A Literature Review of Network Advancements for Future Adaptive Systems

1st Trevon Williams, UNC Charlotte Qualifer

***Abstract*—The management of information technology system architecture and its design considerations is continually challenged by evolving system security, service resource consumption, development velocity and increasing complexity. Software defined networking (SDN), orchestration applications and light weight virtual machines have served as building blocks to enable robust and agile networked computer systems. The driving motivation for developments converge on moonshots like autonomic computing systems research. SDN is a key component for enabling autonomic computing environments but there is much work to be done to enable security and reliability in interconnected network applications. In this paper, we will outline and analyze the core ideas inspired by a future vision of secure and resilient systems architectures.**

***Index Terms*—Cyber Threat Intelligence (CTI) Software Defined Networking (SDN) Network Control Plane Autonomic Computing Proactive Computing**

I. INTRODUCTION

In this paper, we view referenced technologies from a systems perspective to describe why our pursued research field requires further advancement. We define our system perspective to contain the points of integration of computer network services, applications, and appliances traditionally described as IT managed infrastructure. Systems literature contains many strategies and goals for guiding the future direction of computer systems. Computing system architecture research is composed of complementary efforts and strategies to advance the state of computer maintenance and security.

There are two notable ideas presented in the early 2000's, proactive computing [1] and autonomic computing [2]. Each system design methodology attempts to address management and complexity problems as computer systems scale in size and interconnectivity, both with inherit security considerations. Scale can be considered as the growing number of networked devices and services contained in a high interconnected computer system. Today, cloud computing is used to deliver services without worrying about complete server maintenance but this level of abstraction does not completely solve the issue of growing complexity in managing computing systems [2].

As these problems are discussed, proactive and autonomic computing solutions hypothesize the idea of allowing the system to secure itself. Both systems in their purest form, operate without human intervention. Within each approach, high-level objectives are defined to abstract components to hypothesize how to reach either moonshot. Each strategy contains ideas from the fields of distributed systems, ubiquitous computing, spatial computing, fault tolerant computing and symbiotic computing research with other ideas derived from totally complementary fields such as economics or biology.

In this paper we explore the trajectory of autonomic and proactive computing to better understand the role of SDN. We then analyze distributed SDN solutions and complimentary protocols to evaluate the state of SDN's impact on the goals of autonomic and proactive computing and we conclude by describing areas for future work that advance autonomic solutions.

II. BACKGROUND

Self-managing computer systems is a vision sought after by IT researchers. Man-Computer symbiosis [3] presents a vision of computing in which computers and human operators both play a part in general decision-making. In 1998, Burgess [4] stressed the need for self-healing in modern computing systems, due to their inherent fragilities and demanding requirement for operator maintenance. Intel's proactive computing [1] advanced this idea by calling for a general restructure of computer science research to remove human operators out of the control loop. In 2001, DARPA funded researchers presented one of the first, complete autonomic architecture's SARA [5], which focused on the inherent security features of autonomic computing. IBM later released a manifesto [2] detailing the IT industries complexity crisis and their own researched architecture [6]. Industry leaders created varying research initiatives, signaling a commitment of research goals and product solutions. A few of these solutions include:

- IBM's Autonomic computing [2]
- Microsoft's Dynamic Systems Initiative [7]
- Hewlett Packard's Adaptive Enterprise [8]
- Intel's Proactive Computing [1]

The proposed "self-managing" infrastructure collective of initiatives has continued to be driven by industry, academia and government entities. We have taken the time to include notable examples of reference architectures and initiative to provide a general background on the described field.

Computer Immunology

Computer Immunology [4] provides motivation for autonomic system advancement by juxtaposing computers and organisms that live in hostile environments. This point illustrates the need for organisms to adapt and heal in order to survive. Burgess then introduces cfengine [4], a robust provisioning engine, that automates and manages services based on high-level abstractions and semantics that describe the managed system. This notable technology is arguably the grandfather to the software that is powering DevOps today.

Proactive Computing

Proactive computing [1] also known as human-supervised computing, is a modern vision that seeks to synergize symbiosis focused computer science research similarly to the work of Licklider [3]. Proactive computing builds an argument for reducing how much influence people have in the computer system's control loop. This is supported by the fact that human error is the number one cause for reduced system integrity.

