Using the RFC 7575 and Models at Runtime for Enabling Autonomic Networking in SDN

Felipe A. Lopes

Instituto Federal de Alagoas – Campus Arapiraca
Arapiraca, Brazil

felipe.alencar@ifal.edu.br

Abstract. *The programmable network architectures that emerged in the last decade have opened new ways to enable Autonomic Networks. However, there are several open issues to address before making such a possibility into a feasible reality. For instance, defining network goals, translating them into network rules, and granting the correct functioning of the network control loop in a self-adaptive manner are examples of complex tasks required to enable an autonomic networking environment. Fortunately, architectures based on the concept of Models at Runtime (MART) provide ways to overcome such complexity. This paper proposes a MART-based framework – using the RFC 7575 as reference (i.e., definitions and design goals for autonomic networking) – to implement autonomic management into a programmable network. The evaluation shows the proposed framework is suitable for satisfying the functional and performance requirements of a simulated network.*

1. Introduction

About ten years ago, the emergence of the Software-Defined Networking (SDN) paradigm rekindled the debate over programmable networks. With the advances made by the network community since then, it was possible to conceive that programmable networks are a fundamental piece of achieving autonomous networks, in which network management and adaptations take place autonomously – a concept commonly known as Autonomic Network Management (ANM) [Samaan and Karmouch 2009].

When separating the control plane from the forwarding devices (i.e., one of the pillars of SDN) [Kreutz et al. 2014], self-adaptive actions can be implemented into the software (centralized or distributed) that manages the network [Wickboldt et al. 2015]. However, few proposals demonstrate how a programmable network architecture (e.g., SDN) can be integrated with the definitions and design objectives of autonomic networks – even in simulated scenarios. According to [Jiang et al. 2015], translating high-level objectives into code to be executed on routing devices, reacting autonomously to changes in the network based on knowledge, and defining where autonomic loops should be implemented and controlled are some of the open challenges in adopting ANM in SDN scenarios. In addition to these difficulties, the network community has no standard widely accepted to implement ANM systems.

Fortunately, some of the challenges related to the ANM systems development already have solutions applied in the area of autonomic computing. One of these solutions is known as Models at Runtime (MART), which uses a high-level abstraction associated with code generation technologies to monitor, control and execute autonomic systems

[Blair et al. 2009]. Besides, in the context of standards, IETF's joint efforts created the RFC 7575 [Behringer et al. 2015] to define a common language and design objectives for autonomic functions in a network. Thus, based on the guidelines defined in RFC 7575, this work aims to propose and analyze a MART framework for implementing an ANM. The main contribution of this paper is in demonstrating the applicability of MART to implement the RFC 7575's reference model, using the proposed framework as the basis for such implementation. Besides, this paper evaluated the proposed framework's performance and found that its execution could enable autonomic network functions, such as network discovery and flow forwarding functions.

This paper is organized as follows: Section 2 presents the basic concepts related to the target problem and the proposal; Section 3 discusses related work; Section 4 presents the proposed MART framework using the RFC 7575 guidelines; in Section 5, there is the performance evaluation of the framework in a simulated environment; Finally, Section 6 brings conclusions perceived from the integration between MART and ANM for programmable networks.

2. Basic Concepts

This section presents the basic concepts that are fundamental to the understanding of the work. It also defines the scope for which the present MART-based proposal was designed to enable ANM.

2.1. ANM, SDN, and the RFC 7575

According to [Samaan and Karmouch 2009], Autonomic Networking Management should anticipate, diagnose, and prevent any fault that may be caused to the underlying network. Besides, such capabilities should be achieved using an independent and autonomic manner, guided by a set of high-level objectives. Recent research papers might refer to ANM concepts using the terms Cognitive Network Management [Ayoubi et al. 2018], Self-Driving Networks [Kalmbach et al. 2018], or Knowledge-Defined Networks [Mestres et al. 2017]. Despite the conceptual definition, there is still no consensus on how these diagnosis and prevention actions should occur. In this context, an RFC 7575 provides guidelines and a reference model that aims to pave the way for autonomic behavior to be implemented in networks.

According to [Behringer et al. 2015], the original design goals of autonomic systems also apply to Autonomic Networks. The main objective is to achieve self-management, which involves several *self-** properties (e.g., self-configuration, self-healing, self-optimizing). In this context, joint efforts at the IETF's Network Management Research Group (NMRG) proposed the RFC 7575 – a draft describing NMRG's view on autonomic networking. It defines that *self-** properties must be autonomous functions at the node level, i.e., autonomic nodes – with minimal dependence on management systems or controllers. On the other hand, [Behringer et al. 2015] also emphasizes that the communication between these autonomic nodes must be through an Autonomic Control Plane (ACP), which also already have standardization efforts [Eckert et al. 2020].

