IO VISOR PROJECT

IO Visor Use Cases: Networking Infrastructure

LINUX FOUNDATION COLLABORATIVE PROJECTS

IO Visor Use Cases: Networking Infrastructure

IO function virtualization is now a necessity, not a luxury. Building the right abstractions is the key enabler for this transformation.

Virtual Network Infrastructure as a Necessity

Server virtualization was a catalyst for cloud deployments and in the process, virtualization itself evolved. A key transformation occurred when server functions (such as compute, storage, and I/O) were decoupled from the underlying physical hardware that these functions execute upon. This decoupling created three fundamental value propositions for data center operators:

- **Maximizing utilization** using on-demand resource allocation.
- **Introducing elasticity** so that workloads/VMs can be easily created, migrated, copied, and cloned.
- **Fault isolation** within VMs, which, in terms of cost of recovery, bounds the risk of failure of a physical server.

Today, network functions (such as switches, routers, firewalls/ACLs, and load balancers) are embedded into and coupled with the physical appliances that provide these functions. Decoupling the networking functions from the underlay is a fundamental requirement to support agile, on-demand, and elastic provisioning of network services in multi-tenant cloud data centers. An overlay that provides such decoupling offers the following unique value propositions:

- On-demand provisioning of physical network resources to maximize network utilization.
- A system/IT administrator can define any virtual network topology, which provides network functions per tenant with traffic separation and then this specific overlay topology be created, spawned, managed or migrated, cloned, and copied at will. This automation introduces significant operational efficiencies and portability for data replication and hybrid clouds.
- Containing risks such as misconfigurations, security infringements, policy errors, etc. to an isolated virtual network instance is a key benefit, because failures are isolated within an assigned virtual network resources and cannot propagate to other parts of the physical network.

Challenges of Building Overlays

We are now seeing a transformation where the L2 network functions are being decoupled from the underlying physical network hardware. Many different Distributed Virtual Switches (DVSs) are now available to extend L2 functions (such as switching and VLANs) beyond a single server. This architecture needs a central controller that can encode rules in each DVS so that packets can reach remote servers hosting other VMs of a tenant. Hence, the controller and the DVS together form an L2 overlay.

While DVSs have been very useful in decoupling L2 functions from the underlying physical network hardware, this definition has unfortunately become synonymous with the overall overlay concept when there are clearly many other Network Functions (routing, firewalling, and load balancing, to name a few) that are fundamentally critical for any overlay.

An agile and elastic data center should allow secure provisioning of network functions on a per tenant basis, interconnection of these functions in arbitrary virtual topologies, and application of network wide policies (such as security, SLAs, and QoS) to a tenant's virtual topologies.

This creates a potential roadblock. The underlying physical network based protocols used by the network functions vary across operators and vendors, and they evolve over time. Thus, an effective overlay should allow fast and easy support for all possible physical network wire protocols of the past, the present, and the future.

If such a networking overlay were to be realized, its underlying technology would become a network hypervisor that abstracts away the physical network details, while allowing application evolution and agility with changing business needs, network (topological or policy) state, and wire protocols. In fact, this VNI [overlay] can easily extend beyond network functions to encompass storage protocols, thereby becoming a true IO Visor.

IO Visor Project

IO Visor project provides an extensible data plane that is programmable, elastic, and agile. With IO Visor up streamed to the Linux kernel (as eBPF), users gain many benefits and have a full set of new tools that enables interoperability between manufacturers, portability to different hardware platforms, and widely available talent for software development and operations. Moreover, the Linux kernel can be used directly without adding any vendor-specific library or API framework on top of it, guaranteeing that this architecture is not limited to work with any specific vendor's hardware, or bound by the capabilities that a vendor's API may support.

The Linux kernel has traditionally supported several frameworks providing pieces of what could be turned into fully capable "networking functions". There are low-level filtering tools available (perf, ftrace, stap, ktap), traffic analyzers and monitors (tcpdump, iptraf), tools for monitoring the networking stack at different layers (ss, iptraf, netstat, nicstat, ip), tight control of the network interfaces and the link layer (ethtool, lldptool) and powerful hierarchical queuing and prioritization mechanisms (qdisc).