Autonomic Computing

Autonomic computing is an information technology moonshot that requires the collaboration of interdisciplinary and disjoint STEM fields; it combines the research goals of fault-tolerant computing, self-organizing network design and artificial intelligence into a single regiment that attempts to mimic the self-management facilities found in biological systems. Self-management in IT systems promises to reduce system complexity and maintenance. In order for a system to be self-managing, it must display an ability to self-configure, self-heal, self-optimize and self-protect as defined in IBM's vision of Autonomic Computing [6]. White organizes autonomic computing into high-level abstractions of network and computing components within a framework to provide a scale for assessing the maturity of implementations and evaluating innovative momentum. Each objective is then broken down further into non-exclusive attributes.

Autonomic computing requires non-trivial feature integration in the network layer. Autonomic elements [6] are responsible for managing their own and other managed element behaviors. This requires a new paradigm of computer networking managment technologies and interacting components.

Active Networks (AN)

Active networks [9] is an early area of programmable network research that helped shape the vision of proactive computing [1]. Active networks attempts to address the need of future network innovation. The fundamental idea behind active networks is to allow network switches and routers to perform computations on network data packets in the hopes of increasing the velocity of network technology innovation. Active networks is an active field of research and shares ideas with SDN implementations to improve aspects of computer networking environments.

SARA: SARA [5], survivable autonomic response architecture, proposes an architecture that utilizes the autonomic architecture for defense. Like IBM's 'Vision of Autonomic Computing' [2], SARA provides a well defined architecture. This is accomplished by abstracting components into well-defined elements to better describe integration guidance and to provide nomenclature for describing specific functionality and components.

Software Defined Networking

SDN has been defined as the programmatic management of a network. SDN has commonly been achieved by utilizing openflow [10] or another similar SDN enabling protocol that allows an SDN controller to communicate with supporting switches. The SDN controller can then compute and communicate network flow rules to the network switches based on the network state or enforced policies. SDN controllers can then utilize the power to program switches and its knowledge of global state, to optimize and steer traffic from a central or distributed placement configuration.

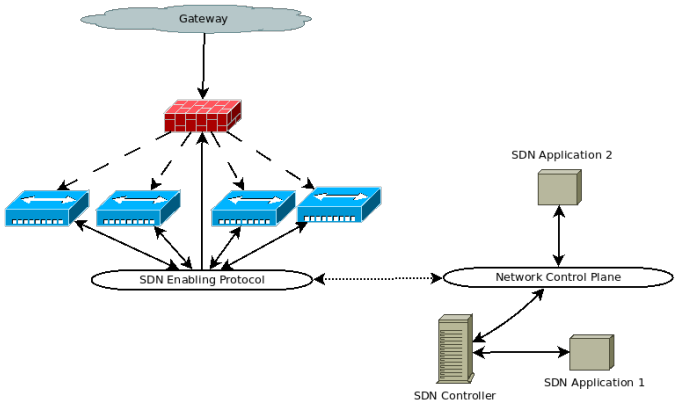

Fig. 1. Map of Software Defined Networking components

SDN is appealing because it can power robust network applications by replacing the need for proprietary router functions with open protocols and APIs, driving innovation. We find that the field of SDN research is composed of sub components and complementary fields of work that are mistakenly thought to be synonymous. This can be visualized using the provided graphic 1. These components are popularly compartmentalized as:

1) Southbound protocols: A southbound protocol is the language of communication between the SDN controller and managed network switches.

2) Network Control Plane: The network control plane is managed by the SDN controller and contains the global view of the network depending on the implementation.

3) Network Applications: Network applications include the points of integration between the network control plane, the northbound application interface and a 3rd party network application.

Openflow [10] is the most popular, open source SDN enabling protocol that allows SDN controllers to communicate with supporting physical and virtual hardware. Openflow has been incredibly successful and seen large industry adoption in both network and virtual appliances. We have provided a table I of other SDN enabling protocols and features.