In addition to the definition of a general self-management objective, RFC 7575 outlines 11 (eleven) design goals for autonomic networking solutions, among them: co-existence with traditional management, decentralization and distribution, simplification

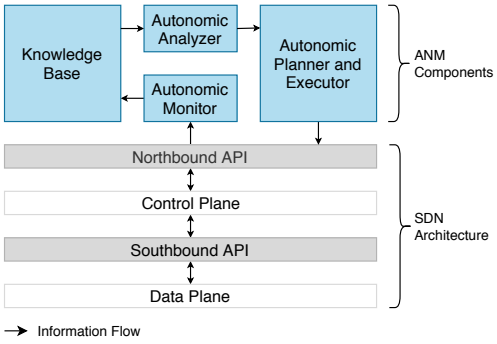


Figure 1. ANM-Based SDN Architecture.

of autonomous node northbound interfaces, abstraction, and monitoring autonomic. The work of [Eckert et al. 2020] extends the RFC 7575, proposing a reference model for autonomic IP networks based on the GRASP protocol. Although [Eckert et al. 2020] could be used as a reference model, the specificity of protocols and the focus on IP networks of this IETF's Internet Draft (ID) hampers the implementation of a more generic solution for alternative network architectures (e.g., SDN). However, the conceptual view on SDN composition with ANM presented here in this paper – depicted in Figure 1 – is similar to such IETF's ID and other authors [Stamou et al. 2019]. Figure 1 shows an ANM system and its components interacting with an SDN network using the Northbound API. Besides, the ANM system executes adaption actions into the SDN network using its knowledge.

2.2. MART

The concept of Models at Runtime discussed here appeared first at the work of [Blair et al. 2009]. The authors defined the term MART as a causally connected self-representation of an associated system that emphasizes the structure, behavior, or goals of the system from a problem space perspective. It is noteworthy that the word "Models" present in MART comes from the Model-Driven Engineering (MDE) discipline and refers to an abstraction or reduced representation of a system that is built for specific purposes. Besides, advanced modeling techniques coupled with MDE capabilities, such as model-to-model transformation and code generation, offer a viable means to capture runtime monitoring information, provide decision-making support through runtime model analysis, and enable runtime adaptation of the system using a model-based approach.

The primary purpose of MART is to enable unanticipated adaptations while ensuring guarantees. It achieves such adaption characteristic by providing abstractions for certain aspects of software code, maintaining runtime models of themselves. Besides, a MART-based system also tries to predict its future behavior when internal or external events occur. According to [Aßmann et al. 2014], the critical characteristic of a system based on MART is then its ability to use some aspects of the reality (e.g., context, behavior, goals) to the modeling space in order to enable tractable decisions and produce decidable plans. As a technology, MART enables various further technologies, especially the possibility to realize *self-adaptive software* (e.g., ANM software for SDN). It is possible due to the reasoner component present in most of the MART architectures [Aßmann et al. 2014, Bencomo and Blair 2009]. The reasoner is responsible for predicting actions in order to provide goal-oriented adaptation.

3. Related Work

Two groups categorize the efforts to enable autonomic networking by means of SDN: (i) architectures and (ii) approaches that investigate part of such management (e.g., resource usage, QoS). For instance, regarding architectures, in [Li et al. 2013] the authors propose the Autonomic Management Architecture (AMA), aiming to offer multiple autonomic networking services specifically in SDN networks. In [Koutsouris et al. 2013], the authors implement the Unified Management Framework (UMF) dealing with an SDN scenario to achieve autonomic properties in SDN management. Both proposals virtualize the SDN infrastructure and implement autonomic blocks to manage the network. Such an idea is also present in [Qi et al. 2014] and [Gelenbe 2013]; however, the second introduces intelligent packets into the SDN network in order to make it autonomous. More recently, [Yahia et al. 2017] proposed a cognitive architecture (named CogNitive) to provide autonomic management in software-defined 5G networks, using the MAPE-K cycle and machine learning techniques. Focusing on QoE, [Volpato et al. 2017] defined an ANM architecture (i.e., the Autonomic QoE Management Architecture) to offer QoE-aware services. The work of [Benayas et al. 2019] describes a semantic data lake architecture for fault management based on Bayesian Reasoning for SDN Environments. In [Bega et al. 2019], the authors propose an architecture based on deep learning for enabling autonomic behavior.

Other authors decided to realize parts of the autonomic management in SDN from implementing algorithms [Tuncer et al. 2015], protocols [Tsagkaris et al. 2015], SDN controllers [Poulios et al. 2014], and frameworks [Barron et al. 2016, Ahmad et al. 2015, Tsagkaris et al. 2015]. The adaptive algorithm discussed in [Tuncer et al. 2015] places SDN controllers integrating the SDN management. In [Zhou et al. 2013], the PindSwitch protocol resembles the OpenFlow protocol but with autonomic actions. The same idea is present in [Tsagkaris et al. 2015, Ochoa-Aday et al. 2019] proposing, respectively, the AUTOFLOW and SHP protocols, which combine SDN with ANM. From the SDN controller perspective, the work of [Poulios et al. 2014] presents the AutoSDN, an SDN controller to enable *self-** characteristics in SDN networks.

