ARTICLE

CONDENSE: Cognitive intent-driven end-to-end network slicing with AI planning agents

Ajay Kattepur[1], Ian Burdick[2], Swarup Mohalik[1], Marin Orlic[3], Leonid Mokrushin[3]

[1] Ericsson Research AI, Bangalore, India, [2] Ericsson SA Cognitive Network Solutions, Texas, USA, [3] Ericsson Research AI, Kista, Sweden

Corresponding author: Ajay Kattepur, ajay.kattepur@ericsson.com

Autonomous network management has gained momentum as a key feature of 5G advanced and 6G systems. Autonomy builds on the use of intent-driven networks that involve intelligent agents to configure various network sub-domains. In addition, the concept of end-to-end network slicing would need to be appropriately managed to provide differentiated Quality of Service (QoS) within 5G/6G networks. However, there is limited work looking at intent-driven end-to-end network slice assurance. In this paper, we propose CONDENSE, a cognitive intent-driven system for network slice fulfilment and assurance. Via the use of Artificial Intelligence (AI) planning agents, we demonstrate the decomposition of high-level intent requirements to individual sub-intents. AI planning makes use of machine reasoning techniques to search for issue resolutions that differs from other data-driven approaches. AI planning agents are employed for resource allocation at the Radio Access Network (RAN), transport and core domains to ensure intent fulfilment. The system is implemented over a cognitive intent management system to demonstrate end-to-end network slicing.

Keywords: AI planning, cognitive reasoning, intent management function, network slicing, neurosymbolic reasoning

1. INTRODUCTION

The evolution of 5G towards 6G has led to multiple use cases requiring service differentiation. A commonly proposed concept over recent years is the use of network slicing [1], to meet differentiated service requirements of mobile broadband, gaming, enterprise and eXtended Reality (XR). Network slicing achieves this differentiation by creating virtual segregated subnets over the same physical network to meet service requirements.

As part of the network slice lifecycle [2], network slices cannot be statically configured and require service assurance [3]. With the large scale deployments of 5G and 6G systems, the end-to-end subnets have to be efficiently designed, managed and assured. This process would involve translating the requirements to individual RAN, transport and core sub-domains. Due to the complexity and large scale deployments, the configurations have to be autonomous rather than expert managed.

Autonomous management of 5G/6G networks is a vision requiring multiple futuristic technologies [4]. To this end, intent-driven [5] systems have been proposed that move away from imperative networks to declarative management. Intents are the formal requirements that must be satisfied by the network. Industry fora such as TM Forum have suggested standardized intent definitions that are interoperable across multiple Communication Service Providers (CSPs) [5]. Standardized intent definition enable intents (or sub-intents) to be passed between CSPs for issue resolution and management. In order to create autonomous networks, another piece of the puzzle are AI agents. These agents can participate in multiple phases of autonomous intent management including data grounding, resolution proposals, prediction and reporting. An implementation of this Intent Management Function (IMF) has been proposed via a cognitive intent management framework [6]. This system also falls in line with 6G AI native architectures [7].

While the intent definitions have reached some amount of maturity [5], the process of handling intents is yet to be fully demonstrated for 5G/6G network slicing assurance. Intents have to be appropriately handled to generate target goals towards RAN, transport and core levels. In addition, the IMFs are traditionally implemented on specific subnets, which require an additional layer of service layer coordination. IMFs also have rule-based agents that are to be replaced by AI-driven agents for superior adaptation.

To clarify these issues, we consider the following three problems.
Problem 1: Decomposition of high-level service expectations to requirements at individual subnets via intent-interfaces has not been demonstrated.
Problem 2: The ability of AI agents to autonomously assure end-to-end slice requirements with appropriate actions has not been suitably studied.
Problem 3: Feedback from lower intent management layers and closed loop control has not been considered.
These problems require novel solutions, that are the main focus of this paper.

To solve these problems, we propose CONDENSE, a cognitive intent-driven framework for end-to-end network slicing. Using intent-driven network concepts, slice service assurance is provided across RAN, transport and core domains. AI planning agents [8] are used to decompose high-level intents to low level requirements that can be handled within each domain. AI planning is once again used to propose efficient resource allocations to meet intent expectations. AI planning makes use of symbolic reasoning techniques to resolve the slice assurance issues, as opposed to traditional data-driven machine learning techniques. Queueing network models [9] are further used to evaluate the efficacy of the proposals prior to network actuation. This approach is demonstrated over a realistic network slicing use case involving multiple intents.

The principal contributions of this paper include:

1. Solve the end-to-end intent-driven network slicing problem using novel AI-driven planning agents and standard-compliant interfaces.
2. Utilize AI planning agents to hierarchically decompose service intents to subnet-specific intents.
3. Demonstrate end-to-end network slicing by using appropriate AI plan proposal and evaluation agents at RAN, transport and core levels. AI planning agents use symbolic reasoning as opposed to data-driven approaches.
4. Evaluate the system over a realistic case study from a mobile network operator.

The rest of the paper is organized as follows: Related work is reviewed in Section 2. Section 3 provides a brief background on network slicing challenges. Slice management mapped to intent management functions are presented in Section 4. Section 5 provides an overview of the CONDENSE system. Section 6 presents the service IMF decomposition. The RAN, transport and core intent management functions and associated agents are described in sections 7, 8 and 9, respectively. Experimental evaluation of the network slicing techniques are presented in Section 10. This is followed by conclusions and future directions in Section 11.

2. RELATED WORK

We provide a brief overview of the state of the art on intent-driven networking, network slice assurance and 6G AI native systems.

2.1 Intent-driven networking

As specified in [5, 29], an intent is associated with two parties: intent owner that creates and manages the intent and intent handler that fulfils the requirements within an autonomous domain. Intents may be formally defined within RDF graphs as a way to structure intent models and extract their semantic correlation. The use of Natural Language Processing (NLP) and large language models is also being explored in the intent space. In [30], the proposed NLP tool can prompt the user to add the missing information or generate intent grammar syntax. In [6], a cognitive core solution is specified that can handle intent-driven specifications. In [3], a comprehensive analysis of the state of the art in intent-driven networks including the intent description models, intent lifecycle management and a generalized architectural framework is presented. In [12], an overview of decomposing standardized intents towards autonomous domains is provided. In [31], intent-specific automation pipelines using closed loop micro-services with self-declared capabilities are presented.

2.2 Network slice assurance

Table 1 provides an overview of the state of the art in intent-driven network slice assurance. The first set of papers [10, 11] in Table 1 focus on the need for end-to-end slicing without specifying intent-driven techniques. The techniques typically cover multiple subnets such as RAN, transport and core. Though the use of automation techniques are mentioned, intent-driven techniques are not covered.

The second set of papers in Table 1 [12, 13] provide an overview of intent-driven network slicing techniques. The focus of these papers is to describe intent specification techniques and how they are broken down to

Table 1 – State of the art in intent-driven network slice assurance.

No.	Paper	Description	Slicing Domains	Intent Techniques	AI Algorithms
1.	End-to-end-slicing [10, 11]	Value proposition for end-to-end network slicing presented with a description of slice lifecycle management.	RAN, transport and core domains	Though automation of slice fulfilment and assurance is mentioned, specific intent-driven techniques not considered.	No specific mention of AI agents.
2.	Intent-driven network slice management [12, 13]	Architecture view and hierarchical decomposition of service intents.	RAN, transport and core domains.	Business intent, service intent, and resource intent, which is useful to address high complexity	Agents not described.
3	RAN slicing [14, 15, 16, 17, 18]	Intent-driven RAN slice design and assurance via dynamic allocation of physical resource blocks	RAN	Service and operational intents towards the RAN domain	AI planning proposal agents with machine learning forecast and grounding. Reinforcement learning for dynamic allocation of RAN partitions.
4	Transport slicing [19, 20], Cisco [21]	To ensure differentiated services within the transport domain, AI-driven strategies and software defined networks have been proposed.	Transport	Intents on service performance mapped to transport domains for throughput and latency	Reinforcement learning agents [19, 20] and policies [21]
5	Core slicing [22, 23], Juniper Astra [24]	Agents are developed for fat tree network management; intent-driven policies for management of data center pods	Core	Intents mapped to requirements on pods and fat tree networks.	Multi-agent reinforcement learning and policies [24].
6	ML agents for slicing [25, 26, 27]	A Deep Reinforcement Learning (DRL)-based network slicing technique to find out the resource allocation policy. Application of ML approaches for autonomous management of resources in the network slicing paradigm.	RAN and transport	Intents not specifically considered	Deep reinforcement learning and machine learning techniques.
7.	LLM intents [28]	Intents translated from natural language for deployment, assurance, feasibility and reporting	RAN, transport, core	Intents described in natural language translated to 3GPP specifications	LLM agents.
8.	CONDENSE	Intents translated from natural language for deployment, assurance, feasibility and reporting	RAN, transport, core	Intents described in TM-Forum specifications are hierarchically broken down.	Symbolic AI planning agents and queueing network models.

requirements at lower subnets. However, the agents and reasoning techniques to implement these intent-driven techniques are not elaborated.

The set of papers 3 – 5 in Table 1 look at domain-specific slicing. RAN slicing is presented in [14, 15, 16] with a focus on reinforcement learning and planning agents. Here, the intents are translated to expectations on throughput or latency at the RAN level. Physical Resource Block (PRB) partitioning, priorities or rate limitations may be applicable techniques. Transport slicing is provided in [19, 20] with the use of reinforcement learning for router/switch configuration. The paper [19] looks at fine-grained changes within the router such as queue load balancing or buffer size changes to ensure that the internal configurations meet intent requirements. Similarly, agents are developed for core slicing in [22, 23]. Actions such as pod load balancing, migration and scheduling may be done via agent frameworks. Multi-agent RL policies are further developed in [24] to control complex data center networks.

The sixth set of papers in Table 1 apply ML and RL strategies to network slicing, without specifically using intent-driven approaches. The use of deep reinforcement leaning agents for network slice resource allocation are specified in [25, 26]. The more recent [27] makes use of Graph Neural Network (GNN) techniques to perform slice assurance within RAN and mobile edge computing nodes. However, it does not make use of standardized intents to change requirements. Furthermore, GNNs require a significant amount of training data. Comparatively, AI planning techniques can start off with limited domain knowledge and proposed solutions.

Recent strategies also look at Large Language Model (LLM) agents [28] for intent translation. Intents can be specified in natural language and converted to 3GPP

or TM Forum specifications. LLMs are also useful for mapping requirements efficiently.

CONDENSE is one of the first papers to look at end-to-end intent-driven slice assurance with prototyped agents. We are compliant with intent-driven standards while incorporating novel AI agents for intent handling. Actions specified at the RAN, transport and core subnets are also done in an autonomous fashion. The system is developed using novel AI planning agents that can use symbolic domain knowledge, to reason about autonomous network slicing actions.

2.3 6G and AI-driven systems

The use of intents, AI-driven automation and data-driven training has received prominence with novel 6G architectures [7]. This process has led to a rethinking of the use of AI within networking architectures. As specified in [32], slicing may be incorporated within Industry 4.0 to create ad hoc customized Network Slices Templates for digital transformation, such as robots and IoT devices. The use of intent-based automation towards 6G open RAN has been presented in [33]. This paper builds on these concepts by providing concrete details on how intents, agents and slice assurance may be integrated within a unified framework.

2.4 Neurosymbolic AI

Data-driven techniques such as deep learning, though efficient in finding patterns for classification and regression, fail to extract compositional and causal structures from data. Symbolic reasoning techniques are efficient in representing these compositional and causal structures. Neurosymbolic AI [34] aims to combine neural networks with symbolic reasoning techniques. This hybrid approach enables machines to reason symbolically while also leveraging the pattern recognition capabilities of neural networks.

Example of neurosymbolic methods include Logic Neural Networks (LNNs) [34]. An LNN consists of a neural network trained to perform symbolic reasoning tasks, such as logical inference, theorem proving, and planning, using a combination of differentiable logic gates and differentiable inference rules. These gates and rules are designed to mimic the operations performed by symbolic reasoning systems and are trained using gradient-based optimization techniques [34]. Knowledge graphs are also a useful formalism to represent numerical embeddings from which patterns could be extracted. Recent studies have tried to integrate expert knowledge into knowledge graphs to enable superior reasoning and learning for neurosymbolic approaches [35].