Network Control Plane: The network control plane is provides a protocol stack that allows network applications to manage network resources through the control of an SDN controller. Mature network control planes are also referred to as a network operating system. The control plane is specific to the controller implementation as each SDN controller typically supports different Northbound API's. Network applications have a varying level of control depending on the placement of SDN controllers, the specific SDN controller deployed and the supported northbound API's. Currently, there is work going into proposing and standardizing northbound protocols. In ta-

Protocol	Features
ForCES [11]	• Forwarding and Control Element Separation • Standardized by the IETF • Extensible • Uses a logical functional Block for flow mapping • Forwarding and control element seperation
PCE [12]	• Path Computation Element • Updates via edge router • Uses segment routing and RSVP • PCE server used as SDN controller
SRv6 [13]	• Segment Routing • Minimizes configuration overhead • Coexists with other protocols • Fast configuration
VXLAN [14]	• Encapsultaion method for overlay networks (MAC-in-UDP) • Scale and complexity issues • High CPU overhead • Tunnels layer 2 network over layer 3
Cisco's ACI [15]	• Application centric infrastructure • Proprietary and tightly coupled with Cisco tooling • Supports OpFlex and RESTful API • All VLAN's are not available in all parts of network • Leverages VXLAN [14]
LISP [16]	• Flexible and supports hybrib deployments • Moderate performance impact of flow mapping • Allows policy enforcement acros domains (DC-to-DC) • Distributed flow mapping database
NETCONF [17]	• Close to native functionality of switch • Reduced complexity • Enhanced performance
OpFlex [18]	• Enhanced Scalability • Distributes network complexity • Utilizes RPC
VMware NSX [19]	• Proprietary and coupled with VMware offerings • Enables distributed switch capabilities (DLR) • Leverages VXLAN [14]

TABLE I
SOUTHBOUND PROTOCOLS

ble II we have included some northbound API and descriptions to highlight current work. Flow installation describes how the API's are designed to issue new flow rules for distribution to the managed network.

OpenFlow [10] is a logically centralized protocol that enables the programming of openflow switches. NOX [24], a highly referenced early controller, was developed to support openflow through a single instance to manage an entire network. This enabled researchers to present use cases on top of an open platform for the future development. The centralized controller placement configuration is still prevalent today as a cluster of centralized controllers but considerations for more resilent and optimized configurations has lead to the development and deployment of distributed SDN controllers. This has allowed for greater network capacities and redudancy in openflow enabled environments.

Distributed Controllers

Distributed controller placement decreases latency between managed network zones and solves scalability and single point of failure issues found in single controller implementations. Distributed SDN controller research is composed of two design methodologies that correlates to the physical placement of each controller. The flat model places controllers on a horizontal plane, relative to each other to allow for the network to be managed in partitions. Visually, the horizontal plane is like a flat network. The hierarchical model on the other handle utilizes a tree structure placement to allow the top level controller to have a global view of the entire network, without having to manage local network partitions. Distributed controllers research effectively highlights the inherit features of mature network operating platforms and northbound network applications integration.

4) Flat-model: The flat level model allows for controllers to share a global network-wide view and is implemented differently amongst SDN controllers for tailored use cases. The common goal of the flat level model is to localize network management into partitions and to allow controllers to manage even partitions of allocated segments of the network. This implementation reduces network congestion, decreases control message latency from controller to switches and improves network failure resiliency. In some implementations, it is found that this model does not scale well, as partitioned network zones face the same problem introduced originally in the single, centralized controller. This occurs when controllers cannot delegate managed network resources to other controllers in the network. Some examples of flat-model implementing controllers include the following.

- Onix [25] introduces a distributed control platform that enables network and datacenter operators to develop management applications through a simple programming interface to enable robust SDN environments. ONIX displays many popular SDN implementation strategies such as a decoupled control and data plane, network API, and centralized network view to simplify control plane application implementations and create a unique case for SDN that makes a case for mature deployment opportunities.

Protocol	Flow Installation	Description
REST API	Depends on implementation	General REST API support of controllers describes programmic interface to send commands to controller and allows for implementation specific features.
Frenetic [20]	Reactive	Enables high-level programming abstractions for network control.
Pyretic [21]	Reactive and Proactive	An upgraded Frentic used for transparent policy handling.
Merlin [22]	Reactive	Delegates the management of policies to tenants.
Kinetic [23]	Reactive and Proactive	Enables the network to be managed dynamically through its domain specfic language

