Demo: Software-defined Virtual Networking Across Multiple Edge and Cloud Providers with EdgeVPN.io

Renato Figueiredo
ACIS Lab, ECE Department
University of Florida
Gainesville, FL
https://orcid.org/0000-0001-9841-6060

Kensworth Subratie
ACIS Lab, ECE Department
University of Florida
Gainesville, FL
https://orcid.org/0000-0001-8248-2856

***Abstract*—This demonstration will showcase EdgeVPN.io, an open-source software-defined virtual private network (VPN) that enables the creation of scalable layer-2 virtual networks across multiple providers - including scenarios where devices are behind Network Address Translation (NAT) and firewall middleboxes. Its architecture combines a distributed software-defined networking (SDN) control plane and a scalable structured peer-to-peer overlay of Internet tunnels that form its datapath. EdgeVPN.io provides a foundation for the deployment of virtual networks that enable research and development in distributed computing. The demonstration will include a brief overview of the architecture, and will show step-by-step how a researcher can deploy EdgeVPN.io networks on devices including Raspberry Pis, Jetson Nanos, and VMs/Docker containers in the cloud. Attendees will be provided with trial resources to allow them to follow the demonstration hands-on if they so desire.**

***Index Terms*—edge computing, fog computing, virtualization, VPN, SDN, peer-to-peer, overlay**

I. Introduction

The nascent area of edge computing encompasses techniques that can complement the widely adopted cloud computing model of deployment of distributed applications: it allows time-sensitive applications and services to leverage low round-trip times to resources deployed physically near mobile and IoT devices, and can significantly reduce the requirements (and cost) of transfer of large datasets to the cloud. In order to fully realize the potential of edge computing, it is key to support deployments that: 1) span across multiple edge regions/providers, 2) enables seamless connectivity among participating nodes, 3) enforces privacy in communications, and 4) leverage a wealth of applications and middleware platforms, such as Docker [3] and Kubernetes [2].

The EdgeVPN.io [1] (Evio for short) software provides a platform to deploy layer-2 virtual private networks spanning multiple edge and cloud providers. With this tool, edge computing researchers and practitioners can easily establish network connectivity at the Ethernet layer among devices that belong to a managed group. Once the virtual network is formed, applications that use TCP/IP (IPv4 and IPv6) can run unmodified.

Evio is a software-defined virtual network that runs on a variety of devices, and does not require any specialized hardware. The open-source software builds on standards and widely-used open-source platforms. Its self-configuring approach enables a quick start process (test virtual networks can be deployed often in minutes), and its self-organizing topology and routing enables the network to accommodate nodes joining and leaving the network without any manual configuration.

This demonstration is intended to target an audience of researchers and practitioners of ICDCS who are interested in experimental research in edge computing. The demonstration will overview the key technical underpinnings of the system (including SDN forwarding, peer-to-peer overlay networking, NAT traversal, and bootstrapping), and provide attendees with documentation, code, and trial environments they can use to deploy their own Evio VPNs - during the demonstration session, and/or afterwards.

II. Use cases

There are several use cases that can benefit from the use of Evio. In particular, edge video stream analytics: EdgeVPN provides a virtual network fabric spanning devices that can support edge-to-cloud analytics workflows for applications including distributed image stream processing, where video streams captured by distributed cameras are processed at the edge (e.g. for AI-based inference), while orchestration and aggregate analytics are performed in cloud resources. In this scenario, Evio facilitates the deployment of existing TCP/IP based software to run across multiple edge and cloud providers. In particular, Evio supports orchestration platforms such as Kubernetes, and the deployment of Docker-based containers.

III. EdgeVPN.io architecture

The Evio architecture is illustrated in Figure 1. It consists of modules that run on each of the edge nodes joining a network,

EdgeVPN.io is funded in part by grants from the National Science Foundation (OAC-2004441, OAC-2004323, and CNS-1951816). 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 NSF.