The present work differs from the previous efforts by considering the entire MAPE-K cycle when defining an ANM system for SDN networks. For instance, none of the previous proposals present a high-level integrated manner to define, check, and execute network goals autonomously. Besides, it appears to be the first ANM implementation based on the RFC 7575 and the first to integrate high-level goals for generating low-level parameters for a machine learning technique – used to enable autonomic network functions.

4. Proposal

In this work, MART and SDN are key enablers to implement ANM systems. Therefore, its proposal is a MART-based framework following the RFC 7575 guidelines to enable ANM in an SDN scenario. This section presents the proposal's architecture, identifying the related RFC 7575's guidelines considered in its design.

4.1. Architecture

The proposed framework's architecture (cf. Figure 2) reflects the three layers proposed by [Aßmann et al. 2014] for MART solutions and can be categorized as a hierarchical

architecture based in [Movahedi et al. 2012], composed of: network model, adaptability, and infrastructure. Each layer has its functional blocks, interacting and exchanging information with components or blocks of other layers. The definition of each block and the inter-block communication follows the RFC 7575's design goals. For instance, as discussed in Section 2, RFC 7575's main design goal is self-management, in which *self-** properties need to be satisfied. The following subsections define each layer, relating their components to the corresponding design goal.

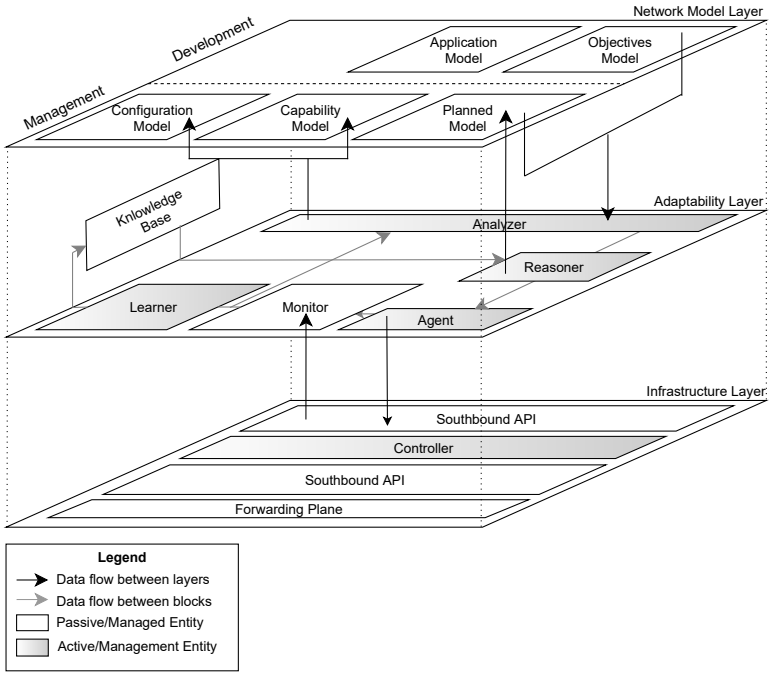


Figure 2. The architecture of the proposed MART-based framework.

4.1.1. Network Model Layer (NML)

NML performs the separation between development models (i.e., application models, objectives) and management models (i.e., configuration model, capacity, and planned model). In such a way, development models could be directly defined by a developer or network operator, while management models might be autonomously defined.

It is worth mentioning that components of this layer are based on meta-models defined for the MART structure (cf. Figure 3). This work defined such meta-models (and their entities) considering the concepts of ANM, SDN, and the RFC 7575's guidelines. These meta-models enable high-level modeling, consequently: (i) representation of network state; and (ii) specification of objectives (i.e., intents) or the application modeling. Besides, following the MDE discipline – that is, the base of the MART paradigm – high-level models are also part of the code generation feature. The role of each functional block of this layer is described as follows:

- **Application Model:** enables network operators and developers to model network applications (e.g., firewall, load balancing, network monitor), resulting in code that SDN controllers execute.

- **Objectives Model:** enables the modeling of objectives[1] (e.g., minimize delay, improve throughput) for a network.
- **Configuration Model:** represents the actual network configuration (e.g., topology, running services).
- **Capabilities Model:** reflects the network potential (or capabilities) to meet requirements (e.g., QoS, bandwidth, delay). It provides information to the Objectives Model.
- **Planned Model:** it describes actions that the network should perform. A planned model represents a sequence of commands or actions describing how agents of lower layers may perform changes in the network environment. Such actions could be autonomously or manually defined. In practice, the planned and objectives model are linked to each other, even though they are architecturally separated.

