Network Self-configuration for Edge Elements using Self-Organizing Networks Architecture (SONAr)

Daniel Ricardo Cunha Oliveira[a], Natal Vieira de Souza Neto[b], Maurício Amaral Gonçalves[c], Flávio de Oliveira Silva[d] and Pedro Frosi Rosa[e]

Faculty of Computing, Federal University of Uberlândia, Uberlândia, Brazil

Keywords: Self-management, Self-configuration, Self-Organizing Networks, SDN, NFV, Cloud Computing.

Abstract: 5G networks deployment is already a commercial reality, while the specification groups move towards 5G standalone standardization. With the need for network slicing and edge computing capabilities – and therefore, inherent SDN and NFV adoption in a cloud environment –, these networks' complexity will require some degree of automation. The concept of Self-Organizing Networks (SON) is a trend for next-generation systems, as it is already used in the mobile radio access network (RAN) context. In this work, we will present a self-configuration (one of the four basic self-* properties of SON) system basic design proposal, within the context of Self-Organizing Networks Architecture (SONAr) framework. We will discuss the generic system architecture and present one practical application example, using the self-configuration platform to automatically configure the network to receive an eNodeB (4G base station) edge component. All the system design is being conceived to work on a carrier-grade production environment, with application versatility, reliability, and scalability.

1 INTRODUCTION

Self-configuration is a key fundamental feature in autonomic computing theory and SON. Essentially, a self-configuration system should automatically adapt the network parameters to changes in the environment. The SON specification states that the self-configuration means actions applied before the network enters an operational state (3GPP, 2018b). In computer networks, there are two distinct contexts where self-configuration may be used: (i) the network bootstrapping; and (ii) the occurrence of a new component.

The network bootstrapping means the transition from a non-operational state to an operational state. Each Network Element (NE) such as switches, routers, eNodeBs, and so on are configured and started by an operator. Typical configurations include the allocation of logical addresses (IPv4 and IPv6), the definition of static routes, the configuration of specific management VLANs management, policy rules, etc. Usually, these operations are executed manually by the network administrator.

The occurrence of a new component stands for the addition or removal of NEs. If a new NE is inserted in the network topology, the operator must configure it, as well as the neighbors' NEs. Sometimes, the introduction of a new NE means that some routes or traffic parameters should be modified to optimize the network behavior.

Some projects are found in the literature aiming attention at the orchestration and configuration of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) resources on the cloud environment, such as network slicing (Foukas et al., 2017). To our knowledge, edge components are not addressed in these projects. Furthermore, (3GPP, 2018a) ranks most of the self-configuration functions as For Future Study (FFS). This problem justifies the new approach proposed in this work: the implementation of self-configuration aspects in the context of a future self-management platform, the Self-Organizing Networks Architecture (SONAr).

In this work, we briefly describe SONAr and its main components, and specify the procedures and flows to adapt its architecture to deal with the auto-

[a] https://orcid.org/0000-0003-4767-5518
[b] https://orcid.org/0000-0001-5047-4106
[c] https://orcid.org/0000-0002-6985-638X
[d] https://orcid.org/0000-0001-7051-7396
[e] https://orcid.org/0000-0001-8820-9113

Oliveira, D., Neto, N., Gonçalves, M., Silva, F. and Rosa, P.
Network Self-configuration for Edge Elements using Self-Organizing Networks Architecture (SONAr).
DOI: 10.5220/0009465104080414
In *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020)*, pages 408-414
ISBN: 978-989-758-424-4

nomic configuration when edge components are inserted on the network. As a use case, we will present the self-configuration case when an eNodeB (4G Base Station) is plugged into a specific network. By using our approach, network operators should connect the new edge components in a physical connection, while all configuration settings will be done automatically by SONAr platforms. Beyond the dynamic parameter settings, the self-configuration introduces an additional security mechanism: the self-configuration module will verify whether the new NE is allowed to get in the network topology.

The remaining of this paper is structured as follows. Section 2 gives some related work. Section 3 presents the self-configuration requirements in future networks and the SONAr model. Section 4 brings our proposed approach, and finally, Section 5 presents concluding remarks and future work.

2 RELATED WORK

Since 2001, IBM was already concerned about the increasing complexity of computer networks and the inherent difficulties to manage them, as well as the companies' OPEX increase with the need of highly specialized professionals to perform trivial tasks. (Ganek and Corbi, 2003) defines the basis and concepts for autonomic computing, including the four self-* features (self-configuration, self-healing, self-optimization, and self-protection) and their characterizations.

