IEICE TRANS. COMMUN., VOL.E103–B, NO.1 JANUARY 2020

INVITED PAPER *Special Section on Internet Architecture, Applications and Operation Technologies for a Cyber-Physical System*

On the Design and Implementation of IP-over-P2P Overlay Virtual Private Networks*

Kensworth SUBRATIE[†a)], **Saumitra ADITYA**[†], **Vahid DANESHMAND**[†], ***Nonmembers***, **Kohei ICHIKAWA**[††], ***Member***, ***and*** **Renato FIGUEIREDO**[†], ***Nonmember***

SUMMARY The success and scale of the Internet and its protocol IP has spurred emergent distributed technologies such as fog/edge computing and new application models based on distributed containerized microservices. The Internet of Things and Connected Communities are poised to build on these technologies and models and to benefit from the ability to communicate in a peer-to-peer (P2P) fashion. Ubiquitous sensing, actuating and computing implies a scale that breaks the centralized cloud computing model. Challenges stemming from limited IPv4 public addresses, the need for transport layer authentication, confidentiality and integrity become a burden on developing new middleware and applications designed for the network's edge. One approach - not reliant on the slow adoption of IPv6 - is the use of virtualized overlay networks, which abstract the complexities of the underlying heterogeneous networks that span the components of distributed fog applications and middleware. This paper describes the evolution of the design and implementation of IP-over-P2P (IPOP) - from its purist P2P inception, to a pragmatic hybrid model which is influenced by and incorporates standards. The hybrid client-server/P2P approach allows IPOP to leverage existing robust and mature cloud infrastructure, while still providing the characteristics needed at the edge. IPOP is networking cyber infrastructure that presents an overlay virtual private network which self-organizes with dynamic membership of peer nodes into a scalable structure. IPOP is resilient to partitioning, supports redundant paths within its fabric, and provides software defined programming of switching rules to utilize these properties of its topology.

key words: *overlay networks, P2P, FOG, EDGE computing, SDN*

1. Introduction

The Internet's core protocol (IP) has been successful at an unprecedented scale. While the use of IPv4 identifiers and lack of security simplified the addition of devices to the Internet in its early days, they have turned into shortcomings as the network scales and security becomes a primary concern. In today's Internet, IPv4 addresses are scarce because of near-exhaustion of the 32-bit address space, and many networks have to resort to the use of private addresses and Network Address Translation (NAT) [1] middleboxes. Security concerns also have led to the proliferation of firewall middleboxes, while the need for authentication, confidentiality and integrity have led to transport-layer protocols (e.g. TLS [2], [3]). As a result, distributed applications that run across the Internet often must deal with devices without public IPv4 addresses that are behind various NAT and firewall middleboxes and must create secure transport sessions for communication. While these issues are relatively easy to handle with client-server applications, they place a burden to applications where peer-to-peer communication is needed.

Emerging distributed applications in edge/fog [4], [5] computing are poised to benefit from the ability for IoT and edge nodes to communicate in a peer-to-peer fashion. However, the connectivity challenges outlined above become an increasing burden on the development of middleware and applications as they move from the cloud to the edge [6], including security and privacy [7]. While the subsequent IPv6 protocol offers a larger address space and built-in security features, it is still not widely deployed despite decades since its release. An approach to address these challenges that does not require a core change to Internet protocols is to create overlay networks [8], [9] that tunnel traffic over the existing infrastructure. Coupled with network virtualization, overlays offer the ability to support existing middleware and applications, while shielding them from dealing with complexities due to private addressing, network address translation, and firewall policies.

This paper describes the design and implementation of IP-over-P2P (IPOP [10]–[15]), an overlay virtual private network that supports tunneling of layer-2 (Ethernet) and above (including IPv4) traffic across peer-to-peer tunnels. IPOP exposes virtual network interface endpoints that integrate with existing operating systems, automatically manages NAT traversal across peers with private IP addresses, self-organizes scalable overlay topologies and, supports encryption of peer-to-peer links, and allows software-defined programming of switching rules. IPOP provides a network virtualization substrate upon which geographically distributed IoT, edge, and cloud computing resources can be logically aggregated to facilitate the design of fog computing applications. This paper describes the evolution of the design and open-source implementation IPOP over several iterations of the project.

The advent of virtualization and cloud computing has

Manuscript received May 20, 2019.
Manuscript revised June 19, 2019.
Manuscript publicized August 5, 2019.

[†]The authors are with ACIS Lab, Electrical and Computer Engineering, University of Florida, USA.
[††]The author is with Division of Information Science at the Nara Institute of Science and Technology (NAIST), Ikoma-shi, 630-0192 Japan.

