2024 IEEE 10th International Conference on Network Softwarization (NetSoft) | 979-8-3503-6958-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/NetSoft60951.2024.10588920

PolyNet: Cost- and Performance-Aware Multi-Criteria Link Selection in Software-Defined Edge-to-Cloud Overlay Networks

Vahid Daneshmand
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Florida, USA
vdaneshmand@ufl.edu

Kensworth C. Subratie
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Florida, USA
kcratie@ufl.edu

Renato J. Figueiredo
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Florida, USA
renatof@ufl.edu

***Abstract*—In the ever-evolving networking landscape, the demand for efficient and adaptable Virtual Private Network (VPN) solutions is growing. Software-Defined Networks (SDNs), particularly Peer-to-Peer (P2P) overlay VPNs, offer a practical approach for networks spanning various edge and cloud providers. However, existing decentralized VPNs, while resilient and scalable, typically utilize a single tunnel type and overlook data plan costs and link performance in their selection processes. This oversight can lead to cost and performance inefficiencies, especially in edge-to-cloud networks where diverse nodes have unique needs that generic solutions fail to meet effectively. Although SDN facilitates the integration of multiple link types in overlay VPNs, existing systems lack efficient policies for selecting favorable tunnels. To bridge this gap, we introduce *PolyNet*, a Multi-Criteria approach designed to make cost- and performance-aware policy decisions in hybrid-link overlay networks. *PolyNet* employs a dynamic link selection policy during runtime that evaluates latency using Vivaldi network coordinates and considers cost, and integrates with SDN-based P2P overlays to enhance link management capabilities and support multiple link types. This paper presents the design of *PolyNet* and evaluates its performance through simulations and prototype testing. Results demonstrate that *PolyNet* achieves up to a 19.1% cost reduction and a 14.1% latency improvement over traditional methods in Symphony P2P topologies. Additionally, tests with a software prototype confirm the advantages of hybrid links, showing that kernel-layer GENEVE tunnels can increase throughput by up to 8.9 times compared to user-layer Nebula and WebRTC tunnels in edge clusters.**

***Index Terms*—overlay network, P2P, VPN, SDN, Hybrid-link VPN, edge computing, cloud computing**

I. Introduction

As the number and variety of computing resources proliferate, gathering various geographically distributed resources—from resource-constrained devices at the network's edge to servers in edge and cloud data centers—under a single management domain becomes more challenging. The contemporary cloud-centric model handles computationally intensive tasks at the cloud servers [40]. However, cloud platforms are not ideal for scenarios requiring low-latency actions [18] or when data transfer cost from the edge to the cloud is high [30]. With edge computing, it is possible to reduce data transfers and round-trip latencies by utilizing the computing power of edge devices to decentralize processing power to the edge of the network and process data in proximity to the physical location of the data sources [28]. Instead of a model where services run solely on the cloud, the edge computing architecture enables edge devices to participate in performing tasks in addition to cloud servers [2]. Edge devices are typically resource- and power-constrained, geographically distributed, and may be connected via metered cellular data plans. While connectivity among them is desirable, avoiding large data costs is critical. Additionally, network characteristics must scale with the number of connected devices. By processing captured data locally on edge devices, the amount of data being transferred to the cloud can drastically decrease [30]. That leads to significantly reduced network latency and cost, especially when using cellular data plans on edge devices [11]. An architecture that supports edge-to-cloud computing is poised to support many emerging applications, such as location-based services, smart cities, connected vehicles, and emergency response [8].

Because such a distributed edge-to-cloud environment can be scattered among multiple providers across the Internet, establishing network connections among the nodes is challenging. The nodes are possibly on different subnets, assigned private addresses, and affected by Network Address Translation (NAT) and network firewall devices that may further limit the connections [9]. While NATs conserve the limited resource of public IP addresses [13], not all tunneling technologies are compatible with NATs and network firewalls [33].

A Virtual Private Network (VPN) expands a private network across a public network and consolidates select distributed resources under the same logical network, enabling nodes in different networks to communicate and providing a foundation to address the above edge-to-cloud networking challenges.

This work is supported by the US National Science Foundation as part of awards OAC-2004441, and OAC-2004323. Any opinions, findings, 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.

While managed VPNs such as WireGuard and OpenVPN are widely used, they require dedicated VPN servers to relay traffic, raising the management burden on the servers as the number of connections increases [33]. Furthermore, they require servers to be publicly addressable and thus cannot directly interconnect private servers that are behind different NATs/firewalls—a case that is common in edge-to-edge scenarios. Additionally, topology and switching rules require management as nodes join or leave the network, which makes these dynamic networks operationally challenging to scale.

An alternative self-managed approach that supports dynamic membership and decentralized switching is a peer-to-peer (P2P) overlay VPN. Compared to traditional VPNs, this approach has advantages, including P2P NAT traversal, self-managed links and switching, forwarding, and dynamic membership. Representative open-source implementations of P2P VPN architectures include Nebula [31] and EdgeVPN.io [34] (Evio [35]), both of which rely on user-level tunneling with NAT traversal. However, P2P VPNs such as Nebula and Evio currently face a limitation in which a single type of tunnel (e.g., the standards-based WebRTC [42] in Evio, and Nebula's custom tunneling protocol) is used for all the overlay links.

While a P2P overlay architecture with decentralized SDN switching built on a single type of tunnel/link is feasible [33] and simplifies the design, it can be inefficient for two distinct reasons. First, it does not account for the cost of maintaining links for metered cellular connections that may be too high due to too many keep-alive messages required for NAT traversal, which can be avoided using fewer or alternative types of links. Second, while an approach built on user-level tunnels is widely applicable, it is inefficient for high-throughput links within edge or cloud data centers, where kernel-mode tunnels can avoid the overheads of user-kernel copies and switching.