Project ANEMA (Autonomic Network Management Architecture) (Derbel et al., 2009) also defines a high-level architecture model for self-configuration and self-optimization in IP networks. It does not, although, addresses self-protection or self-healing features, as well as SDN/NFV capabilities, being restricted to regular IP networks with non-virtualized components.

(Atayero et al., 2014) presents a vision on self-organizing networks applied to 3GPP's Long Term Evolution (LTE). (3GPP, 2018b) is the 3GPP's technical specification (TS) that introduces the concepts and requirements for mobile network SON. The scope of this TS involves only self-optimizations and self-healing, and it is limited to the Radio Access Network (RAN) 3GPP protocols.

(Neves et al., 2016) brings the SELFNET framework, an approach for autonomic management in NFV and SDN networks, mainly designed for 5G networks. Both SONAr and SELFNET have some mutual functions and may work together in the future since SONAr is not dependable on the network type – it was designed to work with NFV, SDN, and even legacy network elements.

Although many efforts to achieve a complete and comprehensive framework and architecture for Self-Organizing Networks, none is already finished and available. There are indeed a lot of requirements left unmet by the several projects in this regard.

3 BACKGROUND

In this section, a review of the ongoing telecommunications and computer networks is presented, along with the reasons that led us to consider self-configuration as an indispensable approach to work on these networks. First, we characterize the requirements of current and future networks and, subsequently, we bring the SONAr project where our proposed self-configuration functions are placed.

3.1 The Need for Self-configuration

Future telecommunications and computer networks are supposed to satisfy new application requirements, such as high-volume traffic, Ultra-Reliable and Low-Latency Communication (URLLC), fault detection, etc (Chen et al., 2018). Current and new approaches (like IoT platforms, cloud computing and 5G networks) are expected to have capabilities to support those requirements (Barona López et al., 2017). In these cases, there are a vast number of network components to support billions of devices, as well as a large number of rules to satisfy the communication requirements. The new set of rules includes edge computing (enabled by cloud and fog computing) network slicing management (SDN), infrastructure orchestration (NFV), distributed resource management, and so on. It can be inferred that it is impossible to manage this complex environment manually.

The autonomic computing concept proposed by IBM (Ganek and Corbi, 2003) already predicted that future systems must have mechanisms to manage themselves autonomously. We believe that the autonomic computing era will arrive for computer networks with the introduction of 5G Standalone (5G SA) networks: with the fully virtualized and SDN enabled core network in cloud infrastructure, 5G will be prepared to deal with several different applications and requirements. The self-adaptation is crucial to achieving this idealization.

Autonomic computing is composed of the four mentioned fundamental features: self-healing, self-optimization, self-protection, and self-configuration. The SONAr approach discussed in Section 3.2 deals

with these four features. However, for this paper, the target is the self-configuration component. Computer networks will require self-configuration functions for several reasons:

- Man made configuration of too many NEs will require a great professional effort;
- Reliability issues due to manual configuration of a complex network;
- The high OPEX costs with professional services could be a disavantage to business;
- Security issues with unauthorized NEs plugged in the network;
- Network environments are changing too fast and, therefore, demands more intense number of interventions.

Within the configuration of a network, the edge NEs represent a special case. For many of them, such as mobile base stations, a specialized configuration have to be done, i.e. special VLANs with different forwarding policies and prioritization must be assigned for different packets types, as it will be seen in Session 4.3. It is not an obvious or natural configuration, as would be expected for regular NEs, such as switches or routers. Therefore, edge NEs require specialized treatment in the network self-configuration.

3.2 SONAr

The SONAr project aims to provide self-* capabilities to computer networks. The first architecture release was designed as a platform that operates in SDN and NFV environments. However, the architecture is prepared to involve legacy networks and protocols as well, in a carrier-grade level of service.

Figure 1 shows the layered architecture design. The Client, Control, and Infrastructure layers hold NEs, SDN control components, NFV management components, and other Operations, Administrations, and Management (OAM) systems. All these mentioned elements are managed by the SONAr platform, which is composed of the Interface Layer, Management Layer, and Administration layer.