*This material is based upon work supported in part by the National Science Foundation under Grants No. 1527415, 1339737, 1234983 and 1550126. 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.

a) E-mail: kcratie@ufl.edu

DOI: 10.1587/transcom.2019CPI0001

fundamentally changed the way in which distributed applications and services are deployed and managed. With the proliferation of IoT and mobile devices, virtualized systems akin to those offered by cloud providers are increasingly needed geographically near the edge of the network [4]. Applications on resources at the edge can perform operations on high-volume data produced by sensors (e.g. real-time high-definition camera feeds) near IoT devices for latency-sensitive, bandwidth-intensive applications – a model referred to as fog computing [5]. Not only is performance important, but also a trustworthy network is key to guarantee privacy and integrity at the network layer across all participating resources (e.g. IoT, cloud VMs, and containers in edge resources [16]).

While software-defined virtual networking systems exist within large-scale cloud data centers [17] - at the core of the Internet and under a single administrative entity - these are not suited for future distributed applications spanning edge resources. Such fog applications are distributed across multiple providers and edge networks, raising more complex security and privacy issues than in cloud environments [6]. Unlike within a cloud, edge virtual networks require traversing multiple (possibly NATed [7]) administrative environments across different providers and enforcing data privacy and integrity in communication. While transport-layer network security (e.g. TLS, DTLS) and VPN tunneling (e.g. IPSEC) technologies exist, they are not readily applicable to systems where resource membership is dynamic, and where most devices are constrained by NAT/firewall middleboxes. Furthermore, the effort associated with developing or porting applications to enforce privacy and integrity in communications comes with significant costs.

To accomplish its required functionality, a fog application requires the ability to deploy, aggregate and process data from sensors on edge and cloud resources in a dynamic fashion. It also needs to support dynamic changes in the membership of participating devices over time, as devices may be mobile (e.g. video cameras in smartphones and vehicles). Fundamentally, the network connecting IoT, edge and cloud resources must provide trustworthy, seamless communication across a dynamic, heterogeneous, mobile set of resources.

The current version of IPOP implements a hybrid overlay/software-defined network (SDN [18]) software that is novel in how it supports dynamic grouping/aggregation of edge and cloud devices into a trustworthy virtual private network that leverages Online Social Network (OSN) interfaces for self-configuration. The following sections outline the core concepts in the design, and the evolution of the implementation leading to its current form.

2. Core Abstractions and Architecture

The core abstraction exposed by IPOP to a computer system using it is of a virtual network. The 1st generation of the designed exposed the abstraction of a layer-3 (IP) virtual network [10]–[12], while the 3rd and 4th generations expose the abstraction at layer-2 (Ethernet) [13], [14]. In both cases, the abstraction is exposed through a virtual network interface (VNIC), such as the TAP pseudo-device available in the Linux and Windows kernels, from which frames/packets that are sent and received are intercepted. IPOP nodes are implemented as a user-space process that runs in each node connected to the overlay; this process reads/writes from the TAP device, using the system call interface, as illustrated in Fig. 1.

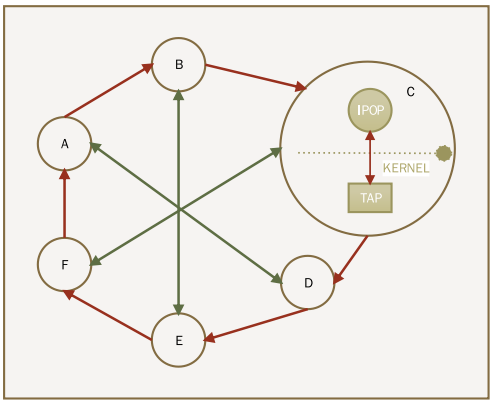


Fig. 1 IPOP's structured P2P architecture and virtualized endpoint. Nodes are connected in successive order based on their integer node identifiers (e.g. A < B < C < D) to form an outer "ring". Additionally, the overlay features shortcut links (e.g. A-D) which reduce routing complexity. In each IPOP node (e.g. C), there is a TAP device to pick/inject packets from the O/S kernel, and a user-mode IPOP application that implements the overlay virtual network functionality.

IPOP nodes form virtual links among each other, where each virtual link is an Internet tunnel that carries encrypted and encapsulated virtual network frames/packets through a transport protocol – typically UDP, which is more amenable to NAT traversal than TCP.

Each IPOP node in an overlay is uniquely identified by a node ID (e.g. A, B, ..., F) in Fig. 1. The set of virtual links among IPOP nodes forms a topology. While different overlay topologies have been implemented in IPOP versions over time, a structured P2P topology has been a feature of the design since its inception. In this topology, nodes form a logical ring, with successor links ordered by their node IDs, and shortcut links across the ring, following a structured P2P algorithm for topology construction and identifier-based routing such as Chord [19] or Symphony [20].

3. Fully Decentralized P2P Design

The 1st generation of IPOP [10] was fully decentralized, following a structured P2P design using the Brunet [21] library and a Symphony-based protocol. Two key motivations for this approach were to avoid any external dependences, and any single point of failure. Each node implemented the functionality to 1) discover other nodes by means of a peer list file, 2) bootstrapping by contacting nodes in the peer list and using them to send messages to the joining peer's left

and right neighbors, 3) providing support for discovery of a node's public IP:port endpoint, and 4) forwarding messages according to a greedy routing algorithm.

