A Distributed and Scalable Compute Platform For Data Networks Emulation

Last Updated: July 31st 2023
Latest Copy:

Introduction

Emulating complex network topologies has always been a challenge. This problem multiplies by a manifold when working with different products from different vendors. Further, due to high end resource requirements for most of the network routers/switches (VNFs), emulating a data network close to reality becomes almost impossible, especially when constrained under a single server.

To solve this problem, a generic distributed platform is envisioned, where in users can build small and large data networks alike using virtualized products from various vendors (VMs and Containers) seamlessly using overlay Software Defined Networking (SDN)

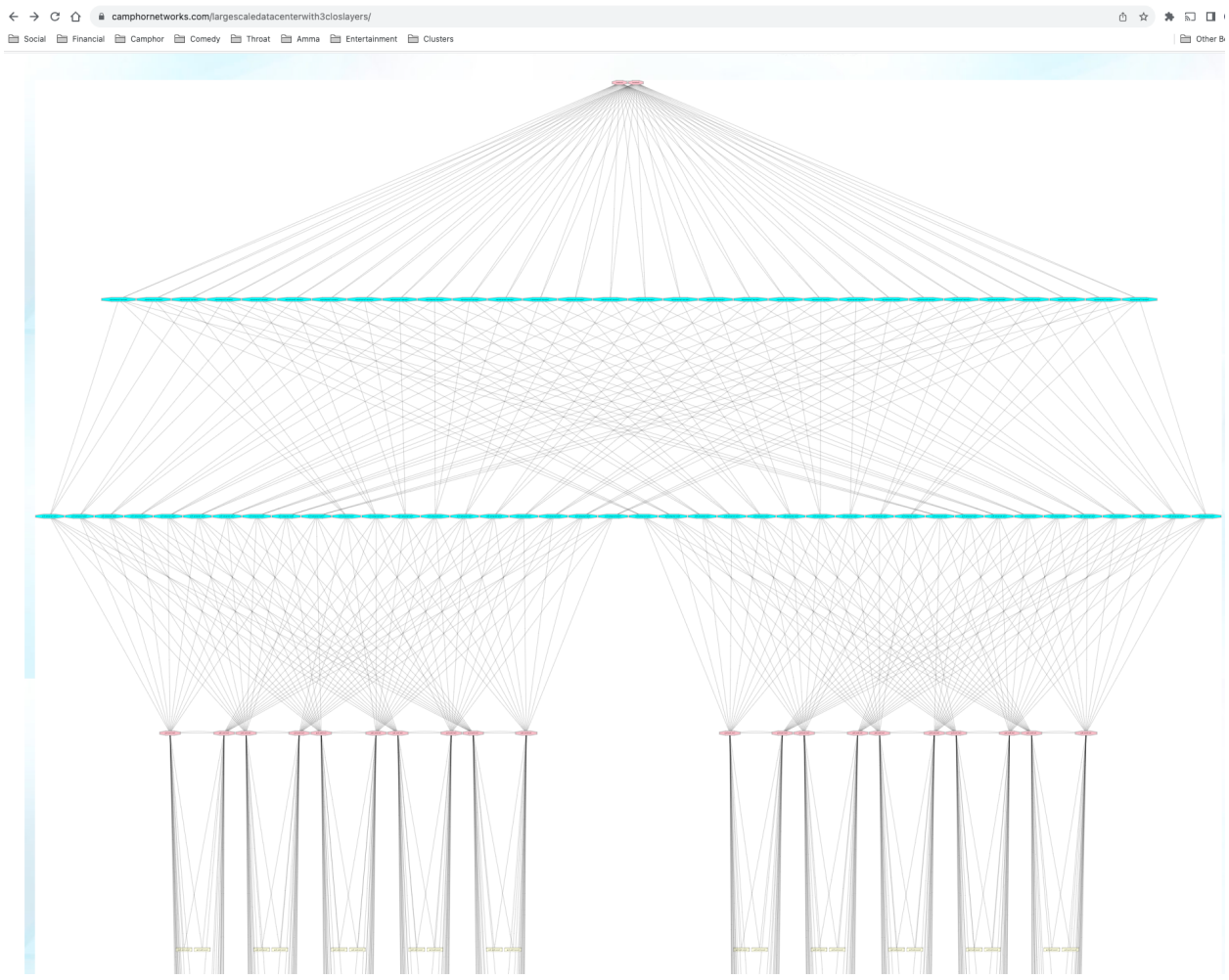
While SDN has played a key role in many production traffic carrying data networks, it has not been effectively employed in the areas of network simulation and testing. Due to this, there seems to be a large gap between the kind of testing that is currently being done in the labs to what indeed can be achieved in reality.