TABLE II
NORTHBOUND PROTOCOLS

- B4 [26] is an SDN powered WAN environment developed by Google to address specific network challenges for routing application data. B4 employes a custom version of B4 and protocol stack to bridge traditional routing and forwarding protocols with openflow to support an application specific traffic engineering in a large WAN environment. Notably, Google's approach to tether forwarding protocols with SDN allows Google to deploy SDN into an environment that also includes legacy networking systems.
- Onos [27] is an open source distributed network operating system that builds on the ideas of ONIX for the specific reason of enabling open communication about implementation details. Onos creates a distributed controller implementation with an open northbound API, based on Floodlight, to enable network application development in parallel with the improvement of distributed network management. Onos is logically centralized and supports cluster implementations or a flat-model placement of SDN controllers.
- HyperFlow [28] focuses on building a physically distributed control plane on top of the NOX controller and was released the same year as Onix. Hyperflow takes a different approach by allowing distributed controllers to compute network paths locally. Then the controller communicates the local network graphs via a publish and subscribing messaging system built on WheelFS to other controllers.
- Disco's [29] approach is tailored for wide-area networks (WAN) and overlay network solutions. The researchers remove the strong requirement of a consistent centralized network-wide view, as seen in Onix, Onos, and HyperFlow. This is accomplished by domain controllers exchanging relevant topology changes in the network using the advanced messaging query protocol (AMQP).

5) Hierarchical model: The hierarchical model on the other hand delegates control to SDN controllers in vertical partitions. This enables a central controller to delegate access and network partition view to lower level controllers connected by networked branched nodes. This model provides advantages that include improved scalability and performance.

- Kandoo [30] deploys a distributed hierarchical model of SDN controllers. Kandoo utilizes two layers of controllers to allow the local management and execution of applications to reduce resource exhaustive events on the control plane. This work is an alternative to the approach taken in DevoFlow [31] and B4 [26]. The alternative approach requires introducing new functionality directly into switches to suppress network control events seen in DevoFlow [31] and DIFANE [32].
- PANE [33] utilizes a distributed hierarchical model to advance the state of the art participatory networks which allow user applications to access network information to guide network action. PANE addresses major shortcomings in participator networks by presenting an approach to resolve conflicts between participant requests and how to decompose the control and visibility of a network safely.
- Orion's [34] approach towards a hierarchical model is motivated by improving the scalability of the control plane and to address the path stretch problem introduced by single root hierarchical placement.

Network Operating System

The research effort behind developing a network operating system; a unified interface for network applications, is motivated by creating a network-wide platform that enables development and serves many use cases. The network operating system provides a programming interface to directly interact with the network control plane and handles translating high-level abstractions into actionable low-level intents. Each network operating system must provide at least one of the following principles:

- Provide applications with the ability to use high-level abstractions.
- System must be able to control interactions between network applications.

This reduces the need for network operators to program network functions requiring low-level targeting attributes. Works in literature that show either one or both of those principles include the following:

- Maestro [35] focuses on orchestrating the network control applications that control network behavior.
- Onix [25], employs proprietary code to allow for centralized management of network assets efficiently.
- Onos [27], composed of open source's tools to enable open development of the network operating system.

Network Applications

SDN applications use the power of SDN controllers to implement novel solutions in a network environment. This is achieved by directly integrating into the controller or utilizing a northbound protocol. We find that most of the applications integrate directly into a specific controller to display a proof of

concept. These solutions include intelligent engines for traffic steering, anomaly detection or the introduction of autonomic facilities. Currently, there is no standard northbound API interface.

Middlebox Integrations

A middlebox appliance is an application that sits in a network and provides extra network functionality using the network control plane. Examples of such an appliance include firewalls or intrusion detection systems. Advancements of SDN focused middlebox architecture are used to distribute and automate the programmability of network applications that can intercept flows in the data plane utilizing the SDN controller. Solutions that fall into this category include BPFabric [36] and CoMB [35].

Containers

Lastly, a fundamental introduction of operating system (OS) containers is important. Containers enable system administrators to deploy applications, services, or user environments into a lightweight, isolated environment that can be managed seamlessly by orchestration technology. Containers are very exciting when analyzing this space because containers provide the ability to quickly and succinctly orchestrate environments with minimal overhead. Engine environments like Kubernetes or Apache Mesos thrive by enabling administrators to deploy, manage and maintain containerized applications in a distributed computing environment. These environments provide cutting edge self-healing and self-orchestrating properties that are extremely powerful when leveraged effectively.

In the next section, we will explore other related work that attempts to further incorporate resilient, autonomic, or security features using SDN.

III. Related Work

Our work in systems research has led us to explore the novel implementations of participatory computer networks, distributed SDN controllers and a unified interface for network applications to embolden deeper integration of computer resource and network management. From a system's perspective, a network architecture composed of distributed SDN controllers and a unified interface for network applications, enables further autonomic behavior integration that bridges the gap of expanding the dynamic management of appliances and utilities. The unified interface for network applications can take many shapes as seen in early sections. Without restricting our perspective to one approach towards a mature unified network interface, we view each attempt in literature that bridges the networking and computer resource gap equally. In this section, we focus on describing the ongoing work that integrates novel mechanisms into a unified networking environment.