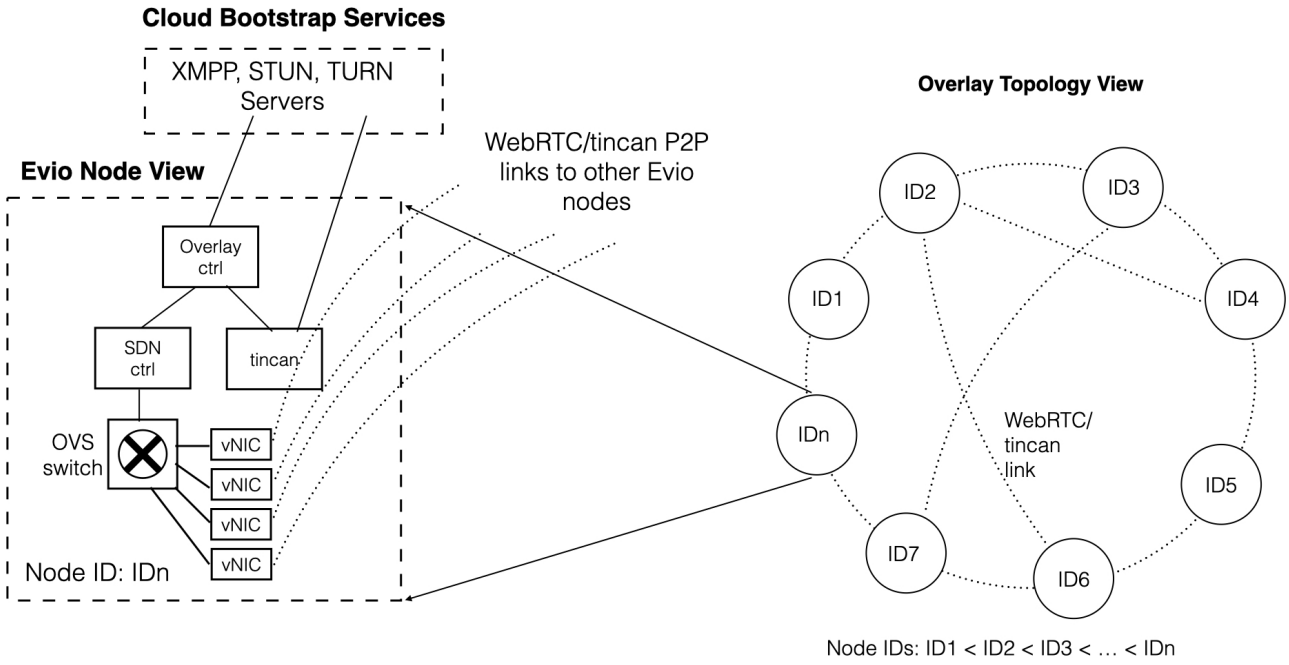

Fig. 1. Software architecture of EdgeVPN.io. Left: each Evio node consists of: 1) a software SDN switch programmed using OpenFlow by an SDN controller, 2) a user-level process (tincan) that tunnels encrypted traffic captured from the host using a tap virtual NIC (vNIC) device, 3) NAT traversal using STUN, TURN and XMPP protocols using WebRTC libraries, and 4) an EdgeVPN.io overlay controller that authenticates the node, discovers other peers in the authorized group of nodes, and creates a scalable P2P topology for overlay routing (shown to the right). The structured P2P topology is ordered by node IDs and follows the Symphony protocol, with an outer ring and log(N) long-distance nodes. The XMPP, STUN, and TURN servers run in cloud resources, while Evio nodes run on edge and.or cloud resources.

as well as modules that run on cloud resources and serve as a basis for bootstrapping and managing networks. These can be divided into the following major components:

- Bootstrapping/messaging service: this is a service that typically is deployed on a cloud resource, and provides the abilities of Evio nodes to authenticate, discover other nodes belonging to the same network, and communicate with them via short messages to establish peer-to-peer tunnels. This is leveraged in the form of a service that supports the XMPP [5] (eXtensible Messaging and Presence Protocol) standard, such as the open-source Openfire and eJabberd platforms.
- NAT traversal services: these services (also typically deployed on cloud resources) support negotiation of NAT traversal endpoints for devices that are in private networks and subject to NAT/firewall middleboxes. This is leveraged in the form of services that supports the standards of STUN [9], TURN [8], and ICE [10] protocols, such as the open-source coturn platform.
- Packet capture and forwarding: each Evio node includes an SDN switch, and software implementations of virtual Ethernet Network Interface Cards (vNICs). Packets from/to the virtual network are picked/injected via the vNICs, and are forwarded through SDN forwarding rules. This is leverage in the form of open-source SDN OpenFlow [7] switches such as Open vSwitch, and tap vNICs that are available in typical Linux distributions. The tap vNICs are bound to the endpoints of Internet tunnels (leveraging the open-source WebRTC [4] framework) via the user-level "tincan" process.
- Overlay management: each Evio node includes both an SDN controller, and an overlay controller. The overlay controller is responsible for creation, monitoring, and tear-down of (NAT-traversed) Internet tunnels through which Evio traffic is forwarded by SDN switches. The overlay controller implements a decentralized, scalable routing algorithm based on a structured P2P topology (with long-distance shortcuts) that requires O(log(N)) links per node and delivers messages over O(log(N)) hops on average. Forwarding tables are programmed automatically in SDN switches via the use of existing mechanisms/protocols (e.g. the ARP protocol)
- SDN controller: the SDN controller is responsible for implementing the flow rules associated with the structured routing algorithm, using the OpenFlow protocol to program the Open vSwitch (OVS). It also is responsible for capturing events that require overlay messaging, such as broadcast messages.

