HUAWEI

Accelerate Network Protocol Stack Performance and Adoption in the Cloud Networking via **DMM**

Waterman Cao
Senior Researcher
Cloud Networking Lab, Huawei

AGENDA

01

What we face

02

DMM Overview

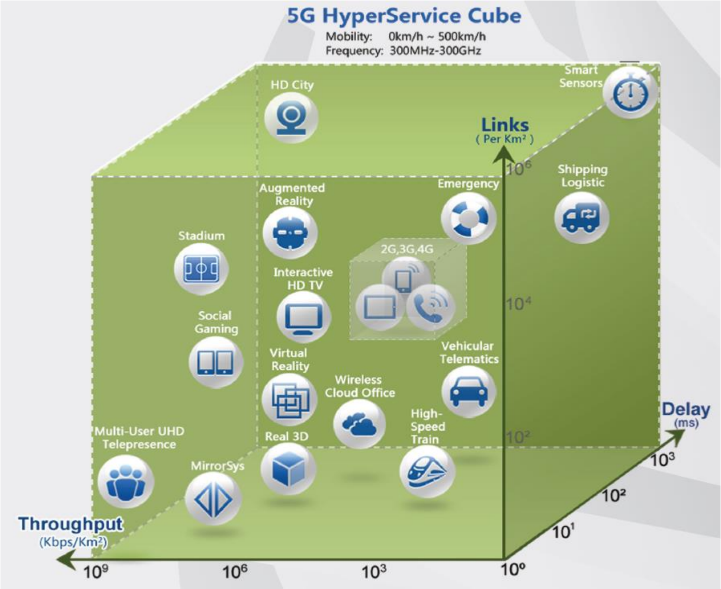
03

Use Cases

Why we need a new way to implement protocol stack?

Diversified and Ultimate QoS req.

- **Diversity:** e.g. >2 million mobile apps with various QoS
- **Performance:** e.g. <1ms, >1Gbps, 10M concurrent , @5G
- **App Density**: e.g. >30 of APP @ smart-phone, 100s containers @ cloud host

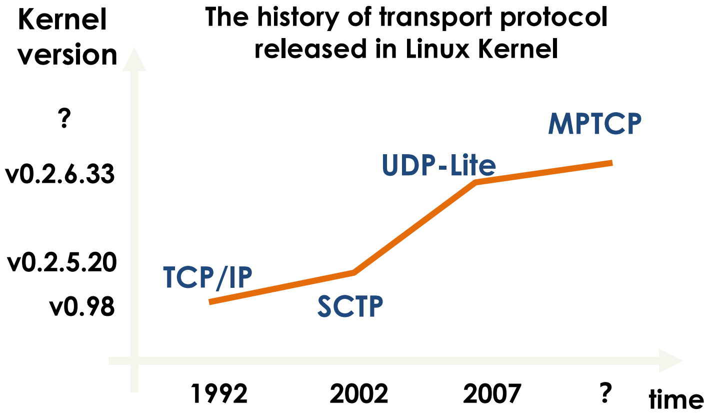


Mismatched

Ossified Kernel Networking Stack

- General-purpose design
- General performance tradeoff
- Hard to customization
- Long protocol/algorithm release cycle…

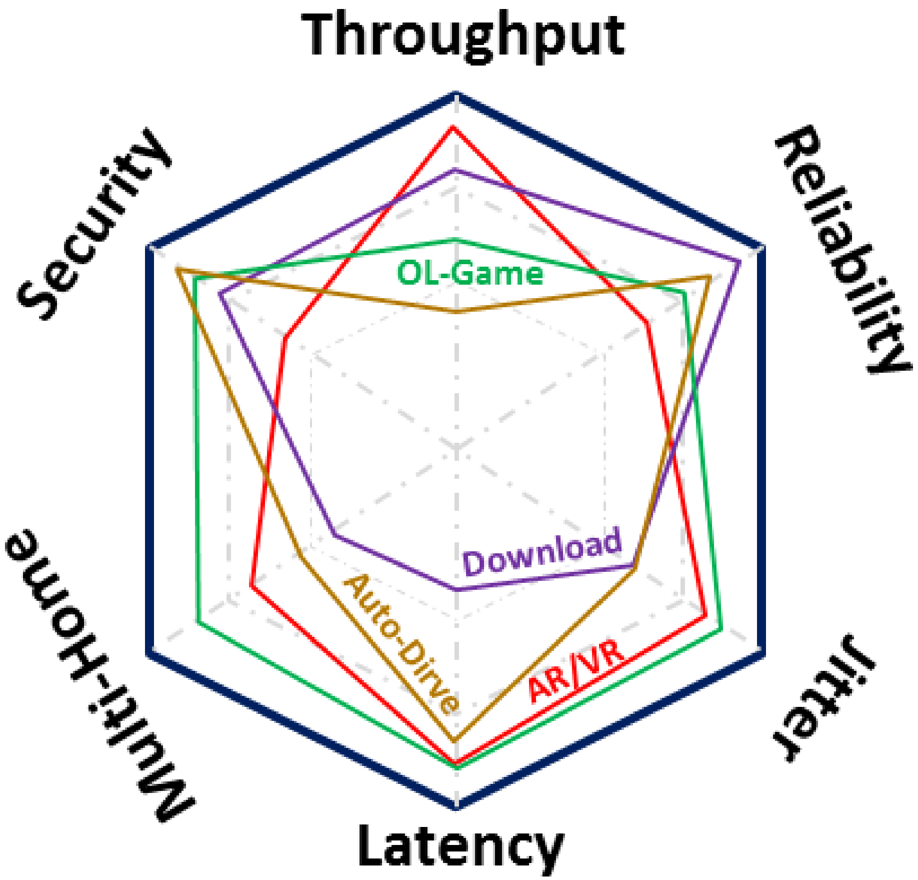
The history of transport protocol released in Linux Kernel



- More than 25 years, <5 transport protocols are released in the Linux kernel
- It has takes 8 years after MPTCP was firstly proposed, but MPTCP is still not released in the Linux kernel

Challenges in Future Transport Protocol Design

- **Extremely high 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 network participants’mobility



HUAWEI

Trends in Future Transport Protocol Design

- **Alternative transport protocols**
  - ✓ Google's QUIC
  - ✓ IBM's FASP
- **User-space network stack**
  - ✓ Improving performance
  - ✓ Protecting intellectual property