In this paper, we make use of symbolic reasoning techniques, such as AI planning, to create intent decompositions and network configurations. The AI planning techniques are combined with grounded forecasting agents that make use of regression methods. The combination of such neurosymbolic agents would lead to scalable and long-term reasoning for the management of 6G networks.

2.5 CONDENSE

When compared to the state of art, this paper is one of the first ones to apply AI planning agents for service intent decomposition, resource allocation and optimization. We further demonstrate end-to-end network slicing by using appropriate proposal and evaluation agents at the RAN, transport and core level. The use of AI-planning is an alternative approach compared to traditional data-driven agents. Such a detailed analysis of AI-driven network slicing would be useful in advanced 5G and 6G deployments. In addition, the standardized intent specifications ensure that the system can be deployed in vendor-agnostic environments.

CONDENSE is shown to solve the following problems: (i) decomposition of high-level service expectations to requirements at individual subnets using agents; (ii) end-to-end slice assurance across the RAN, transport and core; (iii) feedback from lower intent management layers and closed loop control.

3. END-TO-END NETWORK SLICING

Network slicing [1, 10] has been proposed as a key technology to provide differentiated QoS within 5G advanced and 6G networks. The concept of network slicing involves creating virtual dedicated pools of network resources to serve different types of flows such as mobile broadband, extended reality, gaming and Internet of Things (IoT).

3.1 Network slice lifecycle

As seen in Fig. 1, there are multiple phases in instantiating a network slice [2]:

- After the initial stakeholder onboarding and network preparation, the fulfilment phase includes the creation of the slice instance. During the network slice instance creation all needed resources are allocated and configured to satisfy the network slice requirements.
- The operation phase includes the activation, supervision, performance reporting, resource capacity planning and modification of a network slice instance.

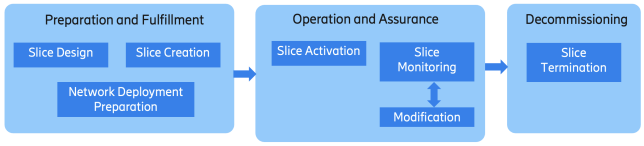

Figure 1 – Lifecycle of network slice provisioning and fulfilment.

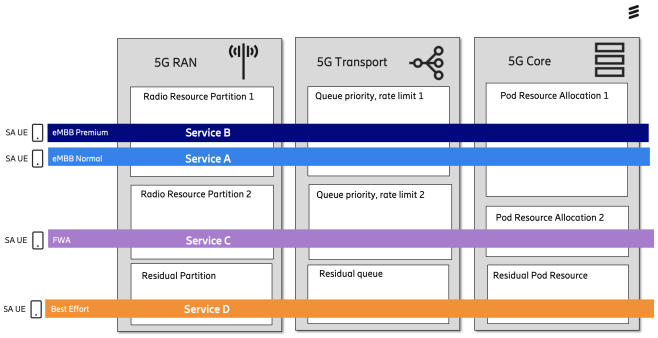

Figure 2 – End-to-end network slicing.

– The slice may be decommissioned to provide additional capacity for other services.

This operation phase can have dynamic intent [3] requirements, that needs to be assured. Resource capacity planning includes any actions that calculate resource usage based on a network slice instance provisioning, and performance monitoring and generates modification policies as a result of the calculation.

As seen in Fig. 2, there can be multiple services such as premium enhanced Mobile Broadband (eMBB), normal eMBB, Fixed Wireless Access (FWA) and best effort services. Resources are to be provisioned at the RAN, transport and core domain levels to assure the required slice service guarantees. These actions could involve radio resource partitioning at the RAN, rate limitation and prioritization at the transport and container pod scaling at the core. The residual pool of resources will be shared among non-sliced traffic.

3.2 3GPP network slice specification

Fig. 3 provides a representation of the 3GPP network slice management architecture [36]. It consists of the following key components:

1. *Communication Service Management Function (CSMF)*: This function acts as the interface between service order management and Operations Support Systems (OSS).
2. *Network Slice Management Function (NSMF)*: manages the lifecycle of the end-to-end slice across the network domains: RAN, core network and transport network.
3. *Network Slice Subnet Management Function (NSSMF)*: manages the lifecycle of the network slice subnets within a network domain.

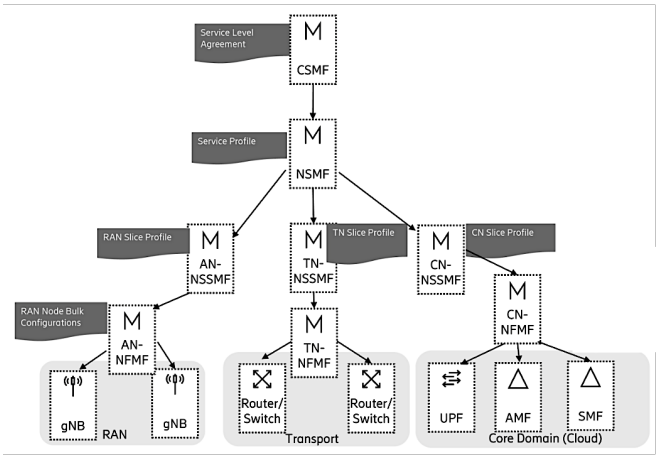

Figure 3 – Network slicing hierarchy.

4 *Network Function Management Function (NFMF)*: centralized platform for managing and orchestrating network functions. Some of the responsibilities of NFMF include: (i) Configuration: provisioning and configuring network functions based on operator requirements; (ii) monitoring: collecting and analyzing performance data from network functions and infrastructure components; (iii) scaling: dynamically adjusting the capacity of network functions to handle varying workloads.

In addition to these specifications, slice instantiation and identification also has standardized terms [1]:

– *Network Slice Instance (NSI)*: a set of network function instances and the required resources (e.g. compute, storage and networking resources) which form a "deployed" network slice.
– *Network Slice Subnet Instance (NSSI)*: an instance of network slice subnet that allocates resources to the slice.
– *Single Network Slice Selection Assistance Information (S-NSSAI)*: it identifies a network slice. S-NSSAI includes the Slice/Service Type (SST) and Slice Differentiator (SD). Requirements to be satisfied by the service may be associated here including throughput, latency and packet loss targets.

As specified in 3GPP standards [37] [38], there are multiple counters that may be collected to measure end-to-end slice metrics. A subset of these relevant metrics are provided in Table 2. These requirements on throughput, latency and resource utilization are the typical constraints provided to intent management functions.

4. NETWORK SLICE MANAGEMENT VIA IMF HIERARCHIES

Intents are the formal specification of requirements to be met by the system [5]. The difference between an intent and policy-based management arises from the level of

Table 2 – 3GPP metrics for network slice performance.

Metric	Description
UL/DL Delay NR–SNw	Average packet transmission delay through the RAN part to the UE.
UL/DL Delay gNBDU Ns	Average packet transmission delay through the gNB-DU part to the UE
Delay E2E UL/DL Ns	Average e2e DL packet delay between the PSA UPF and the UE for a network slice.
UL/DL UE Throughput–Cell	Average DL RAN UE throughput for a NRCellDU.
PDUSesMeanNbr	Mean number of PDU sessions that are successfully established in a network slice .
VirtualResUtilization	Utilization of virtualized resource (e.g. processor, memory, disk) that are allocated to a network slice
UL/DL Total PRB Usage	This measurement provides the usage (in percentage) of physical resource blocks (PRBs) on the downlink for any purpose

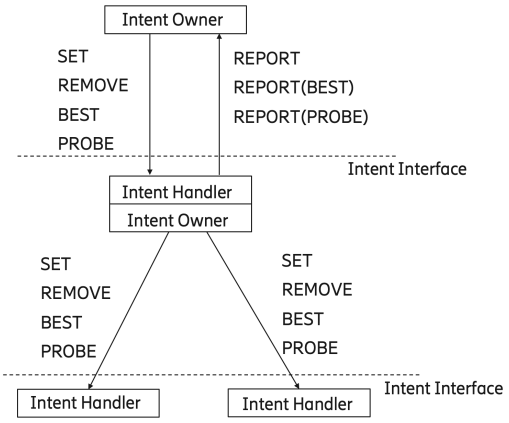

Figure 4 – Intent APIs between the intent owner and intent handler.

detail specified with the defined goals. Intent-based management consists of providing requirements and does not include additional information regarding the implementation of the rules.

To achieve the envisioned concept of autonomous, intent-driven networks, the TM forum standards have proposed Intent Management Functions (IMFs) [5]. The IMFs may be deployed throughout the network and have their own management zones. As seen in Fig. 4, there are two parties within an IMF, the intent owner and the intent handler. The intent is set by the intent owner via the intent interface and complied to with the intent handler. IMFs may be organized hierarchically and communicate with each other using intent Application Programming Interfaces (APIs). An IMF registry provides details of connected IMFs and their capabilities. The operations at the intent interface include (Fig. 4):

1. `SET`: Send a new or modified intent to an intent handler.
2. `REMOVE`: Withdraw an intent.
3. `REPORT`: Report the intent handling status.
4. `PROBE`: Ask the handler to estimate the potential success of an intent.
5. `BEST`: Ask the handler for the best intent expectation that it can successfully handle (i.e., the most severe requirement the handler would be able to successfully comply with).

Mapping the end-to-end network slice management to IMFs has not been done previously. Fig. 5 provides a hierarchy of IMFs that may be exploited for the network slicing use case. The service IMF receives service intent requirements (e.g. throughput, latency) that has to be assured by the system. This view maps to the S-NSSAI requirements to be fulfilled. The service IMF is responsible for decomposing the service intents to the RAN, transport and core intents (further details in Section 6). Once the intents are received by the RAN, transport and core subnets, individual intent resolution actions and sub-slice management (NSSMF) may be done. Actions over the network resource as provided by the NFMF and can be done per sub-domain. Details of the action space, intent management and knowledge graphs related to the RAN, transport and core subnets are specified in sections 7, 8 and 9 respectively.

4.1 Intent Management Function (IMF) internals

In order to implement the IMF, a cognitive intent handling framework has been proposed [6]. It consists of three essential components (Fig. 6):

1. *Knowledge base*: contains the ontology of intents along with domain-specific knowledge such as the current state of the system.
2. *Reasoning engine*: domain-independent reasoning engine serves as the central coordinator function and uses the knowledge graph to orchestrate a number of registered agents for finding solution actions, evaluating their impact and ordering their execution.
3. *Agent interfaces*: allows any number of agent models and services to be used. Agents can contain machine-learned models or rule-based policies for implementing services in the cognitive reasoning process.

In the IMF, the closed loop is implemented by a set of agents operating on the knowledge base and orchestrated by the reasoner. Here are the possible agents (Fig. 6):

– Data grounding agents: These agents are responsible for grounding data external to the IMF.
– Goal setting: These agents map issues with intent expectations to goals that the IMF needs to resolve.
– Proposal agents: These agents analyze the goals, and propose one or more solutions. Each solution can be a combination of direct actions on the underlying network elements or further intents that can be delegated, e.g., to other IMFs.
– Prediction agents: These agents analyze the proposals from the proposal agent and make predictions, based on information stored in the knowledge base and with the help of the reasoner or AI/ML models, about the possible impact of the proposals on all the IMF targets.

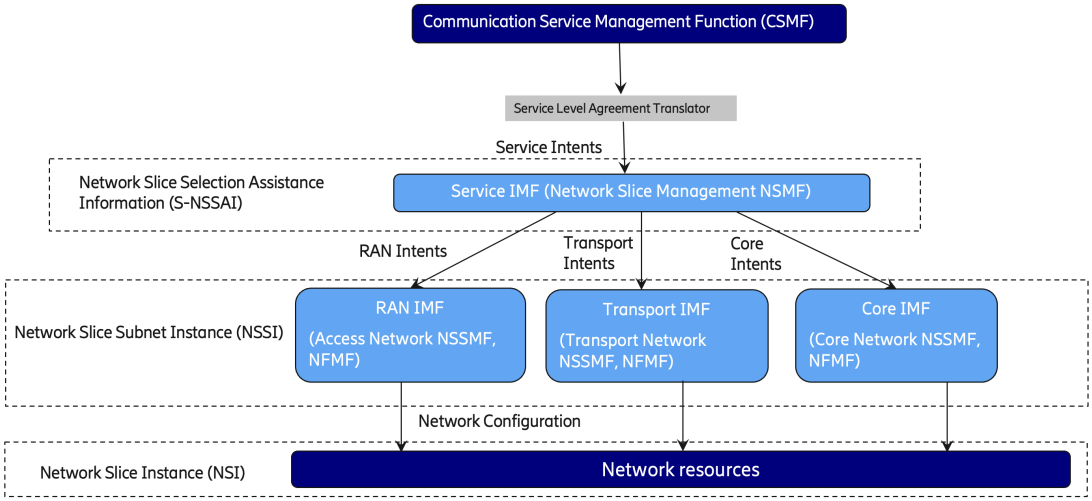

Figure 5 – Hierarchy of IMFs mapped to network slice management.