The overlay topology used by Evio is based on Symphony [6]. The number of long-distance shortcut links is

configurable; the Symphony protocol allows a trade-off between number of links and average routing distance, and links can be configured to be capped to a constant maximum, per node, without compromising correctness. By default, the system creates and maintains O(log(N)) long distance links, where N is the size of the network. This allows Evio to scale to large networks: the average number of links (and vNICs) at each endpoint is O(log(N)), while overlay routing takes place in O(log(N)) hops on average between network endpoints. The O(log(N)) long-distance links in Evio are drawn at random from a harmonic distribution, according to the approach described in Symphony. In addition to Symphony long-distance links, Evio monitors peer-to-peer traffic on links, and creates on-demand links (bound by a constant) between frequently-communicating nodes.

The interactions between Evio nodes and cloud services are limited to bootstrapping and NAT traversal setup, and, except for the case of symmetric NATs (where TURN is needed), the cloud nodes are not in the VPN packet datapath - packets flow across peer-to-peer links as shown in Figure 1 (right). To provide an estimate, the bootstrapping service we use for internal testing with networks of the order of 16-32 Evio nodes utilizes on average 4.3KB/s of network, with an average data transfer cost of 8 cents per day.

IV. Open-source software implementation

The Evio implementation leverages several open-source packages: WebRTC to create tunnels with NAT traversal, Open vSwitch for packet switching, and the Ryu framework for the SDN controller. It also inter-operates with cloud-side software, including coturn for STUN/TURN services, and OpenFire for XMPP messaging. The open-source Evio software itself consists of the three main modules depicted in FIgure 1: tincan, SDN controller, and overlay controller.

The software has been ported to x86 (amd64) and ARM (armhf, arm64) architectures, and both Debian installation packages and Docker [3] images are provided to users. The software has been tested in platforms including amd64 and arm64 Amazon EC2 cloud instances, VMware VMs, Docker containers, armhf and arm64 Raspberry Pi devices, Compulab fitlet2 and Jetson Nano edge devices.

EdgeVPN can be installed and deployed on endpoints in two ways: via a Debian package (hosted in a package repository), or via a Docker container (hosted in DockerHub). The software takes a single JSON configuration file that configures the node's identity, credentials, and various overlay parameters.

Evio has been demonstrated to work with Kubernetes [2], a widely-used platform for workload orchestration using Docker containers that is commonplace in cloud environments, and increasingly used as a platform for edge computing. The Kubernetes Flannel plug-in works seamlessly over Evio tunnels, without requiring any changes to the Kubernetes code.

While Evio performance depends on a variety of factors including the performance of nodes hosting Evio nodes, and underlying link performance, the overhead of running Evio is sufficiently small to make it viable to a variety of edge applications. While a detailed analysis is beyond the scope of this demo, the software footprint is such that it runs on inexpensive edge devices such as Raspberry Pi and Jetson Nano. To provide an estimate of orders of magnitude, on an edge-class device (such as the Compulab fitlet2), the per-packet round-trip latency overhead on an Evio VPN link is of the order of 1ms, and iperf TCP throughput across tincan tunnels is of the order of 140Mbit/s.

V. Demonstration and accommodations

The demonstration flow is outlined as follows; materials will be made available to attendees in advance of the conference:

- Brief Evio architecture overview
- Demonstration of a pre-deployed Evio network that brings together heterogeneous distributed devices - all behind different NAT/firewalls - seamlessly into a VPN. Evio nodes in the demo will include: Amazon EC2 instances; fitlet2, Raspberry Pi, and Jetson Nano edge devices distributed across multiple home ISP networks; Raspberry Pi connected via a cell LTE link
- Demonstration of how to deploy a new Evio network from scratch using a trial service. Users will be provided with credentials to run their own nodes during the demo if they so wish.
- Overview of deployment of Kubernetes and Flannel over EdgeVPN

The demo is designed to work in an environment where the presenter shares their screen, with Zoom or equivalent. It will show introductory slides, and Linux terminals highlighting the user experience with an EdgeVPN. Conference users who have access to a Docker platform will, in addition, be provided the option of deploying their own node to join the demo overlay - however, this is an optional activity.

References

[1] EdgeVPN.io, "Open-source VPN for Edge Computing" [Online]. Available: https://edgevpn.io [Accessed 18- February- 2021]

[2] Kubernetes, "Production-Grade Container Orchestration" [Online]. Available: https://kubernetes.io [Accessed 18- February- 2021]

[3] Docker, "Empowering App Development for Developers — Docker" [Online]. Available: https://docker.com [Accessed 18- February- 2021]

[4] WebRTC, "Real-time communication for the web" [Online]. Available: https://webrtc.org [Accessed 18- February- 2021]

[5] XMPP, "The universal messaging standard" [Online]. Available: https://xmpp.org [Accessed 18- February- 2021]

[6] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," Proc. 4th USENIX Symposium on Internet Technologies and Systems, pp. 127–140, 2003

[7] 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

[8] R. Mahy, P. Matthews, J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", Internet Engineering Task Force (IETF) RFC 5766, April 2010.

[9] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)". Internet Engineering Task Force (IETF) RFC 3489, March 2003

[10] J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", Internet Engineering Task Force (IETF) RFC 5245, April 2010