Control Plane Advancements

An Autonomic Control Plane (ACP) [37] is a well-defined IETF draft for defining the organization and components of an ACP specifically for professionally managed networks.

Security-Focused Solutions

Related security-focused solutions specifically focus on leveraging SDN to introduce more resilient or efficient solutions to current problems. Included solutions introduce security while also considering advancements found in SDN literature.

- Tennison [38] is an ambitious network security framework that utilizes the power of SDN to introduce remediation and monitoring capabilities. Tennison leverages a distributed placement of controllers to reduce DDoS attacks introduced by native system traffic in traditional controller configuration.
- SDN Security Plane [39] introduces a third network plane, the security plane, to display an SDN platform capable of managing network security. The security plane described is responsible for forwarding data packets between SDN switches and rule and data exchange between managed switches and the controller. The security plane can also exchange security related data between third-party agents on the switch and software.
- A Dynamic Composition Mechanism of Security Service Chaining (SSC) Oriented to SDN/NFV-Enabled networks [40] focuses on addressing the dynamic SSC composition problem with an SDN solution. This work seeks to address the nature of middlebox solutions due to their static functionality. Dynamic service chaining addresses this by pairing SDN and NFV solutions to engineer new service traffic to software middlebox solutions after being introduced in computer networks.
- Software Defined Networking for Security (SDN4S) [41] focuses on reducing the time it takes to respond to incidents in enterprise networks coupling automated countermeasures with SDN. This is accomplished by creating incident-specific response countermeasures that are triggered when alerts are received.

Middle-box Architectures

- SIMPLE [42] is a security middlebox management scheme for SDN environments that generates flows for services to security instances. Unfortunately, this scheme breaks down in certain cases of network orchestration.
- BPFabric [36] is a language independent protocol for network operators. It provides the ability to program and monitor the data plane by leveraging eBPF. This solution allows for a high performing network environment with an improved application interface for data plane applications like intrusion detection systems and stateful firewalls.

eBPF Optimizations

- Hybrid kernel-user space virtual network functions (VNF) [43] use eBPF to optimize and deploy network functions commonly found in network hardware in the linux kernel running on network switches. This enables researches to deploy robust packet sniffing and load balancers in their experiment. We found documented limitations of employing network function virtualization using eBPF, but these limitations are shown to be avoided [44].

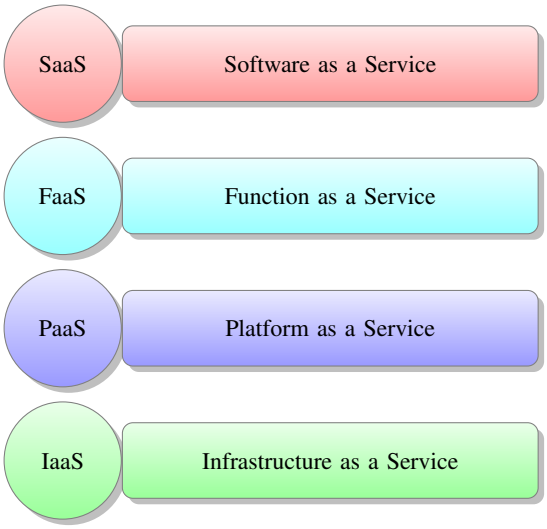

Fig. 2. Diagram of Service Models

IV. Analysis of Literature

Information technology research is driven to embody the features described in both proactive and autonomic computing's vision for computing systems. We find this body of work exciting and important due to the implications of advancements and the difficult nature of presenting a solution that allows for general security improvement over the deployment's lifecycle. In this analysis we provide our understanding of how each work proposes a solution for a sub problem in this meta field and why it is important.

The body of work we analyze incorporates strategies for integrating security in distributed SDN environments. More mature solutions will couple service orchestration solutions through API's with security monitoring software, and SDN optimizations. Current work includes solutions that focus specifically on introducing security frameworks, improving network appliance integration and robust engineering of traffic and network service traffic. Approaches to introduce fine grain traffic engineering and network control exists in varying levels of abstractions. Frameworks for developing and managing different levels of deployment abstraction described in the diagram Figure 2. We believe this is important to note as we analyze tooling and framework.

