.[60]	Hop count, Bandwidth utilization, Latency, Availability	
Khan et al.[43]	Bandwidth utilization, Latency	
Jacobs et al.[37]	Bandwidth utilization	

[56]. Hop count refers to the number of devices/nodes, usually routers, that a data packet passes through from its source to destination. Hop count is used by intents with environment access/visibility non-functional attributes requesting for a security as a service. Kumar et al. [50] specified an intent with hop-count as an SLA parameter with an objective to reduce the number of hops to minimize the cost of security rules placement. As SLA parameters, all these metrics are specified by using their corresponding upper bound values, such that, if the observed value of any of these parameters is more than the specified value then it is considered as a violation of SLAs, and consequently the intent is not fulfilled.
Bandwidth utilization specified in an intent as a service requirement refers to the maximum data transfer rate required over a specific connection [2, 42, 56, 72, 99]. Data rate as an SLA parameter denotes the transmission speed, or the number of bits per second required to be

transferred to fulfill an intent [4, 72]. Availability of a network is a critical SLA parameter which represents the level of accessibility, connectivity and performance of a network in terms of its uptime (network is fully operational) over a specific time interval [56]. Bandwidth utilization, data rate and availability are specified in an intent differently from other networking SLA parameters (delay, latency, hop count). When the obtained bandwidth utilization, data rate or availability is below the intended values, it is considered as intent violation.

- *Computing SLA parameters:* SLAs for computing targets the performance of computing and storage infrastructure provisioned and allocated in domains/sub-systems (ECP, CSP and HCP) selected to satisfy an intent. The parameters of interest are; cost, computing capacity, energy efficiency, storage capacity and cache size (Figure 16). Cost of the service is one of the parameters both service users and providers are most interested to regulate in. Besides cost, other computing SLA parameters used to specify the intents are computing capabilities (CPU count and utilization) and energy consumption of the computing infrastructure provisioned to fulfill an intent. Mehmood et al. [58] proposed a method to regulate the CPU utilization and energy efficiency of the computing infrastructure to meet the profit goals for both service users and providers while fulfilling the intents. Elhabbash et al. [28] exploited storage capacity and cache size as internal SLA parameters to satisfy an intent of a user with minimum cost.

5.2.2 Resource Provisioning and Allocation Techniques/Methods: In order to satisfy the service requirements specified as SLA parameters (discussed in Section 5.2.1) of an intent, RMSOs are required to perform provisioning and allocation of virtual/physical resources across multiple domains/sub-systems identified during the intent decomposition. In this section, such resource provisioning and allocation techniques used by RMSOs to fulfill the intents are discussed.

- *Greedy Method:* It is a simple and intuitive method to design algorithms which makes locally optimal choice at each step in order to obtain an approximate global optimal solution. In crux, it constructs the optimal solution piece by piece. Resource management solutions for IDSM systems based on greedy method choose the best physical/virtual resources available at an instance to host a service request. The solution then extends iteratively to other service request instances to achieve a global optimal solution. Abhashkumar et al [2], Elhabbash et al. [28] and Rafiq [72] used greedy method based algorithms for resource management and allocation to fulfill the intents.
- *Linear Programming Problem Solver:* Linear programming (LP) is a mathematical optimization technique for determining the optimal allocation of scarce resources with having linear objective functions and relations among the variables corresponding to resources. Kumar et al. [50] has formulated and solve the problem of traffic blocking rule placement by using LP with minimum cost while satisfying the security requirements specified as an intent.
- *Genetic Algorithms:* It is a search-based technique inspired from the process of biological evolution and can be used for solving resource optimization problems with linear or non-linear and continuous or non-continuous objective functions. Elhabbash et al. [28] used genetic algorithm based approach to maximize the number of intents being served with optimal selection of services offered by the service provider.
- *Machine Learning:* Resource management methods employing data analytics and model building are covered in this type. Neural Networks and Reinforcement learning are the two commonly used ML methods. Yang et al. [106] used a reinforcement learning based deep Q network (DQN) method for resource composition satisfying the requirements of an intent.

5.3 Monitoring and Awareness

The primary task of monitoring and awareness activity is to provide periodic feedback to intent stakeholders about the status of the intents as well as identify and predict any anomaly (outage/failure or congestion/resource over utilization) in the system that can impact the fulfillment of the intents. Intent-driven service management (IDSM) system performs periodic data collection from the physical and virtual resources and conducts the analytical operations to evaluate the current state of the system. Obtained results are used to determine whether the current performance of the system satisfies the requirements of hosted intents and is able to host new intents. If the telemetry results are found to be satisfactory w.r.t the hosted intent SLA parameters (Section 5.2.1), the existing resource management policy remains unchanged. Otherwise, refinement/remediation activities (Section 5.4) take place autonomously to fix the system's performance and to avoid any foreseeable anomaly which can impact the fulfillment of the intents. Figure 17 provides the taxonomy for monitoring and awareness activity representing various performance monitoring and prediction methods, key performance indicators and performance challenges. Table 8 summarizes the existing research works covering monitoring and awareness activity.

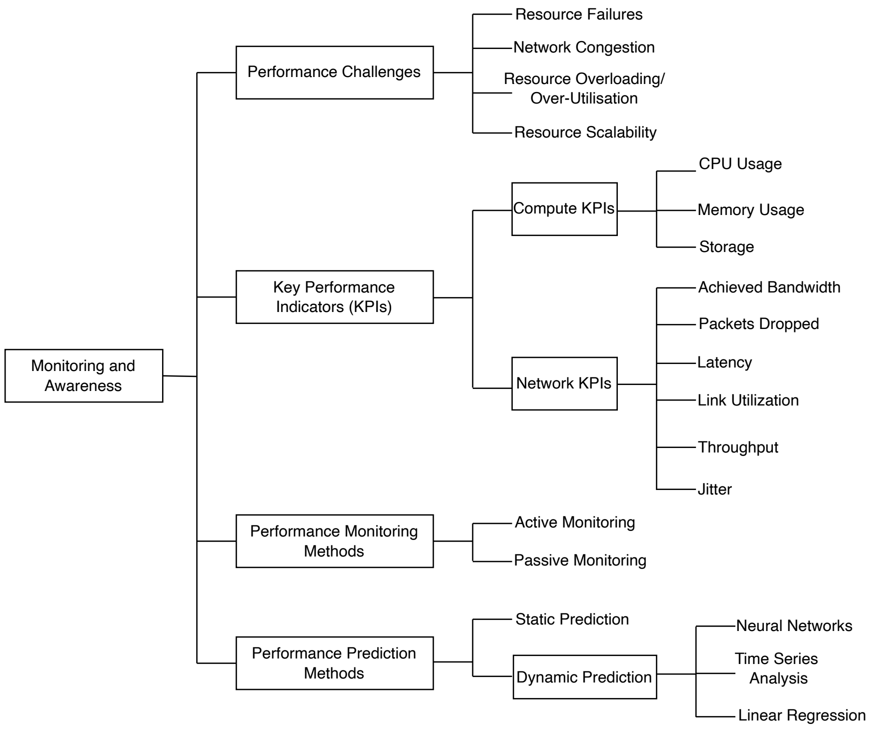


Fig. 17. Taxonomy for Monitoring and Awareness Activity

5.3.1 Performance Challenges: While fulfilling the intents, IDSM systems face various performance challenges, such as, resource failures, network/traffic congestion, resource overloading and resource scalability. Regular monitoring of key performance indicators (KPIs) (Section 5.3.2) is required to avoid/handle the occurrence of events posing such challenges.