This implementation provided a layer-3 virtual network for the IPv4 protocol. There was no encryption, and no authentication of nodes into the overlay. IP addresses were assigned to nodes by leveraging a Distributed Hash Table (DHT [19]) key/value store which was also implemented by IPOP nodes. The DHT used a compound key, comprising of a virtual network namespace and a virtual IP address to map the node's IPOP ID. This allowed multiple virtual networks to share a single overlay without collision of private subnets. IPOP nodes also included an implementation of a user-level DHCP server that handed out IP addresses by randomly assigning an address within a declared virtual network namespace and inserting the mapping into the DHT [22].

While this implementation proved to be useful in many scenarios, and resilient to failures and churn, it had several shortcomings. First, the monolithic design led to a complex node that implemented several modules for discovery and bootstrapping, DHT, overlay routing, NAT traversal, IP address assignment/mapping, as well as virtual network interface bindings. Second, the use of a monolithic design implemented as an application written in C# made it difficult to incorporate standards and functionality in libraries written for different languages into a single process. In particular, the ICE, STUN, and TURN [23], [24] protocols for NAT traversal were not supported, as well as transport-layer security. Third, the design did not provide a mechanism to authenticate peers into the overlay. Fourth, rules for packet forwarding and header manipulation, such as IP mapping, were implemented in the monolithic node, preventing the ability to change them without significant code investment. These shortcomings were progressively addressed in subsequent implementations of IPOP, as described in the next sections.

4. Decoupling Endpoint Discovery from Overlay

To tackle the complexities of overlay membership, endpoint discovery and bootstrapping links - which stem from its initial design - the 2nd generation of IPOP [12] introduced a signaling process reliant on online social networks (OSN). Each host in an overlay is represented and identified by an OSN Identity (ONSI) which exists on an OSN server. The ONSI maintains the notion of a social network (i.e., its private roster of friends) which corresponds to the peer hosts that participate in its overlay.

By using an XMPP [25] compliant instant messaging service, issues of overlay membership and credentialing are handled externally by a service that follows a widely used standard, and that supports the authentication of user accounts and the establishment of trust relationships between users. Additionally, a published service eliminates the need to have prior knowledge of online nodes in order to join the overlay. Instant messaging provides the facility to exchange bootstrapping data that is used in support of other established standards, e.g., ICE, STUN and TURN. Collectively, these changes decouple endpoint discovery and connection bootstrapping from the overlay.

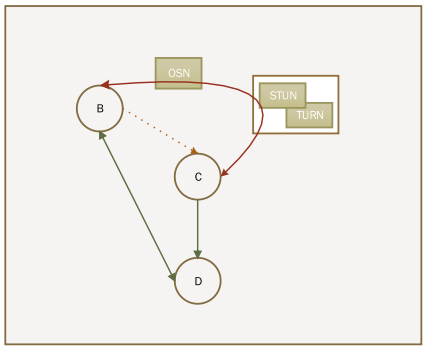


Fig. 2 Illustration of decoupling endpoint discovery from IPOP overlay. Shown is segment (B, C, D) of the overlay in Fig. 1. Provided that nodes B and C have a trust relationship recorded in the OSN, they use external services (STUN, TURN) to discover their addressable endpoints and share them via the OSN. Once endpoint and certificate fingerprints are exchanged, a successor link (in red) is established. The integrity and confidentiality of link communication is enforced by DTLS.

Whereas prior to this approach, anyone with the software could join the single global overlay for all participants, OSNs now provided the necessary mechanism to restrict the participants of an overlay, and subsequently define multiple separate overlays. An ONSI is protected by credentials which requires each identity to authenticate with a centralized OSN server (or federation) prior to using its services. Once authenticated, the OSNI discovers its roster of friends which it interprets as an indication of direct acquaintance and mutual trust. Friends are therefore used to identify the network endpoints that are eligible to participate in the overlay network. In this regard, the overlay network is the realization of an individual's social relationships. The view of social relationships in this approach is taken broadly – on one hand it may be mapped to resources owned by multiple individuals connected by a social network, while on the other hand it may be mapped to resources managed by a single (or federated) administrative domain with trust relationships capturing the membership in an overlay.

When an OSNI signs on, a presence message is broadcasted to all its available friends. This presence awareness is used to trigger the process of negotiating peer links. Establishing peer links using ICE requires each node to discover and share endpoint data with its peer – all of which must occur before a communication channel between the peers is established. The instant messaging facility of the OSN is utilized to exchange messages which indicate the intent to create a P2P channel as well as exchange the necessary bootstrapping data between peers. This data includes the peer UUID, certificate fingerprints, and network address endpoints that will ultimately be used for creating all the facilities of the tunnel.

5. Decoupling Control from Datapath

Another enhancement introduced in 2nd generation IPOP was the separation of concerns between control and data-path. Rather than the monolithic process of its initial implementation, starting in [12], IPOP adopted an SDN-inspired modular design of controller and data-path components.

