Unified Network Policy Management Across Stack

Loic Niragire

TIM-7010 V3: Computer Networks & Mobile Computing

Northcentral University

January 15, 2022

Table of Contents

BACKGROUND 3

NETWORK LAYERS AND FUNCTIONALITIES 3

DATA CENTER NETWORKS MANAGEMENT 4

SERVICE MESH 5

PROBLEM STATEMENT 6

GOAL 9

RELEVANCE AND SIGNIFICANCE 10

LITERATURE REVIEW 11

APPROACH 12

REFERENCES 13

Background

In this proposal I present a context-aware network policy management solution to network policy management in a microservice deployment into the cloud. To minimize the problem scope, I assume that the services are running on Kubernetes and focus on improving deployment models that have proven popular. I make the argument that by augmenting traditional SDN with HTTP context details, a more reliable and secure network can be achieved. This work should be of particular interest to software delivery and network administration teams that are regularly concerned with security and performance measures of their systems. I first present an overview of the network layers and their functionalities, as this will be referenced in subsequent sections. Following that, I discuss data center network management and how the service mesh continues to play an integral part in this area.

Network layers and functionalities

Network protocols govern the exchange of data between computers on a network. The Open Systems Interconnection (OSI) defines seven such layers. L1 layer, the physical layer, encompasses the hardware elements such as cables, network interface cards, and repeaters. L2 layer, known as the data link, on the other hand, is concerned with data pockets exchange between nodes. The network layer, L3, handles routing of network messages by assigning logical addresses to devices on the network. Common protocols at this layer include IP, IPX, and TCP. The transport layer, or L4, ensures efficient and reliable transportation of data packets – popular protocols at this layer include TCP/UDP. Whereas L5, session layer, is concerned with managing connections between applications. L6, the presentation layer, establishes context within the applications. Finally, L7, the application layer encompasses the interface between users and applications. Generally, we use L4-L7 Network services to reference network functions such as load balancing, web application firewalls, service discovery, etc.

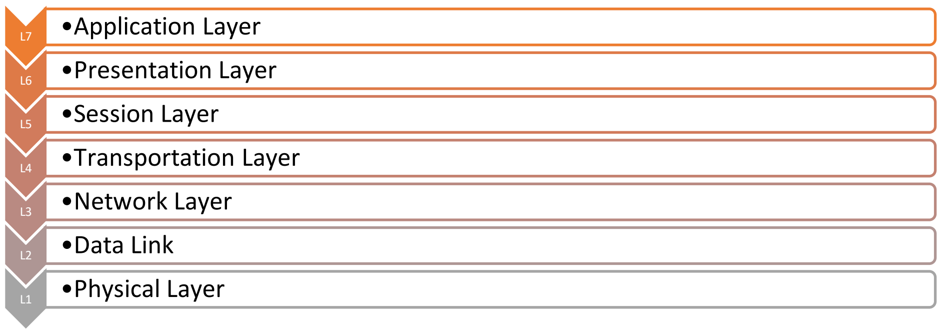


Figure 1 OSI Network Layers

Data Center Networks Management

Cloud adoption over the last decade has further increased management complexities of data center networks. Resulting in the introduction of a dedicated communication overlay at the application layer, the service mesh (Antichi & Rétvári, 2020). In this architecture, each service running on the same server is attached to its own individual sidecar proxy which acts as an L4-L7 network service to manage service-to-service communication. Meanwhile, each sidecar proxy connects to a virtual switch, or vswitch, to access the physical network infrastructure. Communication between services running on different servers is achieved via a data center fabric component.

In modern cloud infrastructure, cloud orchestration frameworks such as OpenStack, Andromeda, or AccelNet, handle the management of the fabric component. Whereas network layer functionalities are provided by containers or orchestration systems like Kubernetes, Mesos, or Docker. The service mesh frameworks such as Istio, linkerd, or Consul, run on top of this to deliver enhanced L7 functionality (Antichi & Rétvári, 2020).

Although this model has endured positive feedback, it poses unresolved security and performance bottlenecks. Multiple services, or applications, in a data center can concurrently utilize the same physical infrastructure whereas the only isolation provided is through network virtual switches, which are still not well understood. For instance, Vswitches violate least privilege principle as they typically execute on server with full kernel privilege (Thimmaraju, Hermak, Rétvári, & Schmid, 2019).