- *Resource Failures:* Occurrence of failures is inevitable and is the biggest challenge that all the systems face, including IDSM systems. There are various reasons that can cause the

Table 8. Comparative Analysis of Existing Solutions based on the Taxonomy of Monitoring and Awareness Activity

Authors	Performance Challenges	Performance Monitoring Methods	Key Performance Indicators	Performance Prediction Methods
Sung et al.[90]	Resource failures, Resource overutilization	Active, Passive	CPU usage, Memory usage, Link utilization	Static method
Tsuzaki et al.[96]	Network congestion, Resource overutilization	Active	Achieved bandwidth	Static method
Sanvito et al.[76]	Resource failures, Resource overutilization	Active	Link utilization	Time series analysis
Davoli et al.[23]	Resource failures, Network congestion	Active	Packets dropped, Latency	Static method
Saraiva et al.[77]	Resource overutilization	Active	Achieved bandwidth, Packets dropped	Static method
Aklamanu et al.[3]	Resource overutilization, Resource scalability	Active, Passive	CPU usage, Memory usage, Achieved bandwidth	Static method
Khan et al.[42]	Resource overutilization	Active	Achieved bandwidth	Neural networks
Yang et al.[106]	Resource failures	Active	Achieved bandwidth, Packets dropped	Neural networks
Wu et al. [105]	Resource failures	Active, Passive	CPU usage	Linear regression
Khan et al. [43]	Network congestion, Resource overutilization	Active	Achieved bandwidth, Packets dropped, Link utilization	Neural networks
Zheng et al. [110]	Resource overutilization	Passive	CPU usage	Neural networks
Dzeparoska et al. [26]	Network congestion	Active	Link utilization	Static method
Abbas et al. [1]	Resource overutilization	Passive	CPU usage, Memory usage, Storage, Throughput	Neural networks
de Sousa et al. [24]	Network congestion		Throughput, Packets dropped, Jitter	Static method

failure of resources (both physical and virtual) and consequently causes the service outage [64, 81, 85, 86]. Identical reasons are found for failures in IDSM systems. Sung et al. [90] identified database application and replicated service failures as the cause of service outage. Sanvito et al. [76], Davoli et al. [23], Yang et al. [106], Wu et al. [105] considered link failures and computing resource failures impacting the service connectivity in their IDSM solutions.

- *Network Congestion:* A spike in the demand of a service increases the data transmission/traffic, which can exceed the capacity of the network and may lead to the network congestion. This impacts the quality of a service and can cause a service outage or makes service inaccessible [32]. Tsuzaki et al. [96] and Davoli et al. [23] considered network congestion as a performance challenge in their intent management solutions.
- *Resource Overloading/Overutilization:* Apart from the overutilization of links causing network congestion, overloading of compute resources contributing to the service, such as, CPU, memory (both RAM and cache) and storage can also significantly causes the performance degradation in IDSM systems. This consequently impacts the fulfillment of the intents. Saraiva et al. [77], Aklamanu et al . [3], Khan et al. [42], Abbas et al. [1] proposed IDSM solutions while dealing with the challenge of resource overutilization.
- *Resource Scalability:* It is the ability of a service management system to provision the resources autonomously to handle the growing amount of work load. However, performing resource scalability in IDSM systems without impacting the intents and; increasing the cost and operational complexity is a challenge. Aklamanu et al . [3] addressed the challenge of resource scalability in the proposed IDSM solution.

5.3.2 Key Performance Indicators (KPIs): In order to get the quantifiable measurements to gauge the compliance of SLA parameters, KPIs play a significant role [84]. Collecting, processing and analyzing the data corresponding to KPIs of interest provides insight into the system performance.

The obtained information is further used to compare against the SLA parameters to measure the level of intent satisfaction and to identify or predict any potential performance challenge. In case, a performance diversion is found or predicted to happen, IDSM system takes performance corrective decisions (Section 5.4.1), autonomously to ensure the fulfillment of the intents. Based on the characteristics of the components (both virtual and physical) involved in serving an intent, we have divided the KPIs in two classes: Compute KPIs and Network KPIs.

- *Compute KPIs:* These KPIs are used to measure the utilization of computing resources, such as, CPU, memory and storage; provisioned and allocated to satisfy an intent. CPU and Memory utilization are the main compute KPIs focused on by IDSM researchers to improve intent satisfaction levels. Sung et al. [90] and Aklamanu et al. [3] monitored and exploited CPU and memory usage KPI values and Abbas et al. [1] used storage KPIs as well to handle the challenges of resource overutilization and failures.
- *Network KPIs:* These KPIs are essential to determine the performance of networking components (both physical and virtual) required to fulfill an intent. Achieved bandwidth [42, 77, 96], packets dropped [106], latency [23], link utilization [76], throughput [1] and jitter [24] are the network KPIs the researchers are using to evaluate the performance of their IDSM solutions.

5.3.3 Performance Monitoring Methods: Two types of monitoring methods are used to monitor the KPIs representing the performance of IDSM systems: Active and Passive Monitoring.

- *Active Monitoring:* This method is also know as the synthetic monitoring which injects test traffic (synthetic traffic) into the system to get the real-time view of its performance, generally with regards to the networking SLA parameters, such as, delay, latency, bandwidth utilization and data rate (Figure 16). Most of the works are found to be using active monitoring method to monitor and analyze the fulfillment of the intents (Table 8) .
- *Passive Monitoring:* This method involves capturing and analyzing the real traffic flow, periodically representing the performance of the serving components of the system. Sung et al. [90], Aklamanu et al. [3], Yang et al. [106], Wu et al. [105], Zheng et al. [110] and Abbas et al. [1] employed passive monitoring to observe the parameters of interest in their proposed IDSM solutions.

5.3.4 Performance Prediction Methods: For an IDSM system to fulfill its intents, a reliable prediction of performance or an event that will affect the performance is critical. Furthermore, having the efficient and accurate performance prediction methods provide a leverage to the service providers during intent negotiation by advising the users about the possible performance degradation if they choose not to select the alternative solutions provided by the service provider (Section 5.1.2). This helps both service users and providers to draft the rich and accurate SLAs and avoid any legal conflicts that can occur because of SLA violations. Performance prediction methods use the monitored KPIs as input to predict any performance challenge (Section 5.3.1) that can impact the fulfillment of intents. We have divided the performance prediction methods in two classes: Static and Dynamic prediction methods.

- *Static Prediction:* In the Static Prediction methods, the occurrence of an event is predicted based on a static threshold value for a variable which remains unchanged until the manual changes are made. The threshold values are obtained and set by the system administrators based on the experience of previous runs, for example, if the system outage is happening at the certain utilization level of a CPU then it will be marked as a threshold value for CPU utilization. When the KPIs (both for compute and network components) under observation reaches the threshold values, performance corrective methods are triggered, autonomously to safeguard the intents. Static prediction methods are the most commonly used methods because of the simplicity of their application. Sung et al. [90], Aklamanu et al. [3], Davoli et al. [23], Saraiva et al. [77], Tsuzaki et al. [96], Dzeparoska et al. [26] and de Sousa et al. [24] used static threshold values to predict the performance challenges.