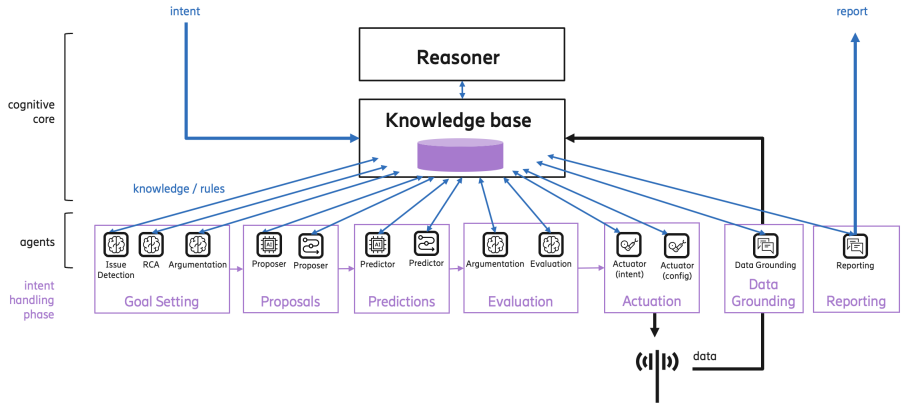

Figure 6 – Cognitive intent management with agents and phases.

– Evaluation agents: These agents take the predictions from the prediction agents and select the more suitable action (e.g., by approving actions maximizing the global utility of the IMF; this global utility is understood as the utility of all the intents accepted by the IMF).
– Actuation agents: These agents receive approved actions from the evaluation agents to implement the solution (e.g., direct actuation or intent setting).
– Reporting: These agents collect the reports from other IMFs and forward them to the cognitive core.

As presented in [39], there are four basic categories of agents.

1. **Simple reflex agent**: These are the simplest agents that select actions based on the current sensory perception. In most cases, these agents may be implemented using condition-action rules that are triggered with events. These agents do not store any knowledge of environment nor do they receive feedback.
2. **Model-based reflex agents**: These agents maintain an internal state of the system dynamics. It does this in two steps: (i) information on how the world evolves independent of the agent; (ii) information of the effect of the agent's action on the world.
3. **Goal-based agents**: Knowledge about the current state of the environment is not always enough for decision-making. Agents require goal information that describes situations that are desirable. The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal. Sometimes goal-based action selection is straightforward; for example, when goal satisfaction results immediately from a single action. Sometimes it will be more involved that requires searching the state space and planning action sequences. In addition, utility functions may be provided, which is a performance measure of the actions of the agent. The agent can try to maximize the utility function outcomes.
4. **Learning agents**: Learning agents have a learning element that uses feedback from a critic to determine improvements in performance. The learning element can suggest actions that may lead to new experiences. Thus, the agent may trade off suboptimal actions in the shorter term, in order to discover much better actions for the long run.

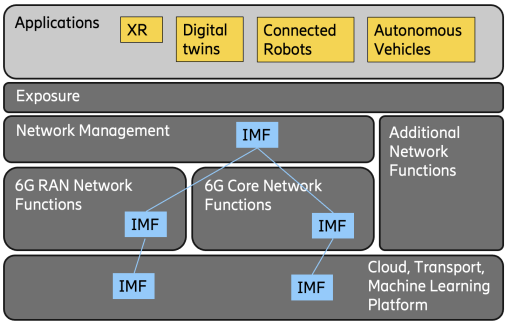

Figure 7 – AI-native 6G architecture with hierarchical IMFs.

In this work, we make use of *goal-based agents* that make use of AI planning and scheduling techniques. Domain knowledge with preconditions and effects of actions are used to determine autonomous actions to resolve intent issues.

4.2 6G native and intent-driven automation

The use of intents and IMFs is an important concept in the future 6G platform architecture, as seen in Fig. 7. As specified in the AI-native view of 6G [7], intents are an important input that need to be handled by autonomous networks. As further seen in Fig. 7, scalable deployments have a hierarchy of IMFs with specific scopes in terms of, for example, a network domain (RAN, transport, etc.) or geographical/administrative domains. This includes transforming the set of global, high-level intents into lower-level intents, and in the end detailed configurations.

In addition, these IMFs would have multiple agents (AI and rule-based) that would participate. This would mean that AI execution and training environments need to be available throughout the network. When the number of AI models in the network grows, their lifecycle management needs to be fully automated, deciding, for example, which model version to use for execution, and where and when to train a model. Models may require data originating from several layers and network domains, which may imply that layer and domain borders blur.

Soto et al. [40] propose a network intelligence stratum for 6G. This stratum proposes an orchestrator that supports closed-loop network intelligence operations across various network domains. Issues related to scalability, conflict resolution and effective data management are mitigated through this hierarchical network intelligence. The use of hierarchical intent decomposition within 6G networks has also been recently presented [41]. Agents are used to efficiently decompose requirements from the network management IMF to the RAN and core subnet IMFs. We draw on these concepts to implement the Condense system.

5. CONDENSE SYSTEM

To provide a cognitive intent-driven solution to end-to-end slice management, Fig. 8 presents the Condense system with the following components:

- The service IMF receives service intents (throughput, latency targets) that the end-to-end network slice has to fulfil. As specified previously, the IMF is implemented via a cognitive intent management framework. Multiple agents may be registered here: (i) grounding agents to provide monitored/forecasted network performance metrics; (ii) AI planning-based proposal agents to decompose the service intents to expectations at lower layers; (iii) an evaluation agent to evaluate the efficacy of the decomposition; (iv) actuation agents to initiate sub-intents to lower subnets. The intent registry information is used to determine the connected IMFs.
- The RAN IMF receives the decomposed intent and is responsible for assuring the RAN subnet performance. This IMF also has different agents to complete actions. Specifically, the AI planning agents would be for allocating RAN resources optimally to meet intent requirements. In addition, queueing network-based evaluation agents will be used to estimate side effects or efficacy of proposals.
- The transport IMF receives the decomposed intent and is responsible for assuring the transport subnet performance. This IMF also has different agents to complete actions. Specifically, the AI planning agents would be for allocating transport resources optimally to meet intent requirements.
- The core IMF receives the decomposed intent and is responsible for assuring the core subnet performance. This IMF also has different agents to complete actions. Specifically, the AI planning agents would be for allocating core resources optimally to meet intent requirements.

Once each of the subnets complete the actions, the service IMF receives feedback on the completion of the intent expectation.

As presented in Fig. 8, there are in fact multiple closed loops that interact with each other. This hierarchy of closed loops has also been presented in [42]. The lower loops at the RAN, transport and core IMFs are responsible for setting new configurations with changes in network state or intents. The upper level loop can set or reset decompositions based on feedback received from lower level IMFs. Note that we are compliant with TM Forum interface specifications between different IMFs in the hierarchy.

Note that while we have showcased just a single IMF at the RAN, transport and core levels, more complex and scalable 6G deployments could have further decomposi-

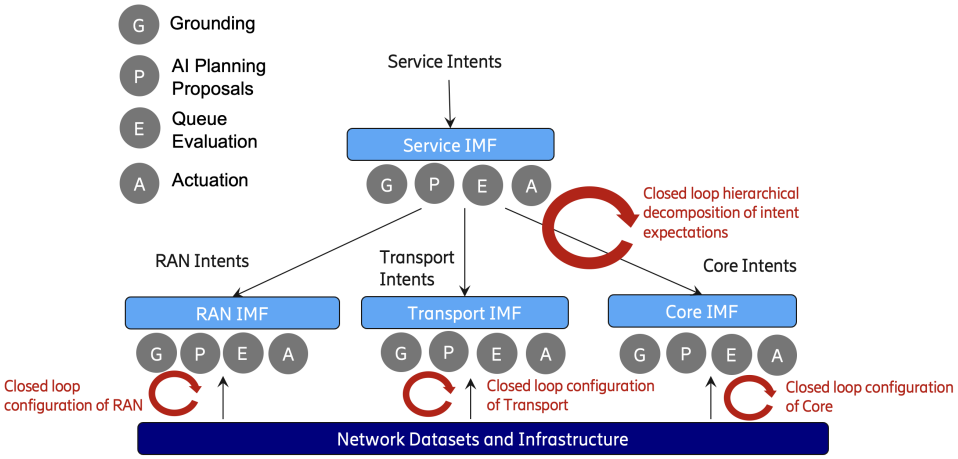

Figure 8 – CONDENSE system with hierarchical intent management.

Table 3 – Agents in the CONDENSE system.

Domain	Agents
Service IMF loop	1. Grounding agent to provide current state of network and **BEST** report from lower IMFs. 2. Proposal agent to decompose expectations. 3. Evaluation agent to rank possible decompositions. 4. Actuation agent to set targets to lower-level IMFs.
RAN IMF loop	1. Grounding agent to provide current or forecasted state of RAN network. 2. Proposal agent to propose planned actions such as radio resource partition change. 3. Evaluation agent to check efficacy and side effects of actions. 4. Actuation agent to set configuration on RAN network.
Transport IMF loop	1. Grounding agent to provide current or forecasted state of transport network. 2. Proposal agent to propose planned actions such as rate limit change. 3. Evaluation agent to check efficacy and side effects of actions. 4. Actuation agent to set configuration on transport network.
Core IMF loop	1. Grounding agent to provide current or forecasted state of core network. 2. Proposal agent to propose planned actions such as pod scaling change. 3. Evaluation agent to check efficacy and side effects of actions. 4. Actuation agent to set configuration on core network.

tions of IMFs based on: (i) geographic or administrative zones that are logically monitored with a RAN or core subnet to ensure monitored network KPIs are reliably captured; (ii) subset of configurations that can be divided between configuring applications such as rApps[1]. The CONDENSE system can be extended to these additional hierarchies as well.

As presented in Table 3, there are 16 agents that interact between the service, RAN, transport and core IMF levels. These agents continuously run in individual closed loops. Agents associated with the service IMF ensure that the intents are decomposed. Agents associated with the RAN, transport and core IMFs ensure configurations of individual subnets.

[1] https://www.ericsson.com/en/ran/intelligent-ran-automation/intelligent-automation-platform/rapps

5.1 AI planning

As many of the proposal agents at the service, RAN, transport and core layers make use of AI planning, we will provide a brief overview of AI planning techniques.

AI planning [8] is a branch of artificial intelligence that concerns the realization of strategies or action sequences, to automate the solution towards a specified goal. AI planning begins with the definition of domains, plans and goals that are to be achieved.

Definition 1 Planning domain: A planning domain is a state transition system $\Sigma = (S, A, \gamma, C)$, where:

- S is a finite set of states of the system.
- A is a set of actions that may be performed by an agent.
- $\gamma : S \times A \rightarrow S$ is the state transition function. If $\gamma(s, a)$ is defined, then action a is applicable to state s, with $\gamma(s, a)$ being the predicted outcome.

– $C : S \times A \rightarrow [0, \infty)$ is a cost function with the same domain as γ. It can represent a cost function modeling monetary cost, latency or parameters within the system.

Definition 2 Plan: A plan is a finite set of actions:

$$\pi = \langle a_1, a_2, \ldots, a_n \rangle$$

A plan π is applicable to a state $s_0 \in S$ if there are states $s_1, s_2, \ldots, s_n$ so that $\gamma(s_{i-1}, a_i) = s_i$ for $i = 1, \ldots, n$. In this case, $\gamma(s_0, \pi) = s_n$.

Definition 3 Planning Problem: A planning problem is specified as a triple $P = (\Sigma, s_0, g)$ where Σ is a state-transition domain, s_0 is the initial state and g is a set of ground literal goals. A solution for P is a plan $\pi = \langle a_1, a_2, \ldots, a_n \rangle$ so that $\gamma(s_0, \pi)$ satisfies g.

