# Demo: EdgeVPN.io: Open-source Virtual Private Network for Seamless Edge Computing with Kubernetes

Renato Figueiredo
Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
Email: renato@acis.ufl.edu

Kensworth Subratie
Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611
Email: kcratie@ufl.edu

***Abstract*—Edge and fog computing encompass a variety of technologies that are poised to enable new applications across the Internet that support data capture, storage, processing, and communication across the networking continuum. These environments pose new challenges to the design and implementation of networks - as membership can be dynamic and devices are heterogeneous, widely distributed geographically, and in proximity to end-users, as is the case with mobile and Internet-of-Things (IoT) devices. We present a demonstration of EdgeVPN.io (Evio for short), an open-source programmable, software-defined network that addresses challenges in the deployment of virtual networks spanning distributed edge and cloud resources, in particular highlighting its use in support of the Kubernetes container orchestration middleware. The demo highlights a deployment of unmodified Kubernetes middleware across a virtual cluster comprising virtual machines deployed both in cloud providers, and in distinct networks at the edge - where all nodes are assigned private IP addresses and subject to different NAT (Network Address Translation) middleboxes, connected through an Evio virtual network. The demo includes an overview of the configuration of Kubernetes and Evio nodes and the deployment of Docker-based container pods, highlighting the seamless connectivity for TCP/IP applications deployed on the pods.**

Keywords: VPN, SDN, Kubernetes, edge, containers

## I. Introduction

This demonstration will highlight EdgeVPN.io [1] (Evio for short), an open-source VPN for edge computing. Edge computing systems require applications to be deployed across distributed resources – often across multiple edge providers, and including cloud computing back-ends. A contemporary approach to orchestrate and manage the deployment of applications leverages the use of virtualization technologies – virtual machines, containers. In particular, Kubernetes [2] is a platform for orchestration of Docker-based [3] container "pods" that is widely used in data centers. Leveraging the features of Kubernetes to deploy pods across edge resources opens opportunities for the reuse of a wealth of existing software to manage edge computing deployments.

However, Kubernetes is often deployed in cloud data centers, where the assumption of the cluster's network model [4] – where all nodes are in the same address space – is satisfied. In such a data center environment, there are no NAT middleboxes between Kubernetes hosts/pods. However, when deploying workloads across multiple edge networks, this is seldom the case - nodes may be served by different providers, and be assigned private, NAT'ed addresses.

Enter Evio: it provides a foundational virtual network layer that exposes the networking model that Kubernetes requires [4], essentially presenting Kubernetes daemons with an environment that is logically the same as they would encounter in a data center. Thus, Evio allows deployments across multiple disparate cloud/edge networks, where private addresses, NATs and firewalls are not uncommon - without any changes. In addition to NAT traversal, the Evio overlay virtual network is scalable, resilient, and self-configuring in response to nodes joining and leaving the cluster, greatly simplifying the management of the cluster's network. In combination with Kubernetes, Evio allows deployment of containerized, micro-service workloads spanning cloud and edge resources to support "fog" computing for processing near IoT sensors/actuators.

Specifically, Evio addresses the technical challenges in creating virtual networks that self-organize into scalable, self-organizing, resilient overlay topologies, with built-in support for interconnecting resources that are constrained by distributed NAT/firewall middleboxes. In doing so, Evio can significantly lower the barriers to deploying a private communication fabric in support of existing and future edge applications.

The architecture of Evio combines Software-Defined Networking (SDN) and peer-to-peer overlay tunneling and is programmable in different layers. At the datapath layer, it employs virtual NICs and Open vSwitch bridges to automatically program OpenFlow [5] rules for packet capture/forwarding within the VPN's virtual address space to support broadcast, unicast and multicast IP. It employs a modular distributed controller design that supports a scalable structured P2P topology and leverages widely used standards (XMPP, STUN, TURN, ICE) for membership authentication and revocation, node discovery, signaling for tunnel bootstrapping, and NAT traversal.

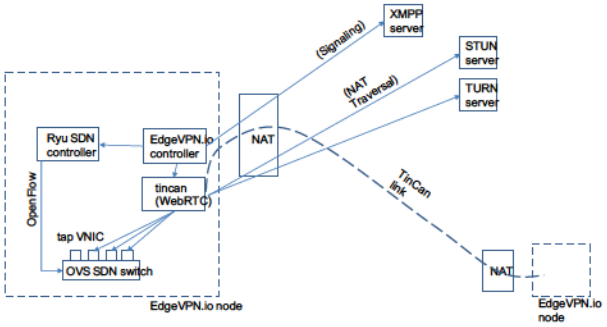


Fig. 1. Software architecture of an EdgeVPN.io node, consisting of: 1) a software SDN switch programmed using OpenFlow by a Ryu controller, 2) a user-level process (tincan) that tunnels encrypted traffic captured from the host using a tap virtual NIC (VNIC) device, 3) NAT traversal using STUN, TURN and XMPP protocols using WebRTC libraries, and 4) an EdgeVPN.io overlay controller that creates a scalable P2P topology for overlay routing