Scope of Area

Some technologies in our related work integrate into IaaS solutions or provide their own service for integration. Apache Mesos [45] , for example, provides an API to distributed resources management and provides well-defined control language to allow for the management of both virtual machines and containers in a distributed environment. In its purest form, Apache Mesos is almost synonymous to the end goal of a mature network operating system that manages network wide resources and not just service resources. We find in SDN literature there have been significant steps to achieving this with the introduction of traffic engineering, consensus algorithms for leader environments, rich control languages, and intent handling optimizations for network events.

There is overlap in our discussed area of work and current software solutions that we did not reference earlier. Cilium [46] is a container network management technology for docker [47] containers and Kubernetes [48] clusters that have developed to greatly expand the idea of SDN in production environments. Cilium is native in Kubernetes [48] environments. Technology related to Cilium includes Weave net [49] and Flannel [50]. These technologies bring powerful capabilities of network and service management to a PaaS, Kubernetes. Our work attempts to provide solutions that can be leveraged in more generic environments.

Integrating Network Management

Secure service and network management is a ripe area for research. The advancements in the distributed SDN control planes address network congestion, network latency, and introduce powerful traffic engineering. We perceive when coupled with the advancements of distributed controller deployments and management applications, truly autonomic and proactive environments will emerge. We believe the next step of research includes improving northbound protocol integrations with robust prototype platforms to improve points of integration between network and service control structures.

Optimizing Middleware

Middleware network optimizations span the fields of active networking and SDN. Notable software advancements currently seek to integrate eBPF programs because of their inherit speed and lightweight footprint. eBPF allows compiled programs to be directly executed in the linux kernel. This allows programs to directly manage hardware interfaces outside of user space. BPFabric [36] develops an openflow like protocol to program the extended data plane functionality. This is not found in openflow flows currrent standard. BPFabric improves the effeciency of middlebox application's management by dynamically steering traffic directly to appliances as new devices are introduced into the network.

Notable Contributions

BPFabric [36] highlights an impressive implementation of managing the network data plane using eBPF. Currently, we find no other work that provides a general implementation to manage the data plane and control plane programmatically.

Areas of Improvement

BPFabric [36] introduces data plane functionality in a protocol disjoint from openflow. This introduces a significant area of work that requires the expansion of openflow's specification and its ability to bring new functionality into environments already containing support for eBPF programs and middleware optimizations. Proposing an update to the openflow specifically would also introduce dataplane programmability into SDN networks.

V. Future Direction

Our immediate research effort will focus on improving the points of integration of service and network orchestration technologies. This will enable us to develop on related work that attempts to address security service chaining, middlebox solutions, and northbound API integration within ONOS [27]. In this section we have broken down what we feel would be our immediate focus into components that build off ideas of current work.

A. A Distribute SDN Testbench

A Distributed SDN testbench would enable the testing and development of northbound API integrations. This would enable us to provide insight into shortcomings and to provide guidance for pain points found in current. We would like to extend the current feature set of containernet [51] to include LXD [52] containers to provide more general testing of both container and virtual machines in management experiments. This would include testing pertaining to container live migration traffic engineering and dynamic service placement.

B. eBPF Engine

Openflow enables the concept of SDN by abstracting the control and data planes of computer networks while providing an API for centralized or distributed controllers to manage the control plane. Currently, openflow does not provide a well defined interface for controllers to abstract high-level controls through northbound API's or directly to network applications. The motivation for this can be seen in BPFabric [36]. In this subsection, we outline future work that we seek to improve upon in this field to enable more efficient network application interfaces.

Service Steering: Security service chaining was referenced in our related work as the ability to improve middlebox software's ability to perform in a computer network. We would like to bring support for eBPF packet filter in supported switches to the openflow protocol. This will enable feature adoption and testing in switches that already include support for openflow and are running linux 3.18+. The goal state of this implementation would be able to deploy and update eBPF tables in a managed switch.

Service Security: After the development of the eBPF engine is completed and enabled in our openflow environment, we would be able to introduce powerful security implementations in our environment. This will include filtering our malformed packets, specifically targeted at our controllers.

Dynamic eBPF Ruleset: To conclude our work, we seeks to development a controller application that dynamically generates and updates eBPF rulesets for use in northbound applications interfaces. This will synergize our outlined research guidance to create a proof of concept for upgradable network application and configuration, outlined in the vision of AN's [9]. We perserve this will also allow for the development of rich AI applications for quickly responding to network activity without disrupting traffic by dropping network flows.