The IPOP datapath (Tincan) builds on WebRTC [26], [27] to create its communication links. WebRTC is an open standard and industry effort to enable direct, real-time media communication between browsers. Tincan utilizes the WebRTC native C++ libraries for data channels which constitutes the virtual link between peers. These virtual links use either direct, reflexive, or relay connections. Direct connection occurs when two nodes have routable endpoints using their local IP address (e.g. within a LAN); reflexive connections occurs when nodes are behind cone-style NATs that are amenable to STUN-based traversal; relay connections occurs when an intermediary server is necessary to exchange messages. Tincan also handles the IO interaction between the VNIC and the virtual link. Ethernet frames read from the VNIC are encrypted using DTLS, encapsulated as the payload of an IP datagram and transmitted on the link. Conversely, incoming messages are stripped of the UDP headers, decrypted and written to the VNIC.

The 3rd generation IPOP Controller uses a modular framework which separates the application framework from the modules which implement specific functionalities [14]. The controller framework loads and initializes a parameterized list of modules at startup, providing an asynchronous task-based messaging service for inter-module communications. The Controller Brokered Task (CBT) abstraction is used for this purpose; it is a self-contained structure that fully describes the task over its lifetime, including all the details pertaining to its request and response. A CBT is created by a module, submitted to the framework, and delivered to the recipient modules work queue. When the requested task is completed, the CBT is returned to the initiator via the framework.

A control module is a component with an application-specific role within the IPOP Controller. By implementing a framework defined interface, modules can be loaded and initialized, sent CBTs for processing, and invoked at periodic intervals. Modules can be created for any purpose to extend the capabilities of the IPOP Controller. A few core modules include signaling, link negotiation and creation, topology definition, and status reporting.

The signaling module leverages XMPP to advertise presence, indicate intent, and exchange connection bootstrap data. At sign-on, and on periodic intervals, a node broadcasts a presence message to all peers on its friendship roster indicating its availability for a communication channel. This module services requests that require peer communication over the XMMP band.

The link management module manages tunnels between peers, mirroring the notion of the link layer between two networked devices. It creates, maintains and destroys tunnels as a service to other modules and utilizes the signal module to indicate the intent to create a link as well as exchanging the link bootstrapping data. Additionally, it tracks the VNIC and link for each tunnel and instructs the data-path to create them – coordinating as an intermediary the CAS exchange handshake between the local and peer data planes.

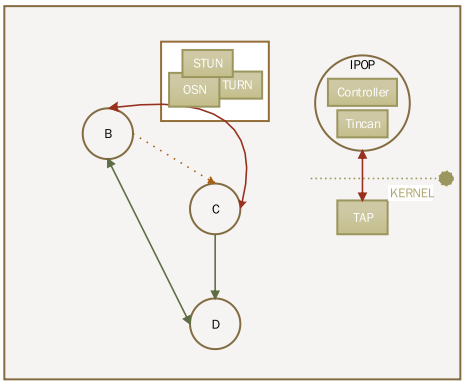


Fig. 3 Decoupling of the IPOP node design into control and data-path modules. The data-path module is responsible for packet capture/injection, as well as for the setup and maintenance of peer-to-peer private tunnels through which virtual network traffic flows. The control module is responsible for signaling, link management, and topology management, among other functions. These modules are implemented as separate processes, written in separate languages (currently, a Python controller and C++ datapath) that execute in the same node and communicate via a localhost network API.

The topology module determines the placement of tunnels and their duration. It utilizes the link management services for the creation of individual tunnels and orchestrates the local node's participation in the construction of the global structure. The 4th generation implementation is a structured P2P topology based on a successor ring with shortcut paths.

Monitoring of the overlay state is accomplished by cooperative work among the reporting module and the other controller. Participating modules periodically submit their respective state to the reporting module which aggregates the data into a node wide representation. The node data is sent to a central collector webservice which aggregates node data into a global view encompassing all the reported overlays and their respective nodes.

6. Software-Defined Switching

In its 4th generation, IPOP moved from a layer-3 to a layer-2 virtual network and implemented an OpenFlow SDN controller. This allows IPOP to support broadcast/multicast protocols other than IP (e.g. ARP), relocatable IP addresses, and a wider variety of applications. Furthermore, supporting the OpenFlow API and SDN-based software switches, opens the virtualized overlay to a variety of possible networking implementations.

IPOP virtualizes a layer 2 broadcast domain via the lo-

cal system TAP device or an OpenFlow software bridge, and by tunneling Ethernet frames. The ring structure with shortcut links provides multiple redundant paths between network switches in the overlay. If utilized, these links provide alternate paths to avoid IO bottlenecks and resilience against link failures. However, typical Ethernet switching does not accommodate a topology with cycles, and network failure from broadcast storms occurs if cycles are not disabled. Approaches such as Spanning Tree Protocol (STP) are used in local-area Ethernet networks to selectively disable links and construct a cycle-free spanning tree. Unfortunately, this approach ignores functional links that could otherwise be used as alternative paths between pairs of communicating hosts. To address the issues stemming from cyclic paths while still retaining the functional benefits of multiple links, an OpenFlow compliant switching protocol, called Bounded Flooding, has been designed and implemented for structured P2P topologies in IPOP.