- *Dynamic Prediction:* The drawback of the static prediction methods is that they do not evolve with time, such that, the threshold values remain the same until they are changed manually. However, due to the autonomous nature of IDSM systems, employment of static prediction methods is not an optimal solution. As an alternate, dynamic performance prediction methods is the solution of choice where the threshold values change with time in an autonomous manner by employing ML based methods. Yang et al. [106], Khan et al. [42], Zheng et al. [110] and Abbas et al. [1] used Neural networks for performance prediction. Sanvito et al. [76] and Wu et al. [105] used time series analysis and linear regression enabled dynamic performance prediction methods, respectively in their proposed IDSM solutions.

5.4 Dynamic Optimization and Remediation

Based on the telemetry results, the intent-driven service management (IDSM) systems autonomically optimize their performance in order to meet the SLA parameters required to fulfill the intents. Performance optimization of IDSM systems include internal reconfiguration of both computing and networking resources to safeguard the intents from any predicted anomaly or to increase the efficiency of the system (Section 5.4.2). Figure 18 provides the taxonomy for dynamic optimization and remediation activity for IDSM systems representing intent guarantee management methods as well as methods to perform optimization and remediation. Table 9 summarizes the existing research works covering dynamic optimization and remediation activity for the fulfillment of intents.

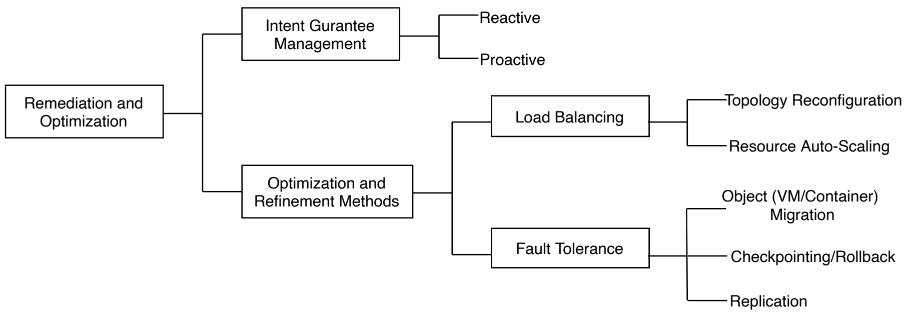


Fig. 18. Taxonomy for Dynamic Optimization and Remediation Activity

Table 9. Comparative Analysis of Existing Solutions based on Taxonomy of Dynamic Optimization and Remediation Activity

Authors	Intention Guarantee Management	Optimization and Refinement Methods
Sung et al.[90]	Reactive	Checkpointing/Rollback
Tsuzaki et al.[96]	Reactive	Topology reconfiguration
Sanvito et al.[76]	Proactive	Topology reconfiguration
Davoli et al.[23]	Proactive	Resource auto-scaling, Replication
Saraiva et al.[77]	Reactive	Topology reconfiguration
Khan et al.[42]	Proactive	Topology reconfiguration
Yang et al.[106]	Reactive	Topology reconfiguration, Object (VM/Container) migration
Dzeparoska et al.[26]	Reactive	Topology reconfiguration
Abbas et al.[1]	Proactive	Resource auto-scaling

5.4.1 Intent Guarantee Management: Methods to guarantee the fulfillment of intents are divided into two categories: Reactive and Proactive.

- *Reactive Management:* In this method, measures are taken after the occurrence of an event. For example, in case of check-pointing used as a fault tolerance method, recovery takes place from the last saved checkpoint after the occurrence of a failure event [82]. Sung et al. [90] and Yang et al. [106] used reactive methods for failure management where as Tsuzaki et al. [96], Saraiva et al. [77] and Dzeparoska [26] used reactive methods to optimize the performance of IDSM systems.
- *Proactive Management:* In contrast to reactive methods, in proactive management, measures are taken before the occurrence of an event. These methods are prediction driven methods where the occurrence of an event is predicted by using machine learning (ML) and data analytic operations. The productivity of the proactive management methods depend upon the accuracy of prediction algorithms. Sanvito et al. [76] and Davali et al. [23] used proactive methods to provide fault tolerance in IDSM systems to safeguard the intents from the failures whereas Khan et al. [42] and Abbas et al. [1] used these methods to optimize the performance of their IDSM solution without impacting the intents.

5.4.2 Optimization and Remediation Methods: The intent guarantee management exploit the load balancing methods for performance optimization of IDSM systems to make them more efficient. Fault tolerance methods are also employed to safeguard the IDSM systems against failures in order to fulfill the intents. The details of both classes of intent guarantee management methods are as follows:

- *Load Balancing:* In general, it is the performance optimization method used to increase the efficiency of the virtual and physical infrastructure by manipulating its resource configuration and migrating the workload. IDSM systems perform load balancing after the detection or prediction of a performance challenge (Section 5.3.1) to avoid the system breakdown or periodically optimize the efficiency of the system in terms of energy consumption [29] and bandwidth utilization [63]. Topology reconfiguration and resource auto-scaling are the two load balancing mechanisms to optimize the performance of IDSM systems. Tsuzaki et al. [96] and Saraiva et al. [77] triggered the topology reconfiguration by re-routing the traffic re-actively if the bandwidth usage of a link exceeded a predefined threshold value. Davoli et al. [23] employed auto-scaling by adding extra resources proactively to avoid any resource scarcity.
- *Fault Tolerance:* To guarantee the fulfillment of intents, IDSM systems need to manage the service failures. In the proposed solutions, various techniques/mechanisms are used to provide fault tolerance in IDSM systems, such as, object (VM or container) migration, checkpointing/rollback and replication. Davoli et al. [23] maintained replicated copy of each transmitted packet to recover from, in case a transmitted packet is lost. Sung et al. [90] used the periodical checkpointing to save the healthy state of the IDSM system serving the intents and used it to recover from the failures. Yang et al. [106] used object migration to migrate the workload from a predicted to be failed computing resource to a healthy one in order to safeguard the fulfillment of the intents.

6 DISCUSSION

This section discusses the principal findings of our systematic review. The discussion covers the critical analysis of all the considered works and highlights the key observations followed by open challenges and future directions for research in service level agreement (SLA) management in intent-driven service management (IDSM) systems.

6.1 Critical Analysis and Key Observations

All studies considered in the survey are critically analyzed and compared in Table 10. The analysis drove the key observations made on the basis of the IDSM activities covered in a solution, its scale (multi-domain or single-domain), area of focus and employment of machine learning (ML) methods. All the observations are supported by the quantitative analysis represented in Figure 19.