VI. Conclusion

Autonomic computing is the vision of self-configuring, self-optimizing, and self-healing computing systems capable of defending themselves. On the journey to developing autonomic systems, we find that it is important to develop and points of integration between network management systems and services that communicate over API. Mature API's will enable systems to be managed by more intelligent engines and human operators. We will focus on improving current network operating systems capabilities and API functionality. We believe that extending openflow to include support for eBPF to manage data plane functions will increase the use of northbound API's, driving improvements and development.

References

[1] D. Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43, no. 5, pp. 43–50, May 2000. [Online]. Available: https://dl.acm.org/doi/10.1145/332833.332837

[2] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003, number: 1. [Online]. Available: http://ieeexplore.ieee.org/document/1160055/

[3] J. C. R. Licklider, "Man-Computer Symbiosis," *IRE Transactions on Human Factors in Electronics*, vol. HFE-1, no. 1, pp. 4–11, Mar. 1960. [Online]. Available: http://ieeexplore.ieee.org/document/4503259/

[4] M. Burgess and O. College, "Computer Immunology," p. 17, 1998.

[5] S. Lewandowski, D. Van Hook, G. O'Leary, J. Haines, and L. Rossey, "SARA: Survivable Autonomic Response Architecture," in *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, vol. 1. Anaheim, CA, USA: IEEE Comput. Soc, 2001, pp. 77–88. [Online]. Available: http://ieeexplore.ieee.org/document/932194/

[6] S. White, J. Hanson, I. Whalley, D. Chess, and J. Kephart, "An architectural approach to autonomic computing," in *International Conference on Autonomic Computing, 2004. Proceedings.* New York, NY, USA: IEEE, 2004, pp. 2–9. [Online]. Available: http://ieeexplore.ieee.org/document/1301340/

[7] Microsoft Corporation, "Microsoft Announces Dynamic Systems Initiative - Stories," Mar. 2003. [Online]. Available: https://news.microsoft.com/2003/03/18/microsoft-announces-dynamic-systems-initiative/

[8] HP Corporation, "HP Unveils Adaptive Enterprise Strategy to Help Businesses Manage Change and Get More from Their IT Investments," May 2003. [Online]. Available: https://www8.hp.com/us/en/hp-news/press-release.html?id=172215

[9] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, Jan. 1997. [Online]. Available: http://ieeexplore.ieee.org/document/568214/

[10] Open Networking Foundations, "OpenFlow Switch Specification," Dec. 2009. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

[11] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework," RFC Editor, Tech. Rep. RFC3746, Apr. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3746

[12] JP. Vasseur and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)," RFC Editor, Tech. Rep. RFC5440, Mar. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5440

[13] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, "SDN Architecture and Southbound APIs for IPv6 Segment Routing Enabled Wide Area Networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1378–1392, Dec. 2018, arXiv: 1810.06008. [Online]. Available: http://arxiv.org/abs/1810.06008

[14] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," IETF, Tech. Rep. RFC 7348, Aug. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7348

[15] Cisco Systems, Inc., "Cisco IT ACI Design," p. 29, 2020.

[16] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The Locator/ID Separation Protocol (LISP)," RFC Editor, Tech. Rep. RFC6830, Jan. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6830

[17] R. Enns, *NETCONF Configuration Protocol*, ser. Request for Comments. RFC Editor, Dec. 2006, no. 4741, published: RFC 4741. [Online]. Available: https://rfc-editor.org/rfc/rfc4741.txt

[18] M. Smith, R. E. Adams, M. Dvorkin, Y. Laribi, V. Pandey, P. Garg, and N. Weidenbacher, "OpFlex Control Protocol," Internet Engineering Task Force, Internet-Draft draft-smith-opflex-03, Apr. 2016, backup Publisher: Internet Engineering Task Force Num Pages: 24. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-smith-opflex-03

[19] "What is VMware NSX - Network Security Virtualization Platform." [Online]. Available: https://www.vmware.com/products/nsx.html

[20] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: a network programming language," p. 13.

[21] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing Software-Defined Networks," p. 13.

[22] R. Soulé, S. Basu, R. Kleinberg, E. G. Sirer, and N. Foster, "Managing the network with Merlin," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, ser. HotNets-XII. New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 1–7. [Online]. Available: https://doi.org/10.1145/2535771.2535792

[23] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, "Kinetic: Verifiable Dynamic Network Control," p. 15.