Each IPOP node instantiate two controller components: an IPOP controller and an OpenFlow controller module – producing a decentralized system. The IPOP controller builds and maintains the topology while the OpenFlow controller programs the switching rules. The OpenFlow controller functionality is topology dependent and relies on structural information queried from the IPOP controller, such as the node's adjacency list. It is designed to be scalable from overlays with two nodes to thousands of nodes.

The IPOP topology is a ring of successors with identifiers increasing clockwise and arcs of shortcut links. Each switching node running IPOP is assigned a UUID, and each node bears the responsibility of identifying its closest neighbor (next larger UUID) and building a successor link between them; a node may select more than one successor for resilience against interruptions from churn. This process continues with each node until the ring is complete. The number of nodes in the overlay are proportional to the average number of switching hops to deliver frames in the overlay and subsequently is used as a measure of the perceived latency in communication. While a complete network would provide constant switching cost, it is infeasible for overlays with more than tens of nodes. For an overlay with n nodes, each new node adds $(n-1)$ edges for a total $n(n-1)/2$ edges. As each tunnel incurs an ongoing communication cost over its lifetime this approach does not scale as maintenance data begins to saturate the network.

Shortcut links can used to bound the average latency, and using $\log_2(n)$ links per node provides an equivalent bound [19]. To choose a suitable peer for the shortcut links the Symphony Long Distance Links [20] selection algorithm is used. The overlay is parameterized to tune the tradeoffs between node degree and latency, and each node can be independently configured.

Topology data provided by the IPOP controller is used to distinguish between directly connected peer bridges and leaf devices, and to associate the leaf devices with their respective root bridge. Knowledge of a device's root bridge is necessary for building on demand tunnels which are a mechanism for eliminating additional switching hops between actively communicating leaf devices.

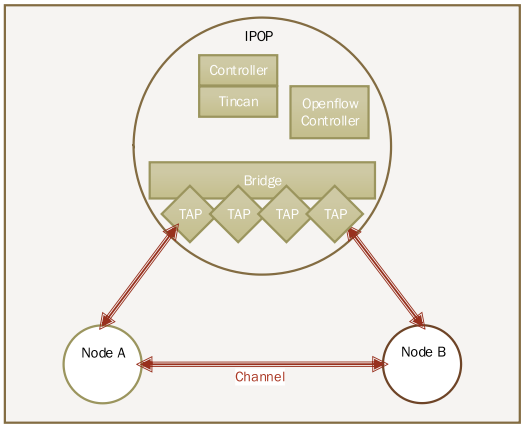


Fig. 4 IPOP functions as a switch by extending a software bridge such as Open vSwitch. Tunnels are added to the bridge as subordinate interfaces as they are created and removed when destroyed.

The OpenFlow controller implements the novel bounded flooding protocol, which is fundamental to building its learning table (a mapping of observed source MAC address and the associated ingress ports) and subsequently programming the data plane flow rules. Whenever a broadcast is necessary, a Flooding Route and Bound (FRB) is used to greedily flood all its peer ports, i.e., egress ports that connect to another bridge. FRB is a custom Ethernet layer protocol implemented and used by IPOP SDN switching to perform link layer broadcasts in dynamic cyclic switched fabrics. The FRB prefixes the original broadcast with a header describing the root bridge and the message's bound. The root bridge is the switch that manages the device that initiated the broadcast. The flooding bound is a closed-open interval $[i, j)$, specifying the recipient i, and the furthest node j, the message should be forward. The recipients are adjacent peers and each bound can potentially differ. On receiving an FRB, a node will deliver the payload on its managed leaf ports, determine if any adjacent peer bridges are within the message bound, recalculate and update the bound as necessary, and retransmit the message. Retransmitting an FRB is done clockwise around the ring on successor and shortcut edges. This approach ensures that broadcasts are never duplicated, are delivered to all devices in the overlay, and will eventually terminate. As FRBs are propagated throughout the overlay, they are tracked at each node and used to update its local learning table. This information collectively provides a return route across the overlay to the FRB initiator. Additionally, an FRB is used to exchange the managed leaf devices for a switch with its peer.

7. Discussion

7.1 Related Work

Two related approaches to virtualized overlay networks are VIOLIN [28] and VNET/P [29]. VIOLIN uses a virtualized

Table 1 Taxonomy of IPOP generations.