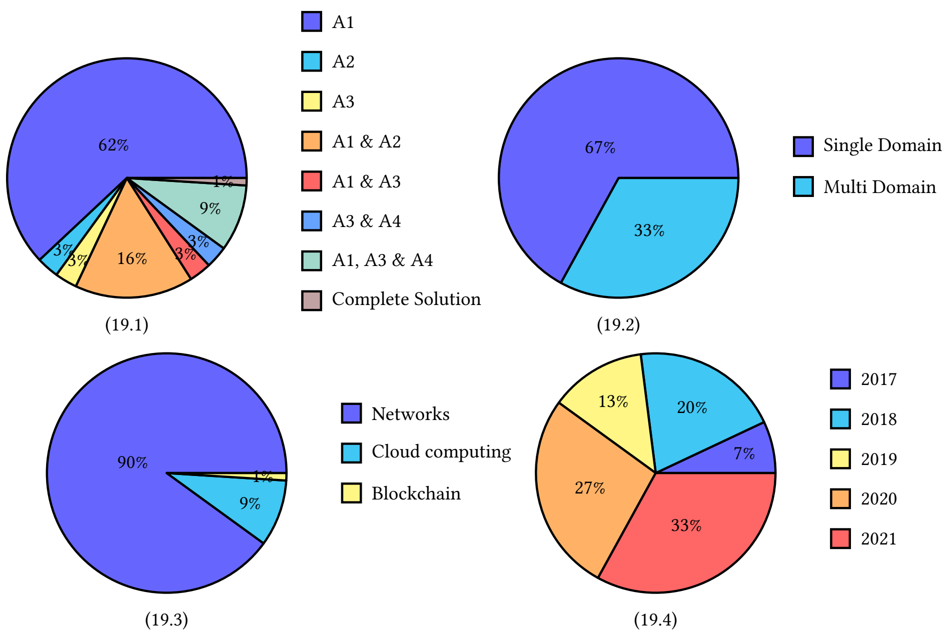


Fig. 19. Quantitative analysis representing the strengths and weaknesses of the current state-of-the-art. Figure shows the distribution of research articles according to the (1) covered IDSM activities (2) scale of the solution (3) area of application (4) use of machine learning.

6.1.1 *Lack of Complete Solution.*

Research in IDSM systems is at early stage; there is a lack of a comprehensive solution covering all four activities of intent management. (IDSM Activities section of Table 10). Figure 19.1 shows the activity wise distribution of the research works considered in this study. Given figure makes it clear that research in IDSM systems has concentrated primarily on Intent Specification and Translation (Activity 1). 43 out of 70 works (62%) covers only intent specification and translation activity in their proposed IDSM solutions. Almost no research has been conducted on the remaining activities. Only one complete solution proposed by Yang et al. [106] covers all the four activities.

6.1.2 *Intent Management in Multiple Domains/Sub-systems.*

Adoption of technologies, such as, intent-driven interfaces, closed loop automation and knowledge driven decision making (based on AI and ML) increases the complexity of IDSM systems. In order to reduce such complexity, IDSM systems can be arranged into layers separating business, service and resource operations; and deployed in multiple domains/subsystems that can operate autonomously. All the layers and domains/sub-systems work together in a closed loop manner and, interact and coordinate with each other by using intent handlers in order to fulfill the intents (Figure 3). However, it has been observed that majority of solutions do not consider the multi-layer and multi-domain architecture

Table 10. Analysis of the research articles considered in this study by highlighting their strengths and weaknesses.

Authors	**Year**	**IDSM Activities**				**Scale of Solution**		**Area of Focus**			**Use of ML**
		A1	**A2**	**A3**	**A4**	**SD**	**MD**	**Networks**	**Cloud computing**	**Blockchain**	
Sung et al.[90]	2016	✓		✓	✓	✓		✓			
Scheid et al.[78]	2017	✓				✓		✓			✓
Tsuzaki et al.[96]	2017	✓		✓	✓		✓	✓			
Abhashkumar et al.[2]	2017	✓	✓			✓		✓			
Kang et al.[40]	2017	✓				✓			✓		
Alsudais et al.[5]	2017	✓				✓		✓			
Sköldström et al.[88]	2017	✓				✓		✓			
Liu et al.[53]	2018	✓					✓	✓			✓
Comer et al.[19]	2018	✓					✓	✓			
Sanvito et al.[76]	2018			✓	✓	✓		✓			✓
Yang et al.[107]	2018	✓				✓		✓			
Elhabbash et al.[28]	2018	✓	✓			✓		✓			
Dzeparoska et al.[25]	2018	✓					✓	✓			
Vilalta et al.[99]	2018	✓	✓				✓		✓		
Tuncer et al.[97]	2018	✓				✓		✓			
Esposito et al.[30]	2018	✓				✓		✓			
Chao et al.[15]	2018	✓				✓			✓		✓
Monga et al.[62]	2018	✓					✓	✓			
Kiran et al.[46]	2018	✓					✓	✓			
Davoli et al.[23]	2018	✓		✓	✓		✓	✓			
Szyrkowiec et al.[92]	2018	✓				✓		✓			
Wang et al.[100]	2019	✓		✓		✓		✓			
Saraiya et al.[77]	2019			✓	✓	✓		✓			
Riftadi et al.[74]	2019	✓				✓		✓			
Wu et al.[104]	2019	✓				✓			✓		
Riftadi et al.[75]	2019	✓				✓		✓			✓
Aklamanu et al.[3]	2019	✓		✓			✓	✓			
Borsatti et al.[11]	2019	✓				✓		✓			
Tian et al.[93]	2019	✓					✓	✓			
Kumar et al.[50]	2019	✓	✓			✓		✓			
Chen et al.[16]	2019	✓				✓		✓			
Chung et al.[18]	2019	✓					✓	✓			
Jacobs et al.[36]	2019	✓				✓		✓			✓
Scheid et al.[79]	2020	✓				✓				✓	
Khan et al.[42]	2020	✓		✓	✓		✓	✓			✓
Chung et al.[17]	2020	✓				✓		✓			
Ujcich et al.[98]	2020	✓				✓		✓			
Alalmaei et al.[4]	2020	✓				✓			✓		
Mahtout et al.[55]	2020	✓					✓	✓			✓
Nagendra et al.[65]	2020	✓	✓			✓		✓			
Gao et al.[33]	2020	✓				✓		✓			
Shi et al.[87]	2020	✓	✓				✓	✓			✓
Ribeiro et al.[73]	2020	✓				✓		✓			
Nazarzadeoghaz et al.[66]	2020	✓				✓		✓			
Kim et al.[45]	2020	✓					✓		✓		
Wang et al.[102]	2020	✓				✓		✓			
Marsico et al.[56]	2020	✓	✓			✓		✓			
Rafiq et al.[72]	2020	✓	✓			✓		✓			
Mehmood et al.[58]	2020	✓	✓				✓	✓			
Yang et al.[106]	2020	✓	✓	✓	✓		✓	✓			✓
Zhang et al. [109]	2021	✓				✓		✓			
Wu et al. [105]	2021			✓		✓		✓			✓
Gritli et al. [35]	2021	✓					✓	✓			
Mehmood et al. [60]	2021	✓	✓			✓		✓			
Khan et al. [43]	2021	✓	✓	✓		✓		✓			✓
Mercian et al. [61]	2021	✓				✓		✓			
Zheng et al. [110]	2021				✓	✓		✓			✓
Bensalem et al. [8]	2021	✓				✓		✓			
Bezahaf et al. [9]	2021	✓				✓		✓			
Ouyang et al. [69]	2021	✓				✓		✓			
Dzeparoska et al. [26]	2021	✓		✓	✓		✓	✓			
Abbas et al. [1]	2021	✓		✓	✓		✓	✓			✓
Karrakchou et al. [41]	2021	✓				✓		✓			
el houda Nouar et al. [27]	2021	✓				✓		✓			
Kuwahara et al. [51]	2021	✓				✓		✓			
Banerjee et al. [7]	2021	✓				✓		✓			
de Sousa et al. [24]	2021	✓		✓			✓	✓			
Jacobs et al. [37]	2021	✓	✓			✓		✓			✓
Gomes et al. [34]	2021	✓					✓	✓			
Curtis-Black et al. [22]	2021	✓					✓	✓			

A1: Activity 1 (Intent specification and translation); **A2:** Activity 2 (Autonomous deployment and orchestration); **A3:** Activity 3 (Monitoring and awareness); **A4:** Activity 4 (Dynamic optimization and remediation); **SD:** Single domain; **MD:** Multi Domain; **ML:** Machine Learning and focused only on the single domain/sub-system solutions (Figure 19.2). In the proposed multi-domain solutions (Table 10), the interaction and intercommunication between the intent handlers of different layers and domains/sub-systems either remained untouched or partly explored.