This paper presents and evaluates *PolyNet*, a novel policy that supports flexible management of links in a P2P VPN based on information about the node and its candidate peers. Cost reduction at the edge is achieved by selecting target nodes to minimize cost-metered traffic. Performance is enhanced with Vivaldi [10] network coordinate estimates and applying kernel-space tunneling protocols (e.g., GENEVE [17]) among nodes within the same private network.

Contributions: This paper presents the research and development of *PolyNet*, a novel approach that supports cost- and latency-aware decision-making for long-distance link selection based on the endpoints' data cost and link latency estimates, and can be integrated with overlays that support hybrid link types. The current implementation of the proposed approach employs a Multi-Criteria Decision-Making method, focusing on these criteria. However, its modular design not only allows for the incorporation of additional criteria but also facilitates the application of alternative decision-making techniques, enhancing its adaptability and applicability in various scenarios.

II. Related Work

Despite the extensive literature on overlay networks [12], software-defined networks (SDNs) [3], [21], [29], and virtual networks, the dynamic multi-criteria decision-making method for hybrid-link overlay P2P VPNs is novel. SDN systems have emerged to address a need for programmable networking elements such as Ethernet switches and have been applied in multiple different scenarios, mainly related to data centers [5]; the focus of our approach is to apply SDN as the core mechanism to implement support for hybrid tunnel types within and across edge and cloud devices and data centers. Network overlays have been applied to improve the resiliency of routing over Internet path outages in RON [1], multicast over the Internet [14], reducing the configuration and supporting the deployment of virtual networks [38], NAT-traversed tunneling [33], and leveraging overlay functionality without modifications to the applications and operating systems [22]. Application-layer overlay networks have also been utilized to expose data storage and lookup [32].

Bilal et al. [4] give a comprehensive overview of SDN multi-controller architecture and compare them based on distribution method and the communication system. Their work highlights the benefits of distributed architecture over centralized architecture in efficiency, scalability, and availability. With Muppet, Uddin et. al. [39]'s multiprotocol edge-based architecture designed for large-scale IoT deployments and services automation, SDN-based switches are inserted between communicating IoT devices under the control of a centralized SDN controller. Contrary to Muppet, *PolyNet* architecture uses distributed SDN controllers as compared to a single centralized controller. Theodorou et al. [37] propose MINOS, a multi-protocol SDN platform for heterogeneous IoT nodes. Their demonstrated platform with CORAL-SDN [36] and Adaptable-RPL [37] routing protocols show improved packet delivery ratio compared to RPL routing protocol.

DOVE [7] is a closely related system; however, it focuses on multi-tenancy in data centers, building on hypervisors that implement dSwitches to handle packet processing. In contrast, our approach employs a model where SDN switches implement packet processing. Moreover, DOVE's design works in distributed data center environments where a single entity controls the virtual network infrastructure. Our work explores a different environment where the infrastructure is distributed across multiple providers in an edge-to-cloud continuum. Also, DOVE does not address the connectivity limitations of edge devices constrained by NAT/firewall middleboxes. Likewise, VirtualWire [41] does not consider NAT-constrained scenarios.

III. Motivating Examples

In the context of edge-to-cloud continuum, while local links in a data center can benefit from high-performance tunneling, cost-metered peers at the edge may be constrained in terms of NATs, data costs, and connection performance. Communications throughout a broad network of scattered resources on the Internet need to be encrypted. On the other hand, encryption may be an avoidable overhead for overlay links within a trusted private network. A P2P VPN overlay supporting a single tunneling technology limits performance, deployability, or both. A

scalable P2P overlay VPN for edge-to-cloud continuum needs to manage both the type and the number of links for each peer, as well as the network topology. The system design of *PolyNet*, which enhances the inherently scalable Symphony topology, includes policies that define the preferred links between peers.

Consider a use case example: an advanced safety response project in a modern urban environment that utilizes a wide range of computing and network resources—from edge nodes embedded in citywide surveillance cameras to local edge data centers and high-performance cloud servers. At the network's edge, each surveillance camera acts as an edge node, using an efficient object detection module such as YOLO [27] tuned to identify and label objects, e.g., firearms. In addition, the edge nodes also perform image processing and object tracking across multiple frames. Whenever a firearm is detected, the system triggers an event that prompts a rapid local response, e.g., increasing frame rate to capture more detailed information and engaging nearby surveillance cameras for additional tracking. For more complex image analysis, such as in-depth firearm classification, trajectory prediction, and behavioral analysis, the video feeds and detected events are relayed to local edge data centers. Cloud servers handle extensive data storage, long-term tracking, and advanced analytics, such as suspect identification and comprehensive behavior analysis. These tasks demand significant processing power and storage, typically available in a cloud setting.

Another potential real-world application is in ad-hoc emergency response. Consider an example response scenario based on urban flooding, an emergency event which can be caused by intense rainfall or storm surge. Such urban flood events can impact not only humans but also infrastructure, and effective and timely response requires coordinated response across multiple agencies (such as the police, fire department, and the National Guard). While a response team may be able to tap into valuable information in cloud resources (e.g., geospatial datasets and hydrologic models aggregated in knowledge graphs), for improved situational awareness responders can benefit from live streams available from cameras distributed across the area affected. However, the cameras themselves are often owned and operated by different agencies (e.g., traffic cameras, body cameras, UAV-mounted cameras, and private cameras owned by businesses), and therefore not readily accessible. A P2P VPN can be deployed as an ad-hoc, cross-provider virtual infrastructure during this event, and the *PolyNet* techniques can lead to improved performance and cost.