Each block has its code generation action, which may occur autonomously – even though the definition of network intents involves a manual procedure. However, the elimination of human operators is not in the RFC 7575's scope.

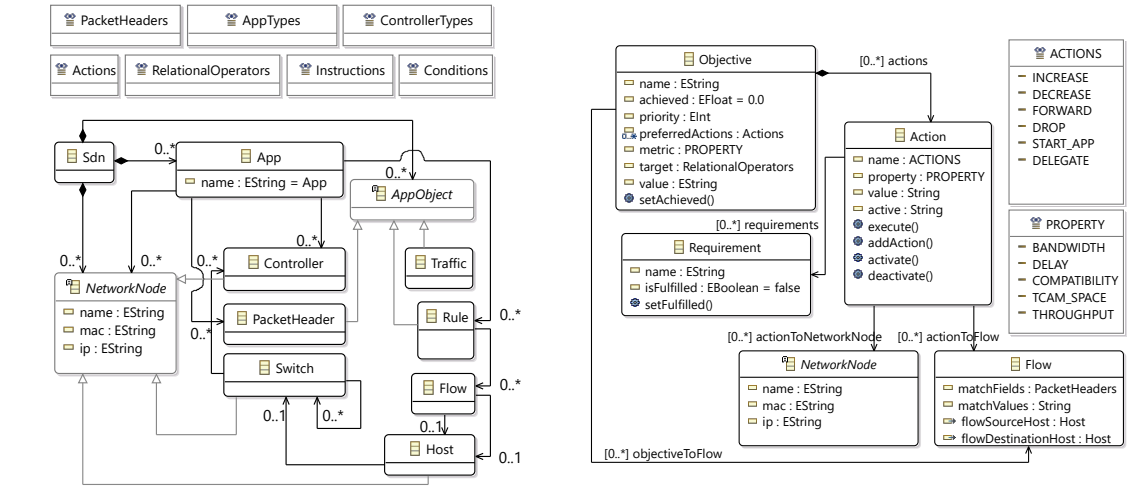
Figure 3a depicts the meta-model used for describing network goals (based on the ECore language [Gronback 2009]). It is the Objectives Model block base and consists of five main classes for representing a set of objectives, requirements, and actions. The class Flow and the abstract class *NetworkNode* are reusable classes from previous work [Lopes et al. 2015] (cf. Figure 3a). From the Objectives meta-model (cf. Figure 3b), a network developer can model objectives, their requirements, and corresponding actions. Obviously, in some cases, not all modeled objectives will be achieved by the underlying network. Thus, the Objective class has the **active** attribute for controlling, which are the possible network objectives.The ones possible to be achieved from the modeled objectives will be transformed into the application code to interact with the SDN controller and the underlying network infrastructure.

4.1.2. Adaptability Layer (AL)

AL accommodates components to perform the interaction with NML's models and contain implementations representing the MAPE-K concepts. At this level, there are transformations (e.g., model-to-code, model-to-command) from both high-level models into low-level instructions and from network data into high-level models. The following items detail the AL layer components:

- **Analyzer:** plays two distinct roles in the proposed framework's architecture. First, it categorizes the information received from the network through configuration and capability models. Second, it verifies if the actual network state needs to be changed, considering the knowledge base and the planned objectives.
- **Reasoner:** verifies if future network configurations should be defined, considering the knowledge base and the objectives model.
- **Learner:** continuously feeds the knowledge base, verifying if the reasoner's decision was beneficial for the network.

[1]Hereinafter, the words goal and objective will be used interchangeably, without loss of generality.



(a) Short version of meta-model used by the NML Management part.

(b) Objectives and Planned meta-model.

Figure 3. Examples of meta-models used in the MART-based framework.

- **Agent:** performs changes in the network based on the instructions prepared by the previous components. The agent performs changes by sending generated code through the Northbound Interface (NBI) of SDN controllers.
- **Monitor:** queries the network using parameters that are relevant to achieve the modeled objectives.
- **Knowledge Base (K-Base):** is the component responsible for storing data about the network control knowledge instead of the domain knowledge (since it is reflected in the models). K-Base enables the proposed framework to register application descriptions, beneficial decisions, and actions that caused errors.

In addition to the conceptual definition of each block, it is worth detailing the analyzer, reasoner, and learner blocks from the code generation perspective. The algorithm used in the code generation of these components receives the parameters of NML layer models and must be based on Deep Reinforcement Learning (DRL), i.e., a type of machine learning algorithm – this choice is beyond the guidelines of RFC 7575 and is a design choice advocated in this paper. The reinforcement deep learning algorithm used to allow this behavior has the following definition:

The action selection process – consisting of actions modeled at the Objectives Model – is the set p in a Markov Decision Process (MDP), p is in some state s at time step t, and the decision-maker may choose any action a that is available in state s. After taking action a, p is at the time step t' randomly moving to the state s', and giving the decision-maker a corresponding reward $R_a(s, s')$. Thus, according to [Puterman 2014], MDP is a 5-tuple (S, A, T, R, γ) – where γ is a discount factor that varies between 0 and 1 for rewards at each time t, representing the weight of future rewards (e.g., γ with a value of 0 benefits recent rewards, while a γ closer to 1 benefits older rewards). However, in the current scope, S is a large bounded set, and both S and A are continuous spaces. The probabilities or rewards of R are unknown (which emphasizes the use of DRL to address the action selection issue).