6.1.3 Network-Centric Solutions. As discussed in the background section (Section 2), apart from the networking field, the adoption of IDSM has been explored in other fields, such as, cloud computing. However, from the current state of the art (Table 10), it has been observed that the majority of the proposed intent-driven solutions are focused on the network service management. Very few solutions are available for other fields, such as cloud computing and block-chain (Figure 19.3). Not having intent-driven solutions for such critically important fields will hinder the development of complete IDSM solutions and could limit the value and adoption of the technology.

6.1.4 Use of Machine Learning. ML is becoming ubiquitous owing to the availability of massive data and improvement in computing power and algorithm innovation. Because of this, ML plays an important role in many fields, including computing and network operations. Considering the hierarchical and multi-domain characteristics of IDSM systems, integration of 'operational intelligence' by using ML methods at each layer (business, service and resources) and stage (planning, maintenance, operation and optimization) to achieve closed loop autonomy is an ultimate goal [68]. Despite this, the current state-of-the-art for IDSM systems has the limited use of ML methods and relies heavily on static solutions for intent management (Table 10). However, an increasing trend of employing the ML methods for intent management is seen in the research articles published from 2020 to 2021, which accounts 53% of the total solutions using ML (Figure 19.4).

6.2 Open Challenges and Future Directions

We have identified various challenges which can be used to drive the future research in the area.

6.2.1 Intent Negotiation Framework: An intent submitted by a user may conflict with the service provider intent or with the intents submitted by other users. In order to resolve the conflicts, intent negotiation (Figure 10) takes place either between the human user and intent-handler or among the intent-handlers (either at the same level or different level in the hierarchy). During intent negotiation, alternate intents representing the current capability of the service provider are generated and provided to the user or intent-handler to select from. In order to do so, an intent negotiation framework is required providing a procedure to extract the state of the system and to use it to compose the alternate intents.

6.2.2 Decomposition of Non-Functional Attributes: Decomposition of functional attributes of an intent (for example, selection and chaining of VNFs to satisfy an intent) can be performed by using a knowledge-base consisting of ontologies. However, the decomposition of non-functional attributes and distribute them between the entities obtained after the functional decomposition is a cumbersome process. As shown in Figure 11, set of VNFs/PNFs and their deployment domains/sub-systems (Edge, communication service provider (CSP) and Cloud) are identified to satisfy the intent during functional decomposition. Now, the challenge is to decompose the quantitative values of non-functional attributes (latency, cost and availability) between Edge, CSP and Cloud sub-systems (Figures 12-15) while meeting the service level agreement (SLA) requirements of the original intent. A mechanism is required to perform the efficient decomposition of non-functional attributes of an intent without impacting the aggregated requirements of the original intent.

6.2.3 Comparison of System KPIs and Intent's Non-Functional Requirements: Collection of relevant system key performance indicators (KPIs), their aggregation to get the values corresponding to non-functional attributes of an intent and comparing them with the non-functional attributes of the original intent is a must to measure the SLA compliance. A method is required to carry out such collection, aggregation and comparison operations to measure the quality of experience (QoE) by the intent owners.

6.2.4 Inter-operations between Legacy and Intent-driven Systems: With the advancements of intent-driven service management (IDSM) systems, more service providers will start switching from the traditional methods to intent-driven methods. However, it won't be possible to perform such transition in one go and will happen in a progressive manner. In order to support such transition period, mechanisms are required to enable the inter-operations between the legacy and IDSM systems. The mechanisms can involve the development of integration adapters able to map the requests between both kind of the systems.

6.2.5 Standardized and Generic Intent Specification Method: A standard method and template is required for the intent specification [67]. This will help to remove the current multi-vendor differences, such that, all the existing IDSM solutions have their own intent specification methods. This does not allow the inter-working of these solutions and make them platform dependent, which results in vendor lock-ins. Having a standard and generic intent specification template can simplify the integration of multi-vendor systems required to enable the service.

7 CONCLUSIONS

A concept of Intent-driven service management (IDSM) has recently been proposed to transition from traditional human-driven service management to zero-touch autonomous service management with a goal of simplifying the network and computing services. In IDSM service, requirements are specified in a declarative manner as intents. Its closed control-loop operations then meet the service requirements, autonomously. System and network administrators/operators are relieved of the complex service designing and resource configuration requirements because of autonomy. As a result, the errors and misconfigurations caused by manual operations are reduced significantly, making service deployments faster, cheaper and improves the quality of service (QoS). The realization of IDSM systems include multi-layered vertical arrangement of several actors, such as, intent-handlers, service and resource managers and orchestrators; as well as various horizontal domains/sub-systems, such as, edge cloud, communication service provider (CSP) infrastructure and hyper-scale cloud etc. The interaction and coordination activities which takes place among all these actors and domains of IDSM systems are complex. The complexity raises the concern about the reliability and performance variability of IDSM systems which impacts the intent fulfillment and can cause service level agreement (SLA) violations, impacting the revenues of the service providers. Therefore, there is a need to identify and develop a deep understanding of all IDSM system activities to fulfill the intents and to meet the SLA requirements.

As part of this study, we reviewed existing methods and solutions proposed for IDSM systems and identified four activities the systems perform to fulfill the intents. For each activity, separate taxonomies are proposed. Existing SLA management solutions for IDSM systems are analyzed based on these taxonomies. This allowed us to identify the research gaps in the state-of-the-art and propose various future research directions. As a result, we assert the following conclusions:

- IDSM systems perform four activities to fulfill the intents: intent specification and translation, autonomous deployment and orchestration, monitoring and awareness, and dynamic optimization and remediation.
- Developing a generic IDSM framework to represent intent processing from its specification to its fulfillment is necessary to manage the SLAs effectively. This will standardize the intent processing operations and their interplay.

- To accommodate the diversified needs of the service users and their SLAs, multi-vendor and multi-domain IDSM solutions should be developed by intensifying the interface standardization and development of integration adaptors.

REFERENCES

[1] Khizar Abbas, Talha Ahmed Khan, Muhammad Afaq, and Wang-Cheol Song. 2021. Network slice lifecycle management for 5g mobile networks: An intent-based networking approach. *IEEE Access* 9 (2021), 80128–80146.