Solutions to the planning problem may be developed using forward-search or backward-search techniques, with multiple heuristics proposed to reduce the state space search [8]. The Planning Domain Definition Language (PDDL) [43] is an action-centered language that provides a standard syntax to describe AI planning problems. It consists of two descriptions: (i) the *domain* description that decouples the parameters of actions from specific objects, initial conditions and goals (ii) the *problem* description that instantiates a grounded problem with objects, initialization, goals and metrics. The same domain description may be paired with multiple problem instances, with varying grounded objects, initial conditions and goals. We make use of the PDDL planning agents as for both hierarchical decomposition and resource allocation.

In addition to specifying actions, it is possible to extend plans with resource constraints (e.g. memory, energy or network bandwidth) in order to perform an action. This may be formulated as an optimization requirement: minimize a cost criteria such as achieving all actions as early as possible or using the least costly resources. This may be specified in PDDL as:

```
metric minimize (resource-cost)
```

Solvers such as Metric-FF [44] enable these optimizations to be specified during the planning process. Note that AI-planning solvers use symbolic reasoning. They are thus not as data-dependent as traditional ML or RL techniques.

5.2 Queueing network models

To evaluate efficient allocation of resources within network slices, we make use of queueing network models.

Table 4 – Queueing network metrics.

V_i	Average number of times packet visits resource i
S_i	Mean service time per packet at resource i
U_i	Utilization of resource i
Q_i	Queue length at resource i
X_i	Throughput of resource i
X	Throughput of the system
D_i	Service demand of resource i
N	Average number of packets in the system
R	Average response time of the system
Z	Mean think time of a terminal user

Queueing network models have been used to perform performance modelling and analysis of computer systems and networks. Fundamental laws applicable to queueing networks have been proposed using the metrics in Table 4. We briefly review them; see [9] for further details.

– Utilization law: utilization U is the fraction of time the resource is busy and is dependent on throughput X and service times S. Resources with high utilization cause bottlenecks.

$$U_i = X_i \cdot S_i \tag{1}$$

– Service demand law: total average service time required by a packet at resource i, denoted D_i is dependent on the visits V_i and service times S_i.

$$D_i = V_i \cdot S_i = \frac{U_i}{X} \tag{2}$$

– Little's law: if there are N users in the system, each with *think times* Z (time waiting between interactions with the system) and the throughput rate X producing a wait time R, the following relationship applies:

$$N = X \cdot (R + Z) \tag{3}$$

5.2.1 Java Modelling Tools

We implement the queueing models using the Java Modelling Tools simulator:

– Queueing station: The arriving packets join the queue and wait to receive service from the first idle server.
– Multiple class models consist of C classes, with varying traffic patterns and service demand at each station. Packets are ordered according to their arrival time but packets with higher priority jump ahead of packets with lower priority.
– Routing station: In the routing section, for each class, the generated packets are routed to the devices connected to the analyzed station according to various routing strategies. The routing probability for each outgoing link must be defined.
– Performance parameters: Three typical parameters are studied:

1. Throughput (of a station or of the entire system): At the station level it refers to the rate at which customers depart from a station, i.e., the number of requests completed in a time unit. These values are described per each class.
2. Response time (of a station or of the entire system): At the station level it refers to the average time spent at that station by a customer for a single visit (sum of queue time and service time). At the system level, it refers to the average time a customer spends in the system in order to receive services from the various stations it visits.
3. Utilization (of a station): percentage of time a station is used (i.e., busy), evaluated over the simulation run. It ranges from 0 (0%), when the station is always idle, to a maximum of 1 (100%), when the station is constantly busy servicing customers for the entire simulation run.

– What-if analysis: increase in traffic of all classes keeping constant the population mix.

6. CONDENSE SERVICE IMF DECOMPOSITION

We will now describe the knowledge base, intent reasoner and agents developed at the service IMF level.

6.1 Knowledge base

The first step in initializing the service IMF is to specify the services and 5G QoS Identifier (5QI) requirements [37]. The 5QI values provide standardized templates for service specification QoS related to bit rate guarantees, packet delay budget and packet loss. These are initialized at a particular time of day with eMBB premium, eMBB normal, fixed wireless access and best effort services. The specification is done in resource description format[2].

Listing 1 – Knowledge base RDF.

```
tel:EMBB_Premium_Service
  a tel:EMBB_Premium_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 8 .
tel:EMBB_Normal_Service
  a tel:EMBB_Normal_Service;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 9 .
tel:FWA_Service
  a tel:FWA_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 9  .
tel:Best_Effort_Service
  a tel:Best_Effort_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 12 .
```

As specified in Listing 1, there are multiple types of services such as `EMBB_Premium_Service` and `FWA_Service`. These services are initialized at particular hours, days with a target 5QI value. This knowledge base can be appended or modified with new artefacts.

6.2 Intents and expectations

Once the knowledge base artefacts have been initialized, the intents are specified according to the TM Forum standard [5] composed of the following information:

1. Intent name: an identifier for human readability.
2. Expectations: expectations of observed metrics are expressed with semantics such as `MinMetricExpectation`: minimum value to satisfy the intent; `MaxMetricExpectation`: maximum value to satisfy the intent.
3. Target: target object for this expectation (e.g. network service or slice).
4. Params: metric and value for expectation (e.g. throughput).

Listing 2 – Intent expectations RDF.

```
cc:eMBB-DL-throughput-intent
  a cc:Intent ;
  cc:hasExpectation
  [ a cc:MinMetricExpectation ;
      cc:target tel:EMBB_Premium_Service ;
      cc:params [ tel:averageThroughput 500];
  ] .
cc:eMBB-DL-latency-intent
  a cc:Intent ;
  cc:hasExpectation
  [ a cc:MaxMetricExpectation ;
      cc:target tel:EMBB_Premium_Service ;
      cc:params [ tel:averageLatency 100 ];
  ] .
```

The example in Listing 2 specifies the eMBB premium service, the `averageThroughput` target to be 500 Mbps and the `averageLatency` to be 100 ms. It follows the standardized intent templates that are being specified within TM Forum [5] and 3GPP [29].

6.3 Agents

As presented in Fig. 6, there are multiple agents participating in the process of intent management. The agents register to particular phases (grounding, proposal, evaluation, actuation) to the IMF using python templates that may be converted to RDF specifications. Note that there may be multiple agents that could participate in each phase. Furthermore, the phases may be expanded to other logical steps such as root-cause analysis or predictions, dependent on the use case.

[2] https://www.w3.org/RDF/

Grounding agent

Based on current value of network load, the grounding agent monitors the average throughput and average latency expectations. As seen in Listing 3, the target expectations for `EMBB_Premium_Service` are not met, causing an issue to be raised.

Listing 3 – Grounding agent RDF.

```
cc:issue1
 a cc:UnmetMetricExpectation ;
 cc:condition "=<" ;
 cc:currvalue 108.8 ;
 cc:expectation _:bn2 ;
 cc:metric im_telco:averageLatency ;
 cc:target im_telco:EMBB_Premium_Service ;
 cc:value 100 .

cc:issue2
 a cc:UnmetMetricExpectation ;
 cc:condition ">=" ;
 cc:currvalue 302.0 ;
 cc:expectation _:bn2 ;
 cc:metric im_telco:averageThroughput ;
 cc:target im_telco:EMBB_Premium_Service ;
 cc:value 500 .
```

The objective is to take the current values (`currvalue`) and bring them close to intent targets (`value`).

Proposal agent

As the intents are not met, there are goals on throughput and latency that need to be decomposed and provided to the RAN, transport and core subnets. To subdivide the latency goals, we make use of AI planning agents. The domain and problem files of the agents are shown in Listing 4. Different ranges of latencies may be provided as targets. These latencies at the RAN, transport and core level can be broken down into further granularity. As seen in Listing 4, lines 6-12, the PDDL domains have `precondition` steps, to be checked before triggering the action and `effect` steps to determine the output of actions. We also notice an `action_cost` value that comes from historical statistics on latency targets achievable by each sub-domain (`PROBE` data from IMFs). This value of cost captures the probability of expectations not being fulfilled (e.g. a stringent latency target may be repeatedly violated). The target latency of 100*ms* is then broken down to expectations at the RAN, transport and core level.

Listing 4 – Service IMF PDDL proposal agents.

```
#PDDL Domain

(:action RAN_latency_50
 :parameters
 (?r - RAN ?t - transport ?l - metric )
 :precondition
 (and  (unallocated_ran ?l ?r) )
 :effect
 (and  (not (unallocated_ran ?l ?r))
 (unallocated_transport ?l ?t)
 (decrease (target_latency) 50)
  (increase (action_cost) 20) ))

(:action Transport_latency_30
 :parameters
 (?t - transport ?c - core ?l - metric )
 :precondition
 (and  (unallocated_transport ?l ?t) )
 :effect
 (and  (not (unallocated_transport ?l ?t))
 (unallocated_core ?l ?c)
 (decrease (target_latency) 30)
 (increase (action_cost) 30) ))

(:action Core_latency_20
 :parameters (?c - core ?l - metric )
 :precondition
 (and  (unallocated_core ?l ?c) )
 :effect
 (and  (not (unallocated_core ?l ?c))
 (decomposed ?l)
 (decrease (target_latency) 20)
 (increase (action_cost) 30) ))

#PDDL Problem

(:init
  (unallocated_ran latency RAN1)
  (= (action_cost) 0)
  (= (target_latency) 100))

(:goal
  (and (allocated latency)
  (= (target_latency) 0)))

(:metric minimize (action_cost)) )
```

Outputs of these decomposed planners are further presented in Section 10. The advantage of these planners is that they may be rerun with new problem sets without the need to chain domain files. The decomposition is based on the values of the `BEST` API response that is returned (the most severe requirement the handler would be able to successfully comply with). Thus, there can be multiple ways to decompose the same intent, as long as they are within the stringent limits provided by each subnet IMF. If there are changes to the `BEST` data returned from lower IMFs, the planners can re-decompose the target expectations. Using an AI planner thus brings in more dynamism compared to more static optimization or rule-based techniques.

Planners have stringent preconditions and effects to be maintained for relevant plans. Thus we can encode operators such as min (throughput), + (latency) and × (availability) to ensure valid decompositions. Further details on the decomposition approach may be found in [41]. There is a further evaluation agent that can rank the decompositions based on COST or efficacy of decompositions.

This process demonstrates a solution towards *Problem 1*: decomposition of high-level service expectations to requirements at individual subnets via intent interfaces.

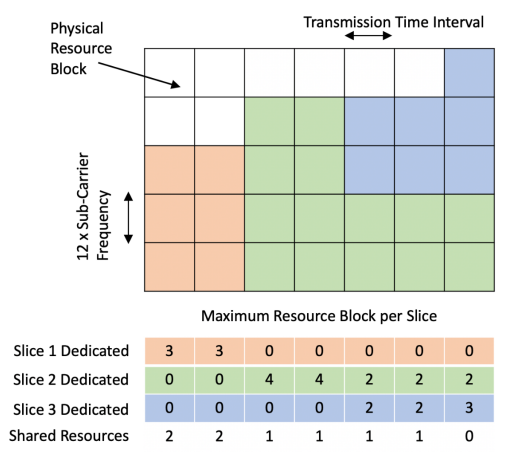

Figure 9 – RAN PRB partitioning.

7. CONDENSE RAN INTENT MANAGEMENT FUNCTION

Once the intents arrive to the RAN, actions may be taken at the RAN subnet level to change configurations.

7.1 Knowledge base

We make use of Physical Resource Block (PRB) partitioning and priorities to achieve the targets for throughput and latency [15, 16]. The 5G throughput and PRB allocations are done according to [45], as represented in Fig. 9. Depending on the spectrum, allocated bandwidth and selected subcarriers, the number of PRBs may be calculated. In addition to the number of PRBs, the throughput is affected by the modulation scheme. The supported data rate is presented in [45] and presented in Appendix A.

An example of the initialized knowledge base is presented in Listing 5 with the slice specifications and allocated partition shares.

Listing 5 – RAN knowledge base.

```
tel:slice1
  a tel:Slice ;
  tel:id "slice1";
  tel:spectrum "n5";
  tel:total_prb 273;
  tel:mimo 2 .

tel:P1_slice
    a tel:Slice_config ;
    tel:day 01042023;
    tel:hour 11;
    tel:prb_share 10 .

tel:EMBB_Premium_Service
  a tel:EMBB_Premium_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 8;
    tel:has_topup "yes";
    tel:slice "P1_slice"  .

tel:EMBB_Normal_Service
  a tel:EMBB_Normal_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 9;
    tel:has_topup "yes";
    tel:slice "P1_slice" .
```

The `total_prb` is initialized to be 273 and we provide 10 per cent of the share to the `P1_slice`. Both `EMBB_Premium_Service` and `EMBB_Normal_Service` are associated with the slice. We notice here that the services have the option to have PRB "top-up" in case of deteriorating QoS. This process would enable adding additional PRBs to the partition to improve service performance.