AGENDA

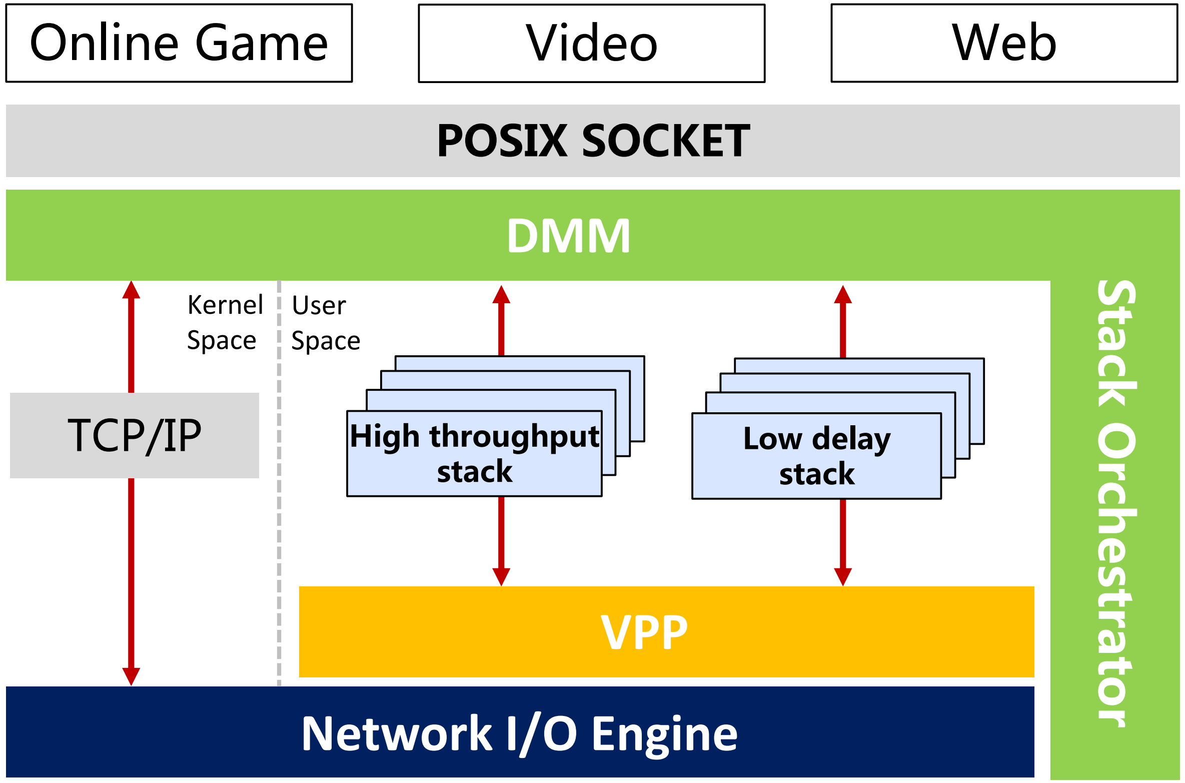
What we face

DMM Overview

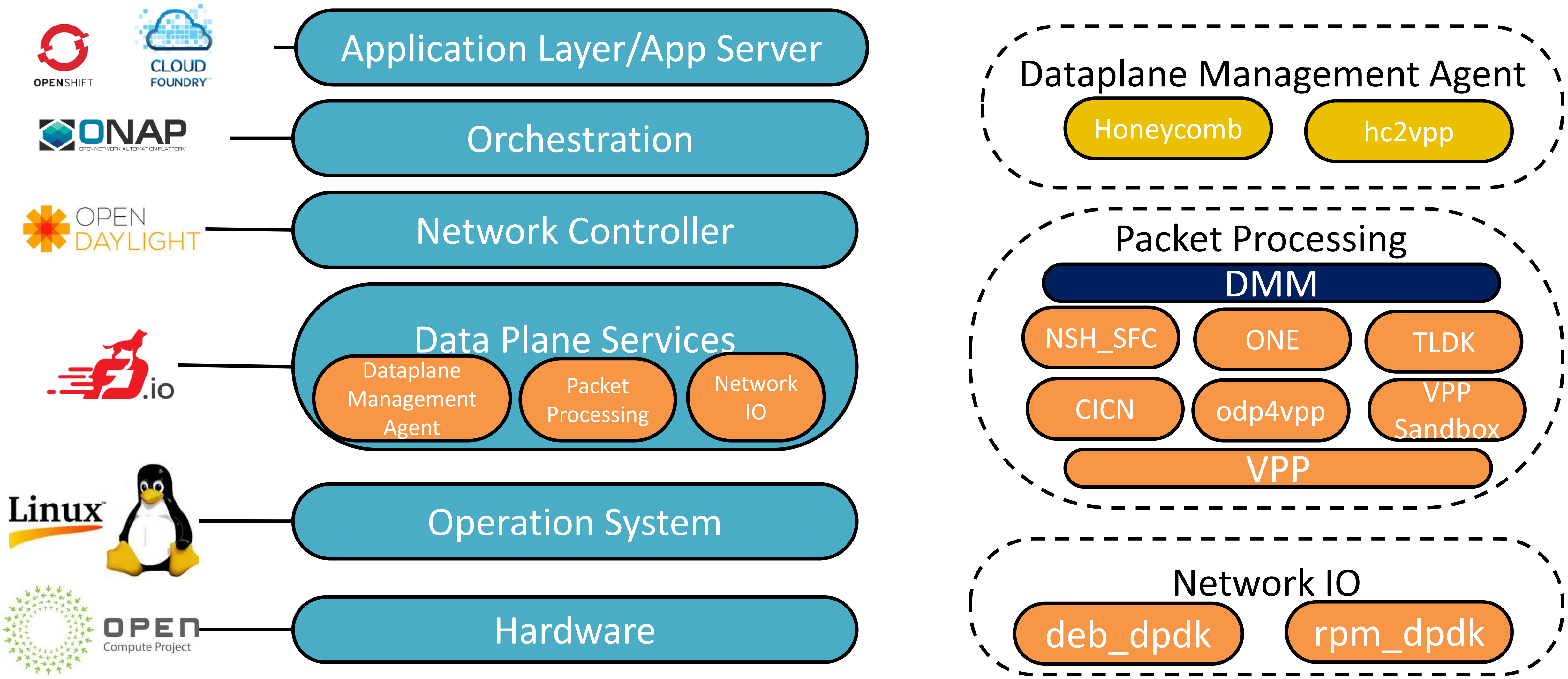
Use Cases

DMM Project : Re-design the Protocol Stack

- DMM (Dual mode Multi-protocol Multi-instance) is DMM is an open source network stack framework under FD.io project, licensed as Apache , which enables:
  - **Dual mode** :Support **Kernel Space** and **User Space**
  - **Multi-protocol**: Simplify **new protocols** adoptions and Integrations with flexible framework
  - **Multi-instance**: concurrent stack instances and Enable "**protocol routing**" in Cloud Networking
- DMM aims to provide the capability to have multiple protocol and multiple stack instances running simultaneously in the same platform.