These tools offer access to different pieces of information about the running networking subsystem. However, in order to build a fully capable networking data plane, the Linux kernel should have the ability to execute atomic networking functions (such as switching, routing, firewalling, etc) and serialize them to form a complex data plane composed of several of those networking functions. This would enable building these atomic networking functions as simple programs, and serializing them to build a full networking topology (the full data plane of the service) in kernel space.

As eBPF, IO Visor provides the in-kernel data plane functionality, designed and optimized for cloud infrastructure to run a dynamic virtual network infrastructure that meets the following requirements:

1. **Self-service provisioning**, fault-resilient network provisioning on a per-tenant basis.
2. **On-demand provisioning**, configuration, and reconfiguration of arbitrary network topologies and policies.
3. **Agile support** of new physical network protocols and network functions, without vendor lock-in.

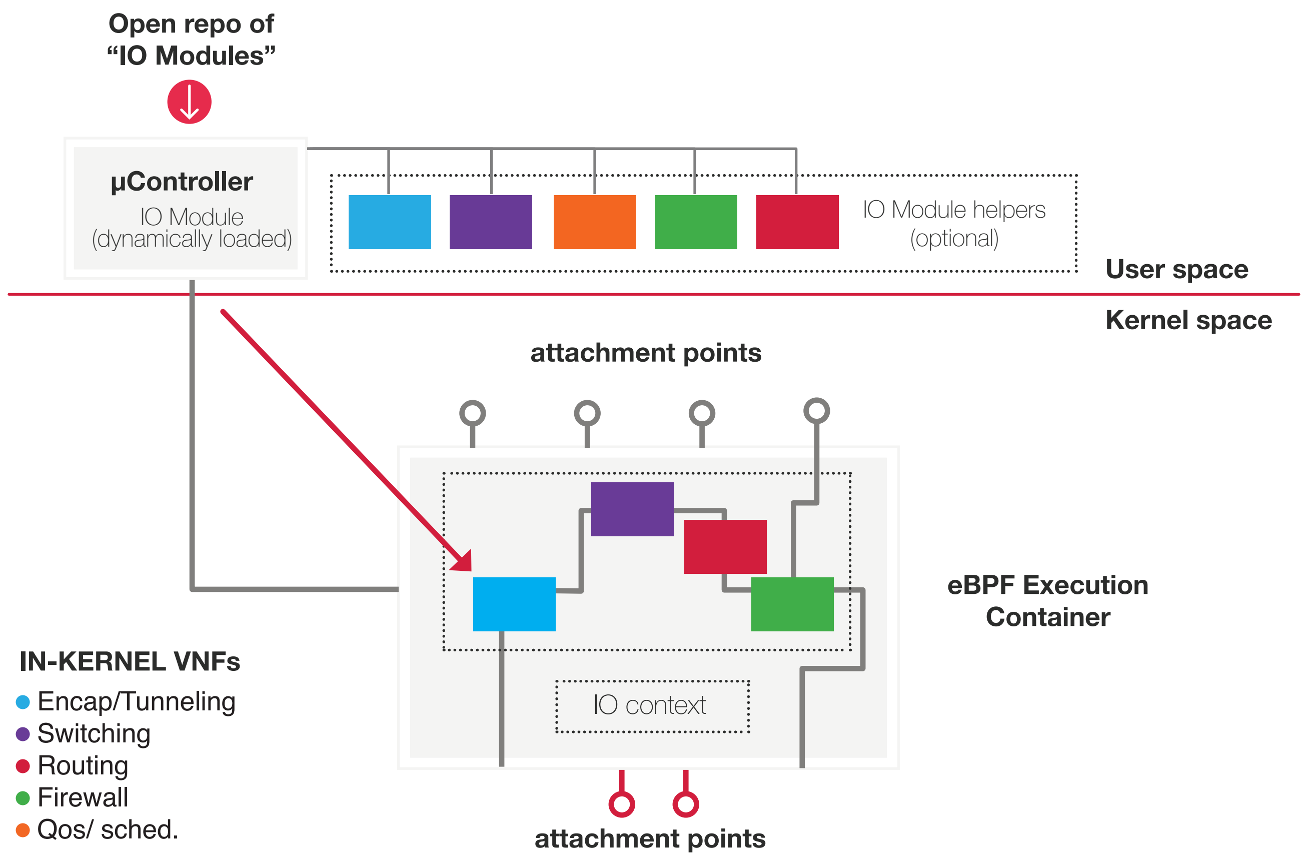

Figure 1: eBPF framework for networking