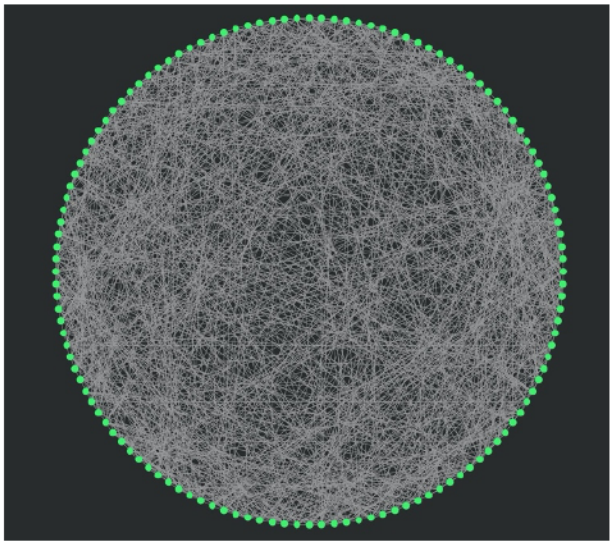

Fig. 2. Visualization of Evio overlay topology with 128 nodes

## II. Edge use cases

Considering a layered architecture for the deployment of edge applications using containers, one can at a coarse granularity divide it across infrastructure layer (underlying computer, storage, and network resources), upon which sits a container orchestration layer (responsible for deployment and lifetime management of containers), upon which sits the application layer. Evio virtualizes the network at the infrastructure layer, exposing the same interfaces (Ethernet, TCP/IP) to the container and application layers. Without such support from the infrastructure virtualization layer, the burden would fall into the container and/or applications layers to deal with the complexity of traversing NATs and/or firewalls.

While Kubernetes has its own network virtualization capabilities (e.g. through the Flannel CNI plugin), it does not on its own handle nodes in different address spaces and constrained by NATs/firewalls. Other VPNs exist, but they require burdensome configuration and/or the use of centralized gateways; Evio is unique in how it self-configures a scalable P2P tunneling fabric (Fig. 2).

A key edge-specific issue that Evio addresses is the ability to aggregate *multiple distinct edge resource providers* into virtual cluster(s) that can then be managed by the container orchestration layer. Two deployment modes can be supported by Evio's virtualization layer: one where there is a single container orchestration manager (i.e. a single Kubernetes cluster), and one where there are federated managers. Aggregating multiple edge providers enables the container and application layers to respond effectively to scenarios that can occur dynamically, such as an edge micro data-center that reaches capacity and cannot absorb additional workloads. A multi-edge deployment with the Evio network virtualization layer allows flexible workload placement, e.g. to a spill-over edge cluster near the resource that has exhausted capacity.

## III. Kubernetes Integration via CNI Plugins

There are two different ways Evio can be used to support Kubernetes deployments, both of which rely on the use of Kubernetes CNI (Container Network Infrastructure) plugins: 1) with Flannel, a CNI plugin that is used in many K8s deployments, and 2) with the Evio CNI plugin. Like Evio itself, Flannel creates an overlay network that exposes a virtual network namespace to K8s pods and uses encapsulation to tunnel messages between pods. Unlike Evio, however, Flannel does not support NAT traversal. Flannel can, however, leverage Evio's NAT traversal and virtualization - it's possible to deploy a Flannel overlay atop the Evio overlay. While Flannel works unmodified atop an Evio overlay, there is a performance price that is paid: double-encapsulation. In essence, messages sent among pods are encapsulated twice (by Flannel, and by Evio). To address this drawback, Evio has its own CNI plugin, which allows messages to be encapsulated only once. The demonstration will highlight Flannel integration, and overview the operation of both plugins.

## IV. Demo User Experience

- An introduction to the Evio software architecture (Fig. 1) and overview of the demo setup, overlay topology and Kubernetes hosts (four VMs across three different sites
- Walk-through: connecting an edge node to the Evio virtual network; configuration and adding a new host to the Evio virtual network
- Walk-through: adding an edge node to Kubernetes cluster, with emphasis on configuration of the Flannel plugin for use with Evio

## Acknowledgment

EdgeVPN.io is funded in part by a grant from the National Science Foundation (OAC-2004441, OAC-2004323). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## REFERENCES


[1] EdgeVPN.io, "Open-source VPN for Edge Computing" [Online]. Available: https://edgevpn.io [Accessed 26- August- 2020]

[2] Kubernetes, "Production-Grade Container Orchestration" [Online]. Available: https://kubernetes.io [Accessed 26- August- 2020]

[3] Docker, "Empowering App Development for Developers — Docker" [Online]. Available: https://docker.com [Accessed 26- August- 2020]

[4] Kubernetes, "Cluster Networking" [Online]. Available: https://kubernetes.io/docs/concepts/cluster-administration/networking [Accessed 26- August- 2020]

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "Openflow: Enabling Innovation in Campus Networks". SIGCOMM Comput. Commun. Rev., 38(2):69-74, Mar. 2008