DMM in the Overall stack



Application Layer/App Server

Orchestration

Network Controller

Data Plane Services

- Dataplane Management Agent
- Packet Processing
- Network IO

Operation System

Hardware

Dataplane Management Agent

- Honeycomb
- hc2vpp

Packet Processing

- DMM
- NSH_SFC
- ONE
- TLDK
- CICN
- odp4vpp
- VPP Sandbox
- VPP

Network IO

- deb_dpdk
- rpm_dpdk

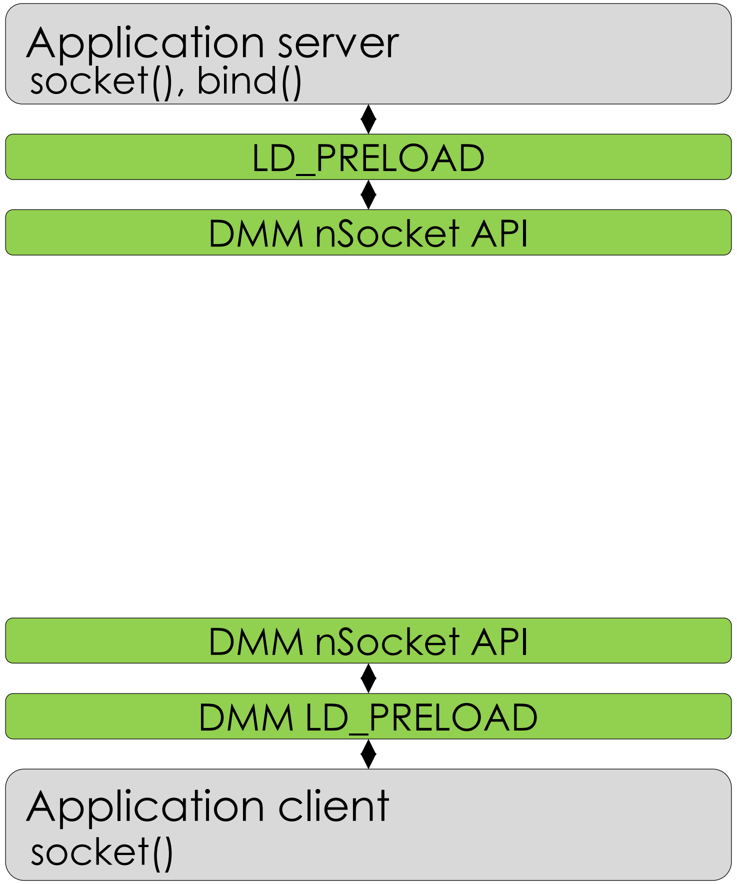
Protocol Routing Workflow

Application server
socket(), bind()

Application client
socket()

1 Application server and client calls socket interface.

Protocol Routing Workflow



Application server
socket(), bind()

LD_PRELOAD

DMM nSocket API

DMM nSocket API

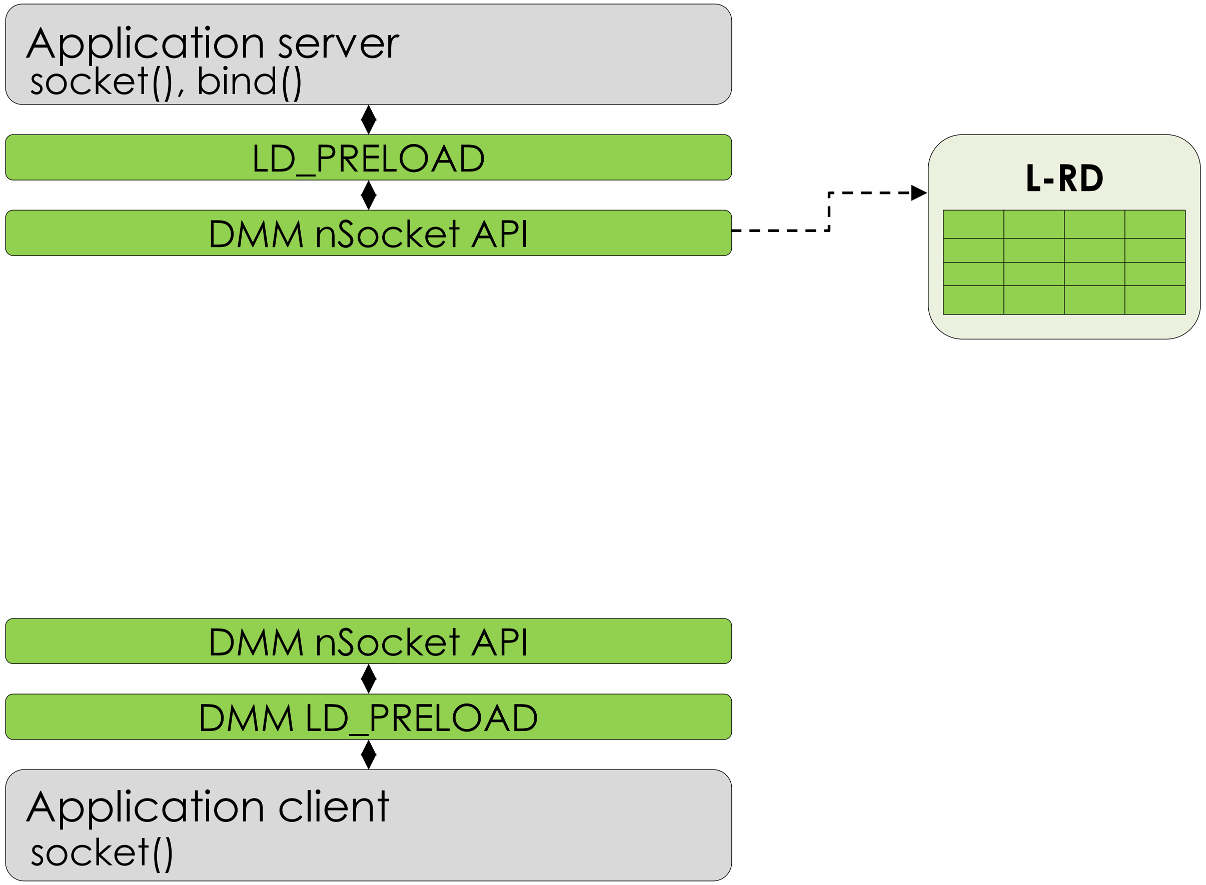
DMM LD_PRELOAD

Application client
socket()

