Empower Diverse Open Transport Layer Protocols in Cloud Networking

GEORGE ZHAO
DIRECTOR OSS & ECOSYSTEM, HUAWEI
Agenda

- FD.io Introduction
- Challenges in Container & Cloud Native Apps
- Proposed Solutions
A Universal Terabit Network Platform
For Native Cloud Network Services

Breaking the Barrier of Software Defined Network Services
1 Terabit Services on a Single Intel® Xeon® Server !!!

- EFFICIENCY
  The most efficient software data plane Packet Processing on the planet

- PERFORMANCE
  FD.io on x86 servers outperforms specialized packet processing HW

- SOFTWARE DEFINED NETWORKING
  Software programmable, extendable and flexible

- CLOUD NETWORK SERVICES
  Foundation for cloud native network services

- LINUX FOUNDATION
  Open source collaborative project in Linux Foundation

Most Efficient on the Planet
Superior Performance
Flexible and Extensible
Cloud Native
Open Source
Packet Processing Software Platform

- High performance
- Linux User Space
- Runs on compute CPUs:
  - And “knows” how to run them well!
- Shipping at volume in server & embedded products since 2004.
FD.io enables:
1. Bare Metal, Cloud NFVi and Container Infra
2. FD.io based VNFs
Kernel space or User space?

**Oops..**
Go through kernel space and user space many times when communication between containers.

**How about**
User space stack instead of kernel TCP/IP stack.

**Challenge:** Applications need to adapt to different APIs while using user space stack.
Apps have different flavors

Oops…

different APP different user space

stack—Some applications required

high I/O throughput(e.g. File transfer, Video on Demand) and Some

applications required low latency(e.g. online game, live streaming).

How about…

Develop different stack to match
different requirement

Challenge: Application can not choose stack dynamic
Hardware is not all the same

Oops...
Didn’t make full use of the speedup ability of different hardware by kernel stack and user space stack

Challenge: How to identify the difference of hardware and how to adapt RDMA
Introducing DMM Project

- DMM Protocol Framework: Dual mode, Multiple protocols & Multiple instances, aim to provide a new solution of diverse protocol stacks for developers.
  - Dual Mode: Kernel Space and User Space
  - Multiple protocols: Simply new protocol adoptions and Integrations
  - Multiple Instances: Enable “protocol routing”
- DMM is a FD.io open source project, first release targets 18.04.
DMM: Protocol Stack Common Framework

- **Web APP**
- **Video Streaming**
- **Online gaming**

**Socket Layer**

- **Posix Socket Compatible API (LD_PRELOAD)**
  - Socket Bridge (SBR)
  - nSocket (MUX)

- **VPP Host Stack**
- **TLDK**
- **F-Stack**

- **Data-Plane HAL**

- **ipv4-input/output**
- **ipv6-input/output**
- **DPDK-input**

**RD**

- **DHT**
- **REST**
- **netconf/yang**
- **Other (BGP)**

**Protocol orchestrator + nRD**

1. Dynamic mapping between application and stack
2. Stack selection rule can be configured by e.g. SDN controllers

**Posix Socket Compatible API + nSocket**

1. Posix compatible and uniform socket API to APP
2. Support both kernel TCP/IP stack and user space stack

**Kernel TCP/IP**

**Nic**

- **DMM**
- **FD.io**
- **Honeycomb**
- **Third stack**

**RD**: Resource Discovery
Key Takeaways

- Stack developers can concentrate on user space protocol innovations;
- Apps can **dynamically choose different protocols.**
- Support both **kernel TCP/IP stack and user space stack**;
- Container network will easily build **E2E communication capacities.**

Common library for develop user space stack

- Posix Socket Compatible API (LD_PRELOAD)
- nSocket (MUX)
- Data-Plane HAL
- VPP Host Stack
- F-Stack
- LWIP
- TLDK

HAL: Hardware Abstraction Layer, a IO adaptor