The proposal presented here train a DRL algorithm proposed by [Tessler et al. 2019] to estimate the actions based on learned states. In this manner, the DRL algorithm uses an agent for choosing and executing actions in network states (continuously monitored by the proposed framework's monitor, cf. Section 4.1.2). It obtains feedback for that action and uses the feedback to update the K-Base. The DRL has a so-called Q-factor for every state-action pair. When the feedback for selecting an action in a state is positive, the associated Q-factor's value is increased, while if the feedback is negative, the value is decreased. Such feedback is the input for the reward function R_a. Algorithm 1 shows a pseudo-code of the DRL used in this proposal.

Algorithm 1: Deep Reinforcement Learning

```
Input: Retrieve data from the K-Base; if exists
Input: Initialize weights for action-value pairs, state s_t, objective g;
Data: reached = false;
Data: episode = 0;
begin
    while !reached do
        for step<learning_rate; step++ do
            Select action a_t in s_t;
            Query network parameters in s_t;
            Select a parameter param_t;
            ε = ε − (step/learning_rate) ∗ ε;
            Execute action a_t for param_t;
            Observe new state s_{t+1};
            Calculate reward r_t and Store experience in K-Base;
            Update param_t in the Planned Model;
            Collect samples of n randomized transitions of K-Base;
            Calculate discount factor d of the reward r_t;
            Update the transition t_t with d;
            Train the Q-Network with the new d values of transition t_t, s_t + 1, and a_t;
            reached = compare(g, better_t, worst_t)
        end
        episode++;
    end
end
```

It is worth emphasizing that AL layer's modules interact with Algorithm 1 and vice-versa. These modules can modify parameters of each step and be modified by the DRL algorithm itself. For instance, when DRL algorithm queries network parameters and takes an action (e.g., install a new flow forwarding rule), such action may change network behavior or performance, which also changes the network models and its parameters at the NML layer.

The investigation of efficient algorithms to provide autonomic behavior of network functions to achieve network objectives is an open issue.

4.1.3. Infrastructure Layer (ILA)

ILA is the managed system (according to the MART concept). It follows the Open Networking Foundation (ONF) architecture for SDN [Schaller and Hood 2017]. This layer is

the target for objectives defined at the NML. Moreover, ILA provides information to the AL (cf. Section 4.1.2) to satisfy such goals.

5. Evaluation and Discussion

The evaluation of this work consists of two main analyses: (i) a qualitative investigation for checking which RFC 7575's design goals could be achieved so far by the proposed framework; and (ii) a quantitative assessment for observing performance factors of the proposed solution in a simulated scenario. Both analyses consider the use case depicted in Figure 4, the high-level modeling enabled by the MART-based framework. In this Figure, there are two models, i.e., a) the Capability Model, consisting of network elements; and b) the Objectives Model, consisting of network objectives and actions. Moreover, the modeled objectives are: i) autonomously enable network forwarding, and ii) minimize the delay. Both objectives should consider `<<Flow1>>` parameters.

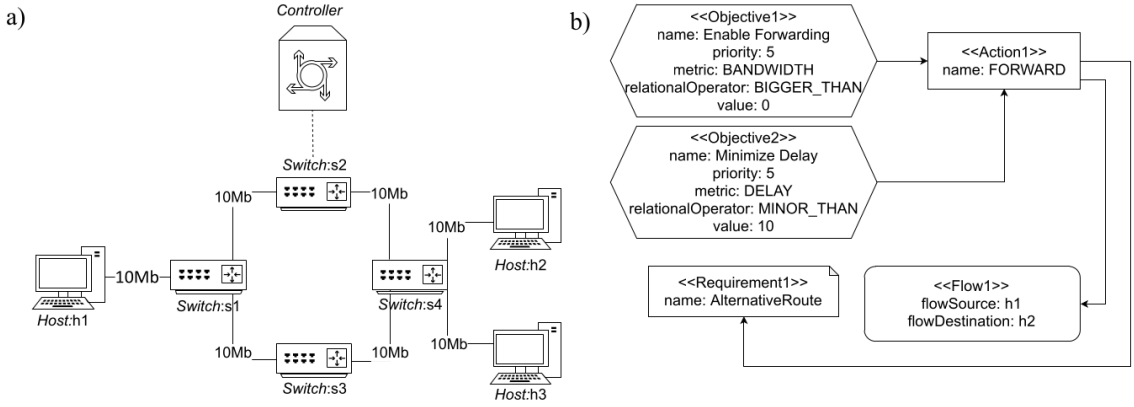


Figure 4. Use case implemented in NML models.