1 Application server and client calls socket interface.

2 Socket APIs are hijacked to DMM nSocket APIs.

Protocol Routing Workflow

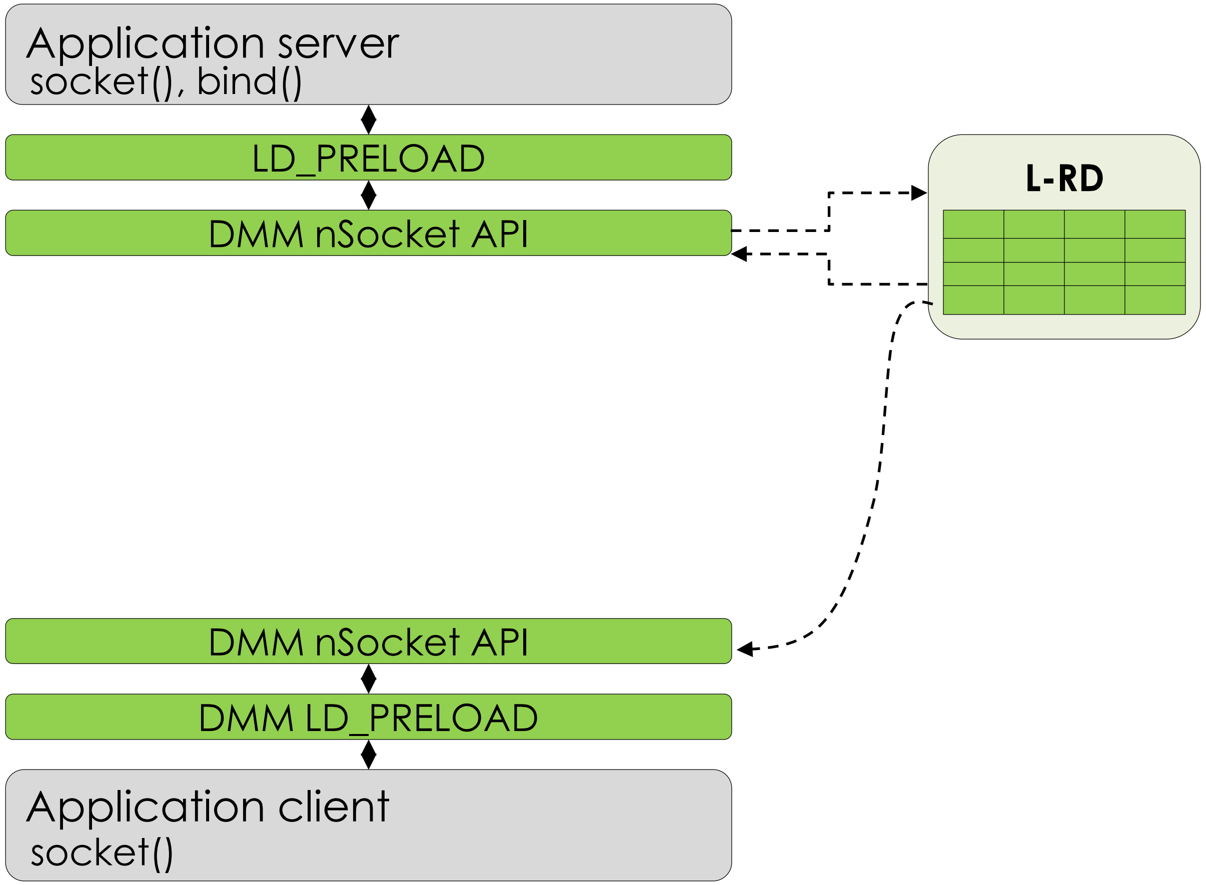


1 Application server and client calls socket interface.

2 Socket APIs are hijacked to DMM nSocket APIs.

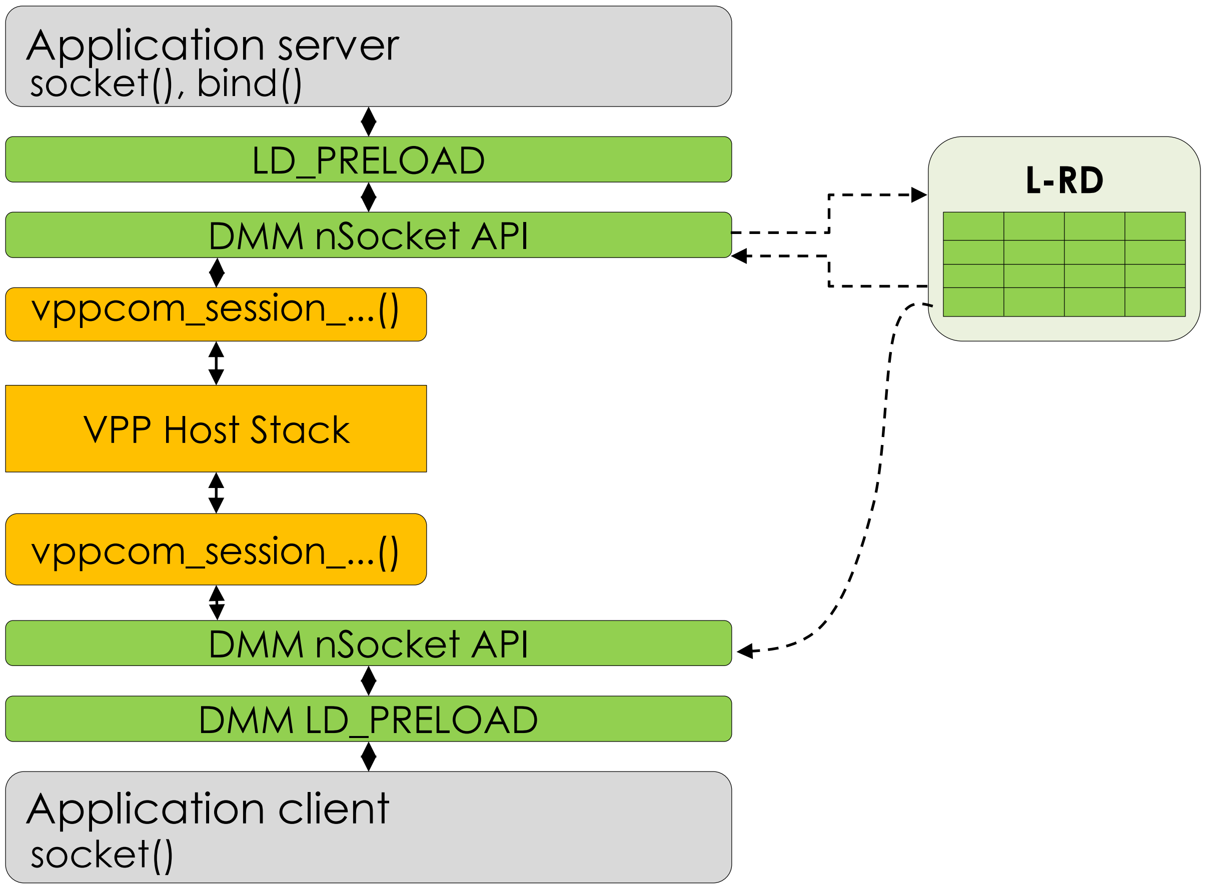
3 Server call listen() triggers L-RD to negotiate protocol policies.
L-RD: manage local DMM Policies and Protocol Configure.

Protocol Routing Workflow



1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call listen() triggers L-RD to negotiate protocol policies.
   L-RD: manage local DMM Policies and Protocol Configure.
4. Server call accept() and client call connect() trigger L-RD to retrieve and resolve protocol stack mapping.

