Multi-Agent Based Autonomic Network Management Architecture

Sisay Tadesse Arzo, *Member, IEEE*, Riccardo Bassoli, *Member, IEEE*, Fabrizio Granelli, *Senior Member, IEEE*, and Frank H. P. Fitzek, *Senior Member, IEEE*

***Abstract*—The advent of network softwarization is enabling multiple innovative solutions through software-defined networking (SDN) and network function virtualization (NFV). Specifically, network softwarization paves the way for autonomic and intelligent networking, which has gained popularity in the research community. Along with the arrival of 5G and beyond, which interconnects billions of devices, the complexity of network management is significantly increasing both investments and operational costs. Autonomic networking is the creation of self-organizing, self-managing, and self-protecting networks, to afford the network management complexes and heterogeneous networks. To achieve full network automation, various aspects of networking need to be addressed. So, this article proposes a novel architecture for the multi-agent-based network automation of the network management system (MANA-NMS). The architecture rely on *network function atomization*, which defines *atomic* decision-making units. Such units could represent virtual network functions. These *atomic units* are autonomous and adaptive. First, the article presents a theoretical discussion of the challenges arisen by automating the decision-making process. Next, the proposed multi-agent system is presented along with its mathematical modeling. Finally, MANA-NMS architecture is mathematically evaluated from functionality, reliability, latency, and resource consumption performance perspectives.**

***Index Terms*—Autonomic networking, multi-agent system, network management, network function virtualization, software defined networking, C-RAN, 5G, future networks.**

Manuscript received November 19, 2020; revised January 21, 2021 and February 1, 2021; accepted February 13, 2021. Date of publication February 16, 2021; date of current version September 9, 2021. This work has been partially funded by NATO Science for Peace and Security (SPS) Programme in the framework of the project SPS G5428 "Dynamic Architecture based on UAVs Monitoring for Border Security and Safety." This work has also been partially funded by the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy Ű EXC 2050/1 Ű Project ID 390696704 Ű Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden. The associate editor coordinating the review of this article and approving it for publication was J.-M. Kang. *(Corresponding author: Sisay Tadesse Arzo.)*

Sisay Tadesse Arzo and Fabrizio Granelli are with the Department of Information Engineering and Computer Science, University of Trento, 38122 Trento, Italy (e-mail: sisay.arzo@unitn.it; fabrizio.granelli@unitn.it).

Riccardo Bassoli is with the Deutsche Telekom Chair of Communication Networks, Institute of Communication Technology, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, 01069 Dresden, Germany (e-mail: riccardo.bassoli@tu-dresden.de).

Frank H. P. Fitzek is with the Deutsche Telekom Chair of Communication Networks, Institute of Communication Technology, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, 01069 Dresden, Germany, and also with the Centre for Tactile Internet with Human-in-the-Loop, Cluster of Excellence, 01067 Dresden, Germany (e-mail: frank.fitzek@tu-dresden.de).

Digital Object Identifier 10.1109/TNSM.2021.3059752

I. Introduction

AUTONOMIC networking is the ability of a communication network to self-manage without (or with minimal) human intervention. In other words, it is the process of enabling the network system to have a self-X property; X can represent managing, configuring, healing, and protecting capability of the network. An autonomic management system has to be able to capture, respond and adapt to the dynamic and evolving behavior of the network, according to the users' and services' demands. In general, the primary aim of autonomic network management systems is not to completely exclude humans from the process. Rather it is aimed at shifting the burden of routine works, such as maintenance, configuration, and management, from network administrator to technology, alleviating tedious and repetitive tasks.

Traditionally, in the networking community, network management was typically performed either manually or by using pre-defined scripts to monitor and control software and hardware. Those solutions implement pre-defined strategies that are triggered by scenarios defined by the network managers. Recently, with the introduction of the concept of network softwarization, the capabilities of network management are enhanced by the programmability and adaptability of the networking infrastructure. However, while modern networks provide a high level of programmability, still automation in network management is supported by limited intelligence in the network.

Therefore, in the next generation networks, autonomic networking is required to scale up the network management capability to meet the expected big growth in the network introducing intelligence in the systems. This is because billions of devices will access and use the network. Thus, it will be extremely complex and cumbersome to continue managing future networks with the current practice. Moreover, from a business perspective, network operators aim at reducing capital expenditure (CapEx) and operating expense (OpEx), while increasing their revenues. Managing next-generation networks would require numerous highly-skilled manpower. Especially, with the arrival of the Internet of Things (IoT), and ultra-reliable machine-type communication (uMTC), there is a dramatic increase in the amount of network information to actively and proactively process. This requires a completely novel approach to network management. Therefore, 5G, and beyond networks [1] are trying to address this issue through some form of network management automation, alleviating the need for a huge number of skilled human network administrators.

Current networking researches mostly focus on network softwarization relying on SDN and NFV. These two emerging technologies enable the traditional static network to be flexible paving the way for network innovation. This opens the door for rapid network evolution and transition to intelligent networking, network automation, and smart networks. Network softwarization consists of mapping hardware-based network functions into software.

Moreover, SDN implies the physical separation of the network control plane from the forwarding data plane [2]. In the SDN architecture, the controller and the forwarding elements in the network are decoupled. This allows for a range of considerably more flexible and effective network management solutions using network programming. SDN enables the network to be programmed and dynamically re-configured, using software to outsource computing to external servers.

NFV is a software implementation of network function, which can run on general-purpose hardware [3]. The architecture of NFV contains three main components, such as

- Network functions virtualization infrastructure (NFVI) consists of the hardware and software that hosts different virtual network functions (VNFs).
- VNFs are the implementation of network functions such as firewall, network address translation (NAT), and packet/serving-gateway(P/S-G), and baseband unit (BBU) in software that could be deployed in an NFVI environment.
- NFV management and network orchestration (MANO) is the place where management and orchestration of VNF are implemented.

SDN and NFV are complementary technologies. Adopting them together provided myriad of innovative possibilities, such as dynamic network configurations, network states estimation and measurement and dynamic network management and control. They are enablers of network automation through the softwarization and orchestration of network functions. A unified architecture for SDN-NFV, which is presented in [4] is depicted in Figure 1.

Another enabling technology for network automation is machine learning (ML)/artificial intelligence (AI), which has recently seen greater advancements. This is expected to play a significant role in the automation of networks [5]. In fact, ML has been applied in various areas of networking such as traffic prediction, resource management, Quality-of-Service (QoS), and network security. ML provides cognition and reasoning capability in automated decision making. The goal of ML is to extract the hidden knowledge from the network, service, and users' behavior using historic data for its training and learning.

SDN and NFV enable the possibility of network programming, virtualization, and orchestration; however, they do not actually automate network management. Therefore, a new framework should be developed to incorporate SDN (for network control and programmability), NFV (for network function virtualization and orchestration), and ML (for knowledge management and cognitive ability) in a single architecture. Moreover, the current architectures typically use a

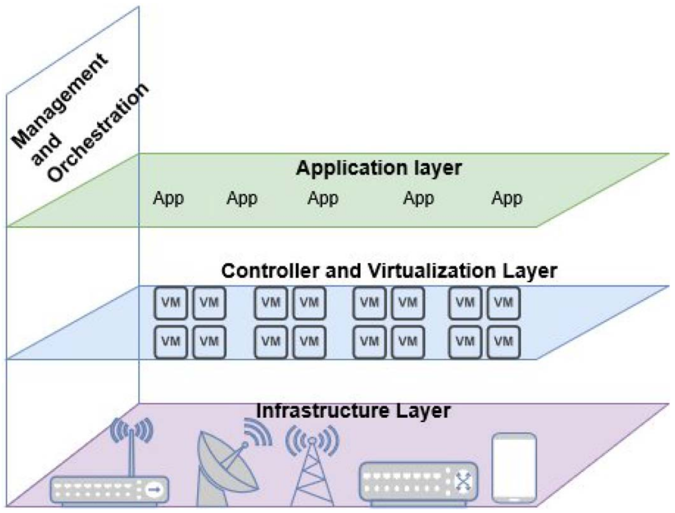

Management and Orchestration
Application layer
App
App
App
App
App
Controller and Virtualization Layer
VM
Infrastructure Layer

Fig. 1. Unified SDN-NFV architectural framework [4].

'monolithic process' in their implementations. However, in the monolithic process, there are strong dependencies between sub-functions. If one of the function fails, it affects or hinders the operation of other sub-functions [6], [7]. In addition, a monolithic SDN controller and an NFV Network Management and Orchestration Architecture (MANO) [3] result to be complex and difficult to scale. This is because a given sub-function can not be modified or scaled independently. Moreover, monolithic implementation of network management is not flexible, reusable, and scalable for complex systems. This makes it unsuitable for network management automation [6], [7]. Sub-system based approach has previously been introduced [6], [7]. However, their discussion is limited to the sub-system creation and distributed deployment. They are limited in the introduction of intelligence in the sub-systems.

Given the above open challenges, this article proposes a MANA-NMS architecture. The core principle behind this architecture is network function division based on autonomic agents. The idea is to subdivide the monolithic network management and control function into sub-functions. It employs agents to be implemented in a virtualized or containerized environments [8], [9]. A given sub-function is deployed as an independent autonomous unit, called agent. Each agent is autonomous and specialized in performing specific network sub-functions, such as network slicing, path computation, and QoS monitoring. Agents are organized to perform the overall network management functionalities. Next, agents could be orchestrated with standard orchestration such as Kubernetes. However, orchestration could also be considered as a sub-function to be implemented as an atomic agent.

Specifically, the overall system behavior and functions should be represented, using multiple interacting autonomous agents. This requires a new architecture that defines the interaction, functionality, communication, and coordination between the agents in delivering the overall system functionality. In general, the overall system behavior needs to encompass different attributes such as agent communication, coordination strategies, decision making, learning, and cognition, in standard architecture.

In this article, we follow European Telecommunication Standards Institution (ETSI) standard's called Generic Autonomic Network Architecture (GANA) reference model [10]. The main goal of the GANA reference model is to prescribe the design and operational principles for Decision Elements (DEs) as the drivers for cognitive, self-managing, and self-adaptive network behaviors. ETSI is performing several GANA instantiations onto various targeted standardized reference network architectures, to enable autonomic algorithm implementation and to design and implement DEs in such architectures, in a standardized way. Our proposed autonomic architecture is presented in comparison with the ETSI's GANA reference model. The authors had already proposed a general architecture for autonomic network management (in terms of the autonomic operator) in [11], [12]. Moreover, they also described and extensively evaluated the self-management of a virtual network based on SDN, via the exploitation of adapted fuzzy cognitive maps at the SDN controller [13].

The main contributions of this article can be summarized as follows:

- Discussing the challenges of autonomic decision, considering decision generation, organization, planning, validation, and execution in the context of network management automation.
- Defining and proposing multi-agent-based autonomic network architecture.
- Mathematical modeling of the autonomic network management system behavior.
- Performance evaluation in terms of functionality and system behavior analysis.

The rest of the paper is organized as follows. Section II reviews existing works. Section III discusses the autonomic network management challenges. The proposed multi-agent-based autonomic management system is presented in Section IV. Next, the mathematical formulation of this multi-agent system is discussed in Section V. Section VI presents a performance evaluation as a proof of concept. Finally, concluding remarks and future research directions are provided in Section VII.

II. Literature Review

Automation is not a new subject since it was applied in several fields such as industrial automation, business process automation, and vehicular-driving automation. For clarity, we mainly divide the literature into two main categories: network automation and architecture, and the application of multi-agents in networking.

A. Autonomic Networking

Autonomics in communication networks can be reviewed considering state-of-the-art before and after softwarization. In fact, SDN and NVF along with cloud computing created a seismic change in the way we design, implement, operate and manage networks.

1) Autonomic Networking Before the Introduction of SDN and NFV: Autonomic computing and networking were first envisioned by IBM in 2001 [14]. Inspired by autonomic computing, numerous research works had been done in autonomic networking since 2004 [15]–[19]. Some works also specifically focused on network management [18], [20]. In [18], the author proposed a system for autonomic network control and management. In [20], the authors presented an interesting approach to enable the self-X capability for network management. An early study of autonomic network architecture is discussed in [21]–[23]. In particular, Bouabene *et al.* [22] aimed at designing a generic architecture by introducing the concept of abstraction. The main idea behind abstraction is to provide a platform for the coexistence of heterogeneous networking styles and protocols. In [23], the authors introduced an architecture called AutoXL. It adopts a knowledge plane (responsible for the management of knowledge) and a Policy plane, (responsible for managing policies). The study in [21] focuses on developing autonomic network monitoring for decision-making through information gathering, using a random number of monitoring devices. The work in [24] developed a specific network architecture for cognitive radio nodes.

There are also other specific network-based autonomic architectures and conceptual models. The works in [25], [26] mainly targeted transport networks. The authors in [25] presented a discussion on an attempt to enable self-optimizing and self-protecting optical burst-switched networks. Similarly, in [26] a model was developed using self-configuration, self-optimization, self-healing, and self-protection for service provisioning in automatically switched transport networks. The developed architecture aimed at enabling autonomic capability through self-governing systems.

Autonomic networking models for wireless networks are also a relatively well-studied subject [27]–[32]. A specific work that focuses on wireless access networks, is presented in [33]. Specifically, it developed a distributed generic and autonomic management paradigm for decentralized management of wireless access networks. Considering the peculiar and inherent challenges of wireless networking, the authors in [27] provided an analysis of autonomic networking. In particular, they aimed at providing autonomic support in the wireless network design process.

In [29], the idea of *function atomization* was presented. Moreover, a consideration of composing autonomic elements was the scope in [30]. This composition aimed at providing a self-governing management system for overlay networks and a self-organizing composition towards autonomic overlay networks.

As per the authors' knowledge, the work in [31] showed the first use of agent-based modeling for automated elements. The authors argued the use of network management automation for wireless sensor networks as a self-recovery model. An interesting conceptual contribution could be extracted from the above works, which is *network element atomization*. This provides the possibility of sub-dividing monolithic network management systems into sub-units. The monolithic network management design has drawbacks such as reusability, scalability, tight coupling between components. Similarly, with a focus on a particular network type, few works attempted to incorporate autonomic in the network operation and management [32], [34]–[36].

Apart from an application on particular network scenarios, there are also works focusing on automating specific functionalities such as QoS monitoring, fault-management, traffic management, and security [37]–[43]. For instance, in [37], a discussion on a model-based approach is presented. The model-based method is to equip the fault-management system with autonomic capabilities. Article [38] studied automation of network operations for fault-management and resilience, while the article in [41] analyzed a memory-enabled autonomic resilient networking.

2) Autonomic Networking After the Widespread Adoption of SDN and NFV: SDN and NFV introduced a dramatic change in the legacy static network, paving the way for network software-based evaluation. From an architecture point of view, SDN provides centralized network provisioning and management capability, considering a global view of the network. Now, we focus on the most recent literature on autonomic networks that have been done since the widespread acceptance of SDN and NFV.

Ironically, after the introduction of SDN and NFV, there are relatively few organized efforts in the research community for network automation [10], [44]–[47]. In [48], an experimental study on dynamic network reconfiguration in a virtualized network environment, using autonomic management, is discussed. An autonomic management system for a software-enabled network is also presented in [45]. The article in [49] presented a novel management architecture and elaborated the ongoing standardization efforts. Most of the concepts in the reference model, autonomic networking integrated model and approach (ANIMA) are heavily inspired by GANA architecture [10]. This architecture for IPv6 based radio access network (RAN) management. More recently, in [50], the authors presented a demonstration of an open-source MANO framework, an extension within the scope of the 5GTANGO H2020 project. They discussed automated service deployment, considering policy-based service level agreement (SLA) provisioning and monitoring. Richard *et al.* [51] discussed a data and knowledge-driven intelligent network for future network transformation. In [52], a first attempt to integrate SDN and autonomic network management was presented. This white paper shows the challenges and opportunities along with the autonomous network framework. An SDN-based autonomic architecture was envisioned in [53]. This architecture aims at enabling the coexistence of heterogeneous virtual networks, supporting hierarchical autonomic management. In [54], the authors introduced an architectural framework to be integrated with the network device operating system. It can work either independently on the device or can be aggregated across multiple devices in a network deployment.

