Data Plane Acceleration
Where Are We?
Where Do We Need To Go?

Margaret T. Chiosi
VP Open Ecosystem

Dr. Kai Zheng
Director Protocol R&D
Agenda

• Industry Challenges
• FD.IO Review
• Emerging Challenges in Container & Cloud Native Apps
• Proposed Solutions
Latency critical services in 5G

Each use case/application requires a different range of latency.

Moving applications closer to the edge for lower latency.
Movement Toward Real Time

- Applications are moving toward more Real Time → lower latency
- IOT requires mobility support and geo-distribution, location awareness and low latency
- Automation of actions in real time is causing local data analytics and management closer to the edge supplemented by end-end analytics in the cloud core
  - Real time closed loops
  - High availability
  - ....
Cloudification

- Cloudify Edge w/Central Cloud for larger capacity and analytics
- Compute-storage-networking building blocks from edge to core
- Lower Latency with large distribution
- Peer-Peer communication
Application on Different Platforms (Example)
## Proliferation of Open Source Network Operating Systems

<table>
<thead>
<tr>
<th>Objectives &amp; Scope</th>
<th>dNOS (AT&amp;T)</th>
<th>Stratum (ONF)</th>
<th>OpenR (Facebook)</th>
<th>FRR (Linux Foundation)</th>
<th>SONIC (Microsoft)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targeted use-case</td>
<td></td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>• Data-Center Fabric</td>
<td>v</td>
<td></td>
<td>v</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Carrier WAN</td>
<td>v</td>
<td></td>
<td>v</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• CPE</td>
<td>v</td>
<td></td>
<td>v</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>• Applications</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td></td>
<td>v</td>
</tr>
<tr>
<td>• Shared data Infrastructure</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>• Hardware abstraction</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td></td>
</tr>
</tbody>
</table>
Data Plane Acceleration is Critical

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
Agenda

- Industry Challenges
- FD.IO Review
- Emerging Challenges in Container & Cloud Native Apps
- Proposed Solutions
Network Framework

1. Common Overlay strategy for VM/Containers/Bare Metal…
2. Common API to VPP for control forwarding and model driven management planes
3. Create common application API to VPP forwarding
4. Common API to DPDK/ODP to support:
   - Diverse CPU chips
   - Diverse virtualization environments
   - Diverse NIC, NPU, FPGA

Cloud Orchestration:
- Openstack
- K8S
- Docker
- ONAP

Control Plane Application Env:
- Openstack
- K8S
- Docker
- ONAP

Control Forwarding Plane:
- Mgmt Plane
  - Yang/Netconf
- Local Control
- HoneyComb

Data Plane
- Network App
- Policy
- Identity

User space
- NSH_SFC
- ONE
- TLDK/socket
- VPP Forwarding
- IO
- DPDK
- ODP

Hardware:
- CPU (INTEL, ARM), NIC, SoC, FPGA, …

1. Common Overlay strategy for VM/Containers/Bare Metal…
2. Common API to VPP for control forwarding and model driven management planes
3. Create common application API to VPP forwarding
4. Common API to DPDK/ODP to support:
   - Diverse CPU chips
   - Diverse virtualization environments
   - Diverse NIC, NPU, FPGA
FD.io in Overall Stack

- Application Layer/App Server
- Orchestration
- Network Controller
- Data Plane Services
  - Data Plane Management Agents
  - Packet Processing
  - Network IO
- Operating Systems
- Hardware
One Data Plane Multiple Deployment Scenarios

FD.io enables:
1. Bare Metal, Cloud NFVi and Container Infra
2. FD.io based VNFs
3. FD.io Ecosystem
4. Various OS (Kernels/Hypervisors) and Hardware

Hardware
- CPU (INTEL, ARM), NIC, SoC, FPGA, …
- INTEL, ARM

Operating Systems
- VM, VM, Con, Con

Server
- VM

Device
- VM
- FD.io
- Operating Systems
- Hardware INTEL, ARM

FD.io
- Smart NIC
FD.IO: Local and Remote Programmable API for Control

Local interface:
1. shared memory or message queue communication
2. high performance

Remote interface:
1. can customize the API interface to provide flexibility
2. The management agent and VPP decoupling can be replaced.
FD.IO: High Performance, Vector-Based Forwarding

Packet-based forwarding framework

All message information is obtained from the driver at one time to form a message vector, and is processed according to the directed graph of the node. One packet is processed at a time, which can effectively reduce instruction cache jitter and reduce the actual number of data caches.

Vector-based forwarding framework
FD.IO: Modular, Flexible Architecture, Easy to Expand

- Modular software architecture: Using Graph mechanism, all functions are modularized, and nodal and pipeline control routing paths are realized
- Flexible scalability: Easy and flexible expansion with plug-in mode (but not yet hot-pluggable)
- Good hardware compatibility: Supports multiple common CPUs: X86, ARM, PowerPC
- Operating system-independent: Smooth migration.
- User mode implementation
Agenda

• Industry Challenges
• FD.IO Review
• Emerging Challenges in Container & Cloud Native Apps
• Proposed Solutions
How Does Container Network Impact Us?