Taxonomy	1st Generation	2nd Generation	3rd Generation	4th Generation
Bootstrapping	Decentralized - requires peer list	Centralized OSN	Centralized OSN	Centralized OSN
Endpoint Addressing	IP4, fixed to node	IP4, fixed to node	MAC, user configurableIP4 or IP6, relocatable IP	MAC, user configurableIP4 or IP6, relocatable IP
Forwarding	Structured P2P, Greedy	Direct Link	STP with MAC Learning	Bounded Flooding
Overlay	Global with namespaces	Independent overlay per process	Multiple independent overlays in single process	Multiple independent overlays in single process
Scalability	Complete decentralization	Decentralized topology and forwarding	Decentralized topology and forwarding	Decentralized topology and forwarding
Software Architecture	Monolithic	Single purpose Controller, Tincan	Controller framework, Specialized modules, Tincan	Controller framework, Specialized modules, Tincan, SDN Controller
Support for Standards	No ICE, No XMPP, No OpenFlow	ICE, XMPP, No OpenFlow	ICE, XMPP, No OpenFlow	ICE, XMPP, OpenFlow
Topology	Structured P2P/Symphony	Social and All-to-All	All-to-All	Structured P2P/Symphony
Virtualization Layer	Layer 3/IP	Layer 3/IP	Layer 2/Ethernet	Layer 2/Ethernet

layer 2/layer 3 (Ethernet/IP) model, and all components – vHost, vNic, vSwitch, and vRouter – are virtualized. VIOLIN's goal is to isolate the vHost within the overlay from the public Internet. This approach allows the complete network to be orchestrated by software, but it has no mechanism for peer identification, authenticating overlay membership, topology planning, or NAT traversal – all which are necessary for dynamic management. Also, in the absence of programable switching routes, standard layer 2 switching (MAC learning) is implied; furthermore, IP routes must be manually configured at each vHost. By comparison it is less scalable as each layer 2 domain must be a spanning tree which results in bottlenecks at high contention overlay links. The manual specification of topology, switching and routing rules means longer deployment and reconfiguration times. While the communication channels are encrypted, any host with the overlay peer list can join.

VNET/P presents a layer 2 network that is geared at tightly coupled HPC workloads. It implements its switching core in the Palacios hypervisor and a tunneling bridge in the host kernel. The overlay topology, nodes and switching routes are defined in a static configuration file which are generated and validated by a separate centralized management tool with a global view of the network. The configuration must be regenerated to reflect any changes and reapplied to the applicable hosts. VNET/P also supports encrypted communication channels but no authentication method for peer membership. These properties result in less scalable overlays, which are open to any host with the peer list, and increased response times to changes in the overlay when compared to Bounded Flood.

7.2 P2P Architecture

While both unstructured and structure P2P schemes have their respective advantages and disadvantages [30], the structured P2P properties inherent to Symphony are particularly beneficial to IPOP. Specifically, these are with respect to decentralization, topology, system parameters, routing performance and state, and resiliency. Being decentralized the system is highly scalable, supporting very large networks. Its topology is simple to create and maintain in realistic deployments, even in systems with frequent churn where nodes have short lifespans. The system parameters, those that significantly influence the system characteristics, are primarily the number of nodes in the overlay and the number of links per node, k. Furthermore, each node within the system can be independently configured. Routing and state costs are bounded as a function of the systems parameters and at limits that are amenable to deployments at large scale. The routing cost is for symphony $1/k\ log(n)$, and its state varies with k. It is a resilient system such that node failure will not cause enduring network-wide failure. Additionally, the overlay can tolerate up to f node failures without partitioning by using $s = f$ successor links per node.

8. Example Use Cases

PerSoNet [31] built upon the virtualized layer 2 functionality provided by IPOP to create a two-layered, software defined VPN network for community-based collaboration at the edge. The Personal layer connects devices owned and managed by an individual exposing OSI Layer-2 semantics.

Above it, connectivity among devices belonging to different peers is provided by interconnecting the personal networks of individuals via SDN-programmable gateways to create a Community layer overlay network with OSI Layer-3 semantics. PerSoNet abstracts the complexities involved in managing IP layer addressing, device name management, access control and connectivity by a combination of DNS query interception and processing along with reactive programming of software defined gateways to perform address translation and switching.

GRAPLE is an inter-disciplinary collaboration between computer scientists and lake modelers associated with two international networks, GLEON (Global Lake Ecological Observatory Network) and PRAGMA (Pacific Rim Applications and Grid Middleware Assembly). The GRAPLE collaboration's main software product is GRAPLEr [32]. While it is relatively easy to run one lake model simulation on a personal computer, it is more challenging to execute multiple simulations, which requires additional computing and human resources. To overcome this constraint, GRAPLEr, as a distributed computing system, integrates and applies the IPOP overlay virtual network to support high-throughput computing, and Web service technologies. By using IPOP as a virtual network substrate, GRAPLEr is able to reuse existing, unmodified software, including the HTCondor job scheduler, and to enable simple addition of resources across the Internet to the HTCondor pool. GRAPLEr allows submission of hundreds or thousands of General Lake Model (GLM) simulations, runs these lake model simulations efficiently, and retrieves the model output.