Inspired by the cognition process of human beings, a cognitive network implements intelligent and adoptive procedures in responding to network dynamic behavior. The authors in [55] presented an evolutionary road-map of communication protocols towards the cognitive internet. They discussed the introduction of self-aware adaptive techniques combined with reasoning and learning mechanisms. The goal of these techniques is to tackle inefficiency and guarantee satisfactory performance in complex and dynamic scenarios. In [13], the author proposes a cognitive SDN architecture based on fuzzy cognitive maps. A specific design modifications of fuzzy cognitive maps are proposed to overcome some well-known issues of this learning paradigm. The authors also discussed the efficient integration with an SDN architecture.

Significant standardization efforts are also taking place by prominent standardization organizations such as ETSI and Internet Research Task Force (IRTF). The first project launched to develop a network automation model is Autonomic Network Architecture (ANA) [22]. The project aimed at designing an autonomic architecture to mitigate network complexity enabling protocols and algorithms to operate in an autonomic manner. Recently, ETSI has taken a prominent role in autonomic architecture standardization and adaptation. An Industry Specification Group (ISG) on Autonomic network engineering for the self-managing Future Internet (AFI) has been established under ETSI [56]. Various versions of this model were presented since its inception [10], [57], [58]. More recently, ETSI developed an autonomic network standard reference architecture [10], which is called GANA. In [59], SDN enablers in the ETSI AFI GANA Reference Model for Autonomic Management & Control (emerging standard), and Virtualization impact.

In [60], an implementation guide for the ETSI AFI GANA model is presented considering a standardized reference model for autonomic networking, cognitive networking, and self-management. The document in [61] provides standardization of resilience & survivability and autonomic fault-management, in future networks: this is an ongoing initiative recently launched in ETSI. As per the authors' knowledge, despite the tremendous effort to introduce an autonomic architecture and software model, none of the works have given enough emphasis on the concept of function *atomization*. Network function *atomization* and multi-agent-based representation of *atomized functions* are the core contribution of our work. Network function *atomization* is the sub-division of the monolithic process into smaller functions with cognitive and adaptive capability. With this added autonomic capability, sub-functions are represented as autonomic agents. Autonomic agents are *atomic units* that will be used as building blocks for autonomous network management systems.

Finally, the dawn of IoT and machine-type communications has dramatically increased the network size and complexity [62]. 5G and Beyond networks are evolving towards autonomic.

B. Multi-Agent Systems for Autonomic Networking

Now, we present a brief overview of the part of the literature emphasizing the application of multi-agent-based systems in networking. Multi-agent systems have been used in many areas [63]–[65]. A multi-agent-based configuration in a ubiquitous wireless network is presented in [66]. Where agents are used in the Radio Access Network (RAN). The information is loaded and fed back through agent communication. The Q-learning paradigm is used for the agent's cognitive ability to identifying whenever to make load balancing. In [65], the authors proposed the use of multi-agent systems within AuRA-NMS. They presented the use of multi-agent system (MAS) technology for distributed system automation and network control, considering the issues associated with robustness of distributed MAS platforms, the arbitration of different control functions, and the relationship between the ontological requirements.

Other works focused on more specific applications such as security, service discovery, and service migration. The article in [67] proposed an approach for dynamic service discovery in service-oriented architectures. The approach is based on a multi-agent system using the matchmaking technique. Patri *et al.* [68] developed a generic migration algorithm. This algorithm is derived from a search-based rational agent decision process. Such an algorithm can deal with uncertainties to provide the migration path. This migration is computed by using a maximized utility function. For military networks of computing and communication, the author in [69] presented an initial reference architecture. The reference architecture is a cyber defense application of intelligent software agents. Several works also exist showing the use of intelligent autonomic agents for security application such as risk assessment [70], and network intrusion detection [71]. The authors in [72] presented mobility management over the satellite networks based on virtual agent clustering.

None of the previews works considered an organized management architecture, like in [73]–[75], for the overall network system management. These works are closer to our approach towards utilizing agents, hierarchically organized them into specialized *autonomous-atomic units*. The work in [73] presented an attempt to organize a multi-agent hierarchical system for a self-adaptive network service. They discussed a hierarchical division of agents, effectively providing an abstract model, in which multi-agents are divided and organized into principal agents and sub-agents, according to different layers. Each principal agent is responsible for the management and coordination of sub-agents. Similarly, Taoqing and Xiaoying [73] presented a multi-agent-based management system designed for distributed network. They developed an intelligent management architectural framework. This architecture organizes agents into three types: center layer-agent, region-layer agent, and access-layer agent. An interesting work about control systems is presented in [76] from the control research community. Several existing works related to steady-states of the closed-loop systems studied state analysis of multi-agent-based system [77], [78]. Finally, an article developed a mathematical graph representation of agents' coordination [79].

Despite all the effort to develop a specific or general architecture and several applications of multi-agents, there is no work considering multi-agents for network management system automation, combining the concept of SDN, NFV, and ML in the context of the ETSI GANA reference model.

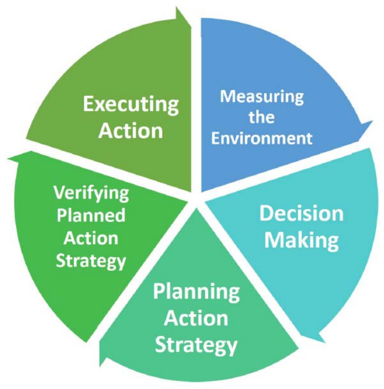


Fig. 2. Network management cycle.

III. Network Management Automation Challenges

Future generation networks are very complex systems with numerous heterogeneous devices, users and services involved. However, they will be very dynamic and consistently advancing. Thus, managing such a dynamic environment involves handling heterogeneous real-time events in a sophisticated manner. This also means dealing with complex events, which require complex event processing. The problem is mainly complicated because of the huge amount of data generated by users, services, and network devices. Managing such an amount of data in such a complex environment, within a short period of time, requires a very tedious, enormous, and sophisticated processing. However, data also contains valuable information about users/services and network states even if it is cumbersome to manage them. Data could systematically be managed and analyzed to extract and use the information for self-management of the network. Extracted information could be used to facilitate services provisioning.

To automate a network management system, a comprehensive understanding of the complete cycle of the automation process is required. This includes measuring and understanding the environment, decision-making process, planning strategy, verification of the action plan, execution of actions, and/or finally monitoring the system behavior [11], as depicted in Figure 2.

- Measuring and understanding the environment determines a network behavior in a reactive or proactive manner, based on continuous observation of the constituent network elements. This enables the accumulation of network knowledge in the knowledge domain. For example, self-monitoring means having the ability to observe own internal state. Observation of this internal state can be a continuous measurement of QoS metrics such as latency, reliability, congestion, and throughput.
- A decision-making process is used to learn, understand and decide what action to take to respond to network events. For example, in a decision-making process, a machine learning model could be used to predict the incoming traffic, which is trained based on the previously accumulated data. Based on the prediction, a decision

could be made on the type and quantity of resources required to satisfy the service's demand.

- Planning action strategy means executing the decision, such as what kind of actions or configurations should be performed on an element to have the required network behavior. For example, once the decision is made to allocate resources, a configuration file needs to be prepared to be executed in network elements or devices. This allocates the required network resources, which are decided in the decision-making process.
- Verification of action plan is the validation of the plan before executing it on the target network elements or devices. This could be a configuration file, which has to be verified and checked for accuracy and consistency.
- Executing action and/or output is implementing the verified or validated action plan. It could be a path configuration for the requested service or reservation of end-to-end network resources for service isolation, as part of QoS provisioning and guaranteeing.
- Monitoring the system's behavior completes the cycle. It continues to analyze the effect of the executed action on the overall system behavior while monitoring the environment for new events and changes. This could be through a loop-back control.

A. Decision Theory From Network Automation Perspectives

Decision-making means selecting one or more courses of action among the available alternatives. Humans make decisions every day. Humans have characteristics of weighing alternatives and deciding based on criteria such as good or bad (i.e., quality). Decisions are generally characterized by gathering knowledge, learning, reasoning, and deciding distinctly. Gathering knowledge is the first step in the decision process. Based on the collected knowledge and experience, humans learn and reason. Humans also learn how to respond without an identifiable reasoning basis, just following instinct. For instance, a child may touch fire just to fulfill his/her curiosity. However, a reasonable decision is based on previous experience through pattern-matching. The human decision-making process is a long-studied subject in research and it is out of the scope of this article. Here, the aim is to get inspired by the human decision process to delegate intelligent decisions in network automation. In general, a decision is the key part of network management in the automation of processes. We aim to emulate and replace human involvement in the network management process [80].

In a similar analogy with the human decision process, intelligent-decision systems could have a knowledge accumulation step to observe and experience the network environment through parametric measurements. Besides, intelligent-decision systems could overcome human cognitive limitations and biases by providing a rational basis for comparison of alternatives [80]. The observation and experience are accumulated in a knowledge domain or *metaspace*. A cognitive process may modify the *metaspace* and the elements of its instances or models [80]. This could be through a learning process, by adding new rules, principles, and concepts to the *metaspace*, or manipulating and modifying the existing instances or models of the *metapace*. This goes with the assumption that the existing instances are already known. They could already be in the knowledge base or they could be added via new instances or models. The *metaspace* and instances constitute a knowledge base of the cognitive process.

A domain expert could be considered to provide knowledge to the knowledge acquisition module. That knowledge is encoded in the knowledge base, usually as part of the development process. The user or decision maker enters the system through an interface. Next, the user may directly access the knowledge such as a reference for past cases, or the inference engine to infer from past cases to a new case. The user may drill down for an explanation of the inference from the explanation module. The intelligent system could have a mechanism to capture, collect and infer knowledge from a domain expert, and pass that expertise to a decision-maker.

Generally, decision making is limited by available information and time to make the decision, together with cognitive ability and resources. Input to a decision-maker includes a *knowledge database* and *model base*. The database contains data relevant to the decision problem, while the knowledge base may include, for example, guidance in selecting alternatives. The model base holds formal models, algorithms, and methodologies for developing outcomes. Processing involves using decision models to simulate or examine different states and find the best solution under the constraints. Feedback from processing can provide additional inputs that may be updated in real-time to improve problem-solving. The output may generate forecasts and explanations to justify recommendations and offer advice. Results are provided to the decision-maker, who may interact with the system to provide additional information or query the results.

Decisions could be categorized as *structured*, *semi-structured* or *unstructured*. An example of a structured decision is the shortest path between two nodes that could be computed as a closed-form solution. However, if we are to provide a routing path for a packet or flow, then the shortest path may not be the best decision. It may not consider the current state of the network that could affect the routing such as congestion and available resources. The current state of the network could be so dynamic that we may not have the complete and exact network state information. This leads to a sub-optimal decision, which is considered semi-structured. In the unstructured decision, the problems have no agreed-upon criteria or solution and rely on the preferences of the decision making elements.

Depending on the time it takes to reach the final decision, decisions can also be categorized as instant decisions and long-term decisions. Examples of instant decisions are routing packets or flows from source to destination. Whereas, an example of long-term decision is the scheduling of system or re-configuring decisions for periodic updates.

In general, taking care of the network without administrators' conscious knowledge requires making an intelligent decision through decision generation, organization, verification, applying, and cheeking system behavior through loop-back control.

1) Data Analytics and Cognition: Managing the network is the ability of the network administrator or autonomic system to maintain the network within the desirable behavior. This is a very complex problem. It requires handling multiple and complex real-time events such as monitoring the network and computing resources which could be a physical channel (wired and wireless channels), encoding techniques, computing resources, stored network, and user data, users, service, and network devices (ports, links, configuration files) so on.

Reasoning and cognition are the key pillars in a decision-making process. Cognition is learning, analyzing, and reasoning to generate decisions. Useful information could be generated about users, services, and networks by analyzing the data. And then, this information could be used for reasoning out and generating management decisions, such as allocating resources, monitoring and guaranteeing QoS, authenticating users, and protecting the network from intruders.

There are various methods to extract information for given data. AI tools provide powerful aids in solving difficult applied problems that are often real-time, involving large amounts of distributed data, and benefiting from complex reasoning. AI tools could be fuzzy logic, case-based reasoning, evolutionary computing, artificial neural networks, and intelligent agents [74], [80].

Currently, machine learning is the most adopted means of extracting valuable information about users, services, and networks. With the interconnection of machine type devices, there is a colossal amount of data to learn from. The goal of machine learning is to extract the hidden knowledge about the network and users through training, using a network data set.

2) Generating Cumulative Decision: Communication networks will be very dynamic and complex. Handling events appropriately to maintain the desired behavior of the network requires making various intelligent and adaptive decisions. Autonomic generation of various decisions through data gathering, analysis, and intelligent reasoning are critical steps. These decisions are dependent on various internal and/or external factors. Events could be triggered by external factors such as service requests, device discovery, new service discovery, topology change discovery, traffic classification, traffic prediction (amount of traffic on each link may be for congestion control), traffic scheduling, fault or anomaly behavior detection, and security breach. Internal decisions could be the network update, scheduled re-configuration for network optimization, available resource monitoring, handling network device failure or misbehavior. All types involve complex correlations. For instance, network re-configuration for optimization of decisions may affect the QoS as it may involve a configuration that may limit the availability of allocated resources at a given time. The QoS for one service may affect the decision for another service, in case of limited network resources.

Therefore, in the network automation process, decisions have to be made considering other interrelated decisions to see the cumulative effect on the overall network behavior. Autonomous decision-making requires the generation of various decisions such as automatic event handling through scheduling of events and tasks. Moreover, the crucial part of autonomous decision-making is intelligent reasoning and cognition that as discussed previously above. That could be supported by AI/ML techniques.

3) Decision Organization: Networks systems consist of heterogeneous devices, which involve cross-layer complex-events such as signal transmission, processing, coding, decoding, routing, load balancing, slicing, QoS monitoring, resource management. Decisions at various layers have to be made and each decision has its peculiarities but also dependencies on one another. For example, the decision made to provide end-to-end QoS for a certain type of traffic involves decisions on protocols, the physical layer, the data-link layer, and the network layer. Moreover, a decision made at the physical layer could impact the once's at the data-link layer or network layer. Hence, abstraction is necessary to isolate and focus on particular aspects. Various approaches could be followed. One approach could be using a standard seven-layer OSI or four-layer TCP/IP model. Another one could be using a new customized version of these layers as followed in the ETSI GANA model. In our proposed architecture, we followed ETSI's GANA model for decision abstraction and organization [10], [74].

4) Overall Autonomic Decision Assessment and Verification: The outcome of an autonomic decision should be assessed according to both the process to and outcome of, decision making. The process of decision making is very critical. And it should be assessed since it has a big implication on the time it takes to decide and the additional computation and communication overhead. The quality of a final decision is also very important that should be assessed based on a defined criterion, which could be specific for a given management demand. Assessing a decision requires defining measurement matrices.

In the network management automation process, we need to have a mechanism to validate the final decision for consistency, accuracy, and optimality. The final decision's measurement and verification mechanisms should consider consistency in collective system behavior. That is because, in a cumulative decision, individual decisions may not be the best or optimal one, since multiple factors should be taken into considerations. It is important to model desired network behavior pattern for proper evaluation of the performance of the automation techniques. Typically, *healthy* network behavior initially is decided by a network service provider or network administrator, through a set of behavior constraints.

Verification mechanisms could be a mathematical procedure to verify and validate intelligent decisions for autonomic networking. It could be choosing the best solution from a set of, sometimes infinite, possible solutions. This may involve making sense out of ambiguity or contradiction in selecting the alternatives. A possible simple example of contradicting or conflicting decisions in autonomic network management could be a security alert with credible threats, such as a DDoS attack on the network that may require network reconfiguration for self-protection and attack mitigation. However, at the same time, the network may be serving critical users or verticals such as remote surgery or an automated car. An early act on the intruder could prevent or reduce the damage. However, the response may require some possible configuration scenarios, which could impact the critical services. Therefore, how to solve such conflicting situations has to be defined or acquired and developed through training, to be embedded in the network automation process. Finally, the generated re-configuration files have to be verified for the impact that it would have on the network (the accuracy of the intended outcome of the actions to be taken). In general, we need to verify and validate the final solution before applying it. And it is beyond the scope of this article to exhaustively discuss the validation and verification mechanisms to each type of decision.