The networking needs of this use case are therefore complex and dynamic, necessitating the ability to manage a diverse range of connections with varying requirements in terms of security, performance, and reliability. This motivating example is similar to VideoEdge [19], with the exception that, as opposed to VideoEdge, we are addressing networking layer challenges. A traditional VPN solution may fall short in these contexts due to its fixed tunneling protocols and link configurations. In contrast, our proposed VPN solution can adaptively select the preferred link for each P2P connection.

IV. Problem Formulation

A. Supporting Multiple Tunnel Types

Tunneling is a protocol to transfer data from one network to another by exploiting encapsulation. Encryption applied to tunnels enables private communications over a public network, such as the Internet. Such a system needs to be architected to support and integrate these multiple types seamlessly. The use of SDN switches can achieve this.

Role of SDN: The Software-Defined Networking (SDN) approach to network management facilitates dynamic, programmatically efficient configuration of networks to enhance their performance. SDN addresses limitations in the static architecture of traditional networks. SDN switches simplify network management, deployment, and operation. This simplification happens by decoupling network control and forwarding procedures from individual switches and routers and placing them in an SDN controller. Being highly programmable, it enables network administrators to develop dynamic, automated programs to specify network behavior under different circumstances. In the context of overlay virtual networks, SDN programming can be employed to allow forwarding across different types of links in the overlay network through the same abstraction—Ethernet switch ports bound to a tunnel (SubsectionVI-C). The communication protocol that SDN controllers use to establish the path of network packets is OpenFlow [25]

B. Decision Making Mechanism

Such a network described above requires a mechanism to select preferred links among a pool of potential candidates based on several criteria, such as cost and performance. *PolyNet* is an effort in this direction. It is a new extensible decision-making system for link selection in P2P overlay networks. The current proof of concept implementation of *PolyNet* employs the Symphony overlay network as a representative structure and a Multi-Criteria Decision-Making (MCDM) method based on various factors for link selection.

C. Decision Making Factors

When determining the preferred link between two nodes in a P2P overlay network, there are several key factors or inputs to consider. The ones specifically addressed in this paper are:

- **Data Transfer Cost:** If nodes are connected via cost-metered networks such as cellular networks, data transfer cost could affect the number and type of links established.
- **Latency:** The round-trip time or latency between nodes is another crucial factor. Lower latency may allow for more efficient usage of specific links and lead to better performance in the whole network.
- **Tunnel Type:** Different tunneling protocols can lead to significant differences in throughput, especially considering kernel-level and user-level implementations.

Several other factors (e.g., availability, throughput) can be considered as criteria in the proposed MCDM model for efficient and desirable link selection in P2P overlay networks. While in this paper we focus primarily on cost, latency, and

tunnel type, by using well-known MCDM methods such as TOPSIS, the design is generalizable to incorporate additional criteria. It also allows users to provide configuration information to drive the weights associated with different criteria.

V. System Design

A. Overview

PolyNet assumes that each node in the overlay can initiate link requests to other nodes by creating tunnels according to the desired overlay topology. As a concrete example, in the EdgeVPN overlay [34], these two responsibilities are divided between a Link Manager (which is responsible for creating, monitoring, and terminating individual tunnels) and a Topology Manager (which is responsible for maintaining a routable topology as nodes join and leave). In particular, EdgeVPN's link manager supports WebRTC-based [42] tunnels and the topology manager supports the Symphony [23] structured P2P overlay. Without loss of generality, the *PolyNet* design assumes that the link manager supports multiple tunnel types and that the topology manager can provide a candidate set of possible link targets that satisfy topology constraints. Specifically, in this paper, we consider two possible tunnel types (WebRTC and GENEVE) and the Symphony overlay topology, where the candidate set is drawn randomly from a universe of large-integer unique node IDs using a harmonic distribution.

Figure 1 illustrates the *PolyNet* policy module in this context. It takes as an event input (left arrow) a request for a link from the topology manager and can use various static and dynamic inputs (top arrows) to drive the MCDM model to select a specific target from a candidate set (right arrow). In this paper, the following specific inputs are considered: 1) Vivaldi [10] network coordinates, 2) data cost estimates, and 3) location (same data center/same NAT, or remote) of the nodes in the candidate set. Without loss of generality, additional inputs can be accounted for by the policy module. Our proposed system design is inherently scalable for two key reasons. First, the inputs used in the link selection module are based on information a node obtains from two scalable, decentralized algorithms: Symphony and Vivaldi. Second, the MCDM decision-making algorithm (TOPSIS) is computationally lightweight and adds minimal overhead. In the evaluation, we compared the performance of a Symphony-based system equipped with the *PolyNet* module against a baseline Symphony system with random assignment.

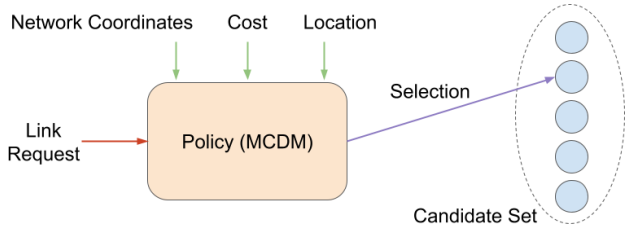

Fig. 1. Policy Management Overview. Link request events (left) trigger the policy module's execution to account for multiple criteria (top) to perform link selection among nodes in a candidate set (right)

B. Symphony Structured Topology

While the concept underlying our proposed method is generally applicable to any P2P VPN, our research specifically zeroes in on a structured Symphony topology. This focus arises from Symphony's inherent self-configuration and scalability, traits that align well with our objectives for *PolyNet*, and from an existing open-source implementation. As such, our exploration of the Symphony topology serves not only as a proof-of-concept for *PolyNet* but also as a potential model for the enhancement of other P2P VPNs.