• E.g. Container network is “a network of applications” (rather than a network of hosts or VMs as the legacy network)

• Container should provide an End-to-End network service model which should ALSO include the virtual switching fabric and the protocol stack

• Many efforts have been put on optimizing the virtual switching fabric, however, few is provided for the protocol stack…
Challenges on protocol stack - Future Transport Protocol Design

- **Ultimate performance**
  - Video – orders of magnitudes higher bandwidth
  - VR/AR – very low latency and jitter
  - IoT – orders of magnitudes more concurrent connections

- **Diversified network QoS/SLA**
  - Applications with different QoS/SLA requirements exist simultaneously on the same platform
  - Any optimization is tradeoff between factors

- **Heterogeneous network environments**
  - Cloud computing and mobile internet turn the network into an extremely complicated system
  - Network environment might change significantly due to mobility
Challenges on protocol stack - Some Answers

- **Alternative transport protocols**
  - Google’s QUIC
  - IBM’s FASP

- **User-space network stack**
  - Improving performance
  - Protecting intellectual property
Agenda

Industry Challenges
FD.IO Review
Challenges in Container & Cloud Native Apps

Proposed Solution - DMM
DMM: Re-designing the protocol stack

DMM is a protocol stack framework and platform, targeting *protocol diversity & dynamic protocol selection*

- **Dual modes:** leverage protocol implementations on both *kernel* and *user* space
- **Multi-protocols:** flexible framework for *new protocol* adoptions & extensions
- **Multi-instances:** concurrent stack instances and adaptive ‘*protocol routing*’

DMM will open-source @ FD.io.

https://wiki.fd.io/view/DMM
DMM: Architecture & key features

- Flexible protocol plug-in framework
- Unified & backward compatible APIs
  - LD_PRELOAD or similar mechanism for transparent acceleration
  - MUX for one-to-many mapping
- Distributed protocol Resource Directory (RD)
  - Proactive protocol orchestration (via conf files or control API)
  - Reactive protocol negotiation and intelligent adoption (via RD peering)
- High perf. & reusable session layer (SBR)
- Flexible pkt I/O for different types of (EAL)
- Virtual Receive Side Scaling (vRSS)
  - Packets to stack instances w/ flow affinity
- Other OAM tools (compliance check, monitoring, etc)
DMM: Benefits to application developers/end-users

User-friendly & transparent acceleration:
✓ Acceleration w/ backward compatible API, friendly to the legacy

Adaptive & customized acceleration:
✓ ‘Protocol Routing’ based on connection profile (network parameters, QoS requirements and host information, etc)

---

**Diagram:**

- **App Server**
  - `sf=socket(AF_INET, TYPE, 0)`
  - `bind(sf,....)`
  - `listen(sf,....)`
  - `csf=accept(sf,....)`
  - ...
  - `read(cs,...)`
  - `write(cs,...)`

- **App Client**
  - `sf=socket(AF_INET, TYPE, 0)`
  - `connect(sf,....)`
  - ...
  - `write(sf,....)`
  - `read(sf,....)`

- **Protocol Negotiate**

---

**‘Protocol Routing’ workflow**

- DMM Stacks sf listens on: TCP, PCC, RDMA...
- DMM Stacks sf connects by: TCP, PCC, RDMA...
DMM: Benefits to protocol researchers/developers

Friendly framework to integrate new protocol stacks
✓ Flexible Pkt I/O NIC/L2/L3/L4 (EAL)
✓ Simplified API (SBR)

Speeding up innovation of new protocol stacks
✓ Modular and reusable function blocks w/ high perf.
✓ Integration w/ both kernel and user space (VPP), making a smooth migration.
DMM: Project roadmap

- v18.04
  - Fd.io open source announcement
  - User Guide
  - EAL (DPDK)
  - Proactive orchestration
  - Dual modes support

- v18.10
  - Support VPP L2/L3
  - Integration of VPP host stack, TLDK, etc
  - Support reactive orchestration
  - Performance optimization

- v19.01
  - Multi-instances support
  - More third-party stacks
  - Enrich nRD knowledge base
Join us

DMM demo @ ONS 2018: Huawei booth #44
FD.io DMM Project Home Page
• https://wiki.fd.io/view/DMM
DMM tech details
Code repository
• https://git.fd.io/dmm (Available in Apr. 20th)
Contact us
• dmm-dev@lists.fd.io

Welcome to contribute or comment~
Thank you

www.huawei.com
Demo: DMM Protocol Routing

• **File transport application**
  - Server in Cloud-DC2
  - Client #1: Internet anywhere
  - Client #2: Same data center
  - Client #3: Different data center

• **Comparison**
  - Kernel TCP/IP stack
  - DMM: Negotiate appropriate stacks (Kernel TCP/IP, RDMA, customized user-space TCP/IP) on different networks
Demo: Protocol Routing for Multi-network Client-Server Application

No one stack/protocol fits all scenario, but by adaptively negotiating stack according to the network environment, DMM achieves significant performance improvement.

![Diagram showing performance improvements with DMM in different network environments: Intra-DC, Inter-DC, and Internet. Reduced by 97% in Intra-DC, Reduced by 69% in Inter-DC, and Almost no Framework overhead in Internet. RDMA and Customized Userspace TCP/IP are highlighted.]