7.2 Intents and expectations

RAN operational intents

In addition to the sub-intents provided by the service layer, the RAN can have additional intents [15, 16].

Listing 6 – RAN operational intents.

```
cc:p1-partition-intent
 a cc:Intent ;
 cc:hasExpectation
 [ a cc:MaxMetricExpectation ;
     cc:target tel:P1_slice ;
     cc:params [ tel:unused_PRB 1 ] ;
 ] ,
 [ a cc:MaxMetricExpectation ;
     cc:target tel:P1_slice ;
     cc:params [ tel:prb_util 95 ] ;
 ] ,
 [ a cc:MinMetricExpectation ;
     cc:target tel:P1_slice ;
     cc:params [ tel:prb_share 10 ] ;
 ] .
```

As presented in Listing 6, the operational expectations can include limits on unused PRBs (optimize usage), maximum PRB utilization (to trigger additional resources) and minimum guaranteed shares (fairness). The proposal agents have to keep these restrictions in account while allocating resources to slices.

7.3 Agents

Grounding agent

The grounding agent forecasts/monitors the current throughput, latency, PRB utilization and Modulation and Coding Scheme (MCS) index for the service. As we notice in Listing 7, the targets for throughput and latency are not met, triggering intent issues and proposals.

Listing 7 – RAN grounding agent.

```
1 Grounding Agent forecasted 302.0 Throughput
      for eMBB DL Premium Service service 13h
      2023 02 01
2 Grounding Agent forecasted 96.20000000000002
      PRButil for eMBB DL Premium Service
      service 13h 2023 02 01
3 Grounding Agent forecasted 24.64 PDUsessions
      for eMBB DL Premium Service service 13h
      2023 02 01
4 Grounding Agent forecasted 11.6 MCSindex for
      eMBB DL Premium Service service 13h 2023
      02 01
5 Grounding Agent forecasted 51.8 latency for
      eMBB DL Premium Service service 13h 2023
      02 01
```

Proposal agents

The AI planning agents for the RAN subnet propose actions such as topping up partition share, ramping down partition share or changing priority of the service (Listing 8). These actions are encoded with preconditions and effects that reflect the operational intent constraints. A plan is produced (Listing 8 line 30) to meet the throughput target of 500 Mbps and intent target of 40 ms.

Listing 8 – RAN PDDL proposal agents.

```
1  (:action topup_partition_percent_share
2    :parameters
3     (?service - service ?band - G5_band
4     ?partition - slice_partition)
5    :precondition (and
6     (suboptimal ?service  ?band  ?partition )
7     (underutilized  eMBB_normal n5 Residual)
8     (<= (PRB_share ?service)
9     (PRB_share_req ?service)))
10   :effect (and
11    (assign (PRB_share ?service)
12    (PRB_share_req ?service))
13    (not (suboptimal ?service  ?band
14    ?partition ))
15    (alloc ?service  ?band  ?partition )))
16
17 (:action change_priority
18   :parameters
19    (?service - service ?band - G5_band
20   ?partition - slice_partition)
21   :precondition (and
22    (low_priority ?service ?band ?partition))
23   :effect (and
24    (not (low_priority ?service ?band
25    ?partition))
26    (high_priority ?service  ?band
27    ?partition)))
28
29 # Plan:
30 0: (change_priority embb_premium n5 p1) [1]
31 0: (topup_partition_percent_share embb_premium
       n5 p1) [1]
32 1: (allocate_partition_n5 embb_premium n5 p1)
      [1]
```

As provided in Listing 8, lines 1-27, actions such as topping up partition and changing priority may be enabled actions at the RAN IMF domain. As provided in Listing 8, lines 29-32, the planned actions to meet targets are changing priority of the eMBB premium service and topping up the partition of the slice associate with the eMBB premium service. Depending on the targets set for each scenario, the planner may suggest alternate actions.

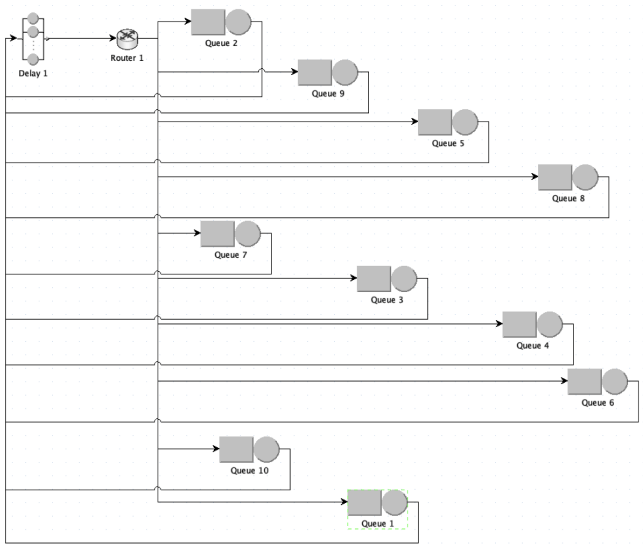

Figure 10 – Queueing network model for RAN PRB allocation.

Evaluation agent

To evaluate the output proposed by the planner, we make use of a queueing network model with multiclass queues. Fig. 10 provides a queueing model in Java Modelling Tools[3] that has 10 queues with each queues representing 10 per cent of PRB shares. Based on router 1, the different classes of services (eMBB, FWA) may be allocated resources. The throughput and latency are affected as additional resources are allocated or removed per slice.

An example of the output of the RAN network slicing is presented later in Section 10.

8. CONDENSE TRANSPORT INTENT MANAGEMENT FUNCTION

We describe the knowledge base, action space and intents specified within the transport IMF.

8.1 Knowledge base

The transport network in typical 5G/6G network slicing use cases consists of edge routers, front-haul switches and backhaul switch connectivity to the core. As seen in Fig. 11, these routers are to be configured effectively to meet the end-to-end slice intent requirements.

[3] https://jmt.sourceforge.net

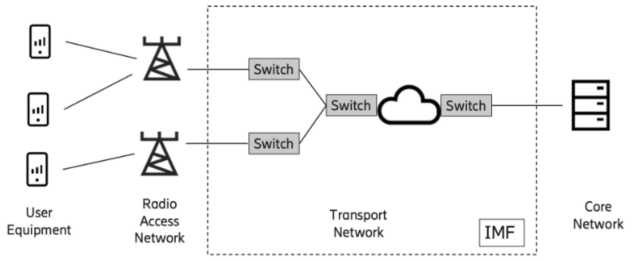

Figure 11 – 5G/6G transport network with multiple switches/routers.

Rather than differentiating network traffic based on the requirements of an individual flow, DiffServ [46] operates on the principle of traffic classification, placing each data packet into one of a limited number of traffic classes. Each router on the network is then configured to differentiate traffic based on its class. Each traffic class can be managed differently, ensuring preferential treatment for higher-priority traffic on the network. DiffServ uses a 6-bit Differentiated Services Code Point (DSCP) in the 6-bit Differentiated Services field (DS field) in the IP header for packet classification purposes.

The DSCP forwarding classes mapped to services may be specified in the knowledge base as in Listing 9, with `class-of-service 1`.

Listing 9 – Transport knowledge base.

```
tel:EMBB_Premium_Service
  a tel:EMBB_Premium_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 8;
    tel:class-of-service 1 .

tel:EMBB_Normal_Service
  a tel:EMBB_Normal_Service ;
    tel:day 01042023;
    tel:hour 11;
    tel:5QI 9;
    tel:class-of-service 1 .
```

8.2 Intents and expectations

The sub-intents to this IMF are provided here, where targets for throughput and latency are decomposed towards the transport domain in Listing 10.

Listing 10 – Transport IMF expectations.

```
cc:eMBB-DL-throughput-intent
  a cc:Intent ;
  cc:hasExpectation
  [ a cc:MinMetricExpectation ;
      cc:target tel:EMBB_Premium_Service ;
      cc:params [ tel:averageThroughput 500];
  ] .
cc:eMBB-DL-latency-intent
  a cc:Intent ;
  cc:hasExpectation
  [ a cc:MaxMetricExpectation ;
      cc:target tel:EMBB_Premium_Service ;
      cc:params [ tel:averageLatency 30 ] ;
  ] .
```

Note that the target `averageLatency` is set to 30 ms based on the decomposition step.

8.3 Agents

Proposal agents

Given the subdivided intent from the service IMF on throughput and latency, the proposal agents set the appropriate DSCP service class, as well as limit the rate of transmission. This ensures that the routers and switches are configured to meet appropriate expectation targets.

Listing 11 – Transport PDDL proposal agents.

```
(:action set_service_class
 :parameters
  (?service1 - service
   ?slice1 - slice_partition)
 :precondition (and
  (< queue_utilization 70)
  (latency_not_met  ?service1  ?slice1))
 :effect
  (and (not  (latency_not_met
  ?service1  ?slice1))
  (latency_met  ?service1  ?slice1)
  (assign (latency ?service1)  30 )))

(:action set_rate_limit
 :parameters
  (?service1 - service
   ?slice1 - slice_partition)
 :precondition
  (and (< queue_utilization 70)
  (throughput_not_met
  ?service1  ?slice1))
 :effect
  (and (not (throughput_not_met
  ?service1  ?slice1))
  (throughput_met  ?service1  ?slice1)
  (assign (throughput ?service1)  500 )))

# Plan:
0: (set_rate_limit embb_normal p1) [1]
1: (set_service_class embb_premium
    assured_forwarding) [1]
```

As provided in Listing 11, lines 1-26, actions such as setting DSCP service class or setting rate limits may be enabled actions at the transport IMF domain. As provided in Listing 11, lines 28-30, the planned actions to meet targets are setting rate limits on the eMBB normal service and setting assured forwarding for the eMBB premium service. These actions ensure that both throughput and latency intent expectations of these services are reached.

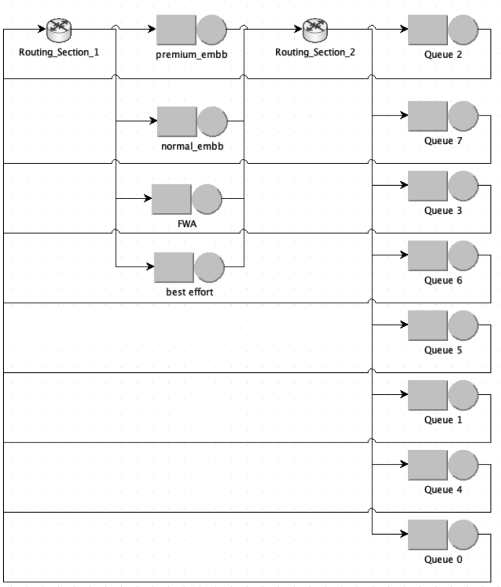

Figure 12 – Queueing network model for a transport router.

Evaluation agent

Once the proposals are made, we simulate the traffic within the queueing network model for a router with eight queues, as shown in Fig. 12 [19]. This queuing model may be used to evaluate the efficacy of the proposals.

An example of the output of the transport network slicing is presented later in Section 10.

9. CONDENSE CORE INTENT MANAGEMENT FUNCTION

We describe the action space, intents and outputs of the core IMF.

9.1 Knowledge base

Intents provided to the core would be used to ensure the appropriate 5QI values [47] and the container capacities needed for the network functions. As seen in Fig. 13, there is a fat tree network assumed at the core that can be connected to multiple pods. The pods can be scaled up/down in conjunction with the 5QI requirements.

9.2 Intents and expectations

In addition to the intents on throughput and latency propagated by the service IMF, there are additional expectations on pod utilization. We see in Listing 12, intents on the maximum and minimum utilization levels that can trigger actions at the pod level.

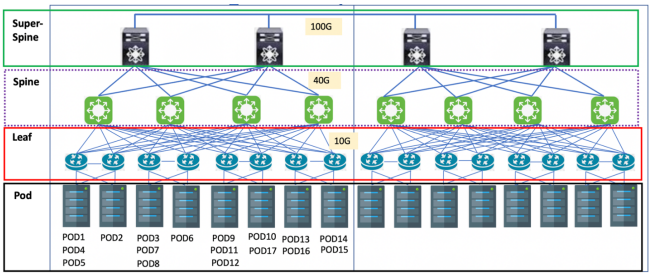

Figure 13 – Fat tree networks and containers.