5.1. Qualitative Evaluation

For analyzing this work qualitatively, RFC 7575's design goals were collected and used for verifying which of them could be achieved by the MART-based framework proposal. Although the framework's design considered RFC 7575 as the reference model, design goals could not be achieved due to limiting factors of a MART-based solution or a limited expressiveness of the meta-models and code generation features implemented at the framework. Thus, the qualitative analysis answers the following question:

Which RFC 7575's design goals the proposed MART-based solution could not achieve?

- **Self-protection**: The self-protection property defines that autonomic functions automatically secure themselves against attacks. Currently, the meta-models have no entities for representing such behavior.
- **Coexistence with Traditional Management**: The final ANM system currently supports only intents. The RFC defines that alternative management methods (e.g., command line, SNMP) could coexist with intents. The proposed framework's meta-models need extensions to support this compatibility.
- **Secure by Default**: Although the proposed framework uses TLS in the communication between NML models and IL components, it still cannot assert the membership of all components as required by the RFC.

- **Common Autonomic Networking Infrastructure**: The current framework's architecture is currently designed to integrate autonomic functions (generated from high-level goals) and IL's network components. Achieving this goal using a unique MART solution that generates an ANM system might not be trivial. This work claims that more such infrastructure and its autonomic functions should be in a MART set separated from the ones used to generate the ANM system.

Table 1. Checking the RFC 7575's design goals achieved.

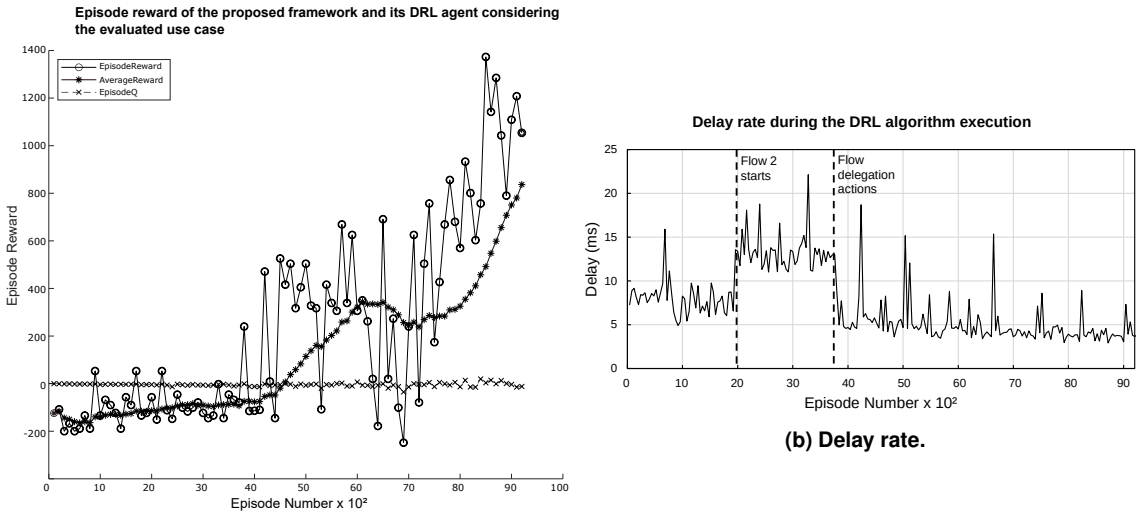
	Achieved?		
RFC 7575 Design Goals	**Yes**	**No**	**Partially**
Self-configuration	X		
Self-healing	X		
Self-optimizing	X		
Self-protection		X	
Coexistence with Traditional Management		X	
Secure by Default		x	
Decentralization and Distribution			X
Simplification of Autonomic Node Northbound Interfaces			X
Abstraction	X		
Autonomic Reporting	X		
Common Autonomic Networking Infrastructure		X	
Independence of Function and Layer	X		
Full Life-Cycle Support	X		

As Table 1 shows, two Design Goals were partially achieved: i) *Decentralization and Distribution*: although the NML models centralize some autonomic logic, the distributes flow tables at the switches receives such logic but they can't adapt themselves in an independent manner; and ii) *Simplification of Autonomic Node Northbound Interfaces*: instead of using low-level and application-centric interfaces, NML enables a high-level interface to define network goals which interact with autonomic nodes. However, representing the autonomic nodes as autonomic *software* nodes is still an open issue.