5) Decision Execution: In network management automation; action or decision execution is needed to fulfill network management demands. An action could be re-configuration, database updating, path re-computing, authentication, authorization, and monitoring [11]. The outcome of decision generation needs to be applied to the system to change the part of the (overall) network system to fulfill service or management demands. The network's behavior change comes by applying the final verified decision to change the network environment so that it behaves as per the network management demands. The response to the change has to be done appropriately and timely. This is very critical. A simple example could be when we make a decision to route a flow based on the currently available path. The path status or available resource changes very quickly, which may invalidate or outdated the decision. This is mostly because, in network automation, the aim is to develop a context-aware response to the demands. Decision execution as apart of the automated process is also a vast and challenging area that needs to be explored.

6) Monitoring System Behavior: After implementing the final decisions, the network behavior has to be monitored to observe how the action affects the system behavior [11]. This is to evaluate the applied solution, which could be for consistency, correctness, and appropriateness. A typical mechanism that is applied in most automation comes from control theory, such as loop-back or feedback control. This also requires data gathering and data processing discipline, which correlates data from multiple sources to identify patterns of events that the network is expected to behave. Exploring this from an autonomic network management perspective is also a critical research area that needs to be investigated.

IV. Multi-Agent Based Autonomic Network Management System

MANA-NMS architecture divides the monolithic management system into sub-functions to be performed by autonomic multi-agents. It can be considered as a way to organize parallel computing or task execution, as it includes the division of a complicated task into several more simple ones. The tasks are performed by the group of agents autonomously. In other words, we aim to provide a systems architecture called MANA-NMS which partitions a complex network management system into *autonomic units* or modules. These autonomic units are *atomic units* capable of performing tasks autonomously while interacting with the environment as well as with other agents. Modularity also helps in reducing network management complexity through the independent implementation of sub-units and components. These units are reusable.

As previously mentioned, we focus mostly on proposing general principles and directions to follow for automated network management. Moreover, the proposed MANA-NMS architecture requires to deal with a vast area and numerous challenging topics. Therefore, studying all potential areas and topics including implementation technologies are beyond the scope of this article. Nevertheless, our article attempts to investigate the feasibility of the proposed architecture through mathematical analysis of a preliminary representative system.

A. Multi-Agent Systems

A network management system is a complex system having multiple events such as multiple-input, multiple-processing, and multiple-output or action. We identify that multi-agent systems are the best candidate in developing such complex, distributed, large-scale heterogeneous systems [64]. In addition to automation, the proposed MANA-NMS architecture provides modularity, re-usability, scalability, robustness, and re-configurable.

As a set of autonomous entities, agents are capable of performing a given task autonomously. Agents have the capability to replicate a single or multiple collaborative decision-makers. In a multi-agent system, agents are the smallest building blocks. The blocks are specialized in performing specific functions. Using the appropriate set of autonomic agents, it is possible to build a complete multi-agent system to replace the complex and distributed monolithic systems like network management systems. The constituting autonomic agents can communicate, collaborate, cooperate, and coordinate to accomplish complex tasks. In building a multi-agent system, several challenging aspects, such as autonomy, collaboration, activity, reactivity, communicativeness, and goal setting, have to be addressed.

Agents could be distributed to perform distributed tasks or assigned to handle different functions in a localized and specialized rule. Or they could be centrally located to perform the tasks arriving into the central computing or task performing multi-agents. Agents could also be homogeneous or heterogeneous. Homogeneous agents are agents with identical characteristics. They could be designed or instantiated to perform workload sharing. They could collaboratively perform large tasks that could be sub-divided into sub-tasks to be assigned to different agents. Moreover, agents could also be organized hierarchically [73], [74]. They could be organized to handle different functionalities.

An example of a multi-agent based system for network automation could be to perform network management functions. For example, some agents could be designed to monitor and analyze the network parameters that are required to provide given services or tasks. Other types of agents could be designed and dedicated for QoS monitoring for a given type of service. Some other types of agents could be decision aid

TABLE I
MICROSERVICE AND MULTI-AGENT COMPARISON

Design Principles	Microservice	Multi-Agent System
Autonomy	Require no human intervention	Require no human intervention
Interaction	Interact with other microservices using RESTful APIs and HTTP	Interact with other agents using ACL
Re-activity Response	Respond to HTTP requests in a timely manner	Measure, understand and perceive the environment to respond in a timely manner
Pro-activity Response	Unable to take the initiative	Respond and take the initiative
Loose Coupling	Monolithic systems are decomposed into loosely coupled sets of highly cohesive co-located services	Autonomous and loosely-coupled solution providers

or classifier or RAN slicer. Each of them is specialized in performing a specific task as per the demand. Finally, the overall system could represent an automated network management system.

A multi-agent system has been applied in implementing complex software systems [64]. They are capable of automating heterogeneous and complex real-time systems. A network management system is a heterogeneous complex system with real-time and complex environments. Therefore, we propose that a multi agent system is an effective approach for the network automation process. However, several challenges need to be addressed to exploit the solutions offered by this approach. This includes decision generation which involves environment monitoring, data analytic, cognition, complex event processing (CEP), and context-awareness.

B. Multi-Agents and Microservices

The industry is heading in the direction of highly decentralized and loosely-coupled softwarized systems such as distributed NFV and mobile edge computing (MEC). Currently, large-scale software development is based on microservices. A microservice is a model, in which systems are built from small and loosely-coupled services. They have the potential to be a tool for building systems such as network management systems. They adhere to the Isolated state, Distribution, Elasticity, Automated management, and Loose (IDEAL) coupling design principles. Multi-agent systems have many similarities with microservices [8], [9]. Their core similarity is related to independence and loose-coupling.

Despite their similarity, agents have an advantage over microservices for dynamic network management system's design. Agents have the autonomic capability to provide multi-agents the advantage in building automated systems. Recent work presented in Collier *et al.* showed the possibility of combining microservices and multi-agent systems [9]. Table I presents a comparison between microservices and multi-agent systems.

C. Proposed Architecture

Currently, there is no universally accepted definition for what is an agent. We define every component in the network management system as agents, in our context. We propose agents to be implemented with full autonomic capability, which is derived from agent-design principles. The overall network management becomes a multi-agent system. In principle, agents are assumed to have multiple algorithms such as cognitive and intelligent algorithms, which could be based on ML or Deep Learning (DL), computational intelligence, and other intelligent algorithms. This provides a cognitive ability for agents in their autonomic operations. In general, the internal components and specification of agents vary depending on the functionality and type of tasks or services they are designed to perform. A specification of a standard agent is discussed later in Section V.

1) Network Functions Atomization: To subdivide and organize functions into agents, let us first elaborate the concept of network function *atomization*. *Network function atomization* (*atomic* agent in our context) is the idea of organizing autonomic functions at different abstraction level. This with the obvious reason of separately dealing with each layer. Functional unit representation is based on autonomic agents. The abstraction is based on the GANA model. Network management functions could be physical-layer, device-level, and network layer functionality. Physical-layer functionality (low-level *atomic* actions) could be network resource block (channel) allocation or physical transmission and reception of frames, error correction, data framing, etc. Device-level functionality could be packet/flow forwarding etc. Network-level functionality (high-level *atomic* actions) could be end-to-end QoS provisioning and monitoring etc. All those functions could be provided by autonomic agents with suitable internal components. Since agents represent various functionalities, they need to be modeled and categorized accordingly.

In a general category, the types of agents could be service-type or device-type agents. Alternatively, to align with ETSI GANA model definition of decision units [10], agents could be categorized into four types: protocol-level agents, representing protocol-level DE, function-level agents representing, function-level DE, device-level agents, representing device-level DE, and network-level agents, representing network-level DE. Each function in each layer could be represented as agents in the proposed architecture. All agents are specialized and dedicated to performing tasks autonomously while coordinating and communicating with other agents. However, we will limit the description to only network-level functions. The following are the most typical network-layer functions, that could be defined as agents:

- Topology Management Agents (TM-AG)
- Routing Management Agents (RM- AG)
- Network Slicing Agents (NS-AG)
- Service Function Chaining Agents(SFC-AG)
- Forwarding Management Agents (FM-AG)
- QoS Management Agents (QoS-AG)
- Mobility Management Agents (MM-AG)
- Security Management Agents (SM-AG)
- Fault Management Agents (FM-AG)

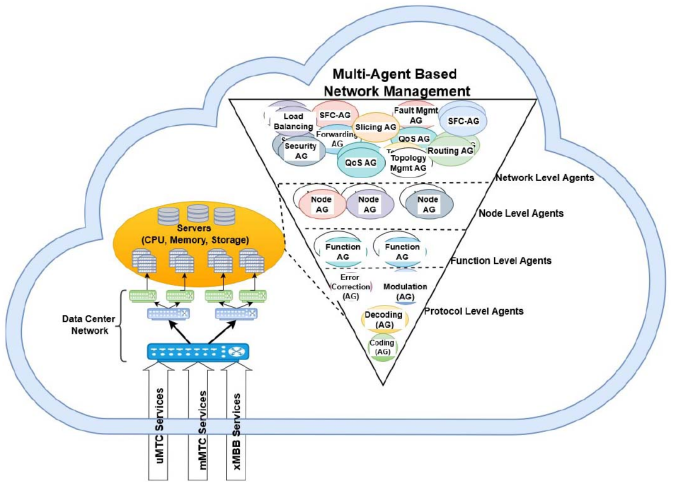


Fig. 3. Multi-agent based representation of MANA-NMS for cloud RAN.

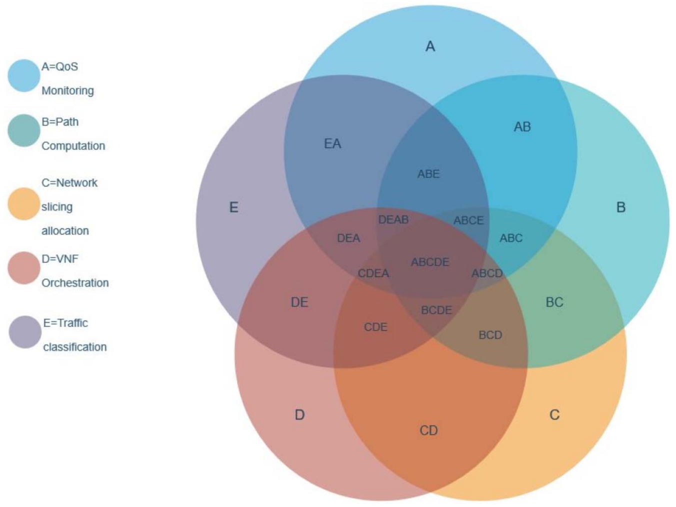


Fig. 4. Simple example for network level agent relationship.

- Resilience and Survivability Agents (RS-AG)
- Service and Application Management Agents (SAM-AG)
- Monitoring Management Agents (MM-AG)
- Generalized Control Plane Management Agents (GCPM-AG)

Now we focus on building the multi-agent-based autonomic network management system. Therefore, by using the above *atomic units* as a building block, we should be able to realize an autonomic network management system, replacing the existing monolithic management system. Figure 3 shows an example of a multi-agent autonomic network management system. The figure also shows functions' abstraction into four layers/levels/ protocol-level agents, function-level agents, node-level agents, and network-level agent.

In the process of building the autonomic management system, there could be a complex relationship between functions/sub-units that should clearly be defined. Each function performs specific service processing. In doing so, there could be an interaction of functions or units to accomplish the overall goal of service processing in the network management system. For example, when a given service arrives at the input of the system, it has a set of service constraints that have to be satisfied by the system for successful processing of the service. To perform that the functions or units collaborate or cooperate or compete to satisfy and perform the required service processing. This relationship could be complex in that in satisfying the service constraints, there could be contradictory requirements. This means that for instance to satisfy the reliability constraints of a given service while maintaining latency and resource constraints could be a complex optimization if we consider the overall system state. An example of the units composition relationship is presented in Figure 4. The overlapping of agents shows the close physical and/or logical proximity.

2) Proposed MANA-NMS Architecture in Comparison With GANA Architecture: As indicated above, agents are used as building blocks, replacing the network function to autonomously perform network management functions. Agents communicate by passing ACL messages to coordinate and they could share information with each other. Moreover, there are also different types of agents that should be organized in layers/levels. Figure 5 shows agents' layers organization in the proposed multi-agent-based architecture. The picture also depicts a comparison between the proposed MANA-NMS architecture and ETSI GANA architecture. Each agent represents an *atomized* network function such as QoS management function. Agents are expected to perform the intended original functionality of the network components or functional units. Multi-agents should represent all softwarized network components and functions as a system. Using the MANA-NMS architecture, agents are categorized as protocol level, function level, node/device level, and network-level agents as depicted in Figure 5.

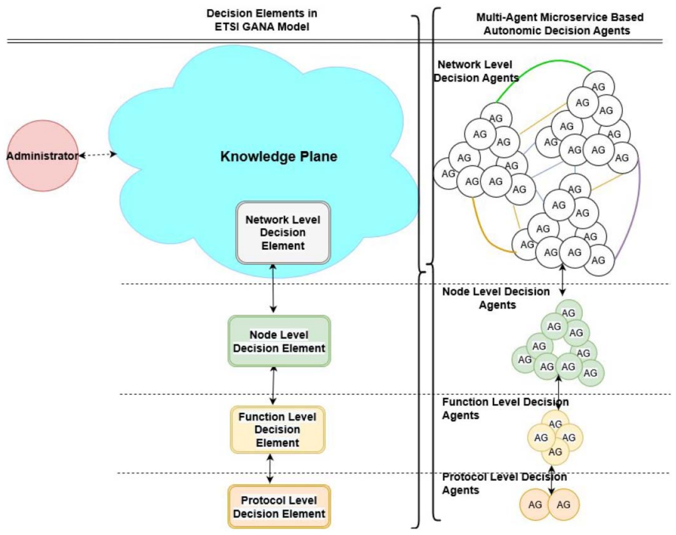

Decision Elements in ETSI GANA Model
Multi-Agent Microservice Based Autonomic Decision Agents
Network Level Decision Agents
Administrator
Knowledge Plane
Network Level Decision Element
Node Level Decision Agents
Node Level Decision Element
Function Level Decision Agents
Function Level Decision Element
Protocol Level Decision Agents
Protocol Level Decision Element
AG

Fig. 5. Multi-agent based representation of GANA architecture.

3) Proposed MANA-NMS Architecture in Comparison With SDN-NFV Architecture: As discussed previously, SDN and NFV are the enabling technologies for network automation. The joint architecture is presented in Figure 1. Here, we present our proposed architecture in comparison with

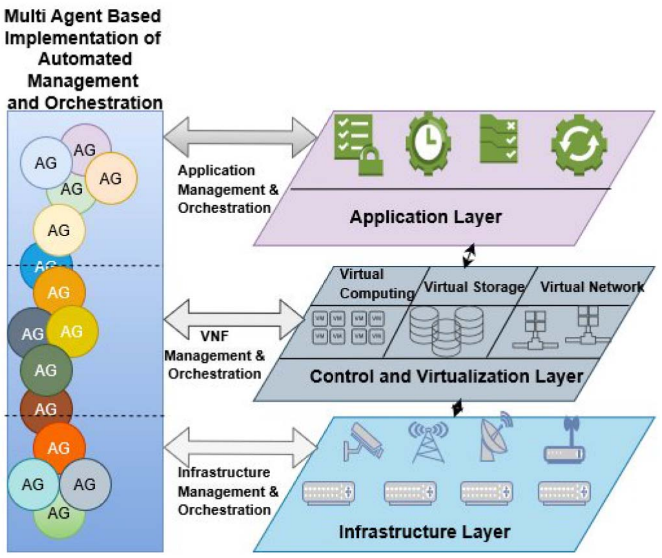

Multi Agent Based Implementation of Automated Management and Orchestration
AG
Application Management & Orchestration
Application Layer
VNF Management & Orchestration
Virtual Computing
Virtual Storage
Virtual Network
Control and Virtualization Layer
Infrastructure Management & Orchestration
Infrastructure Layer

Fig. 6. Multi-agent-based network management automation on unified SDN-NFV architectural framework.