[2] Anubhavnidhi Abhashkumar, Joon-Myung Kang, Sujata Banerjee, Aditya Akella, Ying Zhang, and Wenfei Wu. 2017. Supporting diverse dynamic intent-based policies using janus. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. 296–309.

[3] Fred Aklamanu, Sabine Randriamasy, and Eric Renault. 2019. Intent-Based 5G IoT Application Network Slice Deployment. In *2019 10th International Conference on Networks of the Future (NoF)*. IEEE, 141–143.

[4] Shiyam Alalmaei, Yehia Elkhatib, Mehdi Bezahaf, Matthew Broadbent, and Nicholas Race. 2020. SDN Heading North: Towards a Declarative Intent-based Northbound Interface. In *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 1–5.

[5] Azzam Alsudais and Eric Keller. 2017. Hey network, can you understand me?. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 193–198.

[6] Ioana Baldini, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski, et al. 2017. Serverless computing: Current trends and open problems. In *Research Advances in Cloud Computing*. Springer, 1–20.

[7] Anubhab Banerjee, Stephen S Mwanje, and Georg Carle. 2022. Contradiction Management in Intent-driven Cognitive Autonomous RAN. (2022).

[8] Mounir Bensalem, Jasenka Dizdarević, Francisco Carpio, and Admela Jukan. 2021. The Role of Intent-Based Networking in ICT Supply Chains. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. IEEE, 1–6.

[9] Mehdi Bezahaf, Eleanor Davies, Charalampos Rotsos, and Nicholas Race. 2021. To All Intents and Purposes: Towards Flexible Intent Expression. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 31–37.

[10] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 13–16.

[11] Davide Borsatti, Walter Cerroni, Gianluca Davoli, and Franco Callegati. 2019. Intent-based Service Function Chaining on ETSI NFV Platforms. In *2019 10th International Conference on Networks of the Future (NoF)*. IEEE, 144–146.

[12] Karen Campbell, Liz Cruz, Bob Flanagan, Bill Morelli, O'Neil Brendan, Stephane Teral, and Julian Watson. 2019. *The 5G Economy: How 5G will contribute to the global economy*. https://tinyurl.com/vnyj6j9c

[13] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. 2020. An overview on edge computing research. *IEEE Access* 8 (2020), 85714–85728.

[14] Martin Casado, Michael J Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. 2007. Ethane: Taking control of the enterprise. *ACM SIGCOMM computer communication review* 37, 4 (2007), 1–12.

[15] Wu Chao and Shingo Horiuchi. 2018. Intent-based cloud service management. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 1–5.

[16] Xi Chen, Hongfang Yu, Shizhong Xu, and Xiaojiang Du. 2020. CompRess: Composing overlay service resources for end-to-end network slices using semantic user intents. *Transactions on Emerging Telecommunications Technologies* 31, 1 (2020), e3728.

[17] Chaehong Chung and Jaehoon Paul Jeong. 2020. A Design of IoT Device Configuration Translator for Intent-Based IoT-Cloud Services. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. IEEE, 52–56.

[18] Joaquin Chung, Sean Donovan, Jeronimo Bezerra, Heidi Morgan, Julio Ibarra, Russ Clark, and Henry Owen. 2019. Novel network services for supporting big data science research. *Future Generation Computer Systems* 98 (2019), 512–521.

[19] Douglas Comer and Adib Rastegatnia. 2018. OSDF: An intent-based software defined network programming framework. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. IEEE, 527–535.

[20] IDG Communications. 2020. *IDG Cloud Computing Survey 2020*. https://tinyurl.com/7swkmspd

[21] Armin Cremers and Seymour Ginsburg. 1975. Context-free grammar forms. *J. Comput. System Sci.* 11, 1 (1975), 86–117.

[22] Andrew Curtis-Black. 2021. *Network operator intent a basis for user-friendly network configuration and analysis*. Ph.D. Dissertation. University of Canterbury.

[23] Gianluca Davoli, Walter Cerroni, Slavica Tomovic, Chiara Buratti, Chiara Contoli, and Franco Callegati. 2019. Intent-based service management for heterogeneous software-defined infrastructure domains. *International Journal of Network Management* 29, 1 (2019), e2051.

[24] Nathan F Saraiva de Sousa, Danny Lachos Perez, Christian Esteve Rothenberg, and Pedro Henrique Gomes. 2021. End-to-end service monitoring for zero-touch networks. *Journal of ICT Standardization* (2021), 91–112.

[25] Kristina Dzeparoska, Hadi Bannazadeh, and Alberto Leon-Garcia. 2018. SDX-based security collaboration: Extending the security reach beyond network domains. In *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 63–71.

[26] Kristina Dzeparoska, Nasim Beigi-Mohammadi, Ali Tizghadam, and Alberto Leon-Garcia. 2021. Towards a Self-Driving Management System for the Automated Realization of Intents. *IEEE Access* 9 (2021), 159882–159907.

[27] Nour el houda Nouar, Sami Yangui, Noura Faci, Khalil Drira, and Saïd Tazi. 2021. A Semantic virtualized network functions description and discovery model. *Computer Networks* 195 (2021), 108152.

[28] Abdessalam Elhabbash, Gordon S Blair, Gareth Tyson, and Yehia Elkhatib. 2018. Adaptive service deployment using in-network mediation. In *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 170–176.

[29] Vincenzo Eramo, Mostafa Ammar, and Francesco Giacinto Lavacca. 2017. Migration energy aware reconfigurations of virtual network function instances in NFV architectures. *IEEE Access* 5 (2017), 4927–4938.

[30] Flavio Esposito, Jiayi Wang, Chiara Contoli, Gianluca Davoli, Walter Cerroni, and Franco Callegati. 2018. A behavior-driven approach to intent specification for software-defined infrastructure management. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 1–6.

[31] GSNFV ETSI. 2013. Network functions virtualisation (nfv): Architectural framework. *ETsI Gs NFV* 2, 2 (2013), V1.

[32] Thomas Favale, Francesca Soro, Martino Trevisan, Idilio Drago, and Marco Mellia. 2020. Campus traffic and e-Learning during COVID-19 pandemic. *Computer Networks* 176 (2020), 107290.

[33] Kai Gao, Luis M Contreras, and Sabine Randriamasy. 2020. Bi-directional Network and Application Interaction: Application Intents upon Abstracted Network Information. In *Proceedings of the Workshop on Network Application Integration/CoDesign*. 43–50.

[34] Pedro Henrique Gomes, Magnus Buhrgard, János Harmatos, Swarup Kumar Mohalik, Dinand Roeland, and Jörg Niemöller. 2021. Intent-driven closed loops for autonomous networks. *Journal of ICT Standardization* (2021), 257–290.

[35] Nour Gritli, Ferhat Khendek, and Maria Toeroe. 2021. Decomposition and Propagation of Intents for Network Slice Design. In *2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, 165–170.

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

[37] Arthur S Jacobs, Ricardo J Pfitscher, Rafael H Ribeiro, Ronaldo A Ferreira, Lisandro Z Granville, Walter Willinger, and Sanjay G Rao. 2021. Hey, Lumi! Using Natural Language for {Intent-Based} Network Management. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 625–639.

[38] Christopher Janz. 2015. Intent NBI. *Definition and Principles* (2015).

