Current Autonomic Networking Models and Architectures

Lucas Teixeira, l.teixeira@wustl.edu (A paper written under the guidance of Prof. Raj Jain)

Download

Abstract:

Autonomic networking is an approach for easing the management of large-scale networks with any number of devices. However, the motivation for it stems from the growing complexity of modern networking and the vast number of devices that are now connected to them. This is especially prevalent in cloud networks, but applies to any network of scale with many devices. In this paper, we will discuss in more detail the origins and motivation for autonomic networking, as well as laying out the design goals and desired properties of these systems. Further, we will discuss proposed autonomic systems and architectures, specifically Autonomic Router Redundancy Controller (ARRC), Generic Autonomic Networking Architecture (GANA), and Autonomic Networking Integrated Method and Approach (ANIMA).

Keywords:

Autonomic Systems, Autonomic Networking, Autonomic Computing, Self-Management, Self-Configuration, Self-Healing, Self-Optimization, Self-Protection, Generic Autonomic Networking Architecture, GANA, Autonomic Networking Integrated Method and Approach, ANIMA, Autonomic Router Redundancy Controller, ARRCs

Table of Contents:

- 1 Introduction
- 2 Design Goals
  - 2.1 Abstracting the Infrastructure
  - 2.2 Not Design Goals
- 3 Properties of Self-Management
  - 3.1 Definitions
  - 3.2 Self-Configuration
  - 3.2 Self-Optimization
  - 3.2 Self-Healing and Protection
- 4 Modern Uses and Implications

- 5 Proposed Standards and Systems
    - 5.1 Autonomic Router Redundancy Controller (ARRC)
    - 5.2 Generic Autonomic Networking Architecture (GANA) Model
    - 5.3 Autonomic Networking Integrated Model and Approach (ANIMA)
- 6 Future Considerations
- 7 Summary
- 8 References
- List of Acronyms

1 Introduction

In the earlier days of the internet, it quickly became apparent that a large number of interconnected applications and services presents a lot of overhead for application developers, network managers, and others in the IT sector. In fact, in 2006, approximately 70% of overall budget for the IT sector was labor [EMA06]. Most of this labor is involved in maintaining, configuring, and otherwise managing a wide variety of systems that all need to work together.

The origins of Autonomic Computing stem from a 2001 IBM manifesto from an executive who noticed the exact problem mentioned above coming to fruition – a massive increase in software and network complexity leading to difficult and increased overhead to maintain and configure such systems. Initially, services such as DNS, DHCP, Network Information Service (NIS/NIS+), Lightweight Directory Access Protocol (LDAP), and Service Location Protocol (SLP) were used as methods of host and resource identification over a network, but they all fall short of being able to provide the autonomic configuration and discovery properties desired. This is because they require both technical knowledge and tedious, manual configuration [Melcher04]. Autonomic computing has been described as an, *"approach and blueprint, including a set of products, tools, and services that add self-managing capabilities to Information Technology [IT] systems"* [EMA06]. IBM was largely ahead of the early competition, but for once, the functionality progressed faster than the hype, so these practices have taken longer than usual to get implemented on a wider scale.

The literature on autonomic networking and autonomic systems highlights four desired properties of self-management that form the pillars of an autonomic system. These properties are self-configuration, self-optimization, self-protection, and self-healing. Self-configuration has been discussed since at least 2004 [Melcher04], while the other three properties have been vital to the conversation since at least 2006 [EMA06] and have continued to remain of central importance to the modern day [RFC7575]. These properties will be fully defined and discussed after introducing the design goals of autonomic networking. Overall, the primary goal of autonomic systems is not to remove humans from the process, but rather to shift the burden of maintenance and management from people to technology to alleviate tedious, repetitive tasks.

2 Design Goals

The Internet Research Task Force (IRTF) created RFC 7575, which while not an official standard or specification is an often cited source laying out the foundational information of modern autonomic networking, focusing on high-level design goals rather than implementation specifics. These goals, as presented in the RFC, are as follows:

1. Coexistence with traditional networks and management
2. Security by default
3. Decentralization and distribution
4. Simplification of autonomic node northbound interfaces
5. Abstraction
6. Autonomic reporting
7. Common Autonomic Networking Infrastructure
8. Independence of function and layer
9. Full Life-Cycle Support

[RFC7575]

The reason behind many of these design goals is self-explanatory, such as the need for backward compatibility between the autonomic systems of tomorrow and the existing systems of today. Full life-cycle support is also an intuitive desire – an autonomous network must of course be able to recognize any context it may find itself in and act accordingly – else it is not truly autonomic.