PRAGMA Experimental Networking Testbed (PRAGMA-ENT) is an international SDN testbed that is designed to offer complete freedom for researchers to access network resources to develop, experiment, and evaluate new ideas without interfering with a production network. PRAGMA-ENT connects OpenFlow-enabled switches in different institutions over high-speed networks and builds a large scale OpenFlow-based network testbed. In this use case scenario, IPOP has been used to extend the testbed with a software-defined overlay, in order to enable sites that do not have OpenFlow-enabled switches nor a direct connection to the backbone of PRAGMA-ENT. Using IPOP, each end-user can deploy an access network between their resources and the nearest PRAGMA-ENT endpoint. This approach works effectively for those end-users who just need to connect their resources to PRAGMA-ENT temporarily for their experiments.

9. Conclusion

We have traced the evolution of IPOP from its P2P origins in Brunet to its current hybrid implementation, showing how specific needs of a real-world system have driven pragmatic design changes at each stage. These changes allow IPOP to fill a gap in emergent technologies and seamlessly integrate existing applications. We have also illustrated real world systems that utilize IPOP's capabilities, demonstrating its practical applications.

Generation 1 virtualized a layer 3 network through VNIC host integration and tunneling IP packets within UDP/IP. It was a fully decentralized overlay, utilizing the Brunet library. The architecture reflected a monolithic process that incorporated all services for bootstrapping and packet forwarding which made interoperability with open standards difficult. The global overlay used namespace identifiers and DHT store to provide scope for IP subnetting. Joining the overlay required a peer list and used overlay links to carry bootstrap messages, and there was no support for authentication or encryption.

Generation 2 introduced OSNs to decouple endpoint discovery from the overlay and introduced a client-server model for bootstrapping. While no longer a pure P2P model, using a published OSN server and friendship relationships facilitated independent overlays, simplified bootstrapping and enforced authentication for membership. The monolithic process was separated into a control and data plane and standards such as ICE, STUN, TURN was adopted. Moving from the Brunet library meant losing the Symphony structure; to accommodate this links were create to all friends and IP mapping performed between them. An SDN inspired approach was adopted which split IPOP into controller and data plane processes. However, the coarse granularity of the architectural components still left code maintenance and enhancements challenging.

Generation 3 addressed the architectural problems by introducing the controller application framework. The controller utilized modular components focused on topology, link management and signaling. Improvements to the data plane supported multiple, concurrent, isolated layer 2 overlays within a single process.

Generation 4 reintroduced the Symphony structured ring topology, implemented in an IPOP controller module. To provide the switching routes for the specialized topology, a Ryu/OpenFlow module implementing Bounded Flood is used. Both the algorithms for the topology and switching rules were designed to be parameterized and scalable to function for networks as small as two nodes to hundreds on nodes.

References

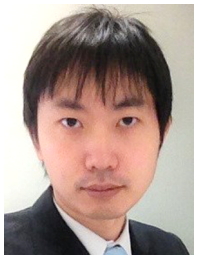
[1] P. Srisuresh and K. Egevang, "Traditional IP network address translator (traditional NAT)," 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc3022. [Accessed: 13-Jun-2019].
[2] E. Rescorla, "The transport layer security (TLS) protocol version 1.3," 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446. [Accessed: 13-Jun-2019].
[3] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6347. [Accessed: 13-Jun-2019].
[4] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J.P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," J. Syst. Architect., vol.98, pp.289–330, Feb. 2019.
[5] N. Bessis and C. Dobre, eds., Big Data and Internet of Things: A