Listing 12 – Core IMF expectations.

```
1  cc:pod-intent
2    a cc:Intent ;
3    cc:hasExpectation
4    [ a cc:MaxMetricExpectation ;
5        cc:target tel:EMBB_Premium_Service ;
6        cc:params [ tel:pod_utilization 80] ;] ,
7    [ a cc:MinMetricExpectation ;
8        cc:target tel:EMBB_Normal_Service ;
9        cc:params [ tel:pod_utilization 20] ;] .
```

9.3 Agents

Proposal agent

The proposal agent at the core level in Listing 13 can suggest pod scale up/down and set 5QI values. These agents are instantiated to resolve issues created by the intents.

Listing 13 – Core PDDL proposal agents.

```
1  (:action pod_scale_up
2   :parameters ( ?service1 - service ?slice1 -
        slice_partition)
3   :precondition (and
4   (throughput_not_met  ?service1  ?slice1)
5   (< (pod_utilization ?service1) 80))
6   :effect (and
7   (not (throughput_not_met  ?service1  ?slice1)
        )
8   (throughput_met  ?service1  ?slice1)
9   (assign (throughput ?service1)  500 )))
10
11 (:action set_5QI
12  :parameters ( ?service1 - service ?slice1 -
        slice_partition)
13  :precondition (and
14  (>= (pod_utilization ?service1) 80))
15  :effect (and
16  (assign (pod_utilization ?service1) 30 )))
17
18 # Plan:
19 1:   (set_5qi embb_premium slice1) [1]
20 1:   (pod_scale_up embb_premium slice1) [1]
```

As provided in Listing 13, lines 1-16, actions such as pod scale up or changing 5QI values may be proposed at the core IMF domain. As provided in Listing 13, lines 18-20, the planned actions to meet targets are setting 5QI and pod scale up for the eMBB premium service. These actions ensure that both throughput and latency intent expectations of these services are reached.

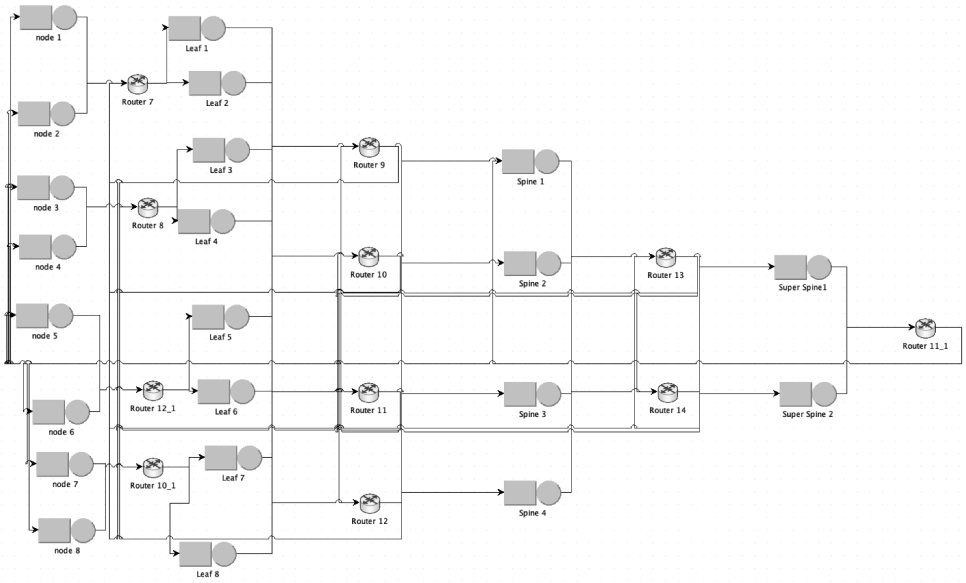

Figure 14 – Queueing network model of a fat tree network.

Table 5 – Experimental setting.

Domain	Parameter	Value
Service	Number of services	4 (eMBB premium, eMBB normal, FWA, best effort)
	eMBB Premium throughput expectation	500 Mbps
	eMBB Premium latency expectation	100 ms
	eMBB Premium users [min, Max]	[40, 160]
	eMBB Premium user arrival	Poisson
RAN	Number of cell sites	1
	Number of PRBs	273
	Number of initial RAN slice partitions	2
	Percentage of PRBs per partition	10
Transport	Number of switches	1
	Number of internal queues	8
Core	Number of containers	8
	Number of fabric layers	3 (leaf, spine, super-spine)
	Leaf interconnect capacity	10 Gbps
	Spine interconnect capacity	40 Gbps
	Super-spine interconnect capacity	100 Gbps

Evaluation

The evaluation is done on the queueing network presented in Fig. 14. It demonstrates the flow of traffic in a fat-tree network and the resource utilization at individual pods. It showcases eight container nodes with leaf, spine and super-spine network fabric. The network load can be load-balanced or make use of actions such as pod scale up to meet target requirements.

An example of the output of the core network slicing is presented later in Section 10.

10. EXPERIMENTAL EVALUATION

To create a realistic scenario for network slicing, we make use of the use case presented in Fig. 2 and described further in Section 6. As specified by a mobile network operator in Asia, the priority eMBB users are expected to meet the target requirements. This instance specifies the `averageThroughput` target to be 500 Mbps and the `averageLatency` to be 100 ms.

Table 5 presents the experimental settings used for the evaluation. Four services are considered with the eMBB premium service expectations chosen for assurance. The RAN has 273 physical resource blocks that may be allocated to radio resource partitions. Transport has eight virtual queues within the switch that has to be configured. The core has eight containers with leaf-spine-superspine network fabric.

Table 6 presents a snapshot of the dataset from an operator eMBB deployment that provides historical PRB utilization, active users and user throughput uplink (UL) and downlink (DL). This dataset is used by the grounding agent to ground the hourly increase in eMBB traffic growth. There can be hourly fluctuations in traffic patterns requiring the assurance agent to resolve the issues.

Table 6 – eMBB service dataset.

Cell	Date	UL PRB util	UL active users	UL user throughput	DL PRB util	DL active users	DL user throughput
NE618	11/01/19	0.932115467	1.879057284	1032.072471	91.16519295	1.457330379	10277.46439
NE618	12/01/19	0.955579017	1.879057284	1032.072471	93.4983892	1.490459507	10277.46439
NE618	01/01/20	0.979042567	1.879057284	1032.072471	95.83158545	1.523588635	10277.46439
NE618	02/01/20	1.002506118	1.879057284	1032.072471	98.1647817	1.556717763	10277.46439

Table 7 – Events seen during intent decomposition.

Report	Events
Grounding Agent	Grounding Agent estimates 302.0 Throughput for eMBB DL Premium Service service 13h 2023 02 01 Grounding Agent estimates 108.8 Latency for eMBB DL Premium Service service 13h 2023 02 01
Proposal Agent	0: ran latency 40 ran1 transport1 latency 1: transport latency 30 transport1 core1 latency 2: core latency 30 core1 latency 3: decompose latency
Evaluation Agent	Decomposition Plan 1 selected for eMBB DL Premium Service 2023 02
Cognitive Reasoner	Intent Decomposed to meet target intents 500 Mbps and 100 ms for service eMBB DL premium

The grounding can be done in two ways to create intent issues:

1. Use the current state of the network to provide an estimate of service metrics.
2. Use the historic load on the network to forecast growth in traffic. The assurance can then be based on predicted deviations that may be hours or days away. Note that there may be uncertainties or suboptimal allocation of resource depending on the time horizon of forecasts.

The assurance actions may be then planned to mitigate issues created due to the grounding in either of these cases.

Service IMF decomposition

Once grounding is done (Table 7) and issues are generated, the first step would be to create decomposed targets (using the PDDL specification in Section 6). Here is a solution using Metric FF [44] plan solvers:

Listing 14 – Service IMF PDDL plan.

```
#Plan
0: ran latency 40 ran1 transport1 latency
1: transport latency 30 transport1 core1
    latency
2: core latency 30 core1 latency
3: decompose latency
```

Listing 14 showcases an example where the latency target of 100 ms is broken down to 40 ms for the RAN, 30 ms for transport and 30 ms for the core. The final decomposed actions to be sent to the lower subnets from the service IMF are provided in Table 7. We see that the targets on throughput are 500 Mbps for all subnets. For the latency, it is additive to meet the 100 ms target.

RAN slicing

The targets for the RAN IMF are average throughput of 500 Mbps and latency of 40 ms. The plan output (based on PDDL specification in Section 7) is:

Listing 15 – RAN PDDL plan.

```
# Plan:
0: (change_priority embb_premium n5 p1) [1]
0: (topup_partition_percent_share embb_premium
     n5 p1) [1]
1: (allocate_partition_n5 embb_premium n5 p1)
    [1]
```

Listing 15 showcases the planned RAN IMF actions, such as changing priority and topping up partitions. Fig. 15 shows the what-if analysis generated by the queueing model (increase in traffic N). We notice in Fig. 15 that the throughput increases with the plan implementation, with positive effect also to the fixed wireless access service. However, this is traded off with latency increase, which is still within the 40 ms target.

Based on the chosen plan, the actuation takes place with 17 per cent PRBs topped up. The issue is resolved and the RAN intents are met.

Listing 16 – RAN events.

```
Plan actuated with new schedule, 17 percent
    Physical Resource Blocks for eMBB DL
    premium in 13h 2023 02 01
Issue Resolved Average throughput 500 Mbps,
    Average Latency 40 ms for eMBB DL premium
    in 13h 2023 02 01
```

Listing 16 showcases that after the PRBs have been topped up, the throughput at the RAN IMF reaches target for the eMBB premium service. As seen in Fig. 18, a top-up for a premium eMBB service is triggered at hour 4, which ensures SLA compliance. An alternative case with no top-up option produces an SLA violation, despite

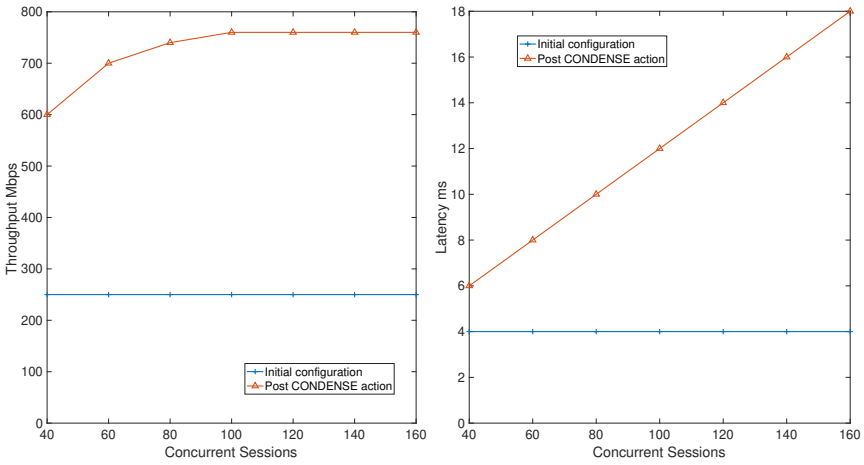

Figure 15 – RAN throughput and latency estimated improvements.

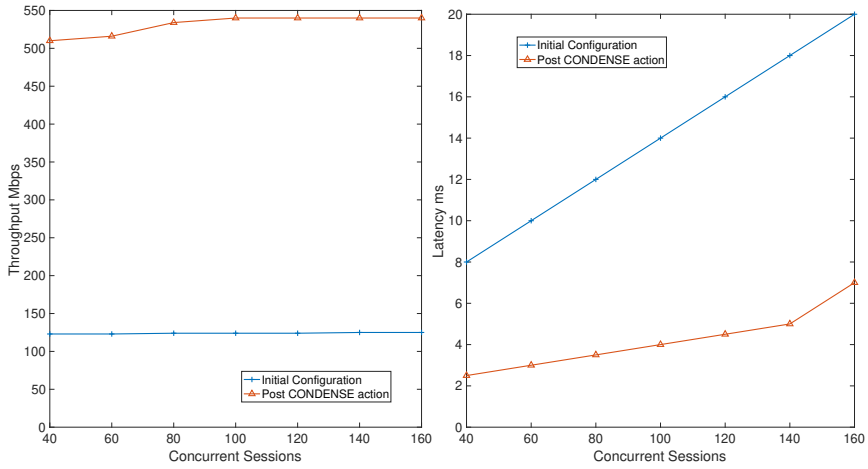

Figure 16 – Transport throughput and latency estimated improvements.

residual resources being available. Thus, it is crucial to dynamically modify the resources for efficient service and slice management.