Decentralization provides the same benefits to an autonomic system that it provides to essentially any system – a massive reduction in reliance on centralized systems, which vastly improves not only redundancy, but performance as well. This is due to the fact that different elements of a decentralized system can separately handle demands. For example, a server hosting company may want to have hypervisors in different geographic locations to prevent all of their customers from being bottlenecked at a single, central server. However, despite all of the benefits that decentralization provides, some interaction with centralized systems will likely always be necessary, as discussed below.

2.1 Abstracting the Infrastructure

In this section we will discuss the design goals of abstraction, autonomic reporting, common Autonomic Networking Infrastructure, simplification of autonomic node northbound interfaces, and independence of function and layer. All of these essentially boil down to the ultimate goal of easing direct management of the infrastructure by allowing an administrator to interact with it at common and high levels of abstraction.

Many autonomic systems that exist are individual and proprietary, so it is an important goal to bridge these various interfaces into a common abstraction that can be used across systems. Additionally it is important that the autonomic functions can act on any layer of the networking stack, so as to be able to serve whatever application is at hand if we truly want to ease the burden on administrators. Otherwise, the burden simply shifts to the complicated management of various autonomic systems rather than alleviating the pain point. In line with this, it is a goal to make management of northbound interfaces of autonomic systems as abstract as possible. This is because, as mentioned previously, decentralization is desirable for autonomic systems. Thus abstraction of interaction with centralized information flows improves the management conditions of the system.

Because of the strong desire for abstraction to aid the management of autonomic systems and networks, it is important that we don't forget about the administrator's visibility into said systems. Since the goal is to empower network administrators rather than eliminate them, it is vital that autonomic systems include autonomic reporting to provide the user with insight into what is happening, because they still need the ability to quickly identify and fix issues when necessary. Robust autonomic reporting also lets an administrator see the tangible effects of their abstracted management actions, allowing them to better tune the system. The next section will discuss aspects that are explicitly not design goals of autonomic systems.

2.2 Not Design Goals

As I have mentioned multiple times in this paper, autonomic systems are not meant to entirely uproot the way our networking infrastructure works today. In fact, ITRF RFC 7575 specifies the following as not being design goals of autonomic networking.

1. Eliminating human operators
2. Eliminating emergency fixes and maintenance
3. Eliminating central control

[RFC7575]

This is simply meant to highlight the fact that autonomic networking is meant to improve the reliability and performance of networks by empowering the people who manage them to do so more easily and more efficiently – not to remove them from the picture entirely.

3 Properties of Self-Management

In this section I will discuss the four properties of self-management that are desired for autonomic networks. These properties essentially encompass the design goals presented above.

3.1 Definitions

Here I will define the properties of self-management described prior. Note that the concept of user-defined intent is important in autonomic networking contexts [RFC7575]. User-defined intent is the concept of a user, likely a network administrator, being able to describe to a computer, in human terms, how it would like the network to behave. For example, a human could tell an autonomous system to simply optimize for power availability over resource efficiency, without needing to manually tune the routers, switches, and other hardware that make up the network. This is a poignant concept because without it, an autonomic system cannot do what the user needs it to.

Self-Configuration – Ability to discover new or changing devices on a network and automatically create routes and other configurations necessary to connect said device seamlessly.

Self-Optimization – Ability to modify its own configurations in order to better align its behavior with the user-defined intent for the system

Self-Protection – Ability to protect itself against a potential attack. In a networking sense, this could be dynamically blocking a host suspected to be contributing to Denial-of-Service (DOS) traffic, or modifying firewall rules based on suspected malicious traffic.

Self-Healing – Ability to rectify any problems that may arise, whether from a result of the system's own attempted actions or not.