The Collector Entity (CoE) collects topology and metrics from the other layers. The Control Primitives Interceptor (CPI) intercepts management primitives in the Southbound Interface (SBI). The Network Service Broker (NSB) integrates with SDN controllers and NFV orchestrators to execute the platform decisions. The Network Event Manager (NEM) allows event-driven communication between the platform components. Finally, the Self-Learning Entities (SLE) and Self-Organizing Entities (SOE) consist of self-* services.

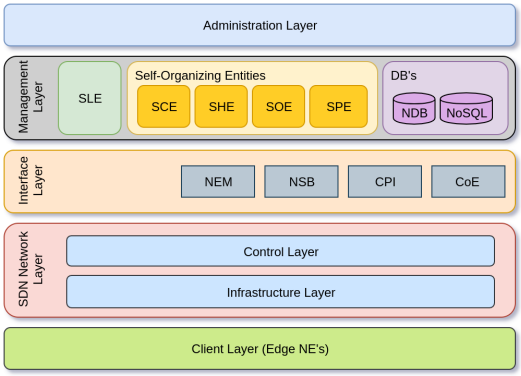

Figure 1: SONAr Basic Design.

The Self-Healing Entity (SHE), the Self-Optimization Entity (SOE) and the Self-Protection Entity (SPE) are out of scope in this document. The SLE implements AI functions such as Machine Learning (ML) to predict the whole structure behaviour. The outputs from SLE may generate configuration tasks in the environment. The Self-Configuration Entity (SCE) is the key component in this role, where the services for network self-configuring are performed. The SCE utilizes the Network Database (NDB) to store topology information, settings, configurations applied, and other relevant information. Section 4 will describe the SCE in details.

4 SELF-CONFIGURATION PROPOSED ARCHITECTURE

In this section, we will present the system's general design, followed by a typical message flow explanation, then an example of application and, finally, the next steps of the work.

4.1 General Design

The general design for the Self-Configuration Entity is shown on Figure 2.

From bottom-up, the Infrastructure Layer represents all the NEs that make up the network, including switches, routers, servers, and other NEs on edge, since the Client Layer is not represented here. These NEs may or may not be Virtual Network Functions (VNFs) running on an NFV platform, such as Open vSwitches deployed on a cloud environment. SDNC is the SDN Controller. In the diagram, we are assuming that all the NEs in the infrastructure layer are SDN compatible, i.e., they receive OpenFlow commands (or any other Southbound Interface protocol

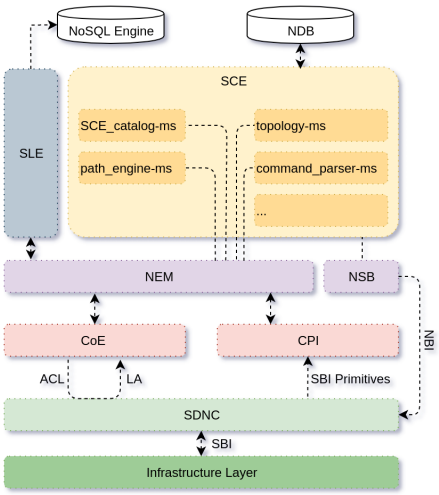

Figure 2: Self-Configuration (SCE) System Basic Design.

commands). In the case of a legacy non-SDN compatible NE, the commands will come from the NSB directly to the NE without passing through SDNC, as will be seen in the next sessions. In this particular case, the NE's proper syntax will be used, instead of the OpenFlow protocol.

CoE and CPI perform network monitoring. The first uses both the Autonomic Control Loop (ACL) and the Local Agent (LA) services to collect network information. ACL monitors the Infrastructure Layer and it's NEs periodically, while the LAs are internal agents inserted into the NEs that summarize data and send it to CoE. CPI is responsible for copying and parsing OpenFlow messages and searching for primitive metrics such as *ofpmp-port-stats*, for example.

NEM component receives the data collected by CoE and CPI and converts them into events, which are published in topics and made accessible to all other elements that subscribe to these topics. The Network Service Broker (NSB) is responsible for receiving messages from SCE, in addition to the other Self-Organizing Entities, and send either Northbound Interface (NBI) commands to the SDNC, or compatible commands directly to the NE in case of legacy one.