Transport slicing

An example output of the IMF is presented below that ensures that the intent expectations (based on PDDL specification in Section 8) are met:

Listing 17 – Transport PDDL plans.

```
# Plan:
0: (set_rate_limit embb_normal p1) [1]
1: (set_service_class embb_premium
    assured_forwarding) [1]
```

Listing 17 showcases the planned transport IMF actions such as setting rate limits and setting service class. We notice from Fig. 16 that the throughput and latency improve for the premium service. Thus the new configuration may be applied to meet the intent requirements.

Core slicing

Similarly, the core slicing is done according to the proposals provided by agents (based on PDDL specification in Section 9):

Listing 18 – Core PDDL plans.

```
# Plan:
1:   (set_5qi embb_premium slice1) [1]
1:   (pod_scale_up embb_premium slice1) [1]
```

Listing 18 showcases the planned core IMF actions such as setting 5QI and scaling up pods. We notice an improvement in latency and throughput in Fig. 17 after action changes.

This process of service IMF decomposition and allocation of resources across the RAN, transport and core subnets

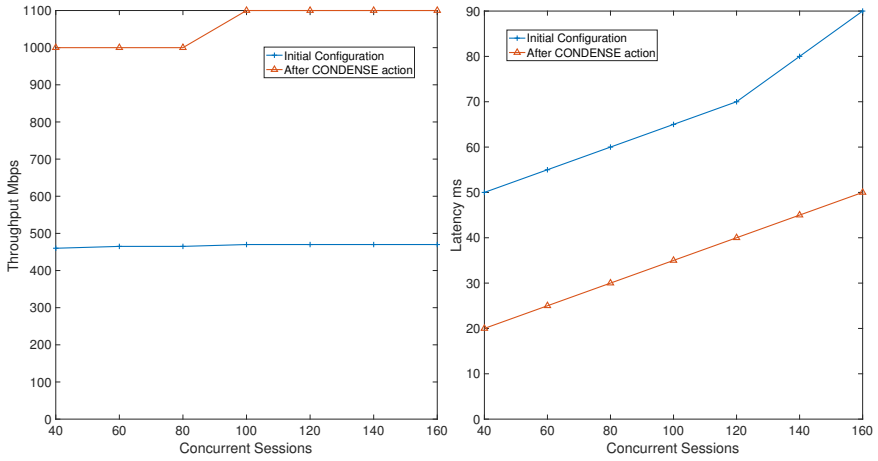

Figure 17 – Core throughput and latency estimated improvements.

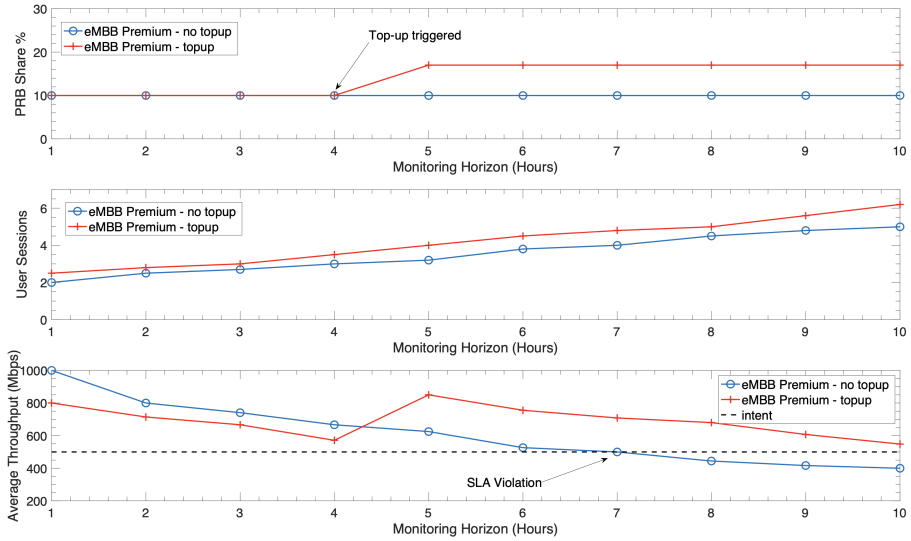

Figure 18 – Service assurance example with PRB top-up.

is envisioned to be non-real time. As slice reconfigurations are not triggered in the seconds time-scale, they can complete the intent reasoning, AI planning proposals, queueing evaluations and actuations in a matter of minutes. The outputs can be written to policy rules that may be applied to near real-time configuration engines.

10.1 Multi-slice PRB partitioning

We further show an example of autonomous decisions taken at the RAN, transport and core levels in the multi-slice scenario of Fig. 2 and in Section 6. The simulator using realistic scheduling decisions is based on partition sizes, priorities and number of user equipment (UE). The initial state of the network is shown in Fig. 19 where the service and level traffic is showcased. Here we notice that there is deterioration in the eMBB premium service target of 500 Mbps average throughput. When the intent of 500 Mbps needs to be met for eMBB, CONDENSE proposes automated actions such as setting partition changes in the RAN, DSCP class in transport and appropriate 5QI at the core. These changes impact the observed outputs in Fig. 20, where the higher priority eMBB traffic reaches the target.

As the intents can be dynamic and be based on traffic patterns, we simulate another case in Fig. 21. With lower traffic in eMBB premium, the system can now satisfy the FWA throughput target of 400 Mbps. Thus, given system capacity, CONDENSE can satisfy multiple intent requirements in parallel.

Via the use of the RAN, transport and core slicing actions, we demonstrate a solution to *Problem 2*: the ability of AI agents to autonomously assure end-to-end slice requirements with appropriate actions.

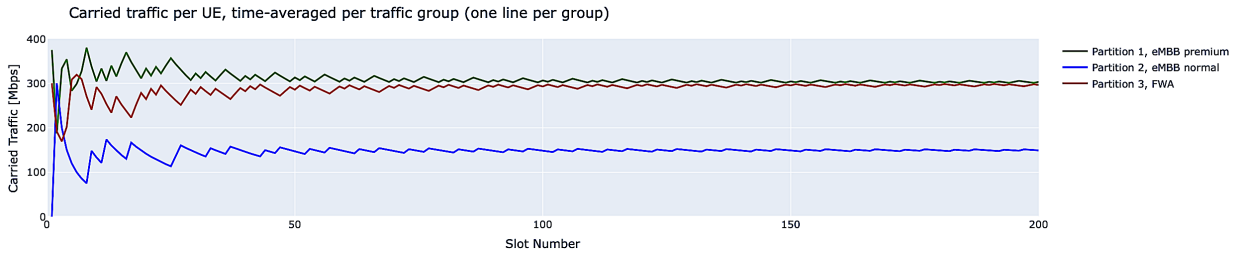

Figure 19 – Initial observations for service and UE level traffic.

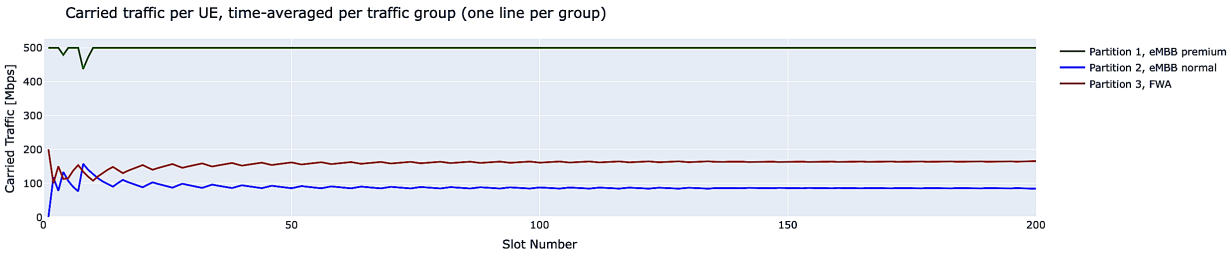

Figure 20 – Condense reconfigured observations for service and UE level traffic (eMBB premium intent).

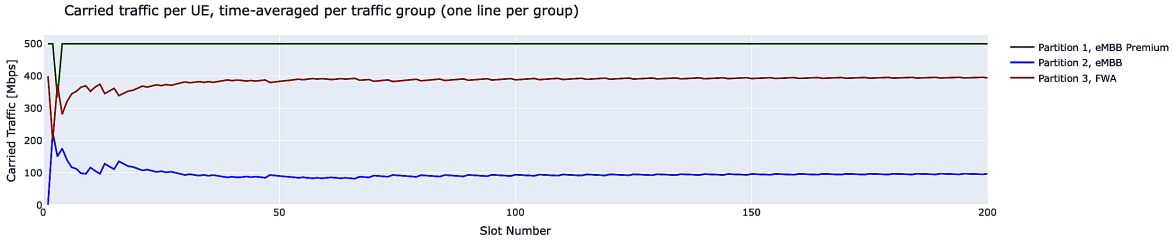

Figure 21 – Condense reconfigured observations for service and UE level traffic (eMBB premium and FWA intent).

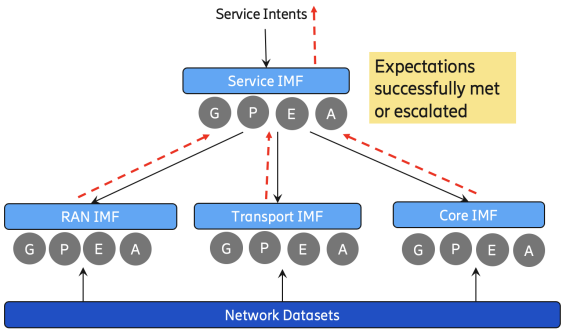

Figure 22 – Expectations at each level aggregated at the service IMF.

10.2 CONDENSE feedback to service IMF

As presented in Table 8, the expectations of the service IMF are met as a result of individual subnets meeting the required targets. These values are reported back via intent reports [48] to the service IMF. However, as seen in Fig. 22, when the expectations are not successfully met, there is a need for replanning. The planner can be used in two ways: (i) If the goal remains the same, the cost function may be updated to reflect that the targets were not met; (ii) in case the targets are changed, the planner can be rerun with the new goals (common domain file). We notice the new plans with changes in target expectations shown in Listing 19.

Listing 19 – Replanning with feedback.

```
## change in cost
0: ran latency 30 ran1 transport1 latency
1: transport latency 40 transport1 core1
    latency
2: core latency 30 core1 latency

## change in latency goal to 140
0: ran latency 50 ran1 transport1 latency
1: transport latency 50 transport1 core1
    latency
2: core latency 40 core1 latency
```

As seen in Listing 19, in the first case, there may be updating of the cost function for planning, resulting in new decompositions. In the second case, the intent targets may change, which would instantiate a new problem file.

This process demonstrates a solution to *Problem 3*: feedback from lower intent management layers and closed loop control.

Table 8 – IMF expectations.

IMF	Min Throughput	Max Latency	Target met?
Service	500 Mbps	100 ms	Y
RAN	500 Mbps	40 ms	Y
Tranport	500 Mbps	30 ms	Y
Core	500 Mbps	30 ms	Y

11. CONCLUSIONS

The evolution of 5G and 6G networks has brought out the need for autonomous management of networks. However, there is limited research work in the end-to-end network slice assurance front dealing with practical implementation of agents. In this paper, we propose CONDENSE, a novel intent-driven system for end-to-end network slice management. The intents are specified using TM Forum compliant standards. The system begins with decomposing intent-driven requirements across multiple intent management function hierarchies. Through the extensive use of AI planning agents, resources may be then efficiently allocated at the RAN, transport and core levels. AI planning agents using symbolic reasoning as opposed to data-driven approaches. In addition, evaluation and replanning agents are proposed to provide a robust framework for intent-driven slicing. The system is demonstrated over a real-world example with multiple types of intents. The CONDENSE system effectively demonstrates the autonomous management of network slices.

In future, we will be deploying this system over a real testbed to collect network statistics and closed loop configuration data. Further improvements and challenges within AI-native 6G network architectures are also envisioned within the framework.