Symphony is a structured P2P overlay network topology known for simplicity, robustness, and scalability. Designed specifically for distributed hash tables (DHTs) but also applicable to SDN-based routing overlays, it features a one-dimensional circular ring or harmonic topology. In Symphony, the network assigns each node a unique random identifier from a large identifier (nodeID) space. Nodes connect to their immediate nodeID neighbors on the ring, as well as to a subset of long-distance nodes. The selection of these long-distance nodes follows a harmonic distribution leading to a higher chance of connecting to nearby nodes and a lower chance to farther ones. The two primary types of Symphony links are **successor links** (a small constant, e.g., k=2, of immediate successors in nodeID space), and **long-distance links** (log(N) links to 'far' nodes in the nodeID space, where N is the overlay size). Symphony supports scalable routing, where, on average, with only a logarithmic number of hops (specifically, O(log N)), a message can be sent from one node to any other node in the network. This efficiency, along with the network's robustness to node failures, makes Symphony an attractive topology for P2P networks and applications.

C. Vivaldi Network Coordinates

Vivaldi [10] is a distributed algorithm for estimating latency between nodes in a computer network. This system represents each node as a point in a multi-dimensional space, and the Euclidean distance in the Vivaldi coordinate space between points serves as an approximation of the network latency between them. By continuously adjusting its coordinates based on latency measurements to other nodes, each node autonomously finds its position in this virtual space. A key feature of Vivaldi is that nodes only need to exchange measurements with a small subset of nodes. Based on these measurements, network coordinates are used to estimate latency between two arbitrary nodes that may not have any measurement. This approach allows for efficient and scalable estimation of network latencies.

D. Data Cost Estimates

Our simulation considered all edge nodes to have cell costs based on pay-as-you-go data plans, meaning they are charged a flat per-byte rate for network traffic they transfer. To simulate the cost of a network route, we assumed each unit of incoming or outgoing network traffic (e.g., 1 GB) has a corresponding cost (e.g., $0.10). Therefore, if traffic passes through a route, the cost at a metered edge node is 2 cost units, while at a

no-cost server node, there is no additional cost. The endpoints of a route, whether handling incoming or outgoing traffic, are associated with a cost of 1 unit each.

E. Location Identifiers

Location identifiers are used to determine whether nodes are within the same data center and thus when NAT traversal is not necessary. These identifiers can be explicitly configured. While it is also possible to implement heuristics to infer co-location (e.g., by using LAN discovery protocols), in the paper, we consider the former in our experiment.

F. Putting It All Together

In this paper, we consider this specific set of inputs for determining the target among a candidate set of nodes that satisfy the Symphony topology:

- **Vivaldi Coordinates:** we simulate these by leveraging the built-in implementation of Vivaldi in Ether [15] to compute the Vivaldi distance between the node and others in the candidate set (Subsection VI-A).
- **Data Costs:** we simulate these using physical topology information from Ether, assigning a cost function to edge nodes connected to cost-metered cell links and zero cost to cloudlet nodes connected to non-metered links.
- **Location Identifiers:** for this criterion we use an enhanced version of the EdgeVPN implementation that supports multiple link types, where nodes in the same data center are configured with the same unique data center location ID (Subsection VI-C).

G. Multi-Criteria Decision Making

Multi-Criteria Decision Making (MCDM) is a valuable approach for decision-makers in many fields, including network tunnel selection in complex environments where multiple factors must be considered. It helps identify the best option among several alternatives based on multiple criteria by using mathematical models and algorithms to evaluate each option's strengths and weaknesses and rank them accordingly. However, as previously used in RODENT [16] for routing in wireless sensor networks, Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) has emerged as an effective variant of MCDM that is well-suited to network tunnel selection. TOPSIS establishes an ideal and anti-ideal solution for the decision problem based on the criteria used to evaluate the options. It then measures each option's distance from the ideal and anti-ideal solutions and ranks them based on their overall suitability. This approach enables decision-makers to compare the options more comprehensively and accurately than traditional MCDM methods. In network tunnel selection, decision-makers can use TOPSIS to consider criteria such as cost, network capacity, availability, and performance.

H. Assigning Weights to Criteria

Assigning appropriate weights to the criteria used in TOPSIS is crucial as it determines the relative importance of each criterion in the decision-making process. Inappropriate or arbitrary weight assignments can lead to biased or inefficient outcomes. This process is inherently subjective, relying heavily on the decision-maker's judgment and their understanding of the decision context. Thus, user or expert input for weight assignment in MCDM is often inevitable. For scenarios with few criteria, weights can be directly assigned, as in [16]. However, with more criteria, a method like the Analytic Hierarchy Process (AHP) ensure weight consistency, as used in [24].

VI. Experimental Setup

This paper uses two methods to evaluate different aspects of *PolyNet*. The first is a simulation environment that implements a TOPSIS-based selection module for long-distance Symphony overlay links on top of Ether [26] to evaluate the cost and performance of *PolyNet* using Vivaldi coordinates and data cost estimates compared to a baseline random link selection policy of Symphony in realistic edge/cloudlet topologies. The second method is a prototype that extends EdgeVPN.io [34] to support two different types of tunnels: WebRTC (user-space tunneling with native NAT traversal) and GENEVE (kernel-space tunneling with no support for NAT traversal). This prototype validates the hypothesis that using location identifiers and selecting between different tunnel types can lead to significant throughput improvements at the link layer. By binding different tunnel types (WebRTC, GENEVE) to switch ports, the SDN-based P2P overlay virtual network can be constructed with hybrid links. This extended overlay implementation is used to quantitatively assess the performance of different tunnels in both cloud and edge resources.