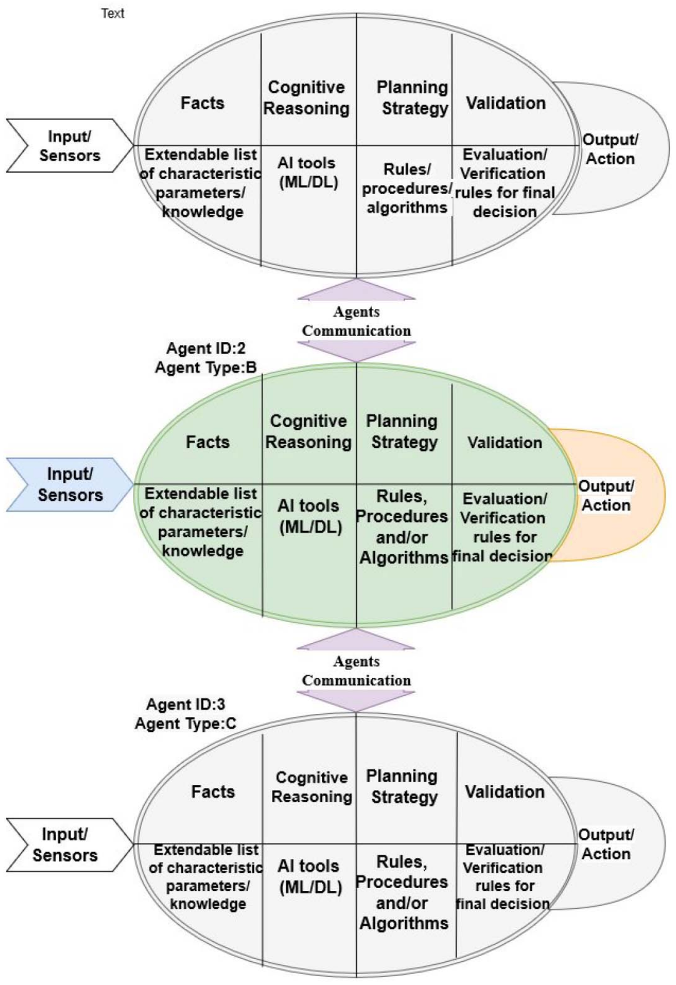

Text
Input/ Sensors
Facts
Cognitive Reasoning
Planning Strategy
Validation
Extendable list of characteristic parameters/ knowledge
AI tools (ML/DL)
Rules/ procedures/ algorithms
Evaluation/ Verification rules for final decision
Output/ Action
Agents Communication
Agent ID:2
Agent Type:B
Rules, Procedures and/or Algorithms
Agent ID:3
Agent Type:C

Fig. 7. Fundamental components of autonomic agent.

SDN-NFV architecture. Figure 6 shows a mapping of the proposed architecture in an SDN-NFV based architecture. As it can be seen from Figure 6, the management units are built using *autonomic units*. These autonomic *atomic units* are organized to cooperate and collaborate, and perform the overall network management functions. Moreover, as indicated in the above sections, the function of SDN and NFV could themselves be built using agents as the building blocks. Each component or modules of the SDN controller could be designed as *autonomic agents*. And the overall controller as a multi-agent system. However, a detailed discussion of this topic is beyond the scope of this article and left as future work.

V. Mathematical Representation of Multi-Agent Autonomic Network Management System

We proposed a multi-agent system for network automation. In the system, several agents are organized and coordinated to perform the network management autonomously, based on underlying control rules, laws, or policies. Developing multi-agent systems with a certain desired behavior requires analyzing the effect of each constituent agents to understand the collective system dynamics. There are few simulation tools available in the research, which could be used to analyze our MANA-NMS architecture. This is due to the complexity and dynamic nature of the network management environment. Therefore, we use mathematical modeling and analysis for the proposed multi-agent-based automated network management system to understand the collective multi-agent-based system behavior.

Let us start with characteristic description and specification of agents, which are considered as the *atomic units* of MANA-NMS. For the sake of simplicity and readability, here we characterize an agent with a very general specification. As depicted in Figure 7: an agent has an ID, for its identification; a *TYPE* describes agent category; an *INPUT*, is a service-request accepting and/or environment measuring points; *FACTS*, are knowledge database of the agent; *COGNITION* (*REASONING*) *UNIT* gives agents the reasoning capability; *PLANNING STRATEGY* organizes the steps or procedures for the action to be taken to satisfy the requested service; *VALIDATION* is the unit that verifies the action plan for consistency; and an *OUTPUT/ACTION* is the final decision (result or action) to be taken by the agent. An agent also has a mechanism to communicate with other agents through agent communication units. Following the definition of Genesereth, in [81], an agent is defined as a software component that is capable of exchanging knowledge and information. We consider the standard agent communication language called ACL messaging [82]. ACL provides a complex knowledge exchange mechanism such as facts, agent's goal, strategy, and plans.

First, we denote an agent Ag with a characteristic specification: $Ag = (A_{ID}, A_{type}, F, R, PS, V)$, where ID is the agent's identification number, A_{type} is its type which could be one of the following $A_{vh}|A_{prtcl}|A_{fun}|A_{nod}|A_{ntk}$. A_{vh} stands for virtualized or containerized agent type, which represents a hardware device such as switches or routers. A_{prtcl} stands for protocol-level agents. A_{fun} stands for function-level agents. A_{nod} stands for node-level agents. A_{ntk} stands for

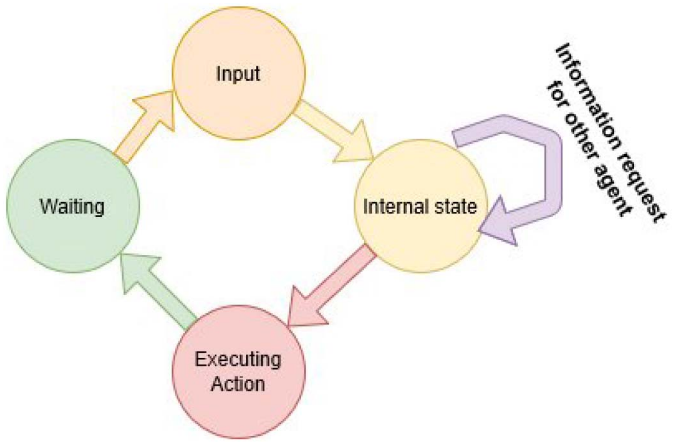


Fig. 8. Agent states.

network-level agents. F stands for facts or lists of characteristics parameters that describe agents' knowledge base, which is extendable. The F list includes information such as the type of provided services and the desired delivery time. R is a reasoning mechanism such as machine learning techniques to generate decisions. *PS* is the action planning strategy such as configuration values and procedures to be performed. Finally, V is the validation technique along with the evaluation criteria to be used to verify the action plan for consistency and correctness.

We employ a stochastic approach for the analysis and mathematical formulation of the multi-agent system behavior [83]. First, let us define a simplified agent states assuming agents with zero probability of failure. Le us assume an agent having four states: *start/input*, *internal execution*, *action* and *waiting* as depicted in Figure 8. This is a very general agent's state definition. In actual scenarios, each of these states may consist of a single state (action or behavior), or a set of states (behaviors). For instance, when an agent is in the internal execution state, it could be performing *reasoning*, *strategy planning*, *validation* or even *communicating* with other agent and *waiting* for responses. To find the collective system behavior of a multi-agent system such as system stability, steady-state, and network equilibrium, we assume the system as Markov process. Agents cannot meet Markov's property directly since they could use the memory of previous actions in decision making. It is commonly right to assume to include the contents of an agent's memory as part of its state to maintain the Markov property. Even if this may increase the number of states, many types of software agents and sensors do satisfy the Markov property with a reasonable number of states.

Before delving into the description, let us restate the Markov property; the configuration of a given system at time $(t+\Delta t)$ depends only on the configuration of the system at time t. Let $p(n,\ t)$ be the probability of an agent be in state n at time t. With this assumption, the change in probability density is written as [83]:

$$\Delta p(n,t) = p(n,t+\Delta t) - p(n,t) \tag{1}$$

The configuration of the system, which is defined by the occupation vector is given by

$$N_k = \sum_{j=1}^{M} n_k \times \tilde{u} \tag{2}$$

where k is a discrete state with value $k = 1, 2, 3 \ldots, M$. N_k is the number of agents in state k. The probability distribution $P(N_k, t)$ is the probability of the system being in the configuration N_k at time t. $\tilde{u}$ is the associated unit vector to each state. The transition rates for a Markov process, which is defined by the conditional probabilities, is given by [83]:

$$W(N_k|\tilde{n},t) = \lim_{\Delta t \to 0} \frac{P(N_k, t+\Delta t|\tilde{n},t)}{\Delta t} \tag{3}$$

The above equation describes the transitions to and from states in the configuration of the system, and it is known as the *Master Equation*.

Let $w_i j$ be the transition rate from state j to state i then the *Rate Equation* is given by:

$$\frac{\partial n_k}{\partial t} = \sum_{j=1} w_{jk}(\langle N_k \rangle) n_j - \langle n_k \rangle \times \sum_{j=1} w_{kj} \times (\langle N_k \rangle) \tag{4}$$

The first term describes an increase in the occupation number n_k due to a transition to state k from other states, while the second term describes the loss due to the transitions from the state k to other states. This equation could be used as a tool to analyze the collective dynamics of an agent-based system. In simple terms, the rate equation is useful to study the system configuration and evolution in time.

A more useful system equation could be found if we consider agents as queue to model the overall system as M/M/1. Let us assume the service arriving at the system has a service arrival rate (λ_{tot}) and a mean serving rate (μ_{tot}). Therefore, in formulating the overall system dynamics, we use the overall system as an input-output system to analyze the resource and reliability constraint, serving capability, and processing latency.

A. Service Arrival, Scheduling, and Admission

Before presenting the formulation, let us define the incoming services to be scheduled for processing in the system. For the purpose of prioritizing service scheduling, we characteristically define services. Each characteristic and preference is represented D_i to be the *demand set*, which defines the requirements in terms of *latency*, *reliability*, *processing demand (workload)*, and *service class (priority indicator)* for that specific application. If the commodity is *elastic* $D_i = \{[\tilde{Th}_{max}], [\tilde{\tau}_{min}, \tilde{\tau}_{max}], [\tilde{\rho}_{min}, \tilde{\rho}_{max}], L_{ij}, \beta\}$. The first two requirements (latency and reliability) are ranges that represent services' *Type 2* and service *Type 3*, otherwise *inelastic* cases which represents service *Type 1*. The set becomes $D_i = \{\tilde{c}, \tilde{\tau}, \tilde{\rho}, \beta\}$, with constant first three members. The priority β is a value in the range $[0, 1]$, which is used to classify the serving priority of the commodity; the sum of all the priorities is 1.

Based on the priority, services are categorized into three type of services: service *Type 1* (*inelastic*), service *Type 2* (*elastic*), and service *Type 3* (with relaxed latency demand). In general, service traffic consists of all the three types of services. The total service workload composition and proportionality equation at a given time is given by

$$\begin{aligned} Srv_{tot} &= \sum_{i=1}^{N_{t1}} Srv_{t1} + \sum_{i=1}^{N_{t2}} Srv_{t2} + \sum_{i=1}^{N_{t3}} Srv_{t3} \\ &= \alpha_1 \times Srv_{tot} + \alpha_2 \times Srv_{tot} + \alpha_3 \times Srv_{tot} \end{aligned} \tag{5}$$

where $\alpha_1 \times Srv_{tot}$, $\alpha_2 \times Srv_{tot}$ and $\alpha_3 \times Srv_{tot}$ refers to Type 1, Type 2 and Type 3 services' workload in GOPS, respectively, N_{t1}, N_{t2} and N_{t3} refer to the number services of each type.

The overall service admission probability of C-RAN system is defined as the ratio between total admitted services and total requested services

$$Ap_{EDC} = \frac{\sum_{i=1}^{N_{srv-Adm}} \times Srv_i}{\sum_{i=1}^{N_{srv-adm}} Sr_i + \sum_{j=1}^{N_{srv-rej}} Srv_j} \tag{6}$$

where Ap_{EDC} is the total service admission probability of a given system (Edge Data Center in our case), $N_{srv-Adm}$ is the total admitted service workload, $N_{srv-rej}$ is the total rejected service workload.

Let us assume service arrival rate to be Poisson distribute. Using Poisson addition property, the total service arrival rate (λ_{tot}) at a given time is given by:

$$\lambda_{tot} = \sum_{i=1}^{Nt1} \lambda_i + \sum_{j=1}^{Nt2} \lambda_j + \sum_{k=1}^{Nt3} \lambda_k \tag{7}$$

where $N_{t(1-3)}$ service generating sources. We assume the critical determining factor for the system processing capacity is only dependent on the service processing agents, which are directly involved in service processing. The total serving rate of the system is given by

$$\mu_{tot} = \sum_{i=1}^{N} \sum_{j=1}^{M} \mu_{ij} = N_{Ag} \times U_{srv} \tag{8}$$

where N is the types of agents, M is the number of instantiated agents of a given type, N_{Ag} represents the total number of service processing agents that are required for a given service to complete its service cycle. We can calculate the mean number of services in the system using

$$\rho_{tot} = \frac{\lambda_{tot}}{\mu_{tot}} \tag{9}$$

The agents have finite resources to perform the required service processing. Hence, we need to consider this in the constraint equations. A constraint equation is developed considering the overall system to have multiple agents, working together performing the tasks. Following the M/M/1 queueing modeling, we can apply Little's Law, which states that the incoming rate of services has to be equal to the service rate to have a stable system with no rejected services. The overall load of the system must be equal to or less than the processing speed of the overall system. That is given by

$$\mu_{tot} \geq \lambda_{tot}. \tag{10}$$

B. Service Latency in Edge Data Center

The overall delay in the system is the sum of mean delay in the serving agents and mean delay in the queues. In the model, the dependence of service processing and service transmission between cascaded agents is eliminated by approximating the arrival process of each ïňĆow at the subsequent agents as a Poisson process. This enables us to employ an M/M/1 queueing model to calculate the average processing latency. Therefore, the total delay is given by

$$D_{tot} = \sum_{i=1}^{K} Qu_D + \sum_{i=1}^{K} Ag_D \tag{11}$$

where Qu_D and Ag_D are mean delay in the agent and mean delay in the service scheduling queue, respectively. The above equation assumes all the required ordering sequences of agents. However, the real application varies from service to service, because of the possibility of splitting services between multiple agents for a faster process of a long single process.

Moreover, to calculate the total delay in passing through the required sequence of service agents, we sum all the delays incurred by the given service in the service function chain (SFC), assuming a queue at each stage of the Agent sequence. This is given by

$$\tilde{\tau}_{queue} = \sum_{i=1}^{N} \frac{1}{M_{sfc} \times U_{Ag} - \lambda} \tag{12}$$

where M_{sfc} is the processing capacity of an agent sequence (the number of agent types that a service require (N) multiplied by each Agents' capacity U_{Ag}), ignoring agents' CPU (overhead) requirements. That means all the allocated capacity of service processing agents in the data center would be used for service processing (without any other processing overhead), given by

$$M_{sfc} = \frac{CPU_{tot-srv}}{N_{Ag}} = U_{Ag} \times N_{Ag} \tag{13}$$

Therefore, to meet the end-to-end delay for a service that is constrained to pass through a given sequence of agents, the data center delay is the sum of SFC decision delay (overhead), queuing delay, processing delay at each agent, and interconnecting links delay (virtual links) between hosts, so that

$$\tilde{\tau}_{Edc} = \sum_{i=1}^{K_{link}} \delta_i + \sum_{i=1}^{N_{Ag}} \frac{L_{ij}}{U_{Ag}} + \sum_{i=1}^{N_{Ag}} \frac{1}{M_{sfc} \times U_{Ag} - \lambda} \tag{14}$$

where K_{link} is the number of links interconnecting different sequence of agents in the SFC. The above equation shows the total delay at edge data center.