In this paper, an architecture is proposed where in distributed systems (aka servers) are used infrastructure to carry one or more emulated data networks. The design shall provide all three - flexible management plane, scalable control plane and high performant data plane like a real network demands.

Architecture

A distributed set of servers typically provide a farm-house of CPUs, Memory and IO. Further, they are connected by a high speed L3 Network so that large amounts of data can be exchanged at high speeds. This provides a great infrastructure for large scale distributed networks emulation. But the main caveat is networking itself. Large scale networks typically encompass many different network elements each with multiple network interfaces. The interfaces are connected in myriad fashion, to carry many different traffic such as IPv4 and IPv6, typically over Layer 2 Ethernet frames.

For example, in a large scale Data Center, one would expect many hundreds of servers, routers and connections among the various elements as depicted in this network topology diagram. This data-center contains 300 devices, 900 connections and around 1800 ports inter-connected in CLOS fashion.



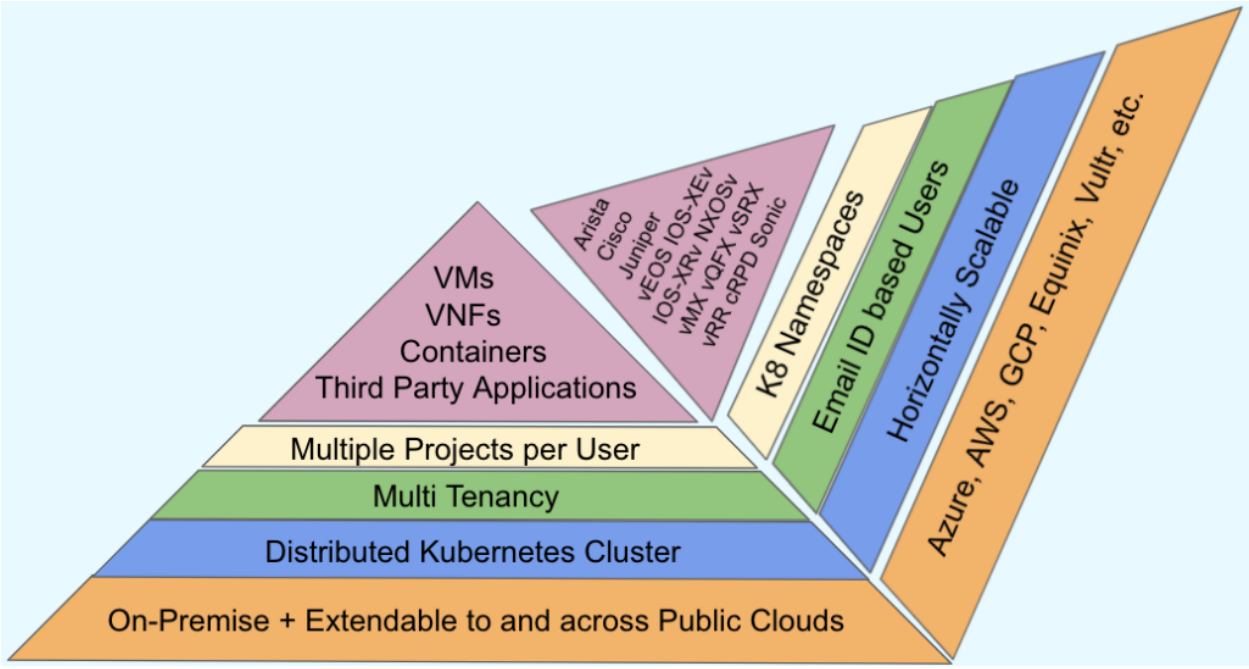
A good analogy would be to use Kubernetes as the distributed compute platform which provides a seamless infrastructure to orchestrate containers across the underlying servers.