A. PHYSICAL RESOURCE BLOCK ALLOCATION

Throughput as a result of PRB allocation:

$$10^{-6} \cdot \sum_{j=1}^{J} \left(v^{(j)} \cdot f^{(j)} \cdot Q_{\mathrm{m}}^{(j)} \cdot R_{\mathrm{m}} \cdot \frac{12 \cdot N_{\mathrm{PRB}}^{(j)}}{T_{\mu}} \cdot \left(1 - OH^{(j)}\right) \right) S \quad (4)$$

- J is the number of aggregated carriers,
- $v^{(j)}$ is the number of MIMO layers per carrier,
- $f^{(j)}$ is the scaling factor per carrier in the range $[0.4, 1]$,
- $Q_{\mathrm{m}}^{(j)}$ is the modulation order per carrier,
- R_{m} is the modulation code rate divided by 2048,
- T_{μ} is the average OFDM symbol duration in a subframe for numerology μ, calculated as $10^{-3}/(14 \cdot 2^{\mu})$,
- $N_{\mathrm{PRB}}^{(j)}$ is the number of physical resource blocks per carrier j, for the given bandwidth and numerology,
- OH is the overhead in range $[0.08, 0.18]$,
- and S is the symbols allocation which determines how much of a slot is dedicated to uplink or downlink.

In typical scenarios, parameters that can be configured are N_{PRB} and the modulation scheme (Q_{m} and R_{m}). Each PRB partition can be configured with a share of resources for which the users belonging to the partition has priority.

REFERENCES

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions". In: *IEEE Communications Surveys and Tutorials* 20.3 (2018), pp. 2429–2453.

[2] A. Kattepur, S. David, S. Mohalik, and S. Terrell. *Cognitive Reasoning for 5G Network Lifecycle Management*. Ericsson White paper, 2022.

[3] K. Mehmood, K. Kralevska, and D. Palma. "Intent-driven autonomous network and service management in future cellular networks: A structured literature review". In: *Comput. Netw.* 220 (2023).

[4] K. Aykurt and W. Kellerer. "Autonomous Network Management in Multi-Domain 6G Networks based on Graph Neural Networks". In: *IEEE 9th International Conference on Network Softwarization (NetSoft)*. Spain, 2023.

[5] TMForum. "Intent Common Model". In: *Specification TR290 3.6.0* (2024).

[6] J. Niemöller, L. Mokrushin, S. Mohalik, M. Konchylaki, and G. Sarmonikas. *Cognitive processes for adaptive intent-based networking*. Ericsson Technology Review, 2020.

[7] Ericsson. *Defining AI native: A key enabler for advanced intelligent telecom networks*. Whitepaper, 2023.

[8] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, 2004.

[9] E. Lazowska. *Quantitative System Performance, Computer System Analysis Using Queuing Network Models*. Prentice Hall, 1984.

[10] Ericsson. *The essential building blocks of E2E network slicing*. Whitepaper, 2023.

[11] M. Chahbar, D. Gladys, A. Dandoush, C. Cérin, and K. Ghoumid. "A Comprehensive Survey on the E2E 5G Network Slicing Model". In: *IEEE Transactions on Network and Service Management* (2020).

[12] M. Xie, P. H. Gomes, J. Niemöller, and J. P. Waldemar. "Intent-Driven Management for Multi-Vertical End-to-End Network Slicing Services". In: *IEEE Globecom Workshops*. Brazil, 2022: Rio de Janeiro, 2022.

[13] K. Abbas, M. Afaq, A. Khan, A. Rafiq, and W. Song. "Slicing the Core Network and Radio Access Network Domains through Intent-Based Networking for 5G Networks". In: *Electronics* 9.10 (2020), p. 1710.

[14] A. Kattepur, S. Mohalik, and I. Burdick. "Praetorian: Probabilistic Planning for Radio Access Network Slice Assurance". In: *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. Japan, 2024: Osaka, 2024.

[15] A. Kattepur, S. Mohalik, I. Burdick, M. Orlic, and L. Mokrushin. "CONRAD: Cognitive Intent Driven 5G Network Slice Planning and Design". In: *In Proceedings of the Third International Conference on AI-ML Systems*. 2023.

[16] A. Kattepur, S. Mohalik, I. Burdick, M. Orlic, and L. Mokrushin. "Convergence: Cognitive Intent Driven 5G Radio Access Network Slice Assurance". In: *IEEE Wireless Communications and Networking Conference (WCNC)*. United Arab Emirates, 2024: Dubai, 2024.

[17] A. Kattepur, S. David, and S. Mohalik. "MUESLI: Multi-objective Radio Resource Slice Management via Reinforcement Learning". In: *IEEE 8th International Conference on Network Softwarization (NetSoft)*. Italy, 2022.

[18] C. Nahum et al. "Intent-Aware Radio Resource Scheduling in a RAN Slicing Scenario Using Reinforcement Learning". In: *IEEE Transactions on Wireless Communications* 23.3 (2024), pp. 2253–2267.

[19] A. Kattepur, S. David, and S. Mohalik. "Automated Configuration of Router Port Queues via Model-Based Reinforcement Learning". In: *IEEE International Conference on Communications Workshops (ICC Workshops)*. Montreal, 2021.

[20] A. Kattepur, S. David, and S. K. Mohalik. "Model-based reinforcement learning for router port queue configurations". In: *Intelligent and Converged Networks* 2.3 (2021), pp. 177–197.

[21] Cisco. *Application Centric Infrastructure (Cisco ACI)*. Solution Overview, 2024.

[22] A. Kattepur and S. David. "Malta: Multi-Agent Reinforcement Learning for Differentiated Services in Fat Tree Networks". In: *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. 2021.

[23] A. Kattepur and S. David. "Madelyn: Multi-Domain Multi-Agent Reinforcement Learning for Data-center Networks". In: *IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. 2022: USA, 2022.

[24] Juniper Networks. *Apstra Architecture*. whitepaper, 2024.

[25] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju, and B. Shim. "Deep Reinforcement Learning-Based Network Slicing for Beyond 5G". In: *IEEE Access* 10 (2022), pp. 7384–7395.

[26] H. Phyu, D. Naboulsi, and R. Stanica. "Machine Learning in Network Slicing - A Survey". In: *IEEE Access* 11 (2023), pp. 39123–39153.

[27] M. Ahmadi, A. Moayyedi, M. Sulaiman, M. Salahuddin, R. Boutaba, and A. Saleh. "Generalizable 5G RAN/MEC Slicing and Admission Control for Reliable Network Operation". In: *IEEE Transactions on Network and Service Management* 21.5 (2024), pp. 5384–5399.

[28] D. Manias, A. Chouman, and A. Shami. "Towards Intent-Based Network Management: Large Language Models for Intent Extraction in 5G Core Networks". In: *20th International Conference on the Design of Reliable Communication Networks (DRCN)*. 2024.

[29] 3GPP. "Intent driven management services for mobile networks (Release 18)". In: *TS 28.312* (2023).

[30] A. Leivadeas and M. Falkner. "A Survey on Intent-Based Networking". In: *IEEE Comms. Surveys and Tutorials* 25.1 (2023), pp. 625–655.

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

[32] E. Chirivella-Perez, P. Salva-Garcia, R. Ricart-Sanchez, J. A. Calero, and Q. Wang. "Intent-Based E2E Network Slice Management for Industry 4.0". In: *Joint European Conference on Networks and Communications and 6G Summit*. 2021.

[33] J. Zhang, C. Yang, A. Anpalagan, R. Dong, and L. Zhang. "IDSM: Intent-Driven Slice Management and Maintenance for 6G RANs". In: *IEEE 22nd International Conference on Communication Technology (ICCT)*. 2022.

[34] P. Shakarian, C. Baral, G. Simari, B. Xi, and L. Pokala. *Neuro symbolic reasoning and learning*. 1st ed. Springer, 2023.

[35] L. DeLong, R. Mir, and J. Fleuriot. "Neurosymbolic AI for Reasoning Over Knowledge Graphs: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 36.5 (2025), pp. 7822–7842.

[36] 3GPP. "Management and orchestration; 5G Network Resource Model (NRM)". In: *Specification Release 18* 28.541 (2024).

[37] 3GPP. "5G end to end Key Performance Indicators (KPI) (Release 18)". In: *TS 28.554* (2023).

[38] 3GPP. "5G performance measurements (Release 18)". In: *TS 28.552* (2023).

[39] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003.

[40] P. Soto et al. "Designing the Network Intelligence Stratum for 6G networks". In: *Computer Networks* 254 (Dec. 2024), p. 110780. ISSN: 1389-1286.

[41] A. Kattepur, S. Das, D. Daroui, S. Mohalik, M. Orlic, and S. Ertas. "DETROIT: Decomposition techniques for a hierarchy of 6G network intent management functions". In: *Computer Networks* 272 (2025), p. 111657. ISSN: 1389-1286.

[42] J. Niemoller, J. Silvander, P. Stjernholm, L. Angelin, and U. Eriksson. "Autonomous networks with multi-layer, intent-based operation". In: *Ericsson Technology Review* (2023).

[43] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. "An Introduction to the Planning Domain Definition Language". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13 (2019), pp. 1–187.

[44] B. Nebel. "The FF Planning System: Fast Plan Generation Through Heuristic Search". In: *Journal of Artificial Intelligence Research* 14 (2001), pp. 253–302.

[45] 3GPP. "User Equipment (UE) radio access capabilities, TR 38.306". In: *Release 17* (2024).

[46] K. Kilkki. *Differentiated services for the Internet*. Macmillan Technical Publishing, 1999.

[47] 3GPP. "System architecture for the 5G System (5GS), 23.501". In: *Release 18* (2024).

[48] TMForum. "Intent Common Model - Intent Reporting". In: *TR290B* (2024).

AUTHORS

Ajay Kattepur is a senior researcher in the Autonomous Intelligent Systems group at Ericsson Research, India. Ajay's focus is on applying automated planning, reinforcement learning and verification techniques for 5G/6G networking and Industry 4.0. Prior to joining Ericsson Research, he was with Tata Consultancy Services (TCS) Research and Innovation Labs in India. Ajay received his Ph.D. in Computer science from the French National Institute for Computer Science and Control (INRIA), Rennes, France, and M.Eng. and B.Eng. degrees in electrical and electronic engineering from Nanyang Technological University (NTU) Singapore.

Ian Burdick is a visionary leader in AI with proven experience as a Lead AI Architect at Ericsson, where he spearheads the integration of artificial intelligence into cutting-edge solutions. With a robust background in technology architecture, Ian bridges the gap between complex algorithms and real-world applications. His expertise lies in machine learning and AI principles: core algorithms, neural networks, and data processing techniques that power AI. Additionally, Ian navigates vast data lakes, excelling in data extraction, transformation, and load (ETL) processes. Leveraging platforms like AWS, Azure, and Google Cloud, he scales AI solutions efficiently. Ian also integrates AI development with continuous deployment pipelines using DevOps practices.

Swarup Mohalik is a principal researcher at Ericsson Research, India. His expertise is in the areas of AI and formal methods, and his work primarily focuses on applying them to telecommunications, service automatization, and the Internet of Things (IoT). He has research experience in formal specification and verification of real-time embedded software and AI planning techniques. Swarup holds a Ph.D. in computer science from the Institute of Mathematical Sciences, Chennai, India, and a postdoctoral fellowship at LaBRI, University of Bordeaux, France.

Marin Orlic is a master researcher in Machine reasoning and Hybrid AI at Ericsson Research, Sweden. Marin works on intent-based management and operations in telecom networks as applications of hybrid AI, especially with symbolic machine reasoning and formal methods. His areas of interest are knowledge representation and reasoning, as well as automated or AI-infused software engineering. After receiving BSc and MSc from the University of Zagreb Faculty of Electrical Engineering and Computing, Marin received a PhD in Computer Science on component-based modelling and analysis of real-time resource constrained systems from the same university.

Leonid Mokrushin is a principal researcher in Cognitive Technologies at Ericsson Research, Sweden. Leonid has been focusing on intent-based management systems for telecom industry based on machine reasoning techniques, especially symbolic AI. With a background in computer science and formal methods, he is currently focusing on knowledge-intensive symbolic AI systems and their practical applications in the telecom domain. Leonid joined Ericsson in 2007 after working as a researcher in Hewlett-Packard Labs in Bristol, UK, and postgraduate studies at Uppsala University in Sweden, where he specialized in the formal verification of real-time systems. He holds an M.S. in software engineering from Peter the Great St. Petersburg Polytechnic University, Russia.