Before further discussing the properties individually, it is important to introduce a reference model for an autonomic control loop that IBM proposed, as much of the research and literature on autonomic networking draws from it. They proposed the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge), intended to implement the four self-* properties central to autonomic networking [Sliem14]. Self-* properties refer to the four properties defined above, and is used to be consistent with other literature on this topic. As described by Sliem et al., these aspects function as follows. The monitoring function collects and aggregates reporting information to form a record of occurrences, that the analysis function then runs through models to find correlations and meaning. This can be anything from simple pattern recognition to a full machine learning layer, and the complexity of the system will depend on the complexity of the analysis function available, since it is the only way for the network to learn about its context. The planning function will then reconcile that analysis with a policy (likely pre-set, but could be machine-learned) to decide on what actions to take, that the execute function will then handle carrying out.

This is not necessarily the reference model that everyone uses when implementing self-* properties in autonomic networks, but it is certainly a model that makes sense and aligns with the goals of these properties. Additionally, this reference model would allow a network manager to simply tweak the policy used by the planning phase to modify the behavior of the network, lending itself nicely to the goal of abstracted control interfaces.

3.2 Self-Configuration

Autonomous systems need to be able to configure themselves, which, as described above, involves the real-time automatic discovery of and route configuration for devices on a network. This becomes increasingly important as the number of connected devices on a network grows drastically, and manual configuration of thousands or more devices quickly becomes intractable.

3.3 Self-Optimization

Self-optimization in autonomic networks is focused around their ability to adjust their own configuration to align with administrator-defined intent. This is very similar to self-configuration, but self-configuration focuses the on automatic discovery and general configuration, so it works, whereas optimization focuses on making that configuration best align with the network administrator’s intent for the network. For example, a network may configure itself differently when asked to optimize for performance to a specific region than when asked to optimize for availability across an entire country.

3.4 Self-Healing and Protection

Self-healing is vital in an autonomous system because it is almost certain that failures will arise, no matter how much self-optimization or protection is involved. Al-Zawi et al. propose a self-healing approach in which the network attempts to reconfigure themselves in order to recover a fault [Al-Zawi11]. This approach essentially boils down to the detection of an issue and re-triggering of self-configuration. For example, if a node on a network goes down, an autonomic network would discover the changed environment and modify the configurations to re-route traffic, accommodating for the lost node. The team also proposes *monitoring, interpretation, resolution, adaptation/reconfiguration* as the four aspects of self-healing, which notably aligns perfectly with the MAPE-K reference model proposed by IBM for self-* properties, showing that this a model that is largely adopted in the industry.

They also bring up the divide in approaches for self-healing between *integrated mechanisms* and *external adaptation* [Al-Zawi11]. Integrated mechanisms are designed such that they are highly specific to the application they were designed for. As such, the method for adapting to problems (i.e. healing) is only applicable to the application at hand and cannot be reused in other contexts or for other systems. An external adaptation approach, contrarily, consists of monitoring and adapting the system in question from an outside application, meaning that different policies and mechanisms can be dropped in place without having to completely rewrite the mechanism, as you would if it were integrated, and giving external approaches a lot more flexibility. In addition, neural networks turn out to be very helpful in deciding how to reconfigure to handle a lost server or other connection, and specifically a pipeline recursive neural network proved to meet the real-time constraints considered by Al-Zawi et al [Al-Zawi11]. This is analogous to a machine-learned analysis and policy as mentioned in the MAPE-K reference model above.

4 Modern Uses and Implications

Autonomic networking practices in recent years have proved essential in facilitating the networking environments that we have all come to rely on. As mentioned previously, autonomic networking is a rare case of functionality coming before the hype, but now the existence of massive cloud platforms such as Amazon Web Services (AWS), Google Cloud Services, and more require robust autonomic networking to function as the world has come to expect them to. In 2015, *“the ability to automatically provision cloud resources did not perform well under unpredictable conditions such as data volume and data rate”* [Ranjan15], thus performant autonomous networking tools are required in order to dynamically adapt to changing workload requirements without sacrificing quality of service (QoS) to anyone.

This is especially important for big data applications, which are increasingly found everywhere around us; since they have highly unpredictable workloads and data ingress rates. Since large clouds need to regularly re-provision, whether it be to accommodate new customers, change the resources allocated to existing once, and be able to ensure that resources are available when and where they are needed, it is thus critical that our infrastructure can adapt to that unpredictability and automatically maintain itself to provide us with the QoS we desire or expect. On top of this, the self-* capabilities continuing to improve through activities such as the performance analysis discussed above will contribute to a reduction in cost and wasted resources as they are more efficiently allocated and managed. The next section will examine various solutions proposed to do exactly that.