By using the power of SDN, small and large data networks can be effectively built literally in minutes, by using the total computing capacity of the distributed cluster in a niche manner.

Design

Proposed solution is to extend Kubernetes to achieve several things.

1. Seamless ability to launch Virtual Machines also, along with Containers
2. Use a Overlay SDN technology (VXLAN, L2TP, MACVLAN, etc.) to provide seamless connectivity among various router interfaces as configured in the topology
3. Ensure multi-vendor products are supported by porting variety of products to the proposed solution
4. Ensure key networking features are met such as
    a. Layer 2 Connections across router interfaces connected
    b. Small and Large sized Packets (aka Jumbo Frames)
    c. High throughput
    d. Low latency
    e. Distributed design for best performance
    f. Ability to deploy many different network protocols such as L3VPN, L2VPN, Multicast, Segment Routing, BGP, MPLS, etc. as applicable
    g. Ease of use despite the complexity involved underneath in the distributed infrastructure that carries the emulated network traffic
5. The solution ideally should be amenable to run on-premise as well as in public clouds for ease of use and quick adaptation
6. Seamless integration with third-party and especially open source based software would bring in a lot more value to the proposed solution. Some examples include OpenNMS, ELK Stack, SuzieQ Network Explorer, etc. Such applications should be able to easily interact with various network elements over the management and/or data plane to realize their full potential for efficient network testing and management



An example of a hierarchical design that meets the proposed goals!

Hybrid Networks

No network emulation model is useful unless it can be intermixed with one or more physical devices. This is so because some devices may not be virtualized at all. Some may not perform as well in virtualized avatar. Hence proposed design should also be able to handle seamless integration with external devices which could be physical/virtual. Key is to be able to connect them together so that the proposed solution can be availed with the best of what both physical and virtual worlds offer.

This can be done by representing the external devices as a native virtual but dummy entity. This entity should be stitched with the physical entity using additional overlay tunnels using technologies as most appropriate. The devices which connect to the external device if running in the virtualized form inside the cluster need not be aware at all about this critical aspect that it is peeing with an external device. The dummy entity present in between should be transparent enough to achieve these key but challenging aspects of network testing and management.

Performance

Networks involve three key parts. Management Plane, Control Plane and Data Plane. A good design shall aspire to provide best performance for all three planes. Usually, the requirements for the three planes vary.

1. **Management Plane**
    a. Seamless ability to interact inbound with the devices via https, ssh, telnet (for console), SNMP, NetConf, etc.
    b. Seamless ability to interact outbound to “Internet” in order to import data (such as images), export telemetry data, etc.
    c. While high performance is desired, a reasonable performance is usually suffice to meet most practical scenarios
2. **Control Plane**
    a. Ability to run many control-plane protocols across the layers of network stack.
    b. Some common examples are BGP, IGP (OSPF/ISIS), MPLS (LDP, RSVP), etc. and L2 and L3 VPN technologies
    c. Low latency is usually desirable especially when BFD protocols are also employed for quick fault detection
3. **Data Plane**
    a. Higher the throughput better obviously
    b. Limitations should be only based on the underlying infrastructure networking capacity
    c. MTUs tuning is crucial to avoid IP fragmentation and achieve maximum throughput
    d. Support for small and large frames alike is always highly desirable

Some performance measurements as gleaned on the Camphor Network Platform can be found in this link.

Live Demo

Design goals listed above can be shown to have been met by looking live at the Camphor Network Platform. Multiple small and large data networks are shown live, emulating many network topologies such as L3VPN, L3CLOS, Data Centers, etc. all on the same distributed compute infrastructure.

References

1. Camphor Networks
2. Demo Videos
3. Frequently Asked Questions (FAQ)
4. Getting Started
5. Introducing Camphor Networks Platform
6. Overview of Camphor Networks
7. Presentation Slides
8. Python Based SDK