A. Ether: The Edge Topology Synthesizer

Ether [15], [26] is an open-source middleware designed to enable researchers and engineers to generate and test plausible edge infrastructure configurations effectively. Addressing the challenges in evaluating edge computing systems (especially in the absence of suitable testbeds), Ether offers a versatile and scalable platform for simulating diverse edge computing environments. Key features of Ether include the ability to simulate a broad range of edge scenarios and the flexibility to design customized network topologies by specifying the number and type of nodes (e.g., IoT devices, edge servers), as well as their interconnections. This tool supports the creation of complex, multi-tier architectures that mimic real-world interactions between edge and cloud components. Additionally, Ether provides a flexible programming interface that facilitates automation and iterative experimentation. This is crucial for conducting empirical research, as it allows for the thorough testing of scenarios under conditions that closely resemble actual deployments.

Ether's pre-parameterized scenarios, validated in [26], enable experiments in this paper to utilize synthesized topologies of edge and cloudlets. This approach allows exploration of environments at scales beyond those feasible in existing testbeds. However, while Ether provides a rich simulation environment, it does not simulate the behavior of SDN switches and hybrid links. Therefore, we use EdgeVPN.io's overlay deployments on

real systems for our link-level performance experiments rather than relying on Ether simulations.

B. Extending Ether with Symphony Overlay Topology

To simulate the Symphony overlay topology on top of the Ether physical topology for our *PolyNet* link selection experiments, we made several extensions to the Ether codebase:

- *Overlay Module:* The classes and methods in the Ether core codebase have been expanded. Symphony-specific properties, such as the Symphony ID and the lists of successor and long-distance nodes, have been added to the Node data structure. New methods to support their initialization and updates have been defined including generating Symphony ID, setting successor and long-distance links, and routing within the Symphony data structure. Additionally, two target selection methods for Symphony long-distance links have been implemented: a TOPSIS-based method and a random selection method.
- *Utility Module:* Helper functions for setting up Symphony overlay and running experiments have been added.
- *Visualization Module:* Symphony ring visualization is facilitated by methods added to the codebase.

C. EdgeVPN.io: Overlay Virtual Network

For the experiments in this paper, the EdgeVPN.io overlay virtual network (hereafter referred to as Evio) has been enhanced to support hybrid tunnels. Specifically, this enhancement involves binding either WebRTC or GENEVE tunnels to Open vSwitch ports via a virtual network interface controller (NIC) tap device, which enables user-space processing. The Evio link manager has also been expanded to manage these two different types of tunnels, as illustrated in Figure 2. This extension allows the overlay to support a model where each link type can be independently selected: the baseline (and lower-performing) WebRTC is supported between any arbitrary pair of nodes due to its support for NAT traversal using ICE, STUN, and TURN protocols, whereas the higher-performing GENEVE tunnels are feasible when nodes are located in the same data center and not behind different NATs.

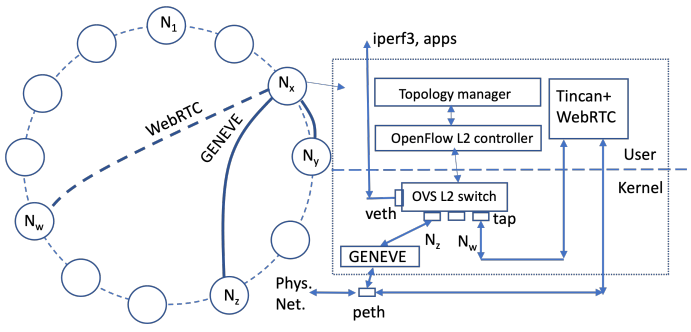

Fig. 2. SDN paths for GENEVE and WebRTC tunnels. Overlay node Nx has a NAT-traversed link to node Nw; in this path, packets are handled in user space by a process (Tincan) that uses WebRTC libraries. Overlay node Nx also has a link to node Nz in the same location identifier; here, packets are handled by GENEVE in kernel space. These tunnel types are abstracted from applications (e.g., iperf3 in our experiments) via the SDN switch.

This extension to support hybrid tunnels only impacts the link layer of the P2P overlay, while the topology and packet switching layers remain unmodified. Thus, the extended hybrid-link VPN supports the same Symphony topology and bounded flood broadcast with unicast path discovery and decentralized SDN controllers as described in [34]. For comparison with the state of the art, the performance of the extended Evio is compared against that of Nebula [31], another open-source modern VPN overlay with P2P links. Unlike Evio, Nebula only supports one link type, which requires all packet processing to be done in user space.

VII. Evaluation

We designed and performed two sets of experiments. In the first, we assess the performance of an SDN-enabled hybrid-link overlay link comparing a prototype implementation based on Evio with the Nebula VPN. In the second, we evaluate the performance of link selection in *PolyNet* using an extended implementation of the Ether simulator.

A. SDN-Enabled Hybrid Link Performance

To highlight the importance of supporting hybrid links in an overlay P2P VPN, we first perform an experiment contrasting the performance of an SDN-based layer-2 overlay (Evio version 24.1.1 [35]) with a non-SDN layer-3 overlay (Nebula [31] version 1.5.2). The former (Evio) supports hybrid links, currently with two tunnel types: GENEVE (kernel-level, no NAT traversal) and WebRTC (user-level, native NAT traversal), while the latter (Nebula) only supports one link type (user-level, native NAT traversal). In the experiments, we measure the iperf3 [20] throughput of a *virtual network link* in two different environments: NSF CloudLab [6] (representative of a cloud data center) and a Raspberry Pi 4 cluster (representative of an edge computing cluster), as follows:

Raspberry Pi Cluster: Processor: 4-core RPi 4, Memory: 8GB RAM, Operating System: Linux 5.4.0-1097-raspi aarch64, Network Interface: 1Gb/s Ethernet (wired)