[39] Peter Jonsson, Steven Davis, Peter Linder, Amir Gomroki, Ali Zaidi, Anders Carlsson P, Miljenko Opsenica, Ida Sorlie, Sebastian Elmgren, Greger Blennerud, Harald Baur, Ritva Svenningsson, Brian Heath, Jim Bugel, Suja John, Stacy Schwartz, and Richard Moller. 2020. *Ericsson Mobility Report.* https://tinyurl.com/ym3cryj6

[40] Joon-Myung Kang, Jeongkeun Lee, Vasudevan Nagendra, and Sujata Banerjee. 2017. LMS: Label management service for intent-driven cloud management. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 177–185.

[41] Ouassim Karrakchou, Nancy Samaan, and Ahmed Karmouch. 2022. Mapping Applications Intents to Programmable NDN Data-Planes via Event-B Machines. *IEEE Access* 10 (2022), 29668–29686.

[42] Talha Ahmed Khan, Khizar Abbass, Adeel Rafique, Afaq Muhammad, and Wang-Cheol Song. 2020. Generic Intent-based Networking Platform for E2E Network Slice Orchestration & Lifecycle Management. In *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 49–54.

[43] Talha Ahmed Khan, Afaq Muhammad, Waleed Akbar, Asif Mehmood, Adeel Rafiq, and Wang-Cheol Song. 2021. Intent-based Networking Approach for Service Route and QoS control on KOREN SDI. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 24–30.

[44] Farzad Khodadadi, Amir Vahid Dastjerdi, and Rajkumar Buyya. 2016. Internet of things: an overview. In *Internet of Things*. Elsevier, 3–27.

[45] Jinyong Tim Kim, Eunsoo Kim, Jinhyuk Yang, Jaehoon Paul Jeong, Hyoungshick Kim, Sangwon Hyun, Hyunsik Yang, Jaewook Oh, Younghan Kim, Susan Hares, et al. 2020. IBCS: Intent-Based Cloud Services for Security Applications. *IEEE Communications Magazine* 58, 4 (2020), 45–51.

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

[47] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[48] Bikas Koley. 2019. *A Primer to Intent Driven Networking.* https://tinyurl.com/et5bydr7

[49] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. 2014. Software-defined networking: A comprehensive survey. *Proc. IEEE* 103, 1 (2014), 14–76.

[50] Himal Kumar, Hassan Habibi Gharakheili, Craig Russell, and Vijay Sivaraman. 2019. Enhancing Security Management at Software-Defined Exchange Points. *IEEE Transactions on Network and Service Management* 16, 4 (2019), 1479–1492.

[51] Takuya Kuwahara, Takayuki Kuroda, Takao Osaki, and Kozo Satoda. 2021. An intent-based system configuration design for IT/NW services with functional and quantitative constraints. *IEICE Transactions on Communications* 104, 7 (2021), 791–804.

[52] A Lerner, J Skorupa, and S Ganguli. 2017. *Innovation insight: Intent-based networking systems.* Technical Report. Gartner, Tech. Rep.

[53] Tong Liu, Franco Callegati, Walter Cerroni, Chiara Contoli, Maurizio Gabbrielli, and Saverio Giallorenzo. 2018. Constraint programming for flexible Service Function Chaining deployment. *arXiv preprint arXiv:1812.05534* (2018).

[54] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. 2018. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*. Springer, 103–130.

[55] Hocine Mahtout, Mariam Kiran, Anu Mercian, and Bashir Mohammed. 2020. Using Machine Learning for Intent-based provisioning in High-Speed Science Networks. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics.* 27–30.

[56] Antonio Marsico, Marco Savi, Domenico Siracusa, and Elio Salvadori. 2020. An automated negotiation framework for application-aware transport network services. *Optical Switching and Networking* 38 (2020), 100571.

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

[58] Asif Mehmood, Afaq Muhammad, Talha Ahmed Khan, Javier Jose Diaz Rivera, Javed Iqbal, Ihtesham Ul Islam, and Wang-Cheol Song. 2020. Energy-efficient auto-scaling of virtualized network function instances based on resource execution pattern. *Computers & Electrical Engineering* 88 (2020), 106814.

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

[60] Kashif Mehmood, HV Kalpanie Mendis, Katina Kralevska, and Poul E Heegaard. 2021. Intent-based Network Management and Orchestration for Smart Distribution Grids. In *2021 28th International Conference on Telecommunications (ICT).* IEEE, 1–6.

[61] Anu Mercian, Faraz Ahmed, Puneet Sharma, Shaun Wackerly, and Charles Clark. 2021. Mind the Semantic Gap: Policy Intent Inference from Network Metadata. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft).* IEEE, 312–320.

[62] Inder Monga, Chin Guok, John MacAuley, Alex Sim, Harvey Newman, Justas Balcas, Phil DeMar, Linda Winkler, Tom Lehman, and Xi Yang. 2018. SDN for end-to-end networked science at the exascale (SENSE). In *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS).* IEEE, 33–44.

[63] Ali Muhammad, Long Qu, and Chadi Assi. 2020. Delay-Aware Multi-Source Multicast Resource optimization in NFV-Enabled Network. In *ICC 2020-2020 IEEE International Conference on Communications (ICC).* IEEE, 1–7.

[64] Mina Nabi, Maria Toeroe, and Ferhat Khendek. 2016. Availability in the cloud: State of the art. *Journal of Network and Computer Applications* 60 (2016), 54–67.

[65] Vasudevan Nagendra, Arani Bhattacharya, Vinod Yegneswaran, Amir Rahmati, and Samir Das. 2020. An intent-based automation framework for securing dynamic consumer iot infrastructures. In *Proceedings of The Web Conference 2020.* 1625–1636.

[66] Navid Nazarzadeoghaz, Ferhat Khendek, and Maria Toeroe. 2020. Automated Design of Network Services from Network Service Requirements. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN).* IEEE, 63–70.

[67] Jörg Niemöller, Kevin McDonnell, James O'Sullivan, Dave Milham, Vinay Devadatta, Azahar Machwe, Wang Lei, Tayeb Meriem Ben, and Leonid Mokrushin. 2021. Intent Common Model Version 1.1.0. https://www.tmforum.org/resources/standard/ig1253a-intent-common-model-v1-1-0/

[68] Jörg Niemöller, Róbert Szabó, András Zahemszky, and Dinand Roeland. 2022. Creating Autonomous Networks with intent-based closed loops. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/creating-autonomous-networks-with-intent-based-closed-loops

[69] Ying Ouyang, Chungang Yang, Yanbo Song, Xinru Mi, and Mohsen Guizani. 2021. A Brief Survey and Implementation on Refinement for Intent-Driven Networking. *IEEE Network* 35, 6 (2021), 75–83.

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

[71] Jrgen Quittek, P Bauskar, T BenMeriem, A Bennett, and M Besson. 2014. Network functions virtualisation (nfv)-management and orchestration. *ETSI NFV ISG, White Paper* (2014), 0733–8716.

[72] Adeel Rafiq, Muhammad Afaq, and Wang-Cheol Song. 2020. Intent-based networking with proactive load distribution in data center using IBN manager and Smart Path manager. *Journal of Ambient Intelligence and Humanized Computing* (2020), 1–18.