The Self-Learning Entities (SLE) will not be discussed here in details due to lack of space, but, in brief, it is responsible for providing AI-based mechanisms to overcome the absence of human intervention. It also subscribes to topics on NEM and receives information about the network, storing and retrieving them in/from the NoSQL Engine. Then it will evaluate improvements that will be implemented by all self-* entities.

Finally, SCE is composed of several microservices (ms) that will act together to provide the right configuration to the network. They are:

- SCE_catalog-ms: this service subscribes to NEM's topics and becomes aware when a new edge NE is plugged in the network. Its function is to determine whether a new edge NE is in the authorized NE database and, if so, discover the type, vendor and version of this element;
- topology-ms: the function of this service is to build the topology between all the network components, including all the NEs and the SDN controller;
- path_engine-ms: this service must calculate all paths between two nodes on the network, including the best path between a node and its controller;
- command_parser-ms: identifies all the NEs in the path (discovered by path_engine service) between the NE on edge and its final endpoint, matching them to the specific configuration syntax. More details of this procedure will be presented in the next subsection.

At last, NDB is where all the network information tables are stored, and the NoSQL Engine Database stores all metrics collected by SLE, involving a high volume of data storage.

In the next subsection, we will present a brief description of a typical message flow between these components when a generic edge NE is plugged into the network.

4.2 Typical Message Flow Diagram

To describe the process of a typical network self-configuration, let's assume that an edge NE is plugged into the network for the first time. Figure 3 shows how the message flows between the SONAr and SCE components.

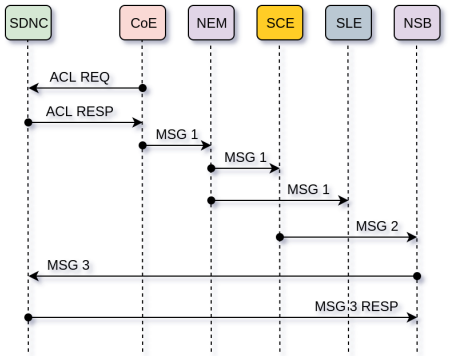

Figure 3: Self-Configuration Flow Diagram.

First of all, CoE (through ACL) sends an update to NEM (MSG 1), informing that a new NE was detected. NEM publishes the event on a specific topic.

The service SCE_Catalog, which subscribes to this NEM topic, receives the information that a new NE was plugged, and then it receives its MAC address (MSG 1). SLE component also gets this update, but its subsequent actions are out of the scope of this paper. Then SCE_Catalog service consults the edge_catalog.db table (on NDB) to verify if: (1) this specific MAC address exists and is authorized, providing an additional security measure against unauthorized network intrusion, and, if so; (2) retrieving the type and version of the NE. Then it starts the command_parser service.

The command_parser identifies every element between the NE on edge and its final destination NE (determined by path_engine service), relating every SDN NE with their configuration syntax. For this, it uses both syntax.db, which contains the commands grammars for each NE network version for each edge NE type, and resources.db, a resource pool where the service may get available IP addresses, for example. It mounts the OpenFlow commands, for example, and send them to the NSB (MSG 2). The command field in syntax.db may be defined using any compatible format, but in this work, Jinja2 is being used as a template engine for describing the command's syntaxes.

Finally, NSB will send the configuration commands to the SDNC through the NBI (MSG 3). SDNC responds, acknowledging the execution of the commands. In case of legacy elements, NSB will send non-SDN proper configuration commands directly to these NEs.

In short, that is the primary communication flow between all components during the self-configuration process.

4.3 Possible Applications

In this subsection, we intend to demonstrate a high-level application example for the self-configuration procedures. We do not intend to enter specific OpenFlow details, mobile network-specific protocols or message analysis.

One typical application, presented here as an example, is the network self-configuration when an eNodeB is plugged into a switch connected to this network. We assume here that all the NEs are SDN compatible and an SDNC (not shown) is controlling all the elements and it uses OpenFlow protocol on SBI. Some of them may be, as mentioned before, Open vSwitches VNFs if part of the infrastructure is cloudified. Also, for simplification, the SONAr components are not represented in the diagram. Since a 4G eNodeB is considered for this example, the core elements may or may not be VNFs running on an NFV structure. The example network, in a simplified topology, is shown in Figure 4.

Just after the eNodeB is plugged on S1, the SDNC receives an OFPT_HELLO message, and the following messages exchange will provide the SDNC with the eNodeB MAC address (as mentioned, the OpenFlow message flows will not be detailed here). Following that, the CoE will request network updates through ACL Request message and the SDNC will respond with an ACL Response, containing information about the new edge NE. CoE will then send a message to NEM informing about the network modifications, and NEM will update its topics with them.