Like SDN, the service mesh operates by assuming a central authority over the global L7 network policies through sidecar proxies. The only exception being that service mesh functions at higher network layers in comparison to SDN, which is typically constrained to L2 and L3 layers. Therefore, the service mesh can be adopted as an L7-SDN by using sidecar proxies as the programmable L4-L7 data plane while leaving the control plane to a service mesh framework such as Istio. (Antichi & Rétvári, 2020).

Service Mesh

A service mesh is a dedicated infrastructure layer for transparently adding observability, traffic management, security, and reliability features to distributed microservices deployed on Kubernetes. It offers L4-L7 features such as load balancing, service-to-service authentication and monitoring to services deployed on the same cluster. This is accomplished by utilizing sidecar proxies attached to each service to intercept all network traffic. The proxies interact with a control plane to enforce network policies without code change within the application. The control plane converts network policies into configurations for related sidecars and deploys them via the data plane management APIs (Li, Wang, & Chen, 2020).

The mesh is regarded as a new network layer within the network stack between application and transport. An App built using the service mesh communicates via service mesh

API instead of directly opening direct open transport connections to remote services. Note that the standard OSI model depicts the session and presentation layers, layer 5 and 6 respectively, exactly between the application and transport layer as well. Although the service mesh shares similar functionalities with these standard layers, it provides additional capabilities such as security policy enforcement and performance-aware load balancing (Ashok, Godfrey, & Mittal, 2021).