5.2. Quantitative Evaluation

The testbed used for the following experiment was a commodity hardware with an Intel Core i7-8550U 1.80GHz processor and 16 GB of memory. To verify the proposed MART-based framework's suitability for modeling an ANM system for SDN, consider the previous use case depicted in Figure 4, a simple network topology in which the objectives are: (i) enable simple packet forwarding; and (ii) achieve maximum delay rates of 10ms in the communication between two hosts. The topology consists of three hosts (i.e., h1, h2, and h3) connected by four switches (i.e., s1, s2, s3, and s4) that form two different paths. The communication among the hosts may occur through one of these paths. For the sake of simplicity, all the links have 10Mb of bandwidth.

For implementing the NML layer and enabling creating its models, this experimental evaluation used the Graphical Modeling Framework (GMF) – a well-known framework for modeling and creating model transformations [Kalnins et al. 2007]. GMF supports the generation of a graphical editor – used at runtime in this work – for updating the models and managing the underlying network. This step enables network operators to



(a) Learning rate for the use case.

(b) Delay rate.

Figure 5. Performance results from the DRL algorithm's execution.

use a graphical editor and the NML concrete syntax to model the Objectives, Capability, and Configuration models. These models instances generate a OpenFlow-based Mininet simulation, controlled by a Ryu Controller, with the modeled network topology and its parameters. As discussed in Section 4.1.2, the NML models also instantiate the AL layer's components (i.e., analyzer, reasoner, learner, agent, monitor, and K-Base) as Ryu's applications. For enabling the principles of MART in this implementation, it is necessary verifying the format of GMF models to update the NML models after executing actions on the network. In this use case, when the Reasoner block decides for executing an action, there is a need to update XML files that represent the NML models. The update might consist of adding new entities and actions to these models.

5.2.1. Autonomic action selection impact

For verifying the model transformations and reasoning in the action selection process, this evaluation step introduced congestion traffic from host h3 to host h1. The action selection mechanism tries to find the best action for achieving a state (i.e., Planned Model). Such tries follow the algorithm discussed in Section 4.1.2. In this way, to achieve the `<<Objective1>>`, for example, the algorithm arbitrarily chooses modeled actions and switches to enable flow forwarding between hosts `h1` and `h2`. Figure 5a depicts the learning process for the present use case, its peaks reveal the overhead of each new flow rules set tried and learned by the algorithm. The framework's DRL algorithm was instantiated with an initial learning rate of 0.0001 and ReLu as the activation function for all neurons. The discount factor is set to $\gamma = 0.99$, and the target network is updated every 20 steps. The reward function measured how much the delay rate increased or decreased after each action selection decision. Besides, we clarify that the action space involved in the training phase encompasses all the actions available at our meta-model. Figure 5a also shows the DRL algorithm's behavior when selecting actions for achieving the modeled objectives. In Figure 5b, note the increasing delay when Flow 2 starts and the corresponding episode number in the DRL average reward. When the algorithm receives peaks of rewards, i.e.,

it is learning which action should be executed, the delay also decreases.

6. Conclusions and Final Remarks

This paper presented a distinguished framework considering the RFC 7575's guidelines for enabling complete autonomic management in SDN networks. The proposed framework enables a network operator to define high-level objectives that will be achieved using autonomic actions based on MART concepts. High-level models and network measurements perform the creation and selection of such actions. The framework also provides prediction blocks to define which actions will positively impact the network behavior to satisfy high-level objectives defined by an operator.

The experiments revealed that a MART-based framework achieved success in autonomously choosing actions, reducing the network delay for the simulated scenario by 17%. Thus, the results obtained so far demonstrated the suitability of MART models and RFC 7575's guidelines for providing a framework to perform autonomic management of SDN networks. Besides, both the proposed framework and the experiments put this paper's proposal as one of the first autonomic frameworks for SDN following a standard reference. The present work demonstrated benefits in using the concept of MART in SDN and outlined challenges (e.g., prediction accuracy, monitoring trade-offs, conflicting design goals) that could be barriers (or opportunities) to the generalization of a self-adaptive solution for softwarized networks.

References

Ahmad, I., Namal, S., Ylianttila, M., and Gurtov, A. (2015). Towards software defined cognitive networking. In *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5.

Aßmann, U., Götz, S., Jézéquel, J.-M., Morin, B., and Trapp, M. (2014). A reference architecture and roadmap for models@ run. time systems. In *Models@ run. time*, pages 1–18. Springer.

Ayoubi, S., Limam, N., Salahuddin, M. A., Shahriar, N., Boutaba, R., Estrada-Solano, F., and Caicedo, O. M. (2018). Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165.

Barron, J., Crotty, M., Elahi, E., Riggio, R., Lopez, D. R., and de Leon, M. P. (2016). Towards self-adaptive network management for a recursive network architecture. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 1143–1148. IEEE.

Bega, D., Gramaglia, M., Fiore, M., Banchs, A., and Costa-Perez, X. (2019). Deepcog: Cognitive network management in sliced 5g networks with deep learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 280–288.