CloudLab Utah m400 Nodes: Processor: 8 64-bit ARMv8 (Atlas/A57) cores at 2.4 GHz (APM X-GENE), Memory: 64GB RAM, Network Interface: Dual-port Mellanox ConnectX-3 10 GB NIC (PCIe v3.0, 8 lanes), Operating System: Linux 5.4.0-139-generic aarch64

As shown in Table I, Evio with WebRTC links and Nebula exhibit lower virtual network throughput compared to other alternatives due to the overhead of user-space tunneling. Conversely, Evio with GENEVE links shows significantly higher throughput—nearly matching physical link performance in the Raspberry Pi cluster—thanks to kernel-space tunneling and Open vSwitch SDN packet processing. User-mode tunneling is necessary for NAT traversal in both overlays when packets flow *across edge/cloud providers*, because existing NAT traversal libraries are user-space. However, within a data center, there are no NATs, and the overhead of NAT traversal for WebRTC and Nebula links becomes significant. Note that two key reasons that the Evio architecture can support the choice between user-mode WebRTC and kernel-mode GENEVE tunnels are

that 1) it operates in layer-2, and 2) links bind to ports of a programmable Open vSwitch (OVS) SDN switch that allows the system to abstract link type through a common interface (i.e., an SDN switch port). Nebula, in contrast, operates in layer-3 and does *not* use an SDN switch and cannot create a kernel-mode tunnel between devices in the same data center.

In summary, with SDN-enabled hybrid link support, the available link type can be provided as a criterion for the MCDM/TOPSIS link selection policy, which can then choose between GENEVE or WebRTC tunnels depending on the location of the overlay endpoints (same data center or across different NATs). Furthermore, for the latter, the MCDM/TOPSIS link selection policy can also consider criteria such as estimated latency distance (using Vivaldi coordinates) and cost (depending on whether nodes are on metered data plans). This is elaborated in the next set of experiments.

TABLE I
IPERF3 OVERLAY VIRTUAL NETWORK THROUGHPUT: PHYSICAL NETWORK, EVIO (HYBRID LINKS), AND NEBULA

	Raspberry Pi 4	**CloudLab**
Physical Link	798 Mbit/s	7.28 Gbit/s
Evio: OVS, GENEVE (kernel-mode)	770 Mbit/s	3.25 Gbit/s
Evio: OVS, WebRTC (user-mode)	80.7 Mbit/s	128 Mbit/s
Nebula (user-mode)	86.5 Mbit/s	131 Mbit/s

B. Latency- and Cost-Aware Long-distance Link Selection

We conducted simulation-based experiments using Ether to compare long-distance link selection in *PolyNet* with a random selection approach used in the original Symphony algorithm and current open-source P2P VPN [34]. *PolyNet* employs the TOPSIS method to select preferred targets for long-distance links. The criteria used in the decision-making process are network latency and cost. On the other hand, the base Symphony algorithm chooses the targets randomly based on a harmonic distribution. We created a simulated random urban sensing topology using Ether, then set up a Symphony overlay network using *PolyNet* with TOPSIS selection and again with random selection. The simulated environment consists of a large number of Raspberry Pi 4 edge nodes with pay-as-you-go cell plans and cost-free server nodes in cloudlets. The parameters for running each experiment include the number of edge and server nodes, and TOPSIS criteria for decision-making. We conducted three separate experiments, each with a different ratio of edge nodes to server nodes, to capture the effects of different relative resource configurations on performance. In each experiment, to compare the methods, we randomly chose source-destination pairs and compared the distributions of latency and cost of paths between these pairs.

This scenario evaluates the distribution of both latencies and cell costs in target selection for long-distance links. The first two experiments (Figures 3 and 4) consider two scenarios with a total of 1024 overlay nodes, where in the first experiment a larger fraction of nodes (80%) is cost-metered. The third experiment (Figure 5) models a smaller network of 256 nodes.

A consistent observation across the three experiments is that the distribution of pairwise latencies achieved with *PolyNet* link selection is more skewed to the left compared to the baseline random selection. In other words, with the same physical network configuration and considering the same set of node pairs, latency and cost are lower when using *PolyNet* compared to the baseline. On average, *PolyNet* reduces the overlay network latency by 12.2%-14.1% and cell cost by 13.0%-19.1%. While *PolyNet* outperforms the baseline in all experiments, the cost reduction is more significant in Experiment 2 (Figure 4), where a smaller portion of the network edge nodes are cost-metered. This can be explained by the fact that *PolyNet* benefits from the availability of a wider range of choices to better perform cost-aware selection. A comparison between Experiment 2 and Experiment 3 shows that changing the network size while maintaining the other aspects of the experiment leads to similar levels of improvements.

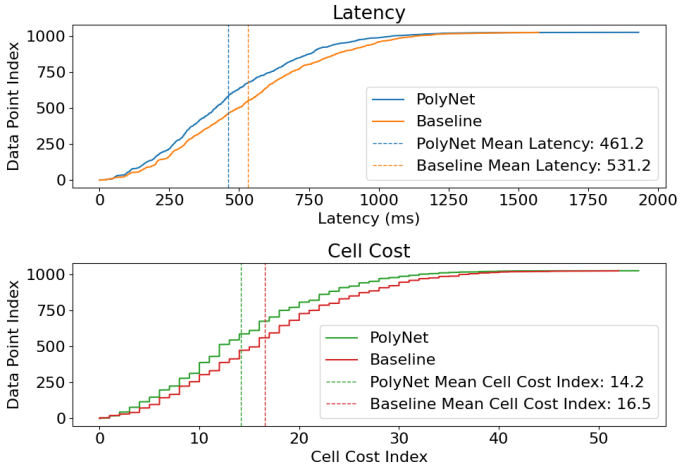

Fig. 3. Experiment 1: Latency and cell cost distributions for 1024 nodes (24 cloudlet and 1000 Raspberry Pi 4 edge nodes, 80% cost-metered), sampled from 1024 random node pairs.