5 Proposed Standards and Systems

Since the inception of autonomic computing in 2001, work has been done to realize the benefits proposed by those early models, largely building on each other. In this section, I will discuss various proposed autonomic systems and how they intend to provide autonomic solutions to networking problems. Though this does not represent all proposed solutions in the autonomic space, it covers the most prominent ones.

5.1 Autonomic Router Redundancy Controller (ARRC)

Autonomic Router Redundancy Controller (ARRC) is an autonomic network management system proposed by Lehocine and Batouche in their 2015 paper "Self-Management in IP networks Using Autonomic Computing." It is based on the Simple Network Management Protocol (SNMP) and the autonomic aspects of ARRC's behavior are heavily inspired by the IBM-proposed MAPE-K reference model mentioned multiple times previously [Lehocine15].

ARRC is motivated by the limitations within traditional First Hop Redundancy Protocols (FHRP). FHRPs are meant to solve issues surrounding hosts only knowing single IP's for the "first hop" to get out of their subnet by providing Layer 3 (generally IP) redundancy. Specifically, traditional FHRPs require that the routers or gateways being used have identical routing tables, which is not necessarily the case in a real network topology. It is quite likely that multiple routers share routes, but also that different routers have different routing destinations, meaning traditional FHRP is not applicable.

"The objective of ARRC is to analyze all the gateways located in the same subnet looking for backup routes to be used when any problem occurs" [Lehocine15]. Essentially, it aims to combine a pre-existing FHRP with static routing to provide multiple master routers rather than the traditional single master router with a backup. The authors chose to implement ARRC on top of an existing FHRP because it allows them to present a general solution that can be used across routers, rather than only working on proprietary or specific equipment. Though protocols such as Hot Standby Router Protocol (HSRP) and Gateway Load Balancing Protocol (GLBP) exist, Virtual Router Redundancy Protocol (VRRP) was chosen as the FHRP to use. This is because it lends itself to a general-purpose solution, as well as

being able to be modified to act as a load balancer, allowing grouping of routers in a way that is necessary for ARRC to achieve its goal of multiple master routers simultaneously.

Because ARRC is based on VRRP, it is worth briefly discussing how VRRP works. In VRRP, a virtual router exists which keeps track of physical routers on the network and their priority. For IPv4, a virtual router with have a MAC of 00-00-5E-00-01-{VRID}, and for IPv6 it's MAC will be 00-00-5E-00-02-{VRID}. The {VRID} at the end is specific to each virtual router on the network, and this address is associated with only one physical router at any given time. Thus, when an ARP (Address Resolution Protocol) for the virtual router's IP is received by the virtual router, it replies with the MAC address currently associated with the master router.

Shown below is a diagram depicting the process of electing a primary router within VRRP, followed by a brief explanation of the process.