C. Data Center Based Multi-Agent System Reliability

Finally, we provide reliability formulation for server and agents. The overall reliability is calculated by considering the reliability of server and agents to be mutually independent events [84]. A given physical server's reliability is given by [85], [86]

$$\tilde{\rho}_{phs} = \frac{MTBF_{phs}}{MTBF_{phs} + MTTR_{phs}} \tag{15}$$

where $MTBF_{phy}$ is mean time between failure, and $MTTRphy$ is mean time to repair. Similarly, a given Agent's reliability is given by [85], [86]

$$\tilde{\rho}_{Ag} = \frac{MTBF_{Ag}}{MTBF_{Ag} + MTTR_{Ag}} \tag{16}$$

where $MTBF_{Ag}$ is mean time between failure and $MTTR_{Ag}$ is mean time to repair. Therefore, the overall reliability of edge data center (EDC) is given by

$$\tilde{\rho}_{Edc} = [(\Pi_{i=1}^{N}\tilde{\rho}_{vlink}) \times (\Pi_{i=1}^{K}\tilde{\rho}_{phs}) \times (\Pi_{i=1}^{N}\tilde{\rho}_{Ag})] \tag{17}$$

To improve agents reliability, we may need to apply backup and overprovisioning of agents. If we apply backup for agents, the reliability increases. Therefore, the reliability of SFC incorporating backup Agents is defined as [87], which is given by

$$\tilde{\rho}_{Ag-with-backup} = \Pi_{i=1}^{N}(1 - \Pi_{i=1}^{N}[1 - \tilde{\rho}_{Ag})] \tag{18}$$

The reliability of edge data center considering the data center links becomes

$$\begin{aligned}\tilde{\rho}_{Edc} = [(\Pi_{i=1}^{N}Pr_{link}) \times (\Pi_{i=1}^{K}Pr_{phs}) \\ \times(\Pi_{i=1}^{N}\tilde{\rho}_{Ag-with-backup}]\end{aligned} \tag{19}$$

Using these techniques, maximizing reliability increases the total number of idle agents. Idle agents could be backup agents, which are waiting to replace working agents in case of failure or waiting to be scheduled in case of sudden and unexpected workload spikes. However, this requires additional physical servers to be activated. This increases the total cost due to power consumption and computing resource utilization. Therefore, we propose to use backup agents for those services that requires ultra reliability. In other words, we only use this technique for Type 1 services. We reserve some number of agents considering the required SFC for Type 1 services.

VI. Performance Evaluation

This section provides a proof-of-concept for the performance evaluation of the proposed multi-agent autonomic network management system, using MATLAB simulation. We rely on the mathematical formulation presented in Section V.

A. System Description and Simulation Environment Specification

The simulation environment is aimed at emulating a MANA-NMS hosted in a cloud-based edge data center that autonomically manages a backhaul network and 5G service processing. A depiction of the simulation environment is presented in Figure 3. As it can be seen from the figure, there are three types of 5G services arriving at the edge data center. These services are ultra-reliable low-latency services (uMTC), machine-type communication (mMTC), and enhanced mobile broadband service (xMBB). The figure also shows an edge data center network that interconnects servers, which contain memory, CPU, and storage. In the edge data center, multiple types of agents could be instantiated to provide different functions. These functions could be service-traffic workload prediction, network-state database management, and synchronization, service authentication and admission, resource slicing and service scheduling, path computation and routing, service QoS monitoring, and security/firewall.

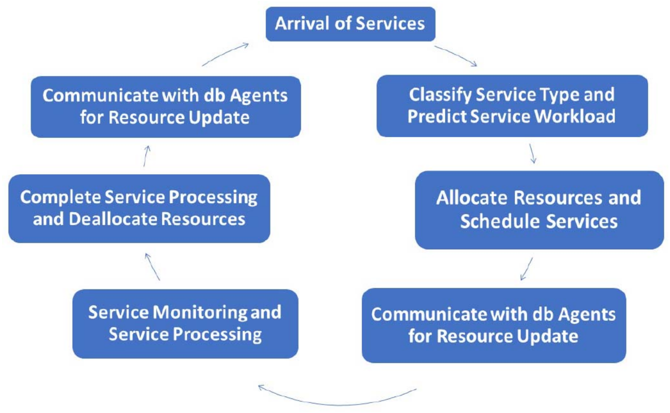

Arrival of Services
Classify Service Type and Predict Service Workload
Allocate Resources and Schedule Services
Communicate with db Agents for Resource Update
Service Monitoring and Service Processing
Complete Service Processing and Deallocate Resources
Communicate with db Agents for Resource Update

Fig. 9. Cyclic step of service scheduling, processing and monitoring.

The evaluation is performed for a simple case of multiple types of agents, performing service processing and network management. For the evaluation purpose, we limit our consideration to only the following type of agents. All of them are network-level agents A_{ntk}.

- Service workload arrival prediction agents
- Event handler and service scheduling agents
- Service-processing agents, performing service processing such as authentication and admission, routing, QoS monitoring, and path computation for the arriving services
- Network state database updates, synchronization and management agents
- Service function chaining and orchestration agents

We categorized agents into two main types with the specification shown in Table II. These two types of agents are system background service agents and user-traffic processing agents. Background service agents provide necessary preliminary functions such as service traffic workload prediction, agents instantiation, and resource allocation, network monitoring, and service-agents orchestration. Such types of agents are assumed to have no direct impact on the service processing (e.g., delay). User/traffic/service workload processing agents perform functions that are directly related to handling the incoming services. These could be path computing, authenticating, admitting, and schedule while monitoring a given service's QoS requirements for processing in the edge data center. Figure 9 shows a complete cyclic step of the overall multi-agent-based autonomic service management and processing system. The figure shows the steps needed to accomplish the scheduling, processing, and monitoring tasks for a given service. Note that the simulation system is built based on the proposed principles that the atomic agents are used as building blocks for the automated network management system.

TABLE II
INTERNAL-FUNCTIONAL SPECIFICATION OF AGENTS

Agent specification	Service Processing Agents	Background Processing Agents
Agent ID	$Ag_s rvm_x$	$Ag_B P_x$
Input	Arrived services	Change in resource/network states
Facts	Service specifications and processing demands	Resources specification and service agents states etc
Cognition	Service identification, and classification	Service workload prediction and monitoring status change etc
Plan	Mechanism of authentication, admission, path computing, routing and monitoring etc	Adaptive resource (de)allocation, seating up of SFC etc
Verification and validation	Check the computed path for consistence, optimality and validity in time and state	Verify resource state before update
Output/Action	authenticating, admitting, and routing of service in a given path or the service chain	(de)allocate resources and resource state database

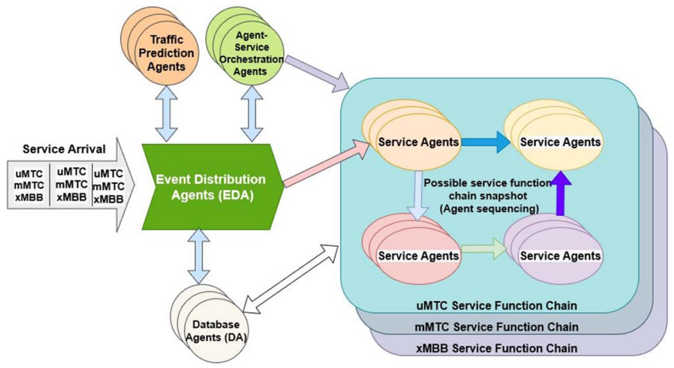
TABLE III
DETAILED AGENTS PARAMETRIC-SPECIFICATIONS USED IN DEVELOPING THE OVERALL SIMULATION SYSTEM

Agent type	Number of Agents Required	Capacity(Gbps)	Processing Latency (ms)
uMTC Service Agents	depends on the incoming service workload amount	100	Service-workload/Agent-Capacity
mMTC Service Agents	depends on the incoming service workload amount	100	Service-Workload/Agent-Capacity
xMBB Service Agents	depends on the incoming service workload amount	100	Service-workload/Agent-Capacity
Traffic Prediction Agents	3 (fixed at the start of the simulation)	100	20
Even distribution Agents	3 (fixed at the start of the simulation)	100	0.03
Orchestration Agents	3 (fixed at the start of the simulation)	100	0.03
Database management agents	3 Agents (fixed at the start of the simulation)	100	0.03

As discussed above, one of the agents is an event handler that responds to incoming services or any network-related events. Based on the events, it reacts and dispatches the events (i.e., arrival of tasks, services, packets, network changes, etc) to appropriate agents for event handling and processing. There are also traffic prediction agents, which predict the incoming service workload. Based on the predicted amount of workload, the required number of service processing agents are instantiated, and service function chains (a sequence of service processing agents) are created. In the simulation, this is performed proactively, every hour. Based on the predicted workload, additional agents are instantiated if an increase in workload is predicted in the next hour compared to the current period. Or some agents are destroyed to reduce and free edge data center resources if a decrease in incoming service workload is predicted.

Depending on the service requirements, the service agents are sequenced as a service function chain for a given service to pass through as a part of service processing. We considered four service agents as a control plane and data plane functions [88]. Since the data plane service function chain normally would not span more than a few service agents, the evaluation considers chains that are limited in size (four agents). An example of agent sequencing is depicted in Figure 10. The service function chain that we assumed is two serially connected agents working in parallel with another two serially connected agents, see Figure 10 for agents sequencing (SFC).

Based on agents' general specification guidelines presented in Section V, let us provide the specification of agents to be used in the evaluation. Agents have a unique identifier number, service capacity, service state (information or facts), adaptive serving methods (mostly a machine learning model), strategy (how to perform service processing), validation/verification techniques (optional), execution(output), which are provided at the agent instantiation stage. The simulation only considered ID, capacity, type, and status (busy, idle, or fail). In the agent specification Table II, the agent IDs are given as $Ag_s rvm_x$ and $Ag_b p_x$. Where srvm indicates service processing type of agents, bp indicates background processing type of agents and X, which could take 1, 2, 3, … N, indicates number used to identify a specific agent instance from a given category. Table III presents all type of agents used in developing the overall simulation system, with detailed parameters. The table also shows different type of agents are instantiated for each type of services. These agents are use in creating SFC. Using the type of agents indicated in Table III, a simulation of an autonomic network management system is developed.


Traffic Prediction Agents
Agent-Service Orchestration Agents
Service Arrival
uMTC mMTC xMBB
uMTC mMTC xMBB
uMTC mMTC xMBB
Event Distribution Agents (EDA)
Service Agents
Service Agents
Possible service function chain snapshot (Agent sequencing)
Service Agents
Service Agents
Database Agents (DA)
uMTC Service Function Chain
mMTC Service Function Chain
xMBB Service Function Chain
Fig. 10. Agent sequencing and scheduling for arrival service processing.

The network topology considered in the simulation is represented in Figure 11. The figure shows a California Research and Education Network (CalREN), to be used as a backhaul or metro network in the evaluation which aggregate traffics to

TABLE IV
SERVICE SPECIFICATION

Services type	Service workload proportion	Expected delay in the edge data center
uMTC (Type 1)	10%	0-2ms
mMTC (Type 3)	20%	2-4ms
xMBB (Type 2)	70%	4-10ms

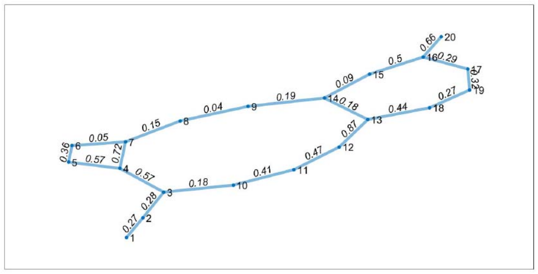


Fig. 11. Considered network topology for path computation.

TABLE V
SIMULATION PARAMETERS

System Specification	System value
Queue	one service queue for each service type
CPU	10,000,000Tbps
Simulation time	120 hours (5 days of a week)
Service arrival process	Random poisson arrival
Service arrival	3 database agents (db agents)

an edge data center. The nodes in the topology are forwarding SDN switches. Aggregate traffic containing multiple types of services such as uMTC, mMTC, and xMBB. The weights on the links and nodes are snapshot values indicating the cost of each element.

In the evaluation, the fraction of service composition is assumed to be 10:20:70 for uMTC, mMTC, and xMBB respectively. This is because there is no representative data to infer the 5G traffic composition. Table IV shows the service ratio and measurement values used in the evaluation. The overall simulation parametric values are presented in Table V.

It should be noted that there are important assumptions made in the evaluation. We assumed equal computational overheads for all service processing types of agents, which may not always be true. It is because agents could perform different computations for each service type. The internal components are designed for the intended function specialization of that agents. Agents could have event monitoring, cognition, planning, verification, and action performing components to make autonomic decisions and dynamic responses. These components require CPU processing and processing delay, which can be added to the virtualization overheads (assuming agents are normally instantiated in a given container or virtualized environment).

Other agents' computational overhead is considered variable, depending on the type of agents. For instance, traffic prediction agents are expected to have different internal components than database management agents service orchestration agents. Therefore, in the evaluation, a given CPU computing overhead is assumed to be added as a computational processing workload. The detailed study of agent computational overhead is left as future work.

In general, the evaluation is based on service workload prediction, path computation, resource allocation, service workload scheduling, service processing, and network and resource state updates. Service arrivals are considered as input to the system and handling agents. The even handling agents may request the orchestration agent to instantiate and create a chain of agents for the services to be processed. The chain of agents represents a service function chain required by the services. This in fact is set proactively every hour by predicting the incoming service type and amount of service workload.

Given service progress through the sequence of service processing agents getting the necessary service treatments; it could be service authentication and authorization to use a given set of resources, SFC path, and service management. A snapshot of agents sequencing from a different type of service agents is provided in Figure 10. In a given type of agents, service agents are scheduled in a round-robin fashion. That means the first arrived service is scheduled to be served by the first agent from a given set of agents of that type, the second arriving services by the second agent if the first agent is still busy, and so on. The evaluation is considered to show agents' interaction to perform a given service, service schedule, and processing.

Arriving services are classified into three categories. And each service is scheduled and stored into three separate queues depending on its service categories. These are uMTC, mMTC, and xMBB queues. The scheduler allocates separate resources and SFCs depending on each queue state. This way services are scheduled maintaining service separation, priority, and service differentiation. The evaluation is considered to show agents' interaction to perform a given service, service schedule, and processing.

Generally, based on the above evaluation specifications and developed simulation environment (emulation of MANA-NMS), the simulation evaluation is performed for: service admission and processing, agent utilization and service processing, resource consumption, processing and queuing delay, agent and server (agent host) failure analysis, and building a resilience system and its impact on resource consumption and processing delay. We performed the simulation for 120hrs, representing a five days traffic pattern. The results are discussed in the following subsections.

B. Service Arrival and Processing Evaluation

Arriving services are allocated a network resource after the best available path is computed. The service schedule is based

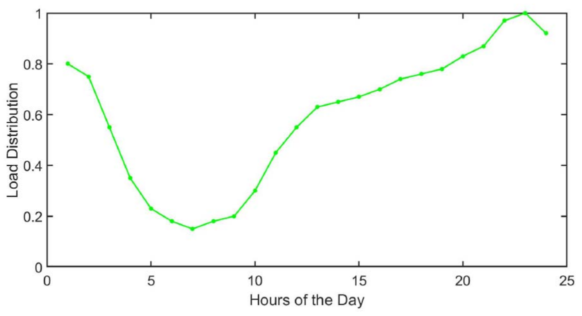


Fig. 12. Workload distribution.