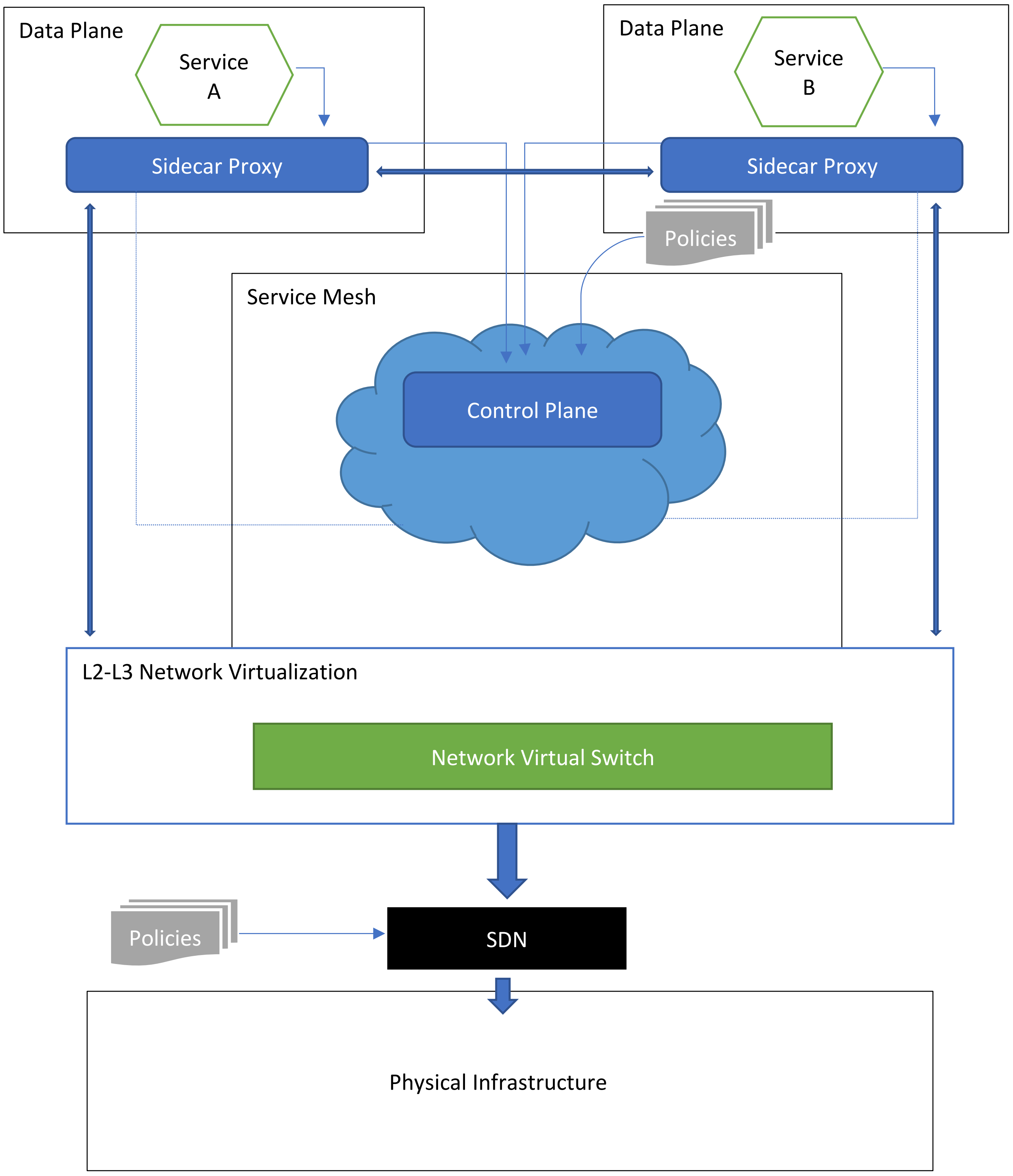
Problem Statement

The presence of multiple control and data planes in data centers increases complexity to data center networks management. On one hand, SDN has proven reliable at managing large networks through its control plane by constraining its functionality to L2-L3 network level. Meanwhile, the adoption of service mesh as an L7 level network function is rapidly showing great promise at controlling service-to-service communication through sidecar proxies.

Enforcing L4-L7 communication policies at the L2-L3 network virtualization layer requires a service proxy to handle communication to and from the application. This functionality is implemented in the sidecar proxy and increases network complexity. For a local package exchange between two applications, three separate phases with three full round trips and six context switches are required. (Antichi & Rétvári, 2020). Could SDN be extended to cover L2 to L7 stack and thereby alleviating sidecar inefficiencies introduced by service mesh?

This challenge is amplified by the fact that visibility into applications running on a network is only possible at the L4-L7 layers of the network. Hence why network policies and functions required to make data centers more efficient and secure are executed at those layers. In other words, by the time an application on the network reaches the network layer, L3, no context data is available onwards. Thus, making SDN policy enforcement oblivious of the application.

Network administrators are affected by this problem as they route traffic, through SDN controller, without adequate insight into the application. As different applications required various network services, knowing application network requirements would facilitate packet transmission within the given constraints (Posch, Rainer, & Hellwagner, 2017), thus leading to high consumer satisfaction.


Data Plane
Service A
Sidecar Proxy
Data Plane
Service B
Sidecar Proxy
Policies
Service Mesh
Control Plane
L2-L3 Network Virtualization
Network Virtual Switch
Policies
SDN
Physical Infrastructure

Goal

There is a desire for a single framework that could manage network policies throughout the entire network stack without incurring a heavy performance and security risk. Currently, the de factor approach has been to utilize SDN at the lower layers of the protocol while relying on the service mesh at the upper network layers. Although these two approaches share a core similarity of splitting the management into a control and data plane. Their interface results in increased security and performance risk.

Relevance and Significance

A full-stack SDN would facilitate the ability to exert fine-grained control over any layer in the network stack in a protocol-agnostic manner (Antichi & Rétvári, 2020). Additionally, by extending this to the application layer, we obtain an HTTP context-aware network management framework. One of the challenges to overcome for this functionality is due to the fact that context information is distributed among network nodes.

The challenge imposed by network context distribution is similar to the integration of services, or functions, distributed over the internet. Which is a problem that the Service-Oriented architecture (SOA) set out the address. SOA introduces the concept of a service delivery platform (SDP) which is responsible for the creation, execution, delivery, and control of services (Lee & Kang, 2012). Moreover, it incorporates the ability to compose one or more distributed component services by orchestrating what is known as a composite service. Thus, allowing service providers a more flexible delivery platform.

Whereas the policies enforced by the service mesh's control plane benefit from HTTP context awareness, the same is not true at the L2-L3 SDN layer. By extending this context to the SDN layer, administrator can leverage additionally acquired insight into the application and offer better QoS by including application constraints into their routing considerations. This would also have an impact on security concerns as administrators only have to maintain a single source of routing policies across the entire network stack.