Protocol Routing Workflow



1 Application server and client calls socket interface.

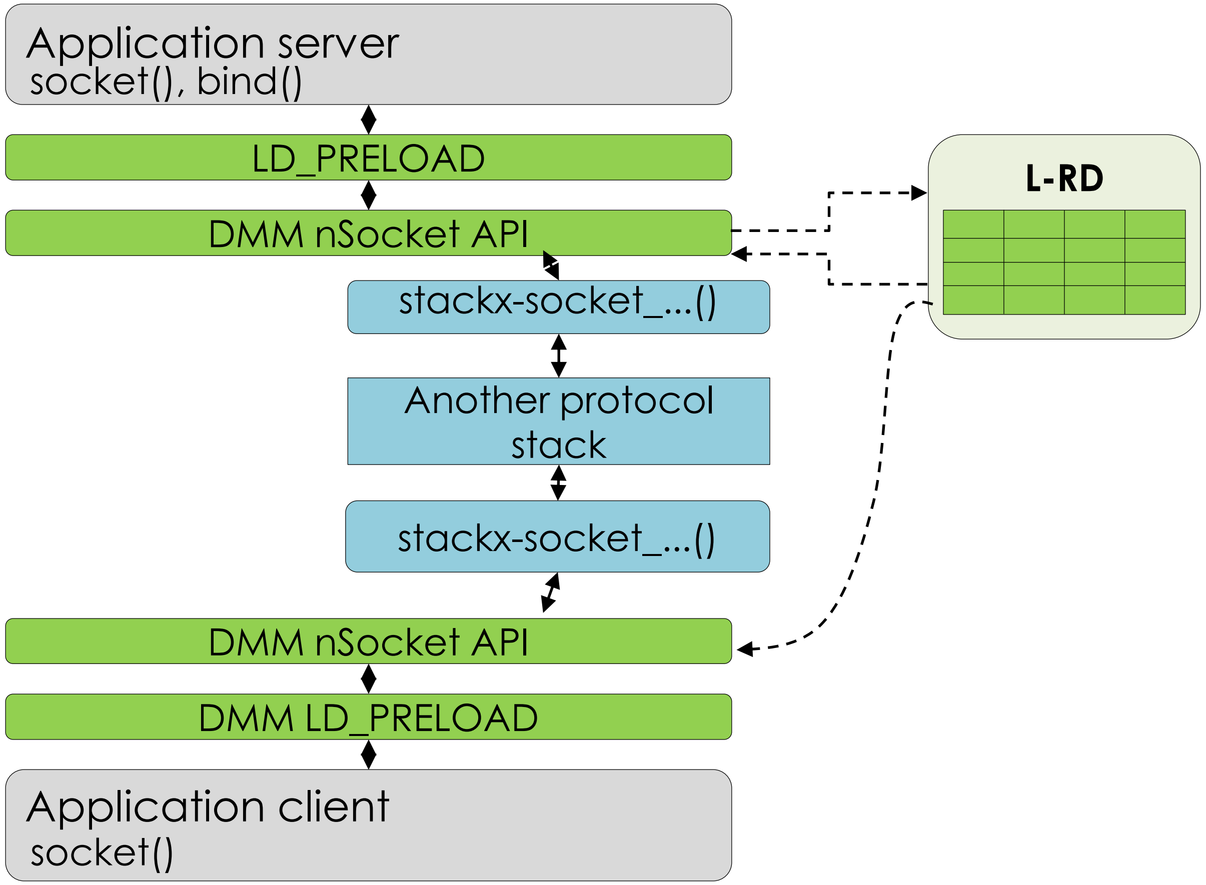
2 Socket APIs are hijacked to DMM nSocket APIs.

3 Server call listen() triggers L-RD to negotiate protocol policies.
L-RD: manage local DMM Policies and Protocol Configure.

4 Server call accept() and client call connect() trigger L-RD to retrieve and resolve protocol stack mapping.

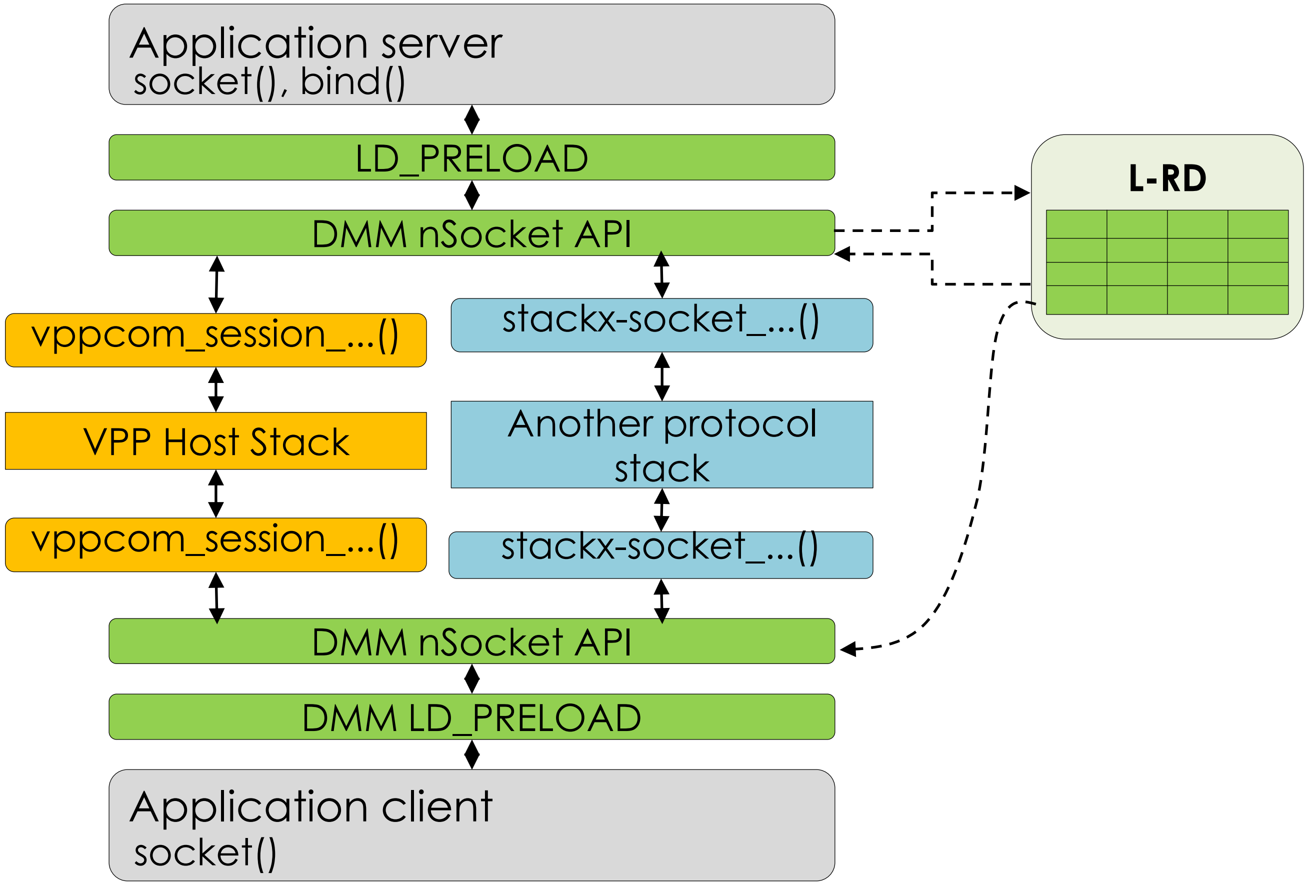
5 According to the mapping, the socket is instantiated to one protocol stack

Protocol Routing Workflow



1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call listen() triggers L-RD to negotiate protocol policies.
   L-RD: manage local DMM Policies and Protocol Configure.
4. Server call accept() and client call connect() trigger L-RD to retrieve and resolve protocol stack mapping.
5. According to the mapping, the socket is instantiated to one protocol stack or Another.