SCE's SCE_catalog service subscribes to this topic in NEM and, therefore, will receive the information about the new eNodeB. It will then consult the edge_catalog.db table on NDB and verify if the eNodeB is an authorized element on the network. Assuming it is, it will retrieve from edge_catalog the type and version of the component, using the MAC address as the key. In this case, it should return something like "eNB_vendorXXX_version2.3.8.001c", for example. The path_engine service calculates the best route from S1 to the final IP destinations. In this case, there are four IP destinations for the packets from and to the eNodeB: control messages should be forwarded to Mobility Mobile Entity (MME), management messages will go to the NE Manager, user plane messages shall be sent to the Enhanced Packet Core (EPC) and, finally, the sync packets should go to the Clock Server. In traditional networks, this configuration is done by manually setting and configuring four VLANs in each element in the path from S1 to S6.

The path_engine service determines that the best route from the eNodeB and the packet's destination should be S1-S2-R1-R2-S6. Therefore, these five elements should be configured. Notice that the redundancy path S1-S5-S4-S3-S2-R1-R2-S6 will not be set, since the self-healing service should treat all the network failures, sometimes in a predictive way. In case of a broken link, SHE should call SCE to reconfigure the new path, but this procedure is also out of the scope of this paper.

The command_parser service is then started and receives the element type information, together with the NEs that shall be configured. In order to do this, it fetches the syntax.db table from the NDB (which relates the edge element type and the NEs types to the command syntaxes) and mounts the command sequences for S1, S2, R1, R2, and S6 elements, also using the resources.db table, when additional parame-

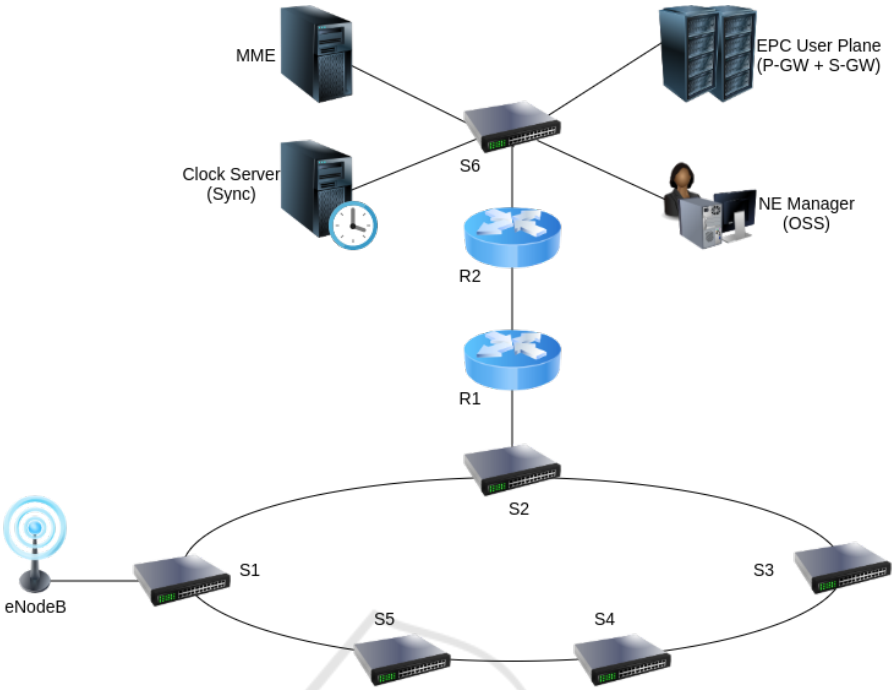

Figure 4: Example of Self-Configuration Application.

ters are needed. Next, command_parser service sends the command sequences to the NSB. NSB then sends the commands to the SDNC, who will, ultimately, create all the flows and other configurations (such as policy rules) for each element. The use of OpenFlow protocol makes it relatively simple to assign specific flows entries for a particular packet, given an IP destination.

At last, SDNC returns a response to NSB, acknowledging that the commands were successful accepted and implemented on all NEs.

