Software-Defined Networking: An annotated bibliography

Loic Niragire

TIM-7010 Computer Networks & Mobile Computing

Dr. Phillip Davis

In this annotated bibliography, I aim to provide an overview of various ongoing research work on the subject of Software-Defined Networks. The central idea behind SDN is the separation of the control plane from the data plane, whereas a software application provides the control and management plane, which results in a centralized path forwarding decision scheme as opposed to the traditional distribution forwarding algorithm found in conventional networks. In other words, a software application handles all forwarding paths decisions implemented by switches. Consequently, this new paradigm gives network operators the ability to efficiently implement network policies across large networks without managing switches independently. The control plane has three components; a control program, a virtualization layer, and a network operating system. The control program operates on an abstract network view graph generated by the virtualization layer. Meanwhile, the network operating system is responsible for communicating to physical network switches. In essence, the SDN paradigm provides the ability to virtualize a physical network.

Mizrahi, T., Saat, E., & Moses, Y. (2015). Timed Consistent Network Updates in Software-Defined Networks. IEEE/ACM Transactions on Networking, 3412-3425.

The authors in this paper explore accurate time synchronization as a practical tool to implement consistent network updates associated with duplicate configurations. SDN routine network policy and configuration updates present a consistency and scalability challenge. Care must be taken to assert that all updates are applied to all devices and in the desired sequence. Update procedures must also scale to the size of the network. The developed method of time-triggered updates allows the controller to apply updates in parallel and thus significantly reducing the update duration. Traditionally, the two commonly used methods to achieve consistent and scalable network updates are “Ordered Updates” and “Two-Phase Updates”, both of which are inherently sequential. The main advantage of the proposed method in this paper is that large network updates can be executed faster and reliably.

Nam, J., Jo, H., Kim, Y., Porras, P., Yegneswaran, V., & Shin, S. (2019). Operator-Defined Reconfigurable Network OS for Software-Defined Networks. IEEE/ACM Transactions on Networking, 1206-1219.

In this paper the authors examine the issue of security, performance, and availability from a Network Operating System perspective. They provide a simplified event-driven integration framework to enhance the Network Operating System through composable modules, thus, facilitating custom controllers with diverse feature combinations. The primary motivation of this work stems from the challenge of managing multiple SDN sub-networks with disparate requirements and policies, where each controller has its own programming architecture and software APIs. Additionally, existing controllers specialize in different aspects such as throughput, scalability, and others. Hence the need for a composable SDN controller to meet a wide range of operational requirements found in many diverse networks, perhaps more

importantly, the ability for the controller to alter functional behavior when network operating conditions change.

Tseng, S.-H., Tang, A., & Choudhury, G. L. (2019). Routing Stability in Hybrid Software-Defined Networks. IEEE/ACM Transactions on networking, 790-804.

The authors explore algorithmic solutions for traffic routing in a hybrid software-defined network. Their algorithm seeks to minimize conflicts between the centralized controller and the local routers' decisions. Given that not all devices support full SDN functionality, the authors aim to obtain a stable routing pattern that leverages both centralized and distributed routings. This dual network control requires a routing consistency between the centralized and distributed controllers. Consequently, such a network is deemed stable. The following three algorithms are devised for establishing a stable routing pattern in a hybrid SDN environment: Global Optimization Kernel Algorithm (GLO), Greedy Kernel Algorithm (GRE), and Local Search Kernel Algorithm (LOC). Moreover, the paper addresses routing decision making under incomplete or inconsistent information. This is particularly critical to the centralized controller given its reliance on data plane.

Xu, H., Huang, H., Chen, S., Zhao, G., & Huang, L. (2018). Achieving High Scalability Through Hybrid Switching in Software-Defined Networking. IEEE/ACM Transactions on Networking, 618-632.

The authors in this paper present a mechanism to integrate traditional and SDN switching to achieve scalability and optimal performance. Traditional switching can achieve scalability through aggregate routing and decentralized control. While on the other hand, SDN relies on per-flow routing and centralized control to achieve optimal network performance and policy-based management but faces scalability challenges. Per-flow routing results in a significant

computational overhead at the controller. Commonly used techniques to manage these issues include a destination-based aggregate routing and decentralization of routing control to avoid a single point of failure. However, the rise of centralized routing control with SDN overshadowed scalability concerns in favor of global policy enforcement and optimal traffic management. Hybrid switching matches received packets both against the traditional routing table and the SDN forwarding table with precedence given to the forwarding table. The authors also discuss path selection strategies for dealing with new entries not found in forwarding tables, such as Traditional Path First (TPF) and SDN Path First (SPF).

Yan, B., Xu, Y., & Chao, J. H. (2016). Adaptive Wildcard Rule Cache Management for Software-Defined Networks. IEEE/ACM Transactions on Networking, 962-975.

The authors in this paper present a rule caching scheme that reduces control channel load and improves flow table efficiency. They achieve this by partitioning the field space into buckets, which are then cached along with all associated rules. Since OpenFlow-enabled switches store network policy rules in flow tables. In effort to guarantee correct packet matching, traditional rule caching schemes often result in unnecessary flow table bloat as they over-cache. This is because rules that depend on a requested rule also need to be cached. These rules can either be stored in an exact-match format or using a wildcard format. Although the wildcard approach offers a flexible storage, wildcard rule matching becomes an expensive operation as the number of rules increases. The authors propose a mechanism to cache wildcard rules on demand, thus, simultaneously improving flow table efficiency and reducing cache misses.

Zhang, P., Wu, H., Zhang, D., & Li, Q. (2018). Verifying Rule Enforcement in Software Defined Networks With REV. IEEE/ACM Transactions on Networking, 917-929.

The authors in this paper address an SDN security challenge associated with the decoupling of the control plane from the data plane. How does the SDN assures that rules issued by the control plane are correctly enforced by the data plane? The decoupling of these layers presents an opening for an adversary to prevent the data plane from enforcing these rules. Therefore, they propose a new security primitive to defend SDNs against rule modification attacks by enabling the controller to verify that rules are correctly being enforced by switches. They also present a mechanism to reduce the switch-to-controller traffic through compressing MAC. Rule Enforcement Verification efficiently ensures that all rules in the network are verified through a heuristic algorithm that selects a small number of flows for verification.

The control plane is responsible for compiling network policies such as routing and access control into rules. Through standard control channels, like OpenFlow, the control plane installs these rules into network switches, located within the data plane. It is these switches that are responsible for enforcing rules from the control plane to realize the network policies. The resulting security challenge is that network switches are subject to compromises from an adversary who may alter original rules. Additionally, many existing network verification tools only assert that a given set of rules are correctly enforced by network switches, but do not match enforced rules at the switch level from original rules at the control plane. The authors in this propose a mechanism to allow the control plane to securely check whether issues rules are correctly enforced by switches.