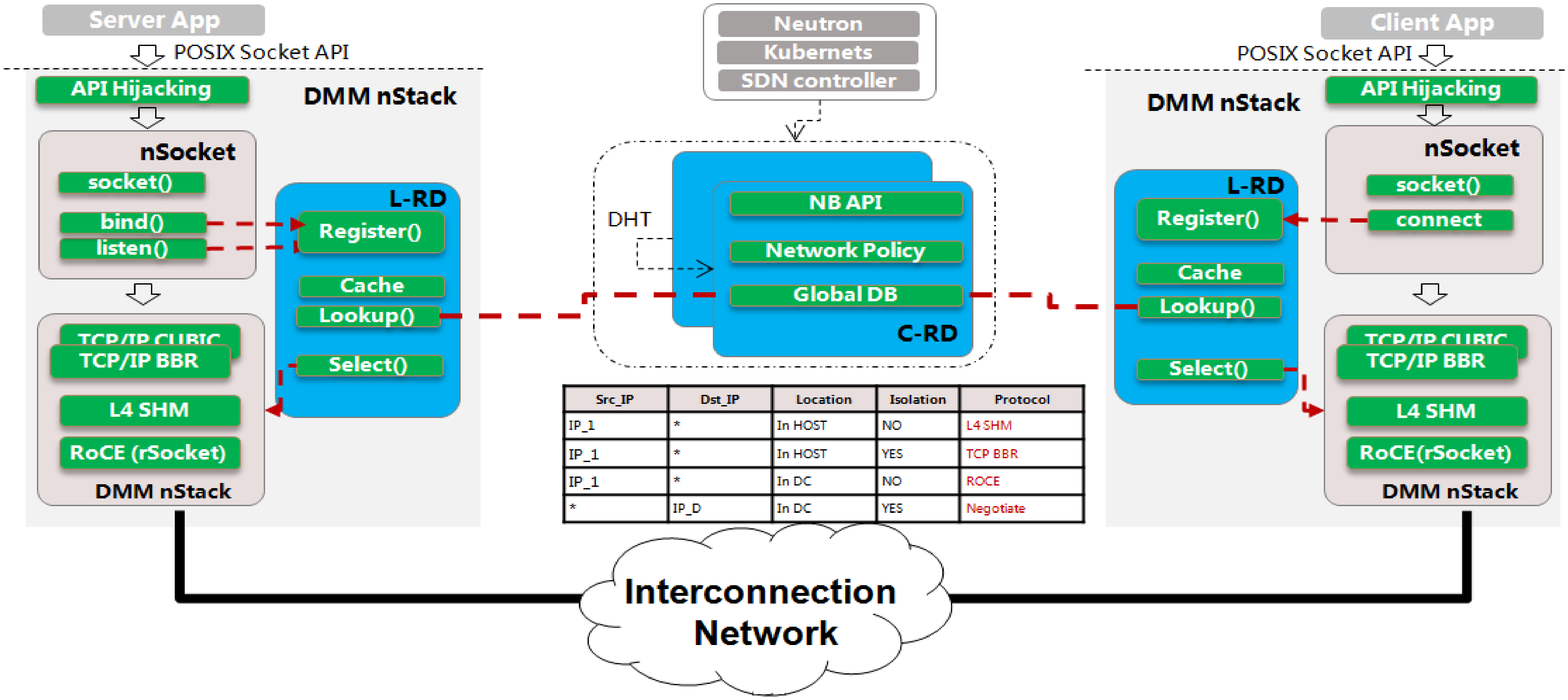
Protocol Routing Workflow



1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call listen() triggers L-RD to negotiate protocol policies.
   L-RD: manage local DMM Policies and Protocol Configure.
4. Server call accept() and client call connect() trigger L-RD to retrieve and resolve protocol stack mapping.
5. According to the mapping, the socket is instantiated to one protocol stack or Another.
6. Dual mode(kernel or user-space), Multiple protocols, Multiple instances can exist simultaneously.

HUAWEI

Protocol Routing Workflow (with Centeral RD)



Src_IP	Dst_IP	Location	Isolation	Protocol
IP_1	*	In HOST	NO	L4 SHM
IP_1	*	In HOST	YES	TCP BBR
IP_1	*	In DC	NO	ROCE
*	IP_D	In DC	YES	Negotiate