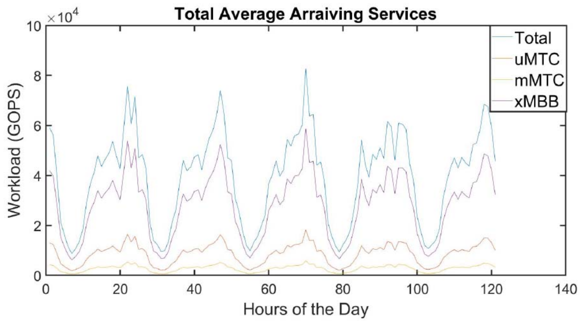


Fig. 13. Average arriving service workload.

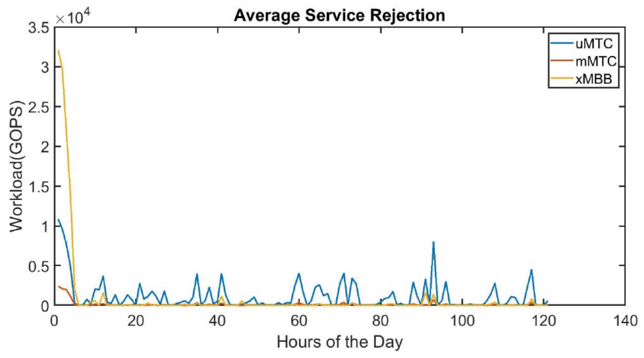


Fig. 14. Service rejection.

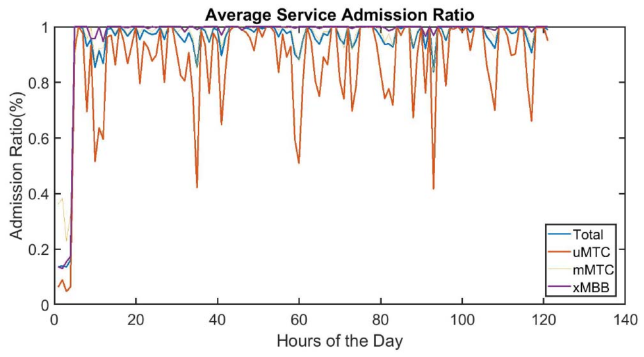


Fig. 15. Service admission probability.

on service priority. It is in the order of URLLC (high priority) –> mMTC –> xMBB (least priority) in a given service scheduling period. Based on the amount of arriving service workload, resources are allocated to agents before they are instantiated. Based on the total serving capability of agents, a service is injected using the workload percentage shown in Figure 12. The evaluation also considers three types of services with a different workload demand to the autonomic network management functions. The arrival is assumed to be Poisson distributed with a workload distribution, shown in Figure 12. The arriving service patter is plotted in Figure showing the total average service workload for each type of service and the overall services workload arriving to the edge data center. Based on the availability of service agents, services are scheduled until all active agents are fully loaded. Figure 13 shows average arriving service workload over a 120hrs evaluation period. Figure 14 shows rejection of services due to agents overload. The overall service admission probability of the multi-agent system is calculated as the ratio of service admitted to service requested. The result is presented in Figure 15. As it can be seen from the Figure, service admission is approximately one during non-peak hours and whereas there is some probability of services rejected at peak hours.

C. Agent Utilization and Edge Data Center Resource Consumption

Agents require CPU resources to process services. The instantiating of multiple agents consumes data center resources such as memory, storage, CPU, and network bandwidth. In our simulation, we considered only CPU, assuming equivalent requirements would be scaled for memory, storage, and bandwidth. As indicated in Table III, there is a fixed number of background processing agents. They are instantiated at the beginning of the simulation. These agents are even handling agents, service workload prediction agents, orchestration agents, and network and resource state database management agents. There are also service agents that are scaled according to the incoming service workload demands. In our simulation, we considered only the number of agents that are scaling with the workload. However, it is also possible that a given agent could scale its processing capability according to the service demands. This could be possible by dynamically adding the necessary resource to the agent. Such kind of agent's capacity scaling is called "horizontal scaling." However, the discussion of such a scenario is out of the scope of this work.

The number of active agents instantiated at a given time is directly related to the service prediction. Service workload is predicted by the service workload predicting agents. On the other hand, the CPU consumption in the edge data center is directly related to the number of active agents instantiated at a given time. Since active agents consume resources, it is necessary to reduce the number of active (but idle) agents to have efficient resource utilization. Figure 16 depicts the average number of required agents in a given hour. This is calculated based on the predicted arriving service workload.

Agents are assigned to provide service scheduling and monitoring functions. Once agents are assigned to accomplish the given task, they will be locked into a busy (internal execution) state before becoming the idle state. The average number of busy agents and idle agent during the 120hrs simulation period are depicted in Figure 18 and Figure 17. The Figures

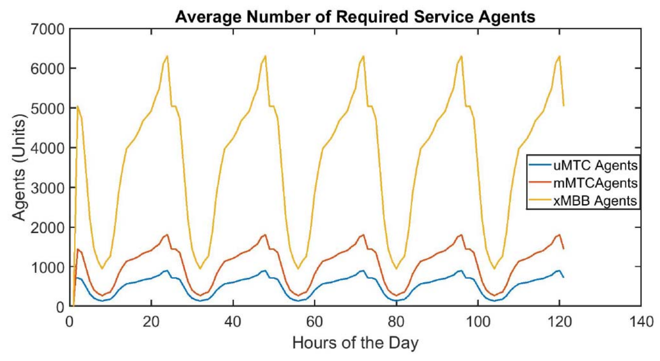


Fig. 16. Average number of required service agents.

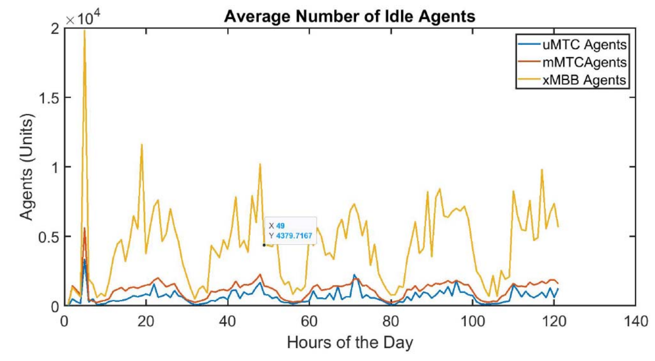


Fig. 17. Average number of idle agents.

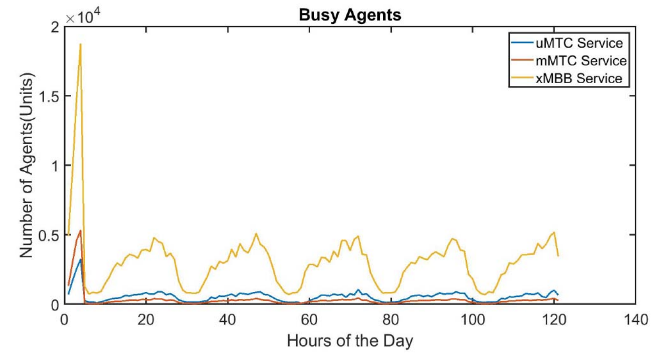


Fig. 18. Average number of busy agents.

indicate the average number of agents that are actively performing service processing and waiting for service scheduling, respectively. The idle number of agents is a total average over the whole period which is approximately 4000 agents. This number indicates idle agents that are actively consuming resources without providing services. This shows the inefficiency of resource consumption which is the consequence of the prediction model accuracy. However, if we reduce these idle agents, we may increase the service rejection and service latency in the edge data center. This is because the service arrival is a random process and an exact prediction is dependent on the prediction model used in the prediction agents. Moreover, given the model used in the prediction agent, the prediction is the optimal value to avoid further service rejection and service delay in the data center. The overall service-processing-agents utilization by each type of services is depicted in Figure 19.

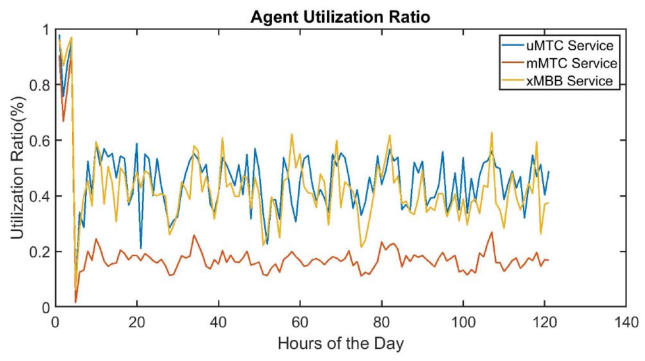


Fig. 19. Agents utilization.

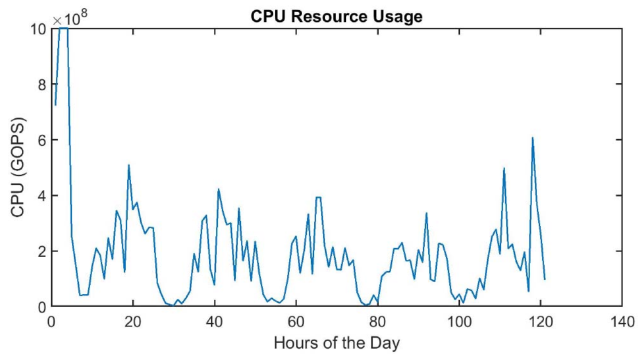


Fig. 20. Edge data center resource utilization.

Figure 20 shows the average amount of resource consumption in the edge data center in terms of the consumed CPU. The maximum available CPU resource in GOPS is given in Table V. As can be seen from the Figure, the resource consumption is directly related to the number of total active agents which is also directly related to the service workload arrival pattern. As indicated the arrival pattern is predicted by the service workload prediction agents.

Moreover, we have evaluated agents' load ratio over 120hrs period for each service type agent. This is an average busy period for a given type of agent. The evaluation shows, on average, how much a given agent is utilized over the 120hrs period. This parameter indicates the efficiency of agents in processing and delivering the given service while the agents are in an active state.

One capability of an autonomic multi-agents network management system is the ability to communicate between agents to accomplish a given task. In our evaluation scenario, we considered agent communication between scheduling and monitoring agents and resource database management agents. Accordingly, the evaluation indicated the communication frequency needed to accomplish the given arriving service workload. Figure 22 shows the average communication frequency over a 120hr period.

D. Service Queuing and Processing Latency Evaluation

The simulation has two main stages of queuing for each type of service. These are service a arrival queue and a service processing queue. As the services arrive at the edge data center, they are first stored in the service arrival queue of the

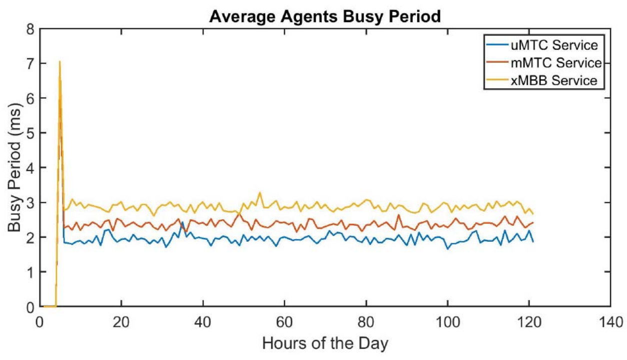


Fig. 21. Agent busy period.

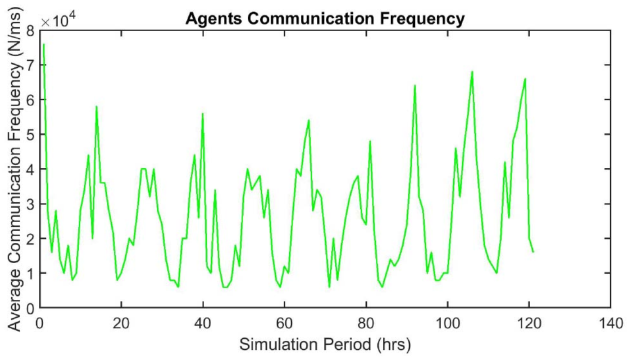


Fig. 22. Agents average communication frequency for database update due to service arrival.

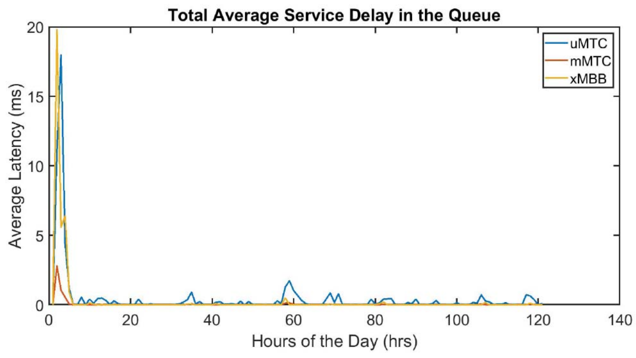


Fig. 23. Arrival service queuing delay.

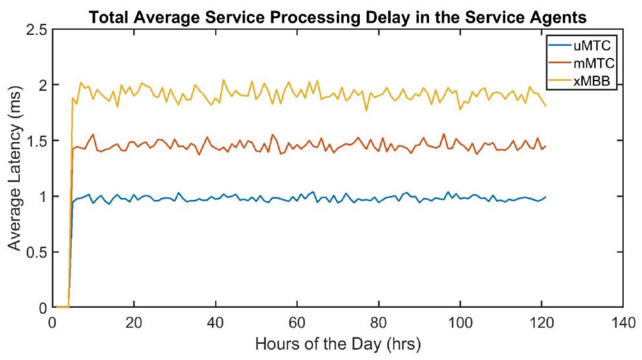


Fig. 24. Service processing delay in the service processing agents.

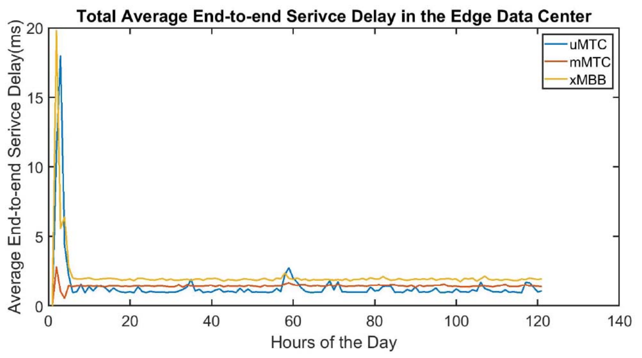


Fig. 25. End-to-end service latency in edge data center.

corresponding service queue before they are scheduled for processing in the service agents. Service is processed sequentially through the service function chain. Since we considered a maximum of two agents connected serially, there is a service processing queue between the two serial connected service agents. In this scenario, the service is scheduled and processed in the service function chain which is connecting service agents. The simulation result depicted in Figure 23 presents the average delay in the queues for each type of service. Figure 24 depicts the average processing delay in service agents passing through the service function chain. The total delay in the edge data center is assumed to be the sum of the two delays as formulated in the above section. Figure 25 shows the total end-to-end latency in the edge data center which ignores the communication link delay between subsequent service agent delay in the SFC.

E. System Evaluation for Fail-Over Scenarios

In the above evaluation scenarios, agents are assumed to be 100% reliable and available. However, this may not be true as agents are software components with a non-zero probability of failure. The agents are an event handler, orchestration, prediction agent, and database management. These agents are very critical for system resilience. Therefore, such agents are redundant with a well-synchronized backup system. Such types of agents are fixed in our simulation, and a few are compared to service processing types of agents. And assumed to be 100% reliable. However, service processing agents are assumed to be less reliable and their number is scaled according to the arrival service workload. In the simulated system, such types of agents are not protected by a backup system. However, overpraising of 1% is considered to accommodate the imperfection of the service workload prediction, which is dependent on the machine learning model incorporated in the service workload prediction agents.