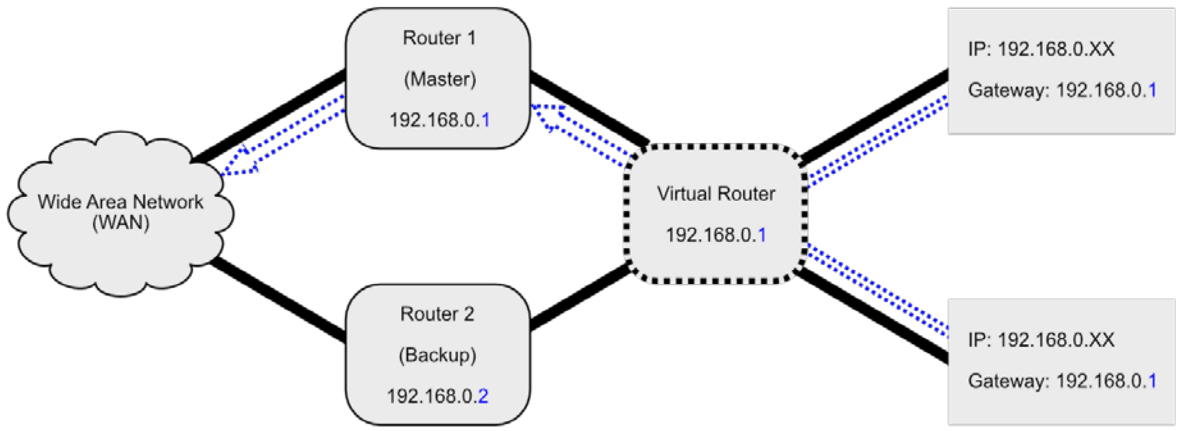

Figure 1. Primary router election in VRRP

Physical routers inter-communicate using multicast IP packets, and backup routers are configured to assume that the master router is dead if a certain amount of time has elapsed without receiving a multicast packet from that physical router. If this occurs, the election process for a new master router begins. At this point, if any physical router with a higher priority than the previous master exists on the network, it will automatically be selected as the new master. Else, the backup routers will send multicast packets which, and the new master is chosen based on the lowest skew time, calculated as

*((256 - priority) * Master_Adver_Interval) / 256* [RFC5798]

The newly elected master will now become the router associated with the virtual router MAC address mentioned above. If multiple backup routers have the same skew time result, the higher IP address number will be chosen to take over the master role.

Now we look at ARRC itself, which is depicted in Figure 2 below and described thereafter.