HUAWEI

AGENDA

What we face

DMM Overview

Use Cases

Use Case 1 : Protocol Routing

❑ File Sync Application

- 3 Clients --> Server

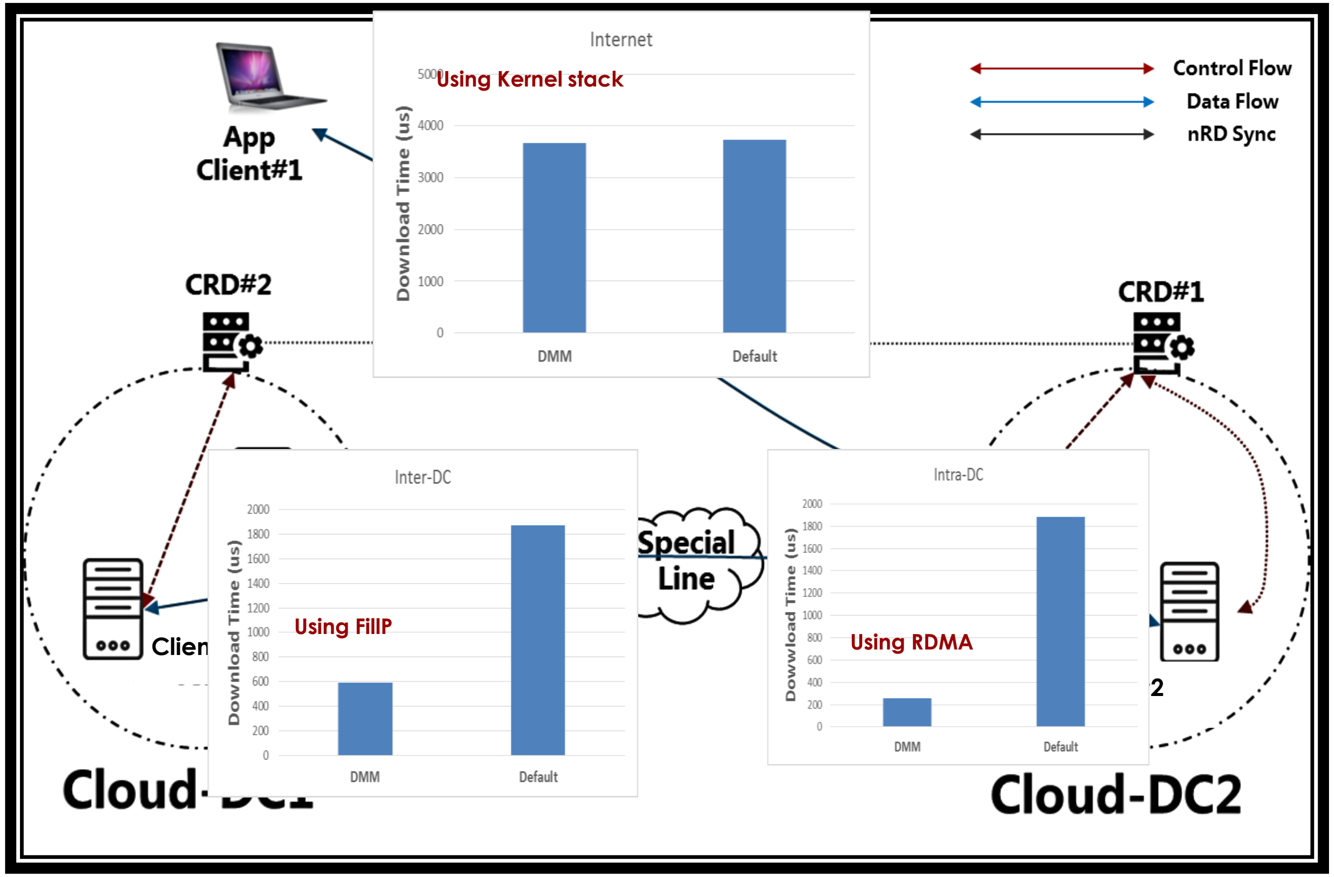
❑ Network Setting

- Internet (Client #1)
- Intra DataCenter (Client #2)
- Inter DataCenter (Client #3)

❑ Comparison scheme

- Default: the kernel TCP/IP stack
- DMM: support kernel TCP/IP stack, RDMA,

FillP (home-grown stack)



By adaptively negotiating stacks according to the network environments, DMM achieves significant performance improvement comparing with the kernel stack by default

For more detail of this demo, please go to Huawei Demo Booth.

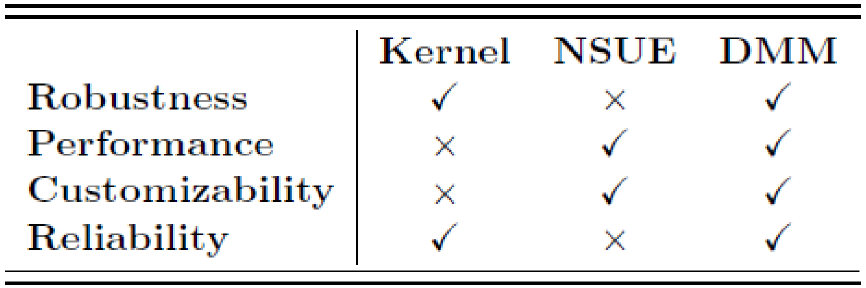
Use Case #2: Dual mode support for Nginx Server

❑ Nginx application

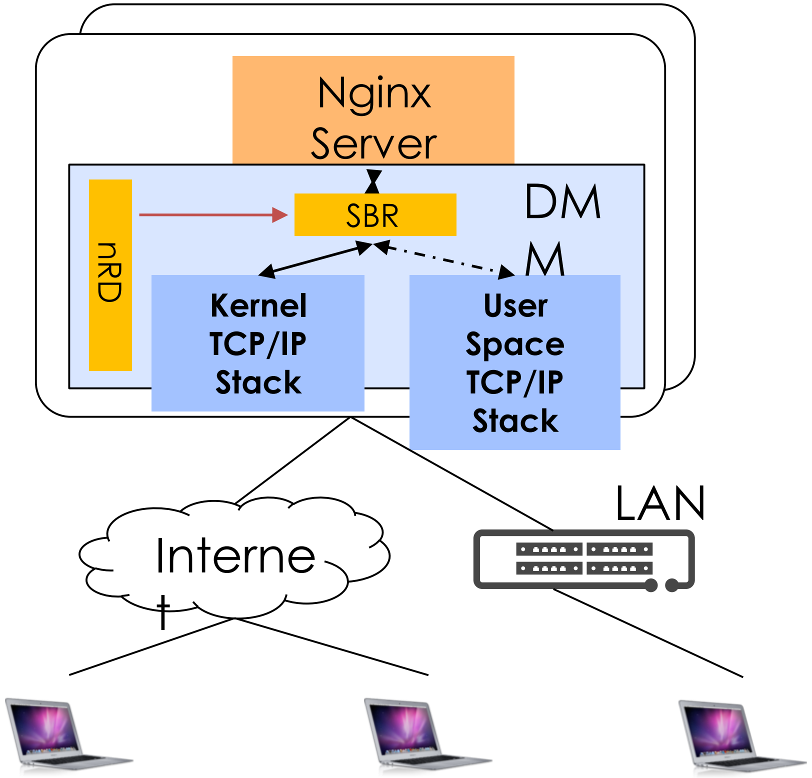
- kernel stack vs user-space stack ?

❑ DMM nRD Policy

- Internet connection ---> kernel stack
- LAN connection ---> user-space stack



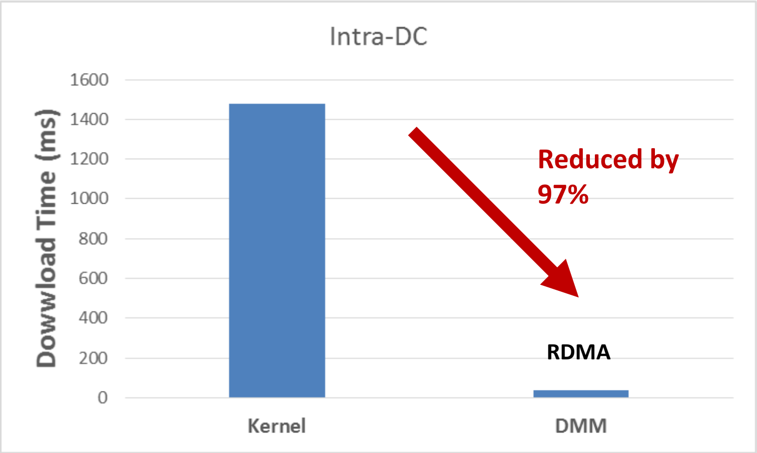
	Kernel	NSUE	DMM
Robustness	✓	×	✓
Performance	×	✓	✓
Customizability	×	✓	✓
Reliability	✓	×	✓