One last remark, the self-configuration platform does not configure the edge NE itself, i.e., in the example, it does not input any specific parameter to the eNodeB, such as Public Land Mobile Network (PLMN), Radio Resource Control (RRC) or Radio Link Control (RLC) paramaters, etc. The self-configuration platform only acts on the access (L2) and transport (L3) network, preparing it for the already configured eNodeB, which has to be done in a separate process (manually or not).

4.4 Next Steps

Both self-configuration module and SONAr as a whole still require some work, which is ongoing. For SONAr, many of the components are already developed and working, such as NEM, NSB, CoE, etc. The network bootstrap procedure is functional. Some of the Self-Organizing Entities are currently being developed, in special SCE and SHE. The SLE components also need to be improved, as some of them still make analytic decisions based on simple statistics trends, instead of more complex and efficient AI techniques.

SCE is currently under construction. The services path_engine and topology are already implemented and working, but SCE_catalog and command_parser are still to be coded and tested.

All the architecture is being developed and tested in a virtual environment based on GNS3 software. This allows us to make adjustments to the system and simulate both traditional and NFV applications. A partnership with a local operator will enable tests to on the real environment, using its lab with physical and virtual network elements.

5 CONCLUSION

In this work, we have proposed a self-configuration platform model, within the context of the SONAr system. We have shown SONAr underlying architecture and, in particular, the Self-Configuration Entity, which will perform the (preferable, but not exclusively) SDN/NFV network configuration automatically. We have presented the main components on the interface layer (NEM, NSB, CPI, and CoE) and man-

agement layer (SLE and SOE) and the overall system structure.

We then presented the self-configuration module architecture, with all the services and databases that compose it, along with its functions and interfaces, followed by the typical message flow through the system and how the components interact with each other.

Finally, we have demonstrated an application example using the network self-configuration procedure when an eNodeB is plugged into an operator's network in a sample topology. In a high-level manner, we have described the information flow and how the services work together to build the configuration needed to accept the new NE rapidly. Some issues have been discussed, primarily related to the network's security and reliability.

There is still some work to be done before the self-configuration module can be thoroughly tested, so this paper has no intention to present final or partial results yet. The tests and measures will be done as soon as the system is complete and will be presented in future works.

ACKNOWLEDGEMENTS

This work is inside UFU-CAPES.PrInt Program. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

REFERENCES

3GPP (2018a). Telecommunication management; Self-configuration of network elements; Concepts and requirements. Technical Specification (TS) 32.501, 3rd Generation Partnership Project (3GPP).

3GPP (2018b). Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements. Technical Specification (TS) 32.500, 3rd Generation Partnership Project (3GPP).

Atayero, A. A., Adu, O. I., and Alatishe, A. A. (2014). Self Organizing Networks for 3GPP LTE. In Murgante, B., Misra, S., Rocha, A. M. A. C., Torre, C., Rocha, J. G., Falcão, M. I., Taniar, D., Apduhan, B. O., and Gervasi, O., editors, *Computational Science and Its Applications – ICCSA 2014*, volume 8583, pages 242–254. Springer International Publishing, Cham.

Barona López, L., Valdivieso Caraguay, Á., Sotelo Monge, M., and García Villalba, L. (2017). Key technologies in the context of future networks: operational and management requirements. *Future Internet*, 9(1):1.

Chen, H., Abbas, R., Cheng, P., Shirvanimoghaddam, M., Hardjawana, W., Bao, W., Li, Y., and Vucetic, B. (2018). Ultra-reliable low latency cellular networks: Use cases, challenges and approaches. *IEEE Communications Magazine*, 56(12):119–125.

Derbel, H., Agoulmine, N., and Salaün, M. (2009). ANEMA: Autonomic network management architecture to support self-configuration and self-optimization in IP networks. *Computer Networks*, 53(3):418–430.

Foukas, X., Patounas, G., Elmokashfi, A., and Marina, M. K. (2017). Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100.

Ganek, A. G. and Corbi, T. A. (2003). The dawning of the autonomic computing era. *IBM systems Journal*, 42(1):5–18.

Neves, P., Calé, R., Costa, M. R., Parada, C., Parreira, B., Alcaraz-Calero, J., Wang, Q., Nightingale, J., Chirivella-Perez, E., Jiang, W., et al. (2016). The self-net approach for autonomic management in an nfv/sdn networking paradigm. *International Journal of Distributed Sensor Networks*, 12(2):2897479.