[73] Rafael Hengen Ribeiro, Arthur Selle Jacobs, Ricardo Parizotto, Luciano Zembruzki, Alberto Egon Schaeffer-Filho, and Lisandro Zambenedetti Granville. 2020. A Bottom-Up Approach for Extracting Network Intents. In *International Conference on Advanced Information Networking and Applications*. Springer, 858–870.

[74] Mohammad Riftadi and Fernando Kuipers. 2019. P4i/o: Intent-based networking with p4. In *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 438–443.

[75] Mohammad Riftadi, Jorik Oostenbrink, and Fernando Kuipers. 2019. GP4P4: Enabling Self-Programming Networks. *arXiv preprint arXiv:1910.00967* (2019).

[76] Davide Sanvito, Daniele Moro, Mattia Gulli, Ilario Filippini, Antonio Capone, and Andrea Campanella. 2018. ONOS Intent Monitor and Reroute service: enabling plug&play routing logic. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 272–276.

[77] Nathan Saraiva, Nazrul Islam, Danny Alex Lachos Perez, and Christian Esteve Rothenberg. 2019. Policy-driven network traffic rerouting through intent-based control loops. In *Anais do XXIV Workshop de Gerência e Operação de Redes e Serviços*. SBC, 15–28.

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

[79] Eder J Scheid, Patrick Widmer, Bruno B Rodrigues, Muriel F Franco, and Burkhard Stiller. 2020. A Controlled Natural Language to Support Intent-based Blockchain Selection. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 1–9.

[80] Azimeh Sefidcon, Martin Julien, Róbert Szabó, Gemma Vall-llosera, and Per Persson. 2019. *Cloud evolution: The era of intent-aware clouds.* https://tinyurl.com/cwnu3j9c

[81] Yogesh Sharma, Bahman Javadi, Weisheng Si, and Daniel Sun. 2016. Reliability and energy efficiency in cloud computing systems: Survey and taxonomy. *Journal of Network and Computer Applications* 74 (2016), 66–85.

[82] Yogesh Sharma, Bahman Javadi, Weisheng Si, and Daniel Sun. 2017. Reliable and energy efficient resource provisioning and allocation in cloud computing. In *Proceedings of the10th International Conference on Utility and Cloud Computing*. 57–66.

[83] Yogesh Sharma, Michel Gokan Khan, Auday Al-Dulaimy, Mohammad Ali Khoshkholghi, and Javid Taheri. 2020. Networking models and protocols for/on edge computing. *Edge Computing: Models, Technologies and Applications* 33 (2020), 77.

[84] Yogesh Sharma, Michel Gokan Khan, Javid Taheri, and Andreas Kassler. 2020. Performance Benchmarking of Virtualized Network Functions to Correlate Key Performance Metrics with System Activity. In *2020 11th International Conference on Network of the Future (NoF)*. IEEE, 73–81.

[85] Yogesh Sharma, Weisheng Si, Daniel Sun, and Bahman Javadi. 2019. Failure-aware energy-efficient VM consolidation in cloud computing systems. *Future Generation Computer Systems* 94 (2019), 620–633.

[86] Yogesh Sharma, Javid Taheri, Weisheng Si, Daniel Sun, and Bahman Javadi. 2020. Dynamic Resource Provisioning for Sustainable Cloud Computing Systems in the Presence of Correlated Failures. *IEEE Transactions on Sustainable Computing* (2020).

[87] Zhan Shi, Ying Zeng, and Zanhong Wu. 2021. Service Chain Orchestration Based on Deep Reinforcement Learning in Intent-Based IoT. In *Proceedings of the 9th International Conference on Computer Engineering and Networks*. Springer, 875–882.

[88] Pontus Sköldström, Stéphane Junique, Abdul Ghafoor, Antonio Marsico, and Domenico Siracusa. 2017. DISMI-an intent interface for application-centric transport network services. In *2017 19th International Conference on Transparent Optical Networks (ICTON)*. IEEE, 1–4.

[89] John F Sowa. 1987. Semantic networks. (1987).

[90] Yu-Wei Eric Sung, Xiaozheng Tie, Starsky HY Wong, and Hongyi Zeng. 2016. Robotron: Top-down network management at facebook scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 426–439.

[91] Stan Szpakowicz, Anna Feldman, and Anna Kazantseva. 2018. Computational Linguistics and Literature. *Frontiers in Digital Humanities* 5 (2018), 24.

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

[93] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, et al. 2019. Safely and automatically updating in-network ACL configurations with intent language. In *Proceedings of the ACM Special Interest Group on Data Communication*. 214–226.

[94] TmForum. 2021. Autonomous Networks Technical Architecture. https://www.tmforum.org/resources/standard/ig1230-autonomous-networks-technical-architecture-v1-1-0/

[95] Adel Nadjaran Toosi, Rodrigo N Calheiros, and Rajkumar Buyya. 2014. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–47.

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

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

[98] Benjamin E Ujcich, Adam Bates, and William H Sanders. 2020. Provenance for intent-based networking. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 195–199.

[99] Ricard Vilalta, Ramon Casellas, Ricardo Martínez, Raul Munoz, Young Lee, Haomian Zheng, Yi Lin, Victor López, and Luis Miguel Contreras. 2018. Fully automated peer service orchestration of cloud and network resources using ACTN and CSO. In *2018 International Conference on Optical Network Design and Modeling (ONDM)*. IEEE, 124–129.

[100] Huazhe Wang. 2019. *Enhancing Automated Network Management.* Ph.D. Dissertation. UC Santa Cruz.

[101] Lin Wang, Lei Jiao, Ting He, Jun Li, and Max Mühlhäuser. 2018. Service entity placement for social virtual reality applications in edge computing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 468–476.

[102] Yuhang Wang, Zhihong Tian, Yanbin Sun, Xiaojiang Du, and Nadra Guizani. 2020. LocJury: An IBN-Based Location Privacy Preserving Scheme for IoCV. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[103] Yiming Wei, Mugen Peng, and Yaqiong Liu. 2020. Intent-based networks for 6G: Insights and challenges. *Digital Communications and Networks* 6, 3 (2020), 270–280.

[104] Chao Wu, Shingo Horiuchi, and Kenichi Tayama. 2019. A Resource Design Framework to Realize Intent-Based Cloud Management. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE Computer Society, 37–44.

[105] Yuming Wu, Nishok Narasimha Mohanasamy, Lalita Jagadeesan, and Muntasir Raihan Rahman. 2021. Changes in Intent: Behavioral Predictions of Distributed SDN Controller Reconfiguration. In *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 433–438.

[106] Hui Yang, Kaixuan Zhan, Qiuyan Yao, Xudong Zhao, Jie Zhang, and Young Lee. 2020. Intent defined optical network with artificial intelligence-based automated operation and maintenance. *Science China Information Sciences* 63 (2020), 1–12.

[107] Jinhyuk Yang and Jaehoon Paul Jeong. 2018. An automata-based security policy translation for network security functions. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 268–272.

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

[109] Jiaming Zhang, Junjie Guo, Chungang Yang, Xinru Mi, Libin Jiao, Xiaoming Zhu, Lihui Cao, and Ruixing Li. 2021. A Conflict Resolution Scheme in Intent-Driven Network. In *2021 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 23–28.

[110] Xiaoang Zheng and Aris Leivadeas. 2021. Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques. In *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 14–20.