Literature Review

Luo in (Luo, Wu, Li, Guo, & Pei, 2015) proposes a context-aware service composition scheme for SDN that effectively utilizes the network resources by inserting a service layer between the control layer and the application layer. The authors consider the dynamic traffic forwarding problem to be a dynamic multi-constrained path optimization problem whereby the capability and cost of a service change frequently. By adopting multiple linear regression analysis method, they successfully made the scheme context-aware.

Wenning in (Wenning, Timm-Giel, & Görg, 2009) gives an overview of context-aware routing approaches known in literature such as CAR (Context-Aware Routing), SCAR (Sensor Context-Aware Routing), and NWAUF (Normalized Weighted Additive Utility Function). Although these approaches are all tailored to a specific use case, the authors propose a generic context-aware routing framework that provides more flexibility using the Multi-Criteria Context-based Decision function (MCCD).

An overview of the next generation service overlay networks (NGSON) is provided in Lee (Lee & Kang, 2012). The overlay layer is positioned between services and transport networks to control the interactions among distributed services. More importantly, this architecture continues to serve as the basis for development of more advanced context-aware routing scheme in existence today.

The case for a full-stack SDN is argued by Antichi in (Antichi & Rétvári, 2020) where they investigated the similarities between SDN and the current service mesh architecture. By using Istio service mesh framework, they concluded that it fits nicely into the SDN paradigm. They also noted that the current state of the art approach of splitting the stack into practically two SDN incurs considerable management complexity.

Approach

The study will focus on the case of HTTP web services deployed on Istio framework running on Kubernetes. This closely resembles the current industry state-of-the-art microservice deployment model and has the potential for a wide adoption. The first hurdle to overcome is the minimization of context switching across network layers for service-to-service communication. This is a major point of concern within the software development community as they continue to embrace the microservice journey. It is worth noting that his study will not investigate beyond service communication across pods.

The second hurdle is related to the simplification of network policies. This is a consequence of unifying policies of various concerns. Policies enforced by the Service Mesh, for instance, are strictly regarding service-to-service relationships. Whereby a mesh operator controls which services are allowed to communicate. Contrary, L2-L3 SDN is only dealing with packets routing through the physical infrastructure without any knowledge of the services involved nor the application nature. In other words, mesh operation as software delivery concern, whereas L2-L3 SDN policy management is a network administration matter.

Complexity management is the biggest threat we envision in this work. This is particularly true in our case as we must remain mindful of security and performance concerns. Various metrics points will be collected and accessed around those two areas to minimize the impact. Additionally, ease of usage is a concern that we plan on tracking as part of complexity assessment. Our approach is to build on what is already proven successful from both the software delivery and the network administration, such as the service mesh and SDN, while improving management of policies across the entire network stack within a single framework.

References

Antichi, G., & Rétvári, G. (2020, March 3). Full-stack SDN: The Next Big Challenge? *Symposium on SDN Research*, pp. 48-54.

Ashok, S., Godfrey, B. P., & Mittal, R. (2021, November 10). Leveraging Service Mesh as a New Network Layer. *HotNets*, pp. 229-236.

Lee, S.-I., & Kang, S.-G. (2012). NGSON: Features, State of the Art, and Realization. *IEEE Communication Magazine*, 54-61.

Li, X., Wang, X., & Chen, Y. (2020, August 10). MeshScope: A Bottom-up Approach for Configuration Inspection in Service Mesh. *SIGCOMM*.

Luo, S., Wu, J., Li, J., Guo, L., & Pei, B. (2015). Context-Aware Traffic Forwarding Service for Applications in SDN. *2015 IEEE International Conference on Smart City*, pp. 557-561.

Posch, D., Rainer, B., & Hellwagner, H. (2017, January). Towards a Context-Aware Forwarding Plane in Named Data Networking Supporting QoS. *ACM SIGCOMM Vol 47*, pp. 6-14.

Thimmaraju, K., Hermak, S., Rétvári, G., & Schmid, S. (2019). MTS: Brining Multi-Tenancy To Virtual Networking. *USENIX Annual Technical Conference*, pp. 521-536.

Wenning, B.-L., Timm-Giel, A., & Görg, C. (2009). A Generic Framework for Context-Aware Routing and its Implementation in Wireless Sensor Networks. *Mobilkommunikation - Technologien und Anwendungen, Vorträge der 14*, pp. 53-58.