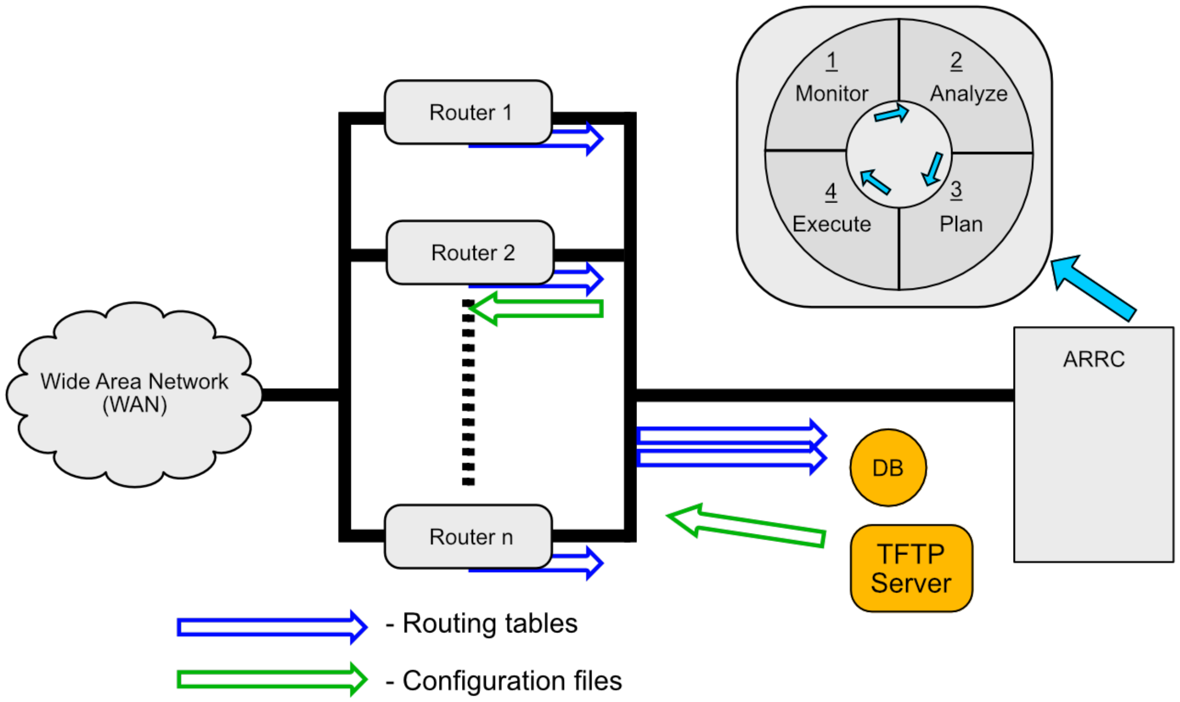

Figure 2. ARRC Management Area

The following is how ARRC's function breaks down into the IBM proposed MAPE-K control loop model that has been discussed. First, ARRC will *monitor* the network using an SnmpWalk request, which it uses to retrieve routing tables from each of the routers it has discovered on a subnet. Then, it will *analyze* that information, identifying duplicated routes both within and outside of each router. After identifying this information, the protocol will *plan* an optimal redundancy configuration plan using identified duplicated routes. Lastly, the protocol will *execute* the plan it designed, copying over changed configuration files to the routers and activating said new configurations. This, in conjunction with the VRRP grouping shown in the diagram above, constitutes the Autonomic Router Redundancy Controller management system's efforts to provide autonomic layer 3 redundancy, encompassing aspects of all self-* properties as it *configures* itself for *optimal* function, achieving *self-healing* and *self-protection* by way of redundant configurations.

5.2 Generic Autonomic Networking Architecture (GANA) Model

In December 2008, Zafeiropoulos et al. proposed Generic Autonomic Networking Architecture (GANA) with the goal of "*identify[ing] autonomic behaviours realised via hierarchical control loops among self-managing elements*" [Zafeiropoulos08]. It is a slightly modified decision, dissemination, discovery, data plane network architecture. In GANA, the decision plane is separated into decision elements (DEs) and managed elements (MEs), where the decision elements are split, from south to north, protocol, function, node, and network elements. DEs further south act as managed elements of northward DEs, as shown in Figure 3 below.

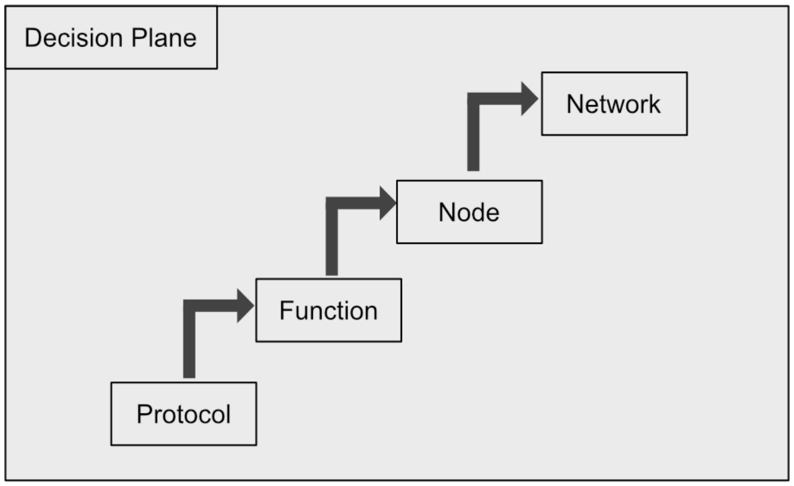

Figure 3. GANA Decision Plane

GANA itself is out of date, but the next approach discussed is heavily inspired by GANA, and thus is worth discussing briefly.

5.3 Autonomic Networking Integrated Model and Approach (ANIMA)

Autonomic Networking Integrated Model and Approach (ANIMA) is a reference model heavily inspired by GANA. It is still at the draft stage of publication within the IETF, and thus is not as much a standard as ARRC and GANA, but is still worth discussing as it is a modern approach to autonomic network solutions. This approach is highlighted by Behringer et al. in their IETF draft "A Reference Model for Autonomic Networking," and is shown at a very high level in Figure 4 below.