Behringer, M. H., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B. E., Jiang, S., and Ciavaglia, L. (2015). Autonomic Networking: Definitions and Design Goals. RFC 7575.

Benayas, F., Carrera, Á., García-Amado, M., and Iglesias, C. A. (2019). A semantic data lake framework for autonomous fault management in sdn environments. *Transactions on Emerging Telecommunications Technologies*, 30(9):e3629.

Bencomo, N. and Blair, G. (2009). Using architecture models to support the generation and operation of component-based adaptive systems. In *Software engineering for self-adaptive systems*, pages 183–200. Springer.

Blair, G., Bencomo, N., and France, R. B. (2009). Models@ run.time. *Computer*, 42(10):22–27.

Eckert, T., Behringer, M. H., and Bjarnason, S. (2020). An Autonomic Control Plane (ACP). Internet-Draft draft-ietf-anima-autonomic-control-plane-30, Internet Engineering Task Force. Work in Progress.

Gelenbe, E. (2013). A software defined self-aware network: The cognitive packet network. In *2013 Ninth International Conference on Semantics, Knowledge and Grids*, pages 1–5.

Gronback, R. C. (2009). *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education.

Jiang, S., Carpenter, B. E., and Behringer, M. H. (2015). General Gap Analysis for Autonomic Networking. RFC 7576.

Kalmbach, P., Zerwas, J., Babarczi, P., Blenk, A., Kellerer, W., and Schmid, S. (2018). Empowering self-driving networks. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 8–14. ACM.

Kalnins, A., Vilitis, O., Celms, E., Kalnina, E., Sostaks, A., and Barzdins, J. (2007). Building tools by model transformations in eclipse. In *Proceedings of DSM*, volume 7, pages 194–207.

Koutsouris, N., Tsagkaris, K., Demestichas, P., Mamatas, L., Clayman, S., and Galis, A. (2013). Managing software-driven networks with a unified management framework. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1084–1085.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Li, H., Que, X., Hu, Y., Xiangyang, G., and Wendong, W. (2013). An autonomic management architecture for sdn-based multi-service network. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 830–835.

Lopes, F. A., Santos, M., Fidalgo, R., and Fernandes, S. (2015). Model-driven networking: A novel approach for sdn applications development. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 770–773.

Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M., Muntés-Mulero, V., Meyer, D., Barkai, S., Hibbett, M. J., et al. (2017). Knowledge-defined networking. *ACM SIGCOMM Computer Communication Review*, 47(3):2–10.

Movahedi, Z., Ayari, M., Langar, R., and Pujolle, G. (2012). A survey of autonomic network architectures and evaluation criteria. *IEEE Communications Surveys & Tutorials*, 14(2):464–490.

Ochoa-Aday, L., Cervelló-Pastor, C., and Fernández-Fernández, A. (2019). Self-healing and sdn: bridging the gap. *Digital Communications and Networks*.

Poulios, G., Tsagkaris, K., Demestichas, P., Tall, A., Altman, Z., and Destré, C. (2014). Autonomics and sdn for self-organizing networks. In *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pages 830–835.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Qi, Q., Wang, W., Gong, X., and Que, X. (2014). A sdn-based network virtualization architecture with autonomie management. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 178–182.

Samaan, N. and Karmouch, A. (2009). Towards autonomic network management: an analysis of current and future research directions. *IEEE Communications Surveys & Tutorials*, 11(3):22–36.

Schaller, S. and Hood, D. (2017). Software defined networking architecture standardization. *Computer Standards & Interfaces*, 54:197–202.

Stamou, A., Dimitriou, N., Kontovasilis, K., and Papavassiliou, S. (2019). Autonomic handover management for heterogeneous networks in a future internet context: A survey. *IEEE Communications Surveys & Tutorials*.

Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR.

Tsagkaris, K., Logothetis, M., Foteinos, V., Poulios, G., Michaloliakos, M., and Demestichas, P. (2015). Customizable autonomic network management: Integrating autonomic network management and software-defined networking. *IEEE Vehicular Technology Magazine*, 10(1):61–68.

Tuncer, D., Charalambides, M., Clayman, S., and Pavlou, G. (2015). Adaptive resource management and control in software defined networks. *IEEE Transactions on Network and Service Management*, 12(1):18–33.

Volpato, F., Silva, M. P. D., Gonçalves, A. L., and Dantas, M. A. R. (2017). An autonomic qoe-aware management architecture for software-defined networking. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 220–225.

Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J., and Granville, L. Z. (2015). Software-defined networking: management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.

Yahia, I. G. B., Bendriss, J., Samba, A., and Dooze, P. (2017). Cognitive 5g networks: Comprehensive operator use cases with machine learning for management operations. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 252–259.

Zhou, T., Xiangyang, G., Hu, Y., Que, X., and Wendong, W. (2013). Pindswitch: A sdn-based protocol-independent autonomic flow processing platform. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 842–847.