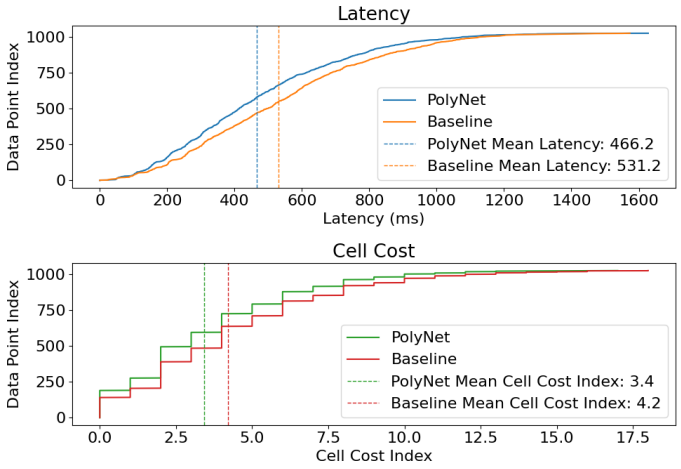

Fig. 4. Experiment 2: Latency and cell cost distributions for 1024 nodes (24 cloudlet and 1000 Raspberry Pi 4 edge nodes, 20% cost-metered), sampled from 1024 random node pairs.

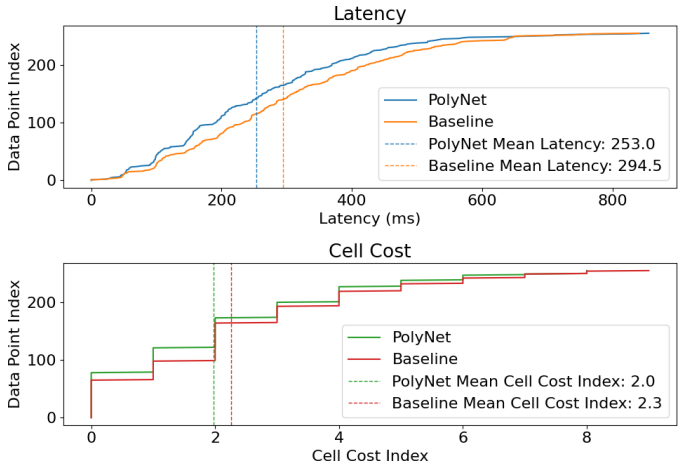

Fig. 5. Experiment 3: Latency and cell cost distributions for 256 nodes (6 cloudlet and 250 Raspberry Pi 4 edge nodes, 20% cost-metered), sampled from 256 random node pairs.

VIII. Conclusions and Future Work

This paper describes a novel approach to selecting overlay links for P2P VPNs that considers data cost and end-to-end latency, and experimentally demonstrates the performance improvements from supporting hybrid (user-level and kernel-level) tunnels bound to SDN switches. The approach utilizes cost information and Vivaldi network coordinates to select a target for long-distance links in the Symphony topology. It employs an MCDM method (TOPSIS), which can be generalized to incorporate additional criteria for making informed link selection decisions. Experimental results show that the TOPSIS-based approach outperforms the baseline random selection of Symphony by 14.1% in average latency and 19.1% in cost, using a simulated network of 1024 nodes representative of an edge-to-cloud continuum environment using a simulated network. Furthermore, experiments with a prototype demonstrate the feasibility of using hybrid links (GENEVE and WebRTC) in an implementation of an open-source P2P VPN, with significant performance improvements for intra-data-center links between edge and cloud devices.

While this study lays the foundation for demonstrating the benefits of using a TOPSIS-based approach to long-distance tunnel selection and SDN-enabled hybrid links for P2P VPNs, the multi-criteria approach can be enhanced and extended to other aspects of network optimization. Future work will aim to improve the decision-making process by incorporating link bandwidth and availability into the TOPSIS criteria and utilizing heuristics to assign weights to each criterion. It will also explore more comprehensive cost functions, taking into account various cellular data plans and the service costs associated with NAT traversal through TURN relays. Additionally, future studies will employ TOPSIS to determine the number of links for each node and assess whether a node should assume the role of an SDN switch within the network topology or function as a non-switched, pendant node connecting to other nodes via an intermediary SDN switch.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 131–145.

[2] K. Arabi, "Trends, opportunities and challenges driving architecture and design of next generation mobile computing and iot devices," in *Proc. MTL Seminar Series*, 2015.

[3] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "Policycop: An autonomic qos policy enforcement framework for software defined networks," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, IEEE, 2013, pp. 1–7.

[4] O. Blial, M. Ben Mamoun, and R. Benaini, "An overview on sdn architectures with multiple controllers," *Journal of Computer Networks and Communications*, vol. 2016, 2016.

[5] C. Chen, C. Liu, P. Liu, B. T. Loo, and L. Ding, "A scalable multi-datacenter layer-2 network architecture," in *Proceedings of the 1st ACM SIGCOMM symposium on software defined networking research*, 2015, pp. 1–12.

[6] "Cloudlab academic testbed." Accessed on: 2024-04-23. (2024), [Online]. Available: https://www.cloudlab.us/.

[7] R. Cohen, K. Barabash, and L. Schour, "Distributed overlay virtual ethernet (dove) integration with openstack," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, IEEE, 2013, pp. 1088–1089.

[8] P. Corcoran and S. K. Datta, "Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 73–74, 2016.