It is important that an IO Visor provides network function programmers with data plane programming SDKs and compiler tool chains for all its supported platforms. This development workflow allows operators to introduce any new network function and wire protocol in their networks, without getting locked into a single vendor. An additional advantage of the IO Visor technology is that the compiler hides data plane specifics from the developer, who can write a network function's data plane using the provided SDK and then recompile it for any supported platform. In other words, the IO Visor technology naturally becomes a Network-as-a-Platform (NaaP), providing the user with a development environment to define new network functions and their corresponding data planes.

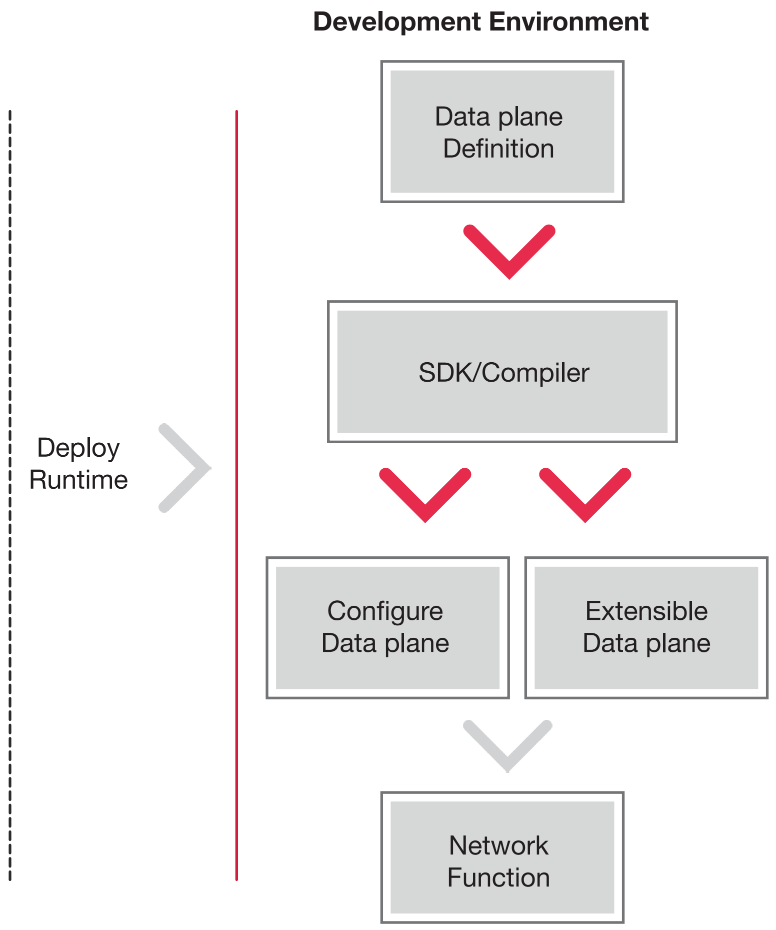

Figure 2: IO Visor SDK-driven development workflow

Summary

IO Visor technology provides all the features of data center automation and virtualization discussed in this whitepaper.

You can use the IO Visor technology to implement isolated virtual network instances for safe isolation of multi-tenant network topologies. Tenants can create, clone, and migrate these virtual network instances on demand. Further you can provide a rich set of virtual network functions (such as routers, switches, and security) and services (such as NAT and DHCP).

In short, IO Visor delivers a true Network-as-a-Platform technology to automate and manage modern cloud enabled data center networks.

Contact Us

IOVISOR PROJECT

contact-us@iovisor.org
events@iovisor.org
www.iovisor.org