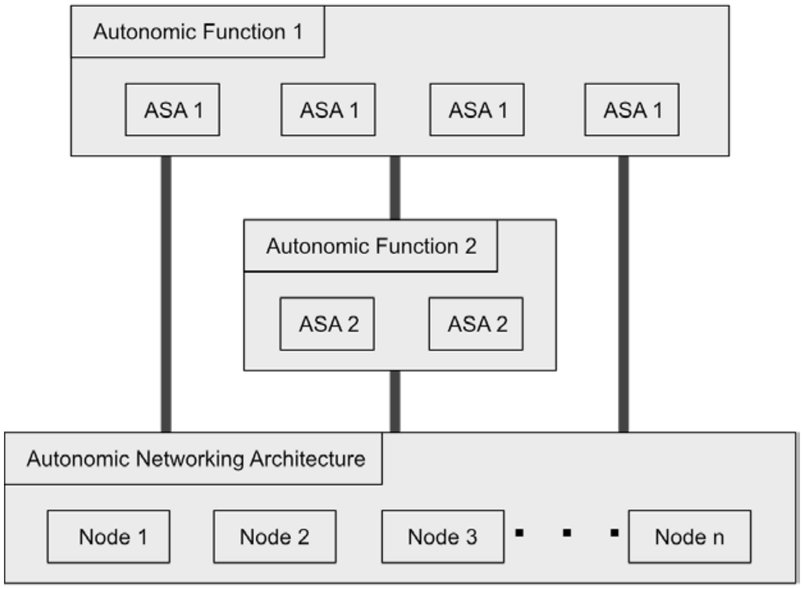

Figure 4. High Level ANIMA Network Layout

In this infrastructure, the series of nodes make up the Autonomic Networking Infrastructure (ANI), providing utilizes like addressing, naming, synchronization, messaging, negotiation, and more, while the Autonomic Service Agents (ASAs) are atomic entities representing autonomic functions of the network [Behringer18]. In order for the ANI to achieve the utilities described above, it is primarily composed of three components: BRSKI (Bootstrapping Remote Secure Key Infrastructure), ACP (Autonomic Control Plane), and GRASP (Generic Autonomic Signaling Protocol) [Long17]. BRSKI allows unconfigured devices (known as the Pledge) to retrieve initial configurations securely (e.g. secure certificate) from an already set up device (known as the Registrar) automatically without a human in the loop. This allows for new devices to be detected and set up autonomously. The ACP, or autonomic control plane, is a self-encompassed control plane for managing communication between the ASAs that make up the autonomic function layers of the network. GRASP is a protocol consisting of four primary mechanisms: discovery, negotiation, synchronization, and flooding, which basically allow for quick operation across a network as it is a protocol solely focused on quick signaling between devices. Of note is the fact that this protocol comes with no baked in security, and is thus often run within the ACP, or must rely on some other external measure to ensure secure communication.

While ARRC is strictly a layer 3 protocol, ANIMA is a more holistic approach as it addresses the needs for autonomic networking across all parts of the stack, as nothing is application specific and the autonomic functions and ASAs can be tailored to the application at hand. Further, while GANA is also a general architecture, ANIMA improves on DE-ME hierarchical nature of GANA.

6 Future Considerations

Moving forward with autonomic networking, there are some key aspects that should be considered in future reference models and specifications to fulfill some of the less-address desires of autonomicity. While the approaches discussed in this paper do mention humans in the process, such as being able to force a new master router in ARRC, they focused largely on systems managing themselves. However, as discussed, the goal of autonomic networks is not to remove humans from the loop, rather to empower them. Thus, I think it is important that future autonomic architectures consider how an abstract interface can be provided to a network manager such that they can easily oversee the system as a whole. Further, while intent was discussed, primarily in terms of self-optimization, it is important that it is factored into the design of the aforementioned abstract interfaces because it allows administrators to manage their networks without manual configuration.

As it stands, it seems manual configuration is needed in the network management process more than desired for true autonomic systems. Deeper integration of intent based management is an important direction for research to follow. Lastly, the discussed approaches mention little in the way of aggregated autonomic reporting, which is important in conjunction with abstract management tools to provide administrators with information to guide their decision making and should be focused on in future research.

7 Summary

In this paper, we have considered autonomic computing from its inception in a 2001 IBM executive's manifesto to ANIMA, a modern and currently proposed reference model. We have discussed the IBM-proposed Monitor, Analyze, Plan, Execute - Knowledge (MAPE-K) reference model for autonomic systems and implementation of self-* properties, which has been widely accepted by many in the autonomic computing and networking research community. We discussed Autonomic Router Redundancy Controller (ARRC) as a network management system providing the benefits of autonomic computing in layer 3, specifically IP, networking. ARRC is important because it shows that a general-purpose autonomic system can be implemented on top of existing infrastructure in a way that aligns with the design goals of autonomic networking, specifically coexistence with traditional networks and decentralization. We also briefly discussed Generic Autonomic Networking Architecture (GANA) to demonstrate the inspiration for Autonomic Networking Integrated Model and Approach (ANIMA), which while still in the draft phase, has been updated many times in the past two to three years. It defines a somewhat high level, comprehensive autonomic architecture that can be applied to a variety of situations, aligning with the design goals laid out in the introduction of this paper, most specifically decentralization, and independence of function and layer.