[9] L. D'Acunto, J. Pouwelse, and H. Sips, "A measurement of nat and firewall characteristics in peer-to-peer systems," in *Proc. 15-th ASCI Conference*, Citeseer, vol. 5031, 2009, pp. 1–5.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 15–26, 2004.

[11] V. Daneshmand, A. Breef-Pilz, C. C. Carey, *et al.*, "Edge-to-cloud virtualized cyberinfrastructure for near real-time water quality forecasting in lakes and reservoirs," in *2021 IEEE 17th International Conference on eScience (eScience)*, IEEE, 2021, pp. 138–148.

[12] D. Doval and D. O'Mahony, "Overlay networks: A scalable alternative for p2p," *IEEE Internet computing*, vol. 7, no. 4, pp. 79–82, 2003.

[13] K. Egevang, P. Francis, *et al.*, "The ip network address translator (nat)," RFC 1631, may, Tech. Rep., 1994.

[14] H. Eriksson, "Mbone: The multicast backbone," *Communications of the ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[15] "Ether: Edge topology synthesizer." Accessed on: 2024-04-23. (2022), [Online]. Available: https://github.com/edgerun/ether.

[16] B. Foubert and N. Mitton, "Rodent: A flexible topsis based routing protocol for multi-technology devices in wireless sensor networks," *ITU Journal on Future and Evolving Technologies*, vol. 2, no. 1, 2021.

[17] J. Gross, I. Ganga, and T. Sridhar. "Rfc 8926 - geneve: Generic network virtualization encapsulation." (2020), [Online]. Available: https://datatracker.ietf.org/doc/rfc8926/.

[18] R. L. Grossman, "The case for cloud computing," *IT professional*, vol. 11, no. 2, pp. 23–27, 2009.

[19] C.-C. Hung, G. Ananthanarayanan, P. Bodik, *et al.*, "Videoedge: Processing camera streams using hierarchical clusters," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, IEEE, 2018, pp. 115–131.

[20] "Iperf - the tcp, udp and sctp network bandwidth measurement tool." Accessed on: 2024-04-23. (2024), [Online]. Available: https://github.com/esnet/iperf.

[21] R. Jain, "Introduction to software defined networking (sdn)," *Washingt. Univ. Saint Louis*, vol. 11, no. 7, pp. 1–44, 2013.

[22] D. A. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle, "Ocala: An architecture for supporting legacy applications over overlays.," in *NSDI*, vol. 6, 2006, pp. 267–280.

[23] G. S. Manku and M. Bawa, "Symphony: Distributed hashing in a small world," in *4th USENIX Symposium on Internet Technologies and Systems (USITS 03)*, 2003.

[24] G. Nie, Q. She, and D. Chen, "Evaluation index system of cloud service and the purchase decision-making process based on ahp," in *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia: Volume 3: Computer Networks and Electronic Engineering*, Springer, 2011, pp. 345–352.

[25] Open Networking Foundation, "Openflow switch specification," Tech. Rep., 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[26] T. Rausch, C. Lachner, P. A. Frangoudis, P. Raith, and S. Dustdar, "Synthesizing plausible infrastructure configurations for evaluating edge computing systems," in *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*, USENIX Association, 2020.

[27] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[28] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[29] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (sdn): A reference architecture and open apis," in *2012 International Conference on ICT Convergence (ICTC)*, IEEE, 2012, pp. 360–361.

[30] S. Singh, "Optimize cloud computations using edge computing," in *2017 International Conference on Big Data, IoT and Data Science (BID)*, IEEE, 2017, pp. 49–53.

[31] SlackHQ. "Nebula: A scalable overlay networking tool with a focus on performance, simplicity and security." Accessed on: 2024-04-23. (2024), [Online]. Available: https://github.com/slackhq/nebula.

[32] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[33] K. Subratie, S. Aditya, V. Daneshmand, K. Ichikawa, and R. Figueiredo, "On the design and implementation of ip-over-p2p overlay virtual private networks," *IEICE Transactions on Communications*, vol. 103, no. 1, pp. 2–10, 2020.

[34] K. Subratie, S. Aditya, and R. J. Figueiredo, "Edgevpn: Self-organizing layer-2 virtual edge networks," *Future Generation Computer Systems*, vol. 140, pp. 104–116, 2023.

[35] K. Subratie, R. Ganesh, V. Daneshmand, P. Nagaraj, and S. Aditya, "Edgevpnio/evio: Release 24.1.1.147," DOI: 10.5281/zenodo.10583025.

[36] T. Theodorou and L. Mamatas, "Coral-sdn: A software-defined networking solution for the internet of things," in *2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*, IEEE, 2017, pp. 1–2.

[37] T. Theodorou, G. Violettas, P. Valsamas, S. Petridou, and L. Mamatas, "A multi-protocol software-defined networking solution for the internet of things," *IEEE Communications Magazine*, vol. 57, no. 10, pp. 42–48, 2019.

[38] J. Touch, "Dynamic internet overlay deployment and management using the x-bone," *Computer Networks*, vol. 36, no. 2-3, pp. 117–135, 2001.

[39] M. Uddin, S. Mukherjee, H. Chang, and T. Lakshman, "Sdn-based multi-protocol edge switching for iot service automation," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2775–2786, 2018.

[40] L. Wang, G. Von Laszewski, A. Younge, *et al.*, "Cloud computing: A perspective study," *New generation computing*, vol. 28, no. 2, pp. 137–146, 2010.

[41] D. Williams, H. Jamjoom, Z. Jiang, and H. Weatherspoon, "Virtualwires for live migrating virtual networks across clouds," *Report by Cornell University and IBM TJ Watson Research Center, New York*, 2013.

[42] World Wide Web Consortium, "Web real-time communications (webrtc)," Tech. Rep., 2023. [Online]. Available: https://www.w3.org/TR/webrtc/.