Using DMM, Nginx server could switch between kernel stack and user-space stack adaptively to use their advantages respectively under different scenario

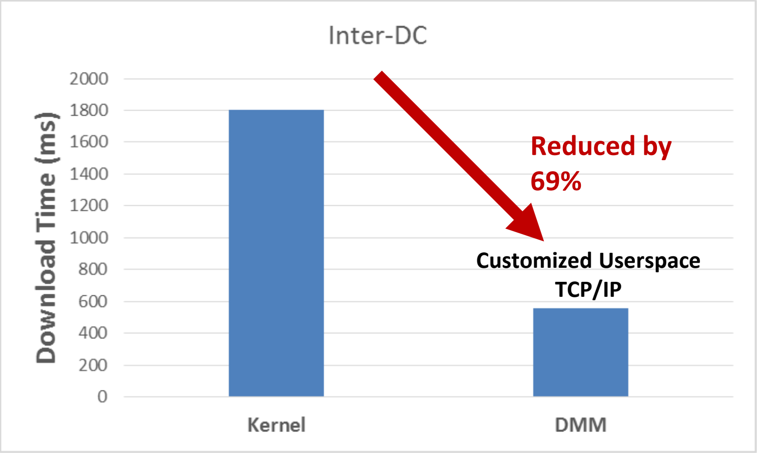
Demo: Protocol Routing for Multi-network Client-Server Application

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


Intra-DC

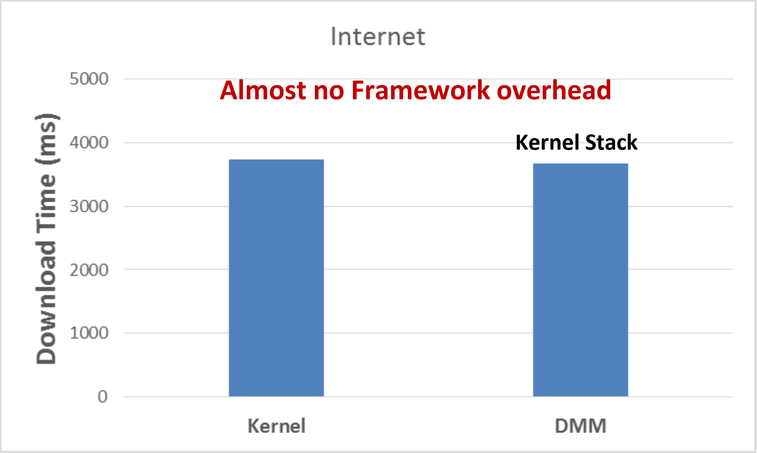
Reduced by 97%

RDMA


Inter-DC

Reduced by 69%

Customized Userspace TCP/IP


Internet

Almost no Framework overhead

Kernel Stack

HUAWEI

Key Takeaway

DMM: Key takeaways

- Flexibility to **dynamically choose different protocols** according to performance and/or functional requirements
- **End-to-end orchestration** to maintain stack instances and the app/socket-to-stack mappings
- Extendable transport protocol plug-in framework to host **multiple stack instances** simultaneously
- Let stack developers concentrate on **user space protocol innovation**

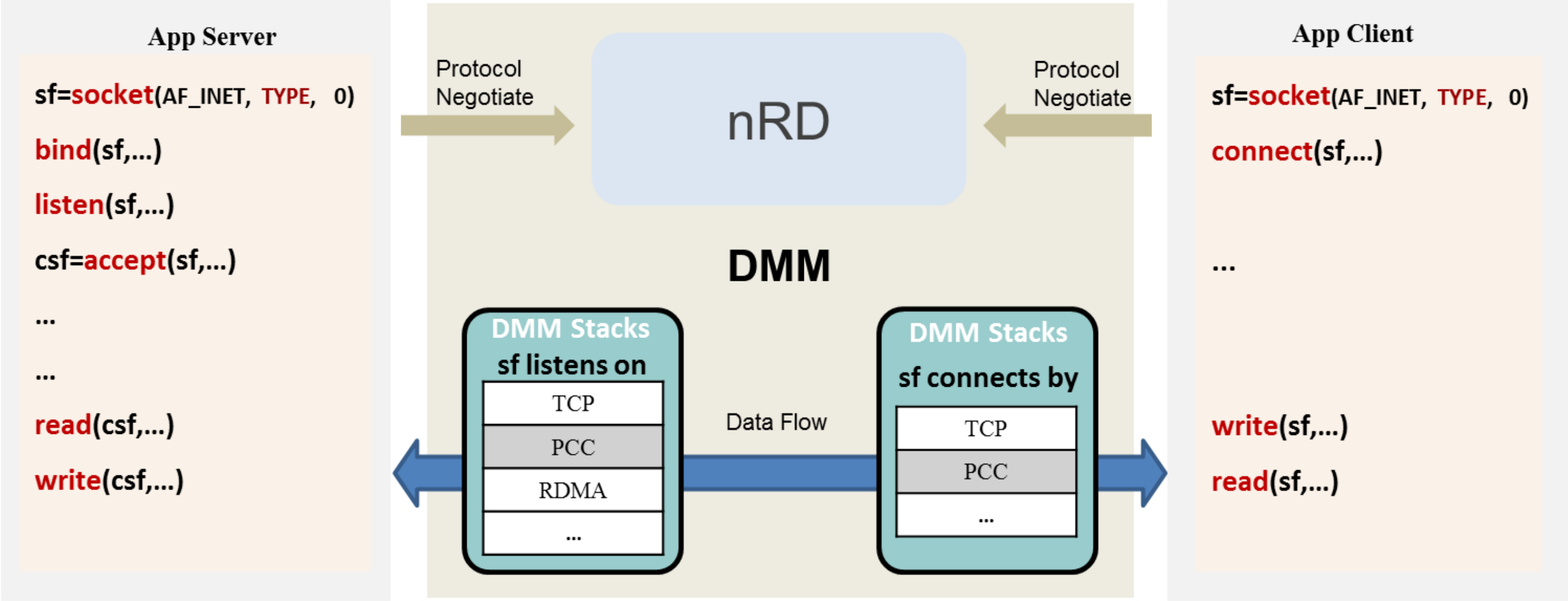
DMM: Benefits to application developers/end-users

- **Friendly Acceleration:**

✓ Acceleration w/ backward compatible API, friendly to the legacy

- **Adaptive and customized Acceleration:**

✓ 'Protocol Routing' based on network env, application requirements and host information



App Server

```
sf=socket(AF_INET, TYPE, 0)
bind(sf,…)
listen(sf,…)
csf=accept(sf,…)
…
…
read(csf,…)
write(csf,…)
```

App Client

```
sf=socket(AF_INET, TYPE, 0)
connect(sf,…)
…
write(sf,…)
read(sf,…)
```

'Protocol Routing' workflow

DMM: Benefits to stack/protocol researchers/developers

- **Friendly interfaces to integrate new protocol stacks**