Different failure scenarios could be analyzed using the developed simulation environment. It could be a backhaul network node failure that affects the link utilization and service admission or server/agent failure which affects the edge data center service admission probability and service latency. A network node failure is handled by recomputing new available paths until the node is back into operation. This is a classical scenario thoroughly considered in the literature. We consider limited scenarios for the multi-agent-based autonomic network management system failure. In simulation two fail-over scenarios are considered; a single or a very few numbers of agents

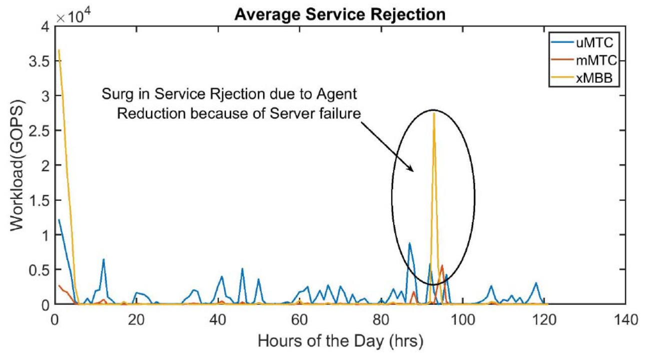


Fig. 26. Impact of server (hosting service processing agents) failure on service rejection.

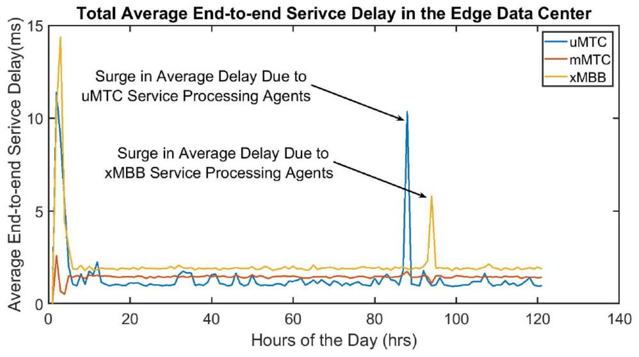


Fig. 27. Impact of server failure on end-to-end latency in the edge data center.

failure and single server failure. A single agent failure has an insignificant impact on the overall service admission and service delay in the edge data center. This is because agents are lightweight service units that can be instantiated easily and in a few more for overprovisioning, with little increase in resource consumption. They can also be easily and quickly instantiated to compensate for the reduction in the number of service agents due to service agent failure.

However, the failure of a server, hosting several agents, could significantly affect the edge data center processing capacity. The extent of the impacts depends on how the SFC is created. That means if a single type of agent is hosted in a single server, and if that server fails, the impact will be the highest. It is because a given service may need to get service treatment from all types of service processing agents in the SFC. The failure of such server would significantly reduce the number of agents of a given type, creating a shortage of such agents for SFC creation. The least impact could be achieved by rightly balancing and distributing the agent type to different servers. With such considerations, the simulation is performed for maximum impact of hosting a single type of agent in a single server. Moreover, a single server failure at a random time within the 120hrs simulation period is introduced. And the impact on service admission probability and service processing latency is evaluated.

As it can be seen from Figure 26, the impact of a server failure is significant in that it increased the service rejection. However, the impact does not last long because the agent monitoring and orchestrating other agents immediately detects the reduction in the number of service processing, agents and next instantiate additional agents of such type. The instantiated agents are updated with the service state and information to take over the service processing. After instantiating the required service processing agents, as a replacement to the failed agents, the orchestrating agents also create the necessary SFC. Figure 27 depicts the overall service latency in the edge data center. As can be seen from the Figure, there is a significant latency on each type of service processing, depending on the time of failure of the server hosting a given type of service processing agents. Figure 28 shows the impact of server failure on the service admission probability. As it can be observer from the Figure, there are significant reduction in the admission probability for all type of services,depending on the time of failure of server hosting a given type of service processing agents.

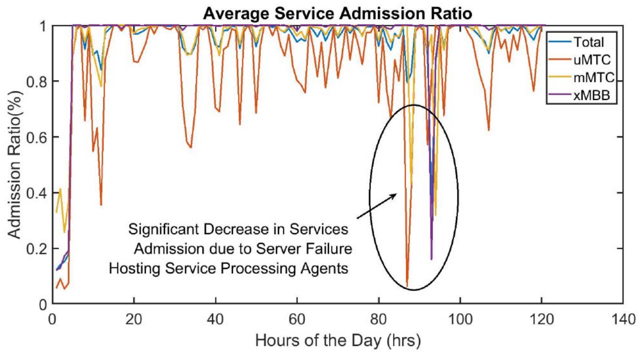


Fig. 28. Impact of server failure on service admission probability.

F. System Reliability Evaluation and Service Differentiation for Resource Efficiency

Reliability of network management system is very important. In Section V, we discussed and presented a reliability improvement technique based on a backup system. We evaluate the system performance by introducing backup service processing agents. Figure 30 shows the admission probability using backup system. Compared to the result obtained without backup, which is depicted in Figure 15, the admission probability has increased. Moreover, Figure 29 shows the reduction of service rejection compared to the system without back service processing agents, see Figure 29 and Figure 14.

The spike (seen in the figure) in-service rejection is due to the introduced server failure impacting the available service processing agents. This shows that even if we have a backup system, the impact is significant for a small duration. This is because, after the active serving agents' unexpected sudden failure, it needs time until new service agents are instantiated and taking over the service processing abandoned by the failed agents. Figure 30 shows the admission probability improvement by using backup system.

Figure 31 shows the increase in resource consumption compared to Figure 20, which is plotted for the system without backup service processing agents. Since there is a large number of service processing agents, having a backup for all of

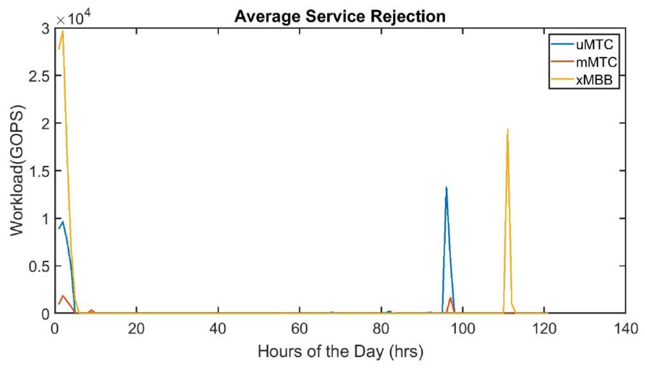


Fig. 29. Service rejection with backup service processing agents.

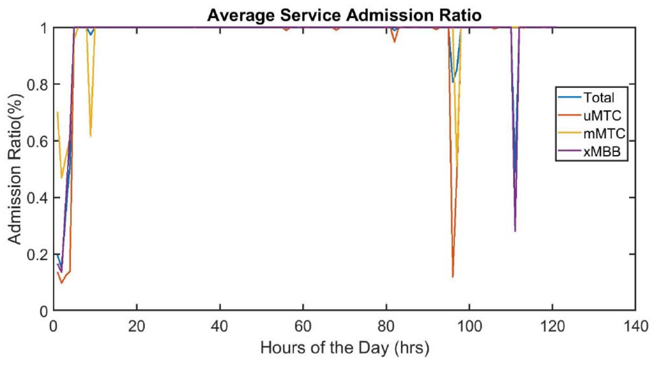


Fig. 30. Service admission probability.

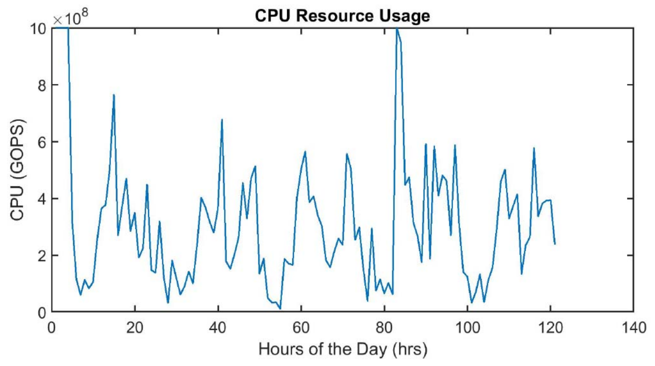


Fig. 31. CPU resource consumption with backup system.

them approximately doubles the resource consumption. This indicated that a backup system has a huge cost of resources and energy consumption. To reduce such consumption, we propose service differentiation in providing a backup system by providing backup agents for only critical services or by providing flexibility to provide such options to use by the network operator to decide. This makes the system more efficient in resource utilization and resilient compared to a system without any backup mechanism. A more detailed analysis is left for future work. Figure 32 shows the reduction in CPU consumption compared to a backup system without service differentiation. However, more critical services are admitted with a few increase in resource consumption compared to the system without service backup.

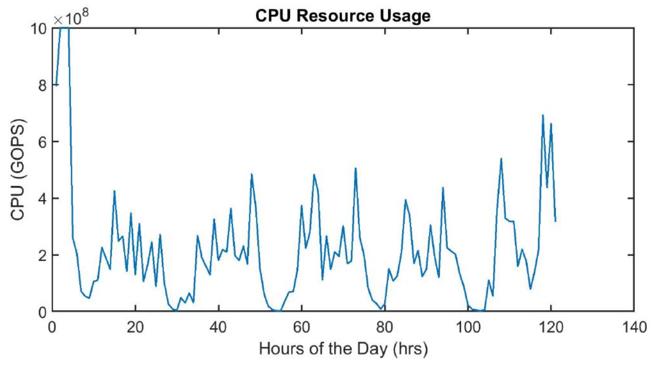


Fig. 32. CPU consumption with backup system for uMTC service processing agents.

VII. Conclusion and Future Works

Network softwarization enables myriad innovative possibilities. The most important innovations of network softwarization are design flexibility and programmability. SDN and NFV architecture miss network automation. The advancement and explosion of machine learning would further enable the realization of full network automation. Autonomic networking is becoming the necessity for future networks, that are aiming at interconnecting billions of heterogeneous device, with stringent services' requirements. In this article, we presented a review of autonomic networking since 2004, categorizing literature into before and after the introduction of SDN and NFV. We then discussed the challenges of network automation. Multi-agent based network automation architecture is presented along with the Markovian mathematical model of the system. Finally, we presented the performance of the system for a simplified case. The performance analysis showed the possibility of utilizing a multi-agent system for network management system automation. What we can guess from the preliminary analysis is that the use of multi-agent in automating network management provides a scalable, flexible, dynamic, and resource-efficient system. However, due to the distributed nature of the system, the effect of virtualization and communication overhead has to be evaluated as future work. Testbed implementation of MANA-NMS and the exact implementation of agents for specific network functions are also left as future works.

Moreover, we also aimed at performing a detailed multi-agent study focusing on designing agents and their internal components.

References

[1] *Consensus on 6G Is Gradually Forming*. Accessed: Mar. 24, 2020. [Online]. Available: https://telecoms.com/503142/consensus-on-6g-is-gradually-forming/

[2] *Software-Defined Networking: The New Norm for Networks*, Open Netw. Found., Menlo Park, CA, USA, 2012.

[3] *NFV Technology*. Accessed: Oct. 30, 2019. [Online]. Available: https://www.etsi.org/technologies/nfv

[4] Á. L. V. Caraguay, P. J. Ludeña-González, R. V. T. Tandazo, and L. I. B. López, "SDN/NFV architecture for IoT networks," in *Proc. 14th Int. Conf. Web Inf. Syst. Technol. Vol. 1 (ITSCO)*, 2018, pp. 425–429.

[5] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Serv. Appl.*, vol. 9, no. 1, p. 16, Jun. 2018. [Online]. Available: https://doi.org/10.1186/s13174-018-0087-2

[6] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony, "A hybrid centralised-distributed network management architecture," in *Proc. IEEE Int. Symp. Comput. Commun.*, Red Sea, Egypt, 1999, pp. 434–441.

[7] W. W. Tseng, P. K. Muller, K. M. Chow, M. W. Meyer, and G. D. Bruno, "Distributed service subsystem architecture for distributed network management," U.S. Patent 6 308 207, 2001.

[8] G. A. Oparin, V. G. Bogdanova, A. A. Pashinin, and S. A. Gorsky, "Distributed solvers of applied problems based on microservices and agent networks," in *Proc. 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, May 2018, pp. 1415–1420.

[9] R. W. Collier, E. O'Neill, D. Lillis, and G. O'Hare, "MAMS: Multi-agent microservices*," in *Proc. Companion World Wide Web Conf.*, 2019, pp. 655–662. [Online]. Available: https://doi.org/10.1145/3308560.3316509

[10] T. B. Meriem, R. Chaparadza, B. Radier, S. Soulhi, J.-A. Lozano-López, and A. Prakash, "GANA—Generic autonomic networking architecture," ETSI, Sophia Antipolis, France, White Paper, 2016.

[11] F. Granelli and R. Bassoli, "Autonomic mobile virtual network operators for future generation networks," *IEEE Netw.*, vol. 32, no. 5, pp. 76–84, Sep./Oct. 2018.

[12] F. Granelli and R. Bassoli, "Towards autonomic mobile network operators," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, Tokyo, Japan, 2018, pp. 1–4.

[13] G. Baggio, R. Bassoli, and F. Granelli, "Cognitive software-defined networking using fuzzy cognitive maps," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 517–539, Sep. 2019.

[14] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[15] J. Strassner, "Autonomic networking—Theory and practice," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp.*, vol. 1. Seoul, South Korea, 2004, p. 927.

[16] J. Strassner and J. O. Kephart, "Autonomic systems and networks: Theory and practice," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Vancouver, BC, Canada, 2006, p. 588.

[17] J. Strassner, "Autonomic networking," in *Proc. 5th IEEE Workshop Eng. Autonom. Auton. Syst. (EASE)*, 2008, p. 3.

[18] A. Tizghadam and A. Leon-Garcia, "AORTA: Autonomic network control and management system," in *Proc. IEEE INFOCOM Workshops*, Phoenix, AZ, USA, 2008, pp. 1–4.

[19] M. Behringer, A. Dutta, Y. Hertoghs, and S. Bjarnason, "Autonomic networking—From theory to practice," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, 2014, pp. 1–17.

[20] H. Gogineni, A. Greenberg, D. A. Maltz, T. S. E. Ng, H. Yan, and H. Zhang, "MMS: An autonomic network-layer foundation for network management," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 1, pp. 15–27, Jan. 2010.

[21] M. May, M. Siekkinen, V. Goebel, T. Plagemann, R. Chaparadza, and L. Peluso, "Monitoring as first class citizen in an autonomic network universe," in *Proc. 2nd Bio-Inspired Models Netw. Inf. Comput. Syst.*, Budapest, Hungary, Dec. 2007, pp. 247–254.

[22] G. Bouabene, C. Jelger, C. Tschudin, S. Schmid, A. Keller, and M. May, "The autonomic network architecture (ANA)," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 1, pp. 4–14, Jan. 2010.

[23] W. Berrayana, H. Youssef, and G. Pujolle, "A generic cross-layer architecture for autonomic network management with network wide knowledge," in *Proc. 8th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Limassol, Cyprus, 2012, pp. 82–87.

[24] S. Wang, H. Zhao, B. Zhang, H. Liu, and L. Xie, "An autonomic communication based conceptual and architecture model for cognitive radio nodes," in *Proc. IET 3rd Int. Conf. Wireless Mobile Multimedia Netw. (ICWMNN)*, Beijing, China, 2010, pp. 200–204.

[25] J. Praveen, B. Praveen, and C. S. R. Murthy, "A first step towards autonomic optical burst switched networks," in *Proc. 2nd Int. Conf. Auton. Comput. (ICAC)*, Seattle, WA, USA, 2005, pp. 306–307.

[26] F. Baroncelli, B. Martini, and P. Castoldi, "Autonomic service provisioning for automatically switched transport network," in *Proc. Joint Int. Conf. Opti. Internet Next Gener. Netw. (COIN-NGNCON)*, Jeju, South Korea, 2006, pp. 15–17.

[27] A. A. F. Loureiro and L. B. Ruiz, "Autonomic wireless networks in smart environments," in *Proc. 5th Annu. Conf. Commun. Netw. Serv. Res. (CNSR)*, Fredericton, NB, Canada, 2007, pp. 5–7.

[28] A. Moursy, A. Aly, B. Xu, D. Perkins, and M. Bayoumi, "Testbed implementation for autonomic network performance management of wireless mesh networks," in *Proc. IEEE Globecom Workshops*, Anaheim, CA, USA, 2012, pp. 903–907.