Roadmap for Smart Environments, vol.546, Springer International Publishing, Cham, 2014.
[6] H. Chang, A. Hari, S. Mukherjee, and T.V. Lakshman, "Bringing the cloud to the edge," 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp.346–351, 2014.
[7] Y. Guan, J. Shao, G. Wei, and M. Xie, "Data security and privacy in fog computing," IEEE Netw., vol.32, no.5, pp.106–111, Sept. 2018.
[8] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," Proc. Eighteenth ACM Symposium on Operating Systems Principles, pp.131–145, New York, NY, USA, 2001.
[9] H. Eriksson, "MBONE: The multicast backbone," Commun. ACM, vol.37, no.8, pp.54–60, Aug. 1994.
[10] A. Ganguly, A. Agrawal, P.O. Boykin, and R. Figueiredo, "IP over P2P: Enabling self-configuring virtual IP networks for grid computing," Proc. 20th IEEE International Parallel Distributed Processing Symposium, p.10, 2006.
[11] D.I. Wolinsky, Y. Liu, P.S. Juste, G. Venkatasubramanian, and R. Figueiredo, "On the design of scalable, self-configuring virtual networks," Proc. Conference on High Performance Computing Networking, Storage and Analysis, pp.1–12, 2009.
[12] P.S. Juste, K. Jeong, H. Eom, C. Baker, and R. Figueiredo, "TinCan: User-defined P2P virtual network overlays for ad-hoc collaboration," EAI Endorsed Trans. Collab. Comput., vol.1, no.2, p.15, Oct. 2014.
[13] K. Subratie and R. Figueiredo, "Towards island networks: SDN-enabled virtual private networks with peer-to-peer overlay links for edge computing," Internet and Distributed Computing Systems, pp.122–133, 2018.
[14] K. Subratie, S. Aditya, S. Sabogal, T. Theegala, and R. Figueiredo, "Towards dynamic, isolated work-groups for distributed iot and cloud systems with peer-to-peer virtual private networks," Sensors to Cloud Architectures Workshop (SCAW-2017), Austin, Texas, USA, 2017.
[15] ACIS, "IP Over P2P Virtual Private Network," IPOP VPN, 2006. [Online]. Available: http://ipop-project.org/. [Accessed: 20-Mar-2015].
[16] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud PaaS architecture based on raspberry Pi clusters," 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp.117–124, 2016.
[17] C. Chen, C. Liu, P. Liu, B.T. Loo, and L. Ding, "A scalable multi-datacenter layer-2 network architecture," Proc. 1st ACM SIGCOMM Symposium on Software Defined Networking Research, pp.8:1–8:12, New York, NY, USA, 2015.
[18] 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., vol.38, no.2, pp.69–74, March 2008.
[19] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Trans Netw., vol.11, no.1, pp.17–32, Feb. 2003.
[20] G.S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," Proc. 4th Conference on USENIX Symposium on Internet Technologies and Systems, vol.4, p.10, Berkeley, CA, USA, 2003.
[21] P.O. Boykin, J.S.A. Bridgewater, J.S. Kong, K.M. Lozev, B.A. Rezaei, and V.P. Roychowdhury, "A symphony conducted by Brunet," ArXiv:0709.4048 Cs, Sept. 2007.
[22] A. Ganguly, D. Wolinsky, P.O. Boykin, and R. Figueiredo, "Decentralized dynamic host configuration in wide-area overlays of virtual workstations," 2007 IEEE International Parallel and Distributed Processing Symposium, pp.1–8, 2007.
[23] J. Rosenberg, "Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols," 2010. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5245.txt
[24] R. Mahy, P. Matthews, and J. Rosenberg, "Traversal using relays around NAT (TURN): Relay extensions to session traversal utilities for NAT (STUN)," 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5766. [Accessed: 13-Jun-2019].
[25] E.P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core," 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3920. [Accessed: 13-Jun-2019].
[26] S. Loreto and S.P. Romano, Real-Time Communication with WebRTC: Peer-to-Peer in the Browser, O'Reilly Media, 2014.
[27] "WebRTC 1.0: Real-time Communication Between Browsers." [Online]. Available: https://w3c.github.io/webrtc-pc/. [Accessed: 29-Jun-2018].
[28] X. Jiang and D. Xu, "VIOLIN: Virtual internetworking on overlay infrastructure," Parallel and Distributed Processing and Applications, pp.937–946, 2005.
[29] L. Xia, Z. Cui, J.R. Lange, Y. Tang, P.A. Dinda, and P.G. Bridges, "VNET/P: Bridging the cloud and high performance computing through fast overlay networking," Proc. 21st International Symposium on High-Performance Parallel and Distributed Computing, pp.259–270, New York, NY, USA, 2012.
[30] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," IEEE Commun. Surveys Tuts., vol.7, no.2, pp.72–93, Second Quarter 2005.
[31] S. Aditya, K. Subratie, and R.J. Figueiredo, "PerSoNet: Software-defined overlay virtual networks spanning personal devices across social network users," 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp.171–180, 2018.
[32] K.C. Subratie, S. Aditya, S. Mahesula, R. Figueiredo, C.C. Carey, and P.C. Hanson, "GRAPLEr: A distributed collaborative environment for lake ecosystem modeling that integrates overlay networks, high-throughput computing, and WEB services," Concurr. Comput. Pract. Exp., vol.29, no.13, p.e4139, 2017.

Kensworth C Subratie is a Ph.D. Candidate at the Department of Electrical and Computer Engineering (ECE) of the University of Florida. His research interests include computer communications and networked systems, software systems design, edge computing and IoT.

Saumitra Aditya is a Ph.D. Candidate at the Department of Electrical and Computer Engineering (ECE) of the University of Florida. His research interests broadly span networked systems and their applications in realization of smart communities.

Vahid Daneshmand is a Ph.D. student at the Department of Electrical and Computer Engineering (ECE) of the University of Florida. His research interests include distributed computing, serverless architecture, and IoT.



Kohei Ichikawa is an Associate Professor in the Division of Information Science at the Nara Institute of Science and Technology (NAIST) Japan. He received his B.E., M.S., and Ph.D. degrees from Osaka University in 2003, 2005, and 2008, respectively. He was a postdoctoral fellow at the Research Center of Socionetwork Strategies, Kansai University from 2008 to 2009. He also worked as an Assistant Professor at the Central Office for Information Infrastructure, Osaka University from 2009 to 2012. His current research interests include distributed systems, Software Defined Networking, and the related information technologies.

Renato J. Figueiredo is a Professor at the Department of Electrical and Computer Engineering (ECE) of the University of Florida. Dr. Figueiredo received the B.S. and M.S. degrees in Electrical Engineering from the Universidade de Campinas in 1994 and 1995, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Purdue University in 2001.