8 References

[RFC7575] M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, Cisco Systems, B. Carpenter, Univ. of Au
Ciavaglia, A. Lucent, "Autonomic Networking: Definitions and Design Goals," Internet Rese
https://tools.ietf.org/html/rfc7575

[EMA06] Enterprise Management Associates "Practical Autonomic Computing: Roadmap to Self Mana
https://web.archive.org/web/20070316012547/http://www-03.ibm.com/autonomic/pdfs/AC_P

[Melcher04] B. Melcher, B. Mitchell, "Towards an Autonomic Framework: Self-Configuring Network Ser
Technology Journal Vol. 8 Issue 4, November 17, 2004,
https://arquivo.pt/wayback/20080224033918/http://download.intel.com/technology/itj/2004/v

[Sliem14] M. Sliem, N. Salmi, M. Ioualalen, "Towards Reliability and Performance Prediction of Auton
2014 International Conference on Cloud and Autonomic Computing, 2014, https://ieeexplore.

[Al-Zawi11] M. Al-Zawi, D. Al-Jumeily, A. Hussain, A. Taleb-Bendiab, "Autonomic Computing: Applica
systems Engineering Conference, 2011, https://ieeexplore.ieee.org/document/6150010

[Ranjan15] R. Ranjan, L. Wang, A. Zomaya, D. Georgakopoulos, X. Sun, G. Wang, "Recent Advances in
Clouds," IEEE, IEEE Transactions on Cloud Computing Vol. 3 pp. 101-104, 2015,
https://www.computer.org/csdl/journal/cc/2015/02/07118807/13rRUygBw1I

[Lehocine15] M. Lehocine, M. Batouche, "Self-management in IP networks using autonomic computing," I
Network of the Future (NOF), 2015, https://ieeexplore.ieee.org/document/7119785

[RFC5798] S. Nadas, Ed. Ericsson, "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and
2010, https://tools.ietf.org/html/rfc5798

[Zafeiropoulos08] A. Zafeiropoulos, A. Liakopoulos, A. Davy, R. Chaparadza, "Monitoring within an Autonomi

	Framework," International Conference on Service Oriented Computing, 2008, https://www.researchgate.net/publication/221050335_Monitoring_within_an_Autonomic_Ne
[Behringer18]	M. Behringer, B. Carpenter, Univ. of Auckland, Futurewei Technologies Inc., L. Ciavaglia, N Reference Model for Autonomic Networking," IETF, 2018, https://tools.ietf.org/html/draft-ie
[Long17]	X. Long, X. Gong, X. Que, W. Wang, B. Liu, S. Jiang, N. Kong, "Autonomic Networking: A Internet Computing Vol. 21 Issue. 5, 2017, https://ieeexplore.ieee.org/document/8039304

List of Acronyms

ACP	Autonomic Control Plane
ANI	Autonomic Networking Infrastructure
ANIMA	Autonomic Networking Integrated Model and Approach
ARP	Address Resolution Protocol
ARRC	Autonomic Router Redundancy Controller
AWS	Amazon Web Services
BRSKI	Bootstrapping Remote Secure Key Infrastructure
DE	Decision Element
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DOS	Denial-of-Service
GANA	Generic Autonomic Networking Architecture
GLBP	Gateway Load Balancing Protocol
GRASP	Generic Autonomic Signaling Protocol
HSRP	Hot Standby Router Protocol
IRTF	Internet Research Task Force
LDAP	Lightweight Directory Access Protocol
MAPE-K	Monitor, Analyze, Plan, Execute - Knowledge
ME	Managed Element
NIS / NIS+	Network Information Service (+)
SLP	Service Location Protocol
SNMP	Simple Network Management Protocol
QoS	Quality of Service
VRRP	Virtual Router Redundancy Protocol

Last Modified: December 10, 2019
This and other papers on latest advances in computer networking are available on line at
http://www.cse.wustl.edu/~jain/cse570-19/index.html
Back to Raj Jain's Home Page