[24] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, Jul. 2008, number: 3. [Online]. Available: https://dl.acm.org/doi/10.1145/1384609.1384625

[25] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks," p. 14.

[26] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: experience with a globally-deployed software defined wan," p. 12.

[27] P. Berde, W. Snow, G. Parulkar, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, and P. Radoslavov, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN '14*. Chicago, Illinois, USA: ACM Press, 2014, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2620728.2620744

[28] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," p. 6.

[29] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–4, iSSN: 2374-9709.

[30] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*. Helsinki, Finland: ACM Press, 2012, p. 19. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2342441.2342446

[31] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-Performance Networks," p. 12.

[32] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," p. 12.

[33] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: an API for application control of SDNs," p. 12.

[34] Y. Fu, J. Bi, K. Gao, Z. Chen, J. Wu, and B. Hao, "Orion: A Hybrid Hierarchical Control Plane of Software-Defined Networking for Large-Scale Networks," p. 8.

[35] Z. Cai, A. L. Cox, F. Dinu, T. Ng, and J. Zheng, "The preliminary design and implementation of the maestro network control platform," Tech. Rep., 2008.

[36] S. Jouet and D. P. Pezaros, "BPFabric: Data Plane Programmability for Software Defined Networks," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. Beijing, China: IEEE, May 2017, pp. 38–48. [Online]. Available: http://ieeexplore.ieee.org/document/7966898/

[37] T. Eckert, M. H. Behringer, and S. Bjarnason, "An Autonomic Control Plane (ACP)," Internet Engineering Task Force, Internet-Draft draft-ietf-anima-autonomic-control-plane-30, Oct. 2020, issue: draft-ietf-anima-autonomic-control-plane-30 Backup Publisher: Internet Engineering Task Force Num Pages: 180. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-anima-autonomic-control-plane-30

[38] L. Fawcett, S. Scott-Hayward, M. Broadbent, A. Wright, and N. Race, "Tennison: A Distributed SDN Framework for Scalable Network Security," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2805–2818, Dec. 2018, number: 12. [Online]. Available: https://ieeexplore.ieee.org/document/8468216/

[39] A. Hussein, I. H. Elhajj, A. Chehab, and A. Kayssi, "SDN Security Plane: An Architecture for Resilient Security Services," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*. Berlin, Germany: IEEE, Apr. 2016, pp. 54–59. [Online]. Available: http://ieeexplore.ieee.org/document/7527815/

[40] Y. Liu, Y. Lu, W. Qiao, and X. Chen, "A Dynamic Composition Mechanism of Security Service Chaining Oriented to SDN/NFV-Enabled Networks," *IEEE Access*, vol. 6, pp. 53 918–53 929, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8466784/

[41] T. Koulouris, M. C. Mont, and S. Arnell, "SDN4S: Software defined networking for security," *Hewlett Packard Labs, Palo Alto, CA, USA, Tech. Rep*, 2017.

[42] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," p. 12.

[43] N. Van Tu, K. Ko, and J. W.-K. Hong, "Architecture for building hybrid kernel-user space virtual network functions," in *2017 13th International Conference on Network and Service Management (CNSM)*. Tokyo: IEEE, Nov. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8256051/

[44] S. Miano, M. Bertrone, F. Risso, M. Tumolo, and M. V. Bernal, "Creating Complex Network Services with eBPF: Experience and Lessons Learned," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*. Bucharest, Romania: IEEE, Jun. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8850758/

[45] "Apache Mesos." [Online]. Available: http://mesos.apache.org/

[46] "Welcome to Cilium's documentation! — Cilium 1.9.4 documentation." [Online]. Available: https://docs.cilium.io/en/v1.9/

[47] "Docker Documentation," Mar. 2021. [Online]. Available: https://docs.docker.com/

[48] "Kubernetes Documentation." [Online]. Available: https://kubernetes.io/docs/home/

[49] "Weave Net: Network Containers Across Environments | Weaveworks." [Online]. Available: https://www.weave.works

[50] "flannel-io/flannel," Mar. 2021, original-date: 2014-07-10T17:45:29Z. [Online]. Available: https://github.com/flannel-io/flannel

[51] M. Peuster, H. Karl, and S. v. Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2016, pp. 148–153.

[52] "Home - LXD - system container manager." [Online]. Available: https://lxd.readthedocs.io/en/latest/