[29] T. Braga, F. Silva, J. M. Nogueira, A. Loureiro, and L. B. Ruiz, "A tiny and light-weight autonomic element for wireless sensor networks," in *Proc. 4th Int. Conf. Auton. Comput. (ICAC)*, Jacksonville, FL, USA, 2007, p. 17.

[30] I. Al-Oqily and A. Karmouch, "A self-organizing composition towards autonomic overlay networks," in *Proc. IEEE Netw. Oper. Manag. Symp. (NOMS)*, 2008, pp. 287–294.

[31] C. Ma, X. Lin, H. Lv, and H. Wang, "ABSR: An agent based self-recovery model for wireless sensor network," in *Proc. 8th IEEE Int. Conf. Depend. Auton. Secure Comput.*, Chengdu, China, 2009, pp. 400–404.

[32] M. ElGammal and M. Eltoweissy, "Towards aware, adaptive and autonomic sensor-actuator networks," in *Proc. IEEE 5th Int. Conf. Self-Adapt. Self-Org. Syst.*, Ann Arbor, MI, USA, 2011, pp. 210–211.

[33] S. Schuetz, K. Zimmermann, G. Nunzi, S. Schmid, and M. Brunner, "Autonomic and decentralized management of wireless access networks," *IEEE Trans. Netw. Service Manag.*, vol. 4, no. 2, pp. 96–106, Sep. 2007.

[34] J. A. McCann and R. Sterritt, "Autonomic pervasive networks (APNs)—Extended abstract," in *Proc. 7th IEEE Int. Conf. Workshops Eng. Auton. Auton. Syst.*, 2010, pp. 145–148.

[35] B. Mathieu *et al.*, "Autonomic management of context-aware ambient overlay networks," in *Proc. 2nd Int. Conf. Commun. Netw. China*, Shanghai, China, 2007, pp. 727–733.

[36] Y. Qiang, Y. Li, and J. Chen, "The workload adaptation in autonomic DBMSs based on layered queuing network model," in *Proc. 2nd Int. Workshop Knowl. Discov. Data Min.*, Moscow, Russia, 2009, pp. 781–785.

[37] Y. Liu, J. Zhang, M. Jiang, D. Raymer, and J. Strassner, "A model-based approach to adding autonomic capabilities to network fault management system," in *Proc. IEEE Netw. Oper. Manag. Symp.*, Salvador, Brazil, 2008, pp. 859–862.

[38] N. Tcholtchev and R. Chaparadza, "Autonomic fault-management and resilience from the perspective of the network operation personnel," in *Proc. IEEE Globecom Workshops*, Miami, FL, USA, 2010, pp. 469–474.

[39] J. C. Nobre and L. Z. Granville, "Towards consistency of policy states in decentralized autonomic network management," in *Proc. IEEE Int. Symp. Pol. Distrib. Syst. Netw.*, London, U.K., 2009, pp. 170–173.

[40] H. Zhang, M. Zhao, W. Wang, X. Gong, and X. Que, "A novel autonomic architecture for QoS management in wired network," in *Proc. IEEE Globecom Workshops*, 2010, pp. 529–533.

[41] B. Mokhtar and M. Eltoweissy, "Memory-enabled autonomic resilient networking," in *Proc. 7th Int. Conf. Collab. Comput. Netw. Appl. Worksharing (CollaborateCom)*, Orlando, FL, USA, 2011, pp. 132–141.

[42] L. Noirie *et al.*, "Semantic networking: Flow-based, traffic-aware, and self-managed networking," *Bell Labs Techn. J.*, vol. 14, no. 2, pp. 23–38, 2009.

[43] M. B. Lehocine and M. Batouche, "Self-management in IP networks using autonomic computing," in *Proc. Int. Conf. Workshop Netw. Future (NOF)*, Paris, France, 2014, pp. 1–3.

[44] R. Sterritt, "Autonomous and autonomic systems: Paradigm for engineering effective software-based systems?" in *Proc. 33rd Annu. IEEE Softw. Eng. Workshop*, Skovde, Sweden, 2009, p. 57.

[45] A. Galis, S. Clayman, and L. Mamatas, "Towards autonomic management of software enabled networks," in *Proc. 8th Int. Symp. Adv. Topics Elect. Eng. (ATEE)*, Bucharest, Romania, 2013, pp. 1–3.

[46] P. Zhang and J. Wang, "SIP based scheme for collaborative autonomic network management," in *Proc. 5th IEEE Int. Conf. Broadband Netw. Multimedia Technol.*, Guilin, China, 2013, pp. 323–326.

[47] L. Wang and E. Gelenbe, "Demonstrating voice over an autonomic network," in *Proc. IEEE Int. Conf. Auton. Comput.*, Grenoble, France, 2015, pp. 139–140.

[48] X. Liu, P. Juluri, and D. Medhi, "An experimental study on dynamic network reconfiguration in a virtualized network environment using autonomic management," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, 2013, pp. 616–622.

[49] X. Long *et al.*, "Autonomic networking: Architecture design and standardization," *IEEE Internet Comput.*, vol. 21, no. 5, pp. 48–53, 2017.

[50] G. Xilouris *et al.*, "Towards autonomic policy-based network service deployment with SLA and monitoring," in *Proc. IEEE Conf. Netw. Func. Virtualization Softw. Defined Netw. (NFV-SDN)*, Verona, Italy, 2018, pp. 1–2.

[51] A. Richard *et al.*, "Autonomous networks: Empowering digital transformation for the telecoms industry," TM Forum, Parsippany, NJ, USA, White Paper, 2019.

[52] K. Tsagkaris, M. Logothetis, V. Foteinos, G. Poulios, M. Michaloliakos, and P. Demestichas, "Customizable autonomic network management: Integrating autonomic network management and software-defined networking," *IEEE Veh. Technol. Mag.*, vol. 10, no. 1, pp. 61–68, Mar. 2015.

[53] Q. Qi, W. Wang, X. Gong, and X. Que, "A SDN-based network virtualization architecture with autonomie management," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Austin, TX, USA, 2014, pp. 178–182.

[54] A. Mercian, P. Sharma, R. Aguiar, C. Chen, and D. Pinheiro, "UDAAN: Embedding user-defined analytics applications in network devices," in *Proc. Workshop Netw. Meets AI ML*, 2019, pp. 70–75. [Online]. Available: https://doi.org/10.1145/3341216.3342216

[55] F. Granelli, D. Kliazovich, and N. Malheiros, "Towards cognitive Internet: An evolutionary vision," in *Cognitive Radio and Networking for Heterogeneous Wireless Networks: Recent Advances and Visions for the Future*. Cham, Switzerland: Springer, 2015, pp. 181–200.

[56] M. Wódczak *et al.*, "Standardizing a reference model and autonomic network architectures for the self-managing future Internet," *IEEE Netw.*, vol. 25, no. 6, pp. 50–56, Nov. 2011.

[57] *Autonomic Network Engineering for the Self-Managing Future Internet (AFI): Scenarios, Use Cases and Requirements for Autonomic/Self-Managing Future Internet*, document ETSI GS AFI 001, ETSI, Sophia Antipolis, France, 2011.

[58] "Network technologies (NTECH): Autonomic network engineering for the self-managing future Internet (AFI): autonomicity and self-management in the broadband forum (BBF) architectures," ETSI, Sophia Antipolis, France, ETSI Rep. TR 103 473, 2018.

[59] R. Chaparadza *et al.*, "SDN enablers in the ETSI AFI GANA reference model for autonomic management & control (emerging standard), and virtualization impact," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, USA, 2013, pp. 818–823.

[60] R. Chaparadza *et al.*, "Implementation guide for the ETSI AFI GANA model: A standardized reference model for autonomic networking, cognitive networking and self-management," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, USA, 2013, pp. 935–940.

[61] R. Chaparadza, M. Wodczak, T. B. Meriem, P. De Lutiis, N. Tcholtchev, and L. Ciavaglia, "Standardization of resilience & survivability, and autonomic fault-management, in evolving and future networks: An ongoing initiative recently launched in ETSI," in *Proc. 9th Int. Conf. Design Rel. Commun. Netw. (DRCN)*, Budapest, Hungary, 2013, pp. 331–341.

[62] S. van der Meer, J. Keeney, and L. Fallon, "5G networks must be autonomic!" in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Taipei, Taiwan, 2018, pp. 1–5.

[63] G. Sakarkar and V. M. Thakar, "Autonomous software agent for localization," in *Proc. Int. Conf. Intell. Agent Multi-Agent Syst.*, Chennai, India, 2009, pp. 1–4.

[64] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Auton. Agents Multi-Agent Syst.*, vol. 1, no. 1, pp. 7–38, Jan. 1998. [Online]. Available: https://doi.org/10.1023/A:1010090405266

[65] E. M. Davidson and S. D. J. McArthur, "Exploiting multi-agent system technology within an autonomous regional active network management system," in *Proc. Int. Conf. Intell. Syst. Appl. Power Syst.*, Kaohsiung, Taiwan, Nov. 2007, pp. 1–6.

[66] Q. Jiang, M. Xu, L. Tang, and Q.-B. Chen, "Research on load balancing based on multi-agent in ubiquitous networks," in *Proc. Int. Conf. Intell. Comput. Technol. Autom.*, vol. 3. Changsha, China, 2010, pp. 10–13.

[67] T. Amudha, N. Saritha, and P. Usha, "A multi-agent based framework for dynamic service discovery and access using matchmaking approach," in *Proc. Int. Conf. Intell. Agent Multi-Agent Syst.*, Chennai, India, 2009, pp. 1–4.

[68] S. K. Patri, E. Grigoreva, W. Kellerer, and C. M. Machuca, "Rational agent-based decision algorithm for strategic converged network migration planning," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 11, no. 7, pp. 371–382, Jul. 2019.

[69] A. Kott *et al.*, "Autonomous intelligent cyber-defense agent (AICA) reference architecture. release 2.0," 2018. [Online]. Available: arXiv:1803.10664.

[70] M. Mohammadian, "Network security risk assessment using intelligent agents," in *Proc. Int. Symp. Agent Multi-Agent Syst. Robot. (ISAMSR)*, 2018, pp. 1–6.

[71] S. Ouiazzane, M. Addou, and F. Barramou, "A multi-agent model for network intrusion detection," in *Proc. 1st Int. Conf. Smart Syst. Data Sci. (ICSSD)*, Rabat, Morocco, 2019, pp. 1–5.

[72] X. Zhang, K. Shi, S. Zhang, D. Li, and R. Xia, "Virtual agent clustering based mobility management over the satellite networks," *IEEE Access*, vol. 7, pp. 89544–89555, 2019.

[73] Z. Taoqing and H. Xiaoying, "Self-adaptive network service research based on multi-agents in hierarchy," in *Proc. 1st Int. Conf. Netw. Distrib. Comput.*, Hangzhou, China, 2010, pp. 314–316.

[74] G.-Z. Liu and D. Juan, "Design of distributed network management system based on multi-agent," in *Proc. 3rd Int. Symp. Inf. Process.*, Qingdao, China, 2010, pp. 433–436.

[75] H. Tianfield, "Multi-agent based autonomic architecture for network management," in *Proc. IEEE Int. Conf. Ind. Informat. (INDIN)*, Banff, AB, Canada, 2003, pp. 462–469.

[76] N. A. M. Subha, G.-P. Liu, and N. H. M. Yusof, "External consensus in networked multi-agent systems with random network delay," in *Proc. 57th Annu. Conf. Soc. Instrum. Control Eng. Japan (SICE)*, 2018, pp. 427–432.

[77] M. Sharf and D. Zelazo, "Analysis and synthesis of MIMO multi-agent systems using network optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 11, pp. 4512–4524, Nov. 2019.

[78] M. Sharf and D. Zelazo, "Network feedback passivation of passivity-short multi-agent systems," *IEEE Control Syst. Lett.*, vol. 3, no. 3, pp. 607–612, Jul. 2019.

[79] A. Y. Yazıcıoğlu, M. Egerstedt, and J. S. Shamma, "Formation of robust multi-agent networks through self-organizing random regular graphs," *IEEE Trans. Netw. Sci. Eng.*, vol. 2, no. 4, pp. 139–151, Oct./Dec. 2015.

[80] G. Uma, B. E. Prasad, and O. N. Kumari, "Distributed intelligent systems: Issues, perspectives and approaches," *Knowl. Based Syst.*, vol. 6, no. 2, pp. 77–86, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/095070519390022L

[81] M. R. Genesereth and S. P. Ketchpel, "Software agents," *Commun. ACM*, vol. 37, no. 7, p. 48, Jul. 1994. [Online]. Available: https://doi.org/10.1145/176789.176794

[82] G. K. Soon, C. K. On, P. Anthony, and A. R. Hamdan, "A review on agent communication language," in *Computational Science and Technology*, R. Alfred, Y. Lim, A. A. A. Ibrahim, and P. Anthony, Eds. Singapore: Springer, 2019, pp. 481–491.

[83] K. Lerman, A. Galstyan, and T. Hogg, "Mathematical analysis of multi-agent systems," 2004. [Online]. Available: https://arxiv.org/abs/cs/0404002.

[84] R. Guerzoni, Z. Despotovic, R. Trivisonno, and I. Vaishnavi, "Modeling reliability requirements in coordinated node and link mapping," in *Proc. IEEE 33rd Int. Symp. Rel. Distrib. Syst.*, Nara, Japan, Oct. 2014, pp. 321–330.

[85] S. Ayoubi, Y. Zhang, and C. Assi, "RAS: Reliable auto-scaling of virtual machines in multi-tenant cloud networks," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Niagara Falls, ON, Canada, Oct. 2015, pp. 1–6.

[86] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: Patterns, causes and characteristics," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Atlanta, GA, USA, Jun. 2014, pp. 1–12.

[87] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 558–573, Mar. 2018.

[88] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. P. Fitzek, "Study of virtual network function placement in 5G cloud radio access network," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2242–2259, Dec. 2020.

Sisay Tadesse Arzo (Member, IEEE) received the B.Sc. degree from Hawassa University, Ethiopia, and the M.Sc. degree in telecommunication engineering from the University of Trento, Italy, where he is currently pursuing the Ph.D. degree. He is a former Visiting Research Fellow with the University of Luxembourg. He has more than five years of industrial experience with Telecom, Huawei Technology and Ethio-Telecom, and five years of academic experience as a Lecturer with Addis Ababa University. His research interest included network softwarization, SDN, NFV, IoT, and network automation.

Riccardo Bassoli (Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunications engineering from the University of Modena and Reggio Emilia, Italy, in 2008 and 2010, respectively, and the Ph.D. degree from 5G Innovation Centre, University of Surrey, U.K., in 2016. He is currently a Senior Researcher with the Deutsche Telekom Chair of Communication Networks, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, Germany. From 2011 to 2015, he was also a Marie Curie Early Stage Researcher with the Instituto de Telecomunicações, Portugal, and a Visiting Researcher with Airbus Defence and Space, France.

Fabrizio Granelli (Senior Member, IEEE) is an Associate Professor with the Department of Information Engineering and Computer Science, University of Trento, Italy. From 2012 to 2014, he was Italian Master School Coordinator in the framework with the European Institute of Innovation and Technology, ICT Labs Consortium. He is the Coordinator of the research and didactical activities on computer networks. He has authored/coauthored more than 250 papers published in international journals, books, and conferences on networking. He was an IEEE ComSoc Distinguished Lecturer from 2012 to 2015, an IEEE ComSoc Director for Online Content from 2016 to 2017, and an IEEE ComSoc Director for Educational Services from 2018 to 2019.

Frank H. P. Fitzek (Senior Member, IEEE) received the Diploma (Dipl.-Ing.) degree in electrical engineering from the University of Technology—Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, in 1997, and the Ph.D. (Dr.-Ing.) degree in electrical engineering from the Technical University Berlin, Germany, in 2002. He became an Adjunct Professor with the University of Ferrara, Italy, in 2002. He is currently a Professor and the Head of the Deutsche Telekom Chair of Communication Networks, Technische Universität Dresden, Germany, and also coordinating the 5G Lab, Germany. He is the spokesman of the DFG Cluster of Excellence, CeTI. In 2003, he joined Aalborg University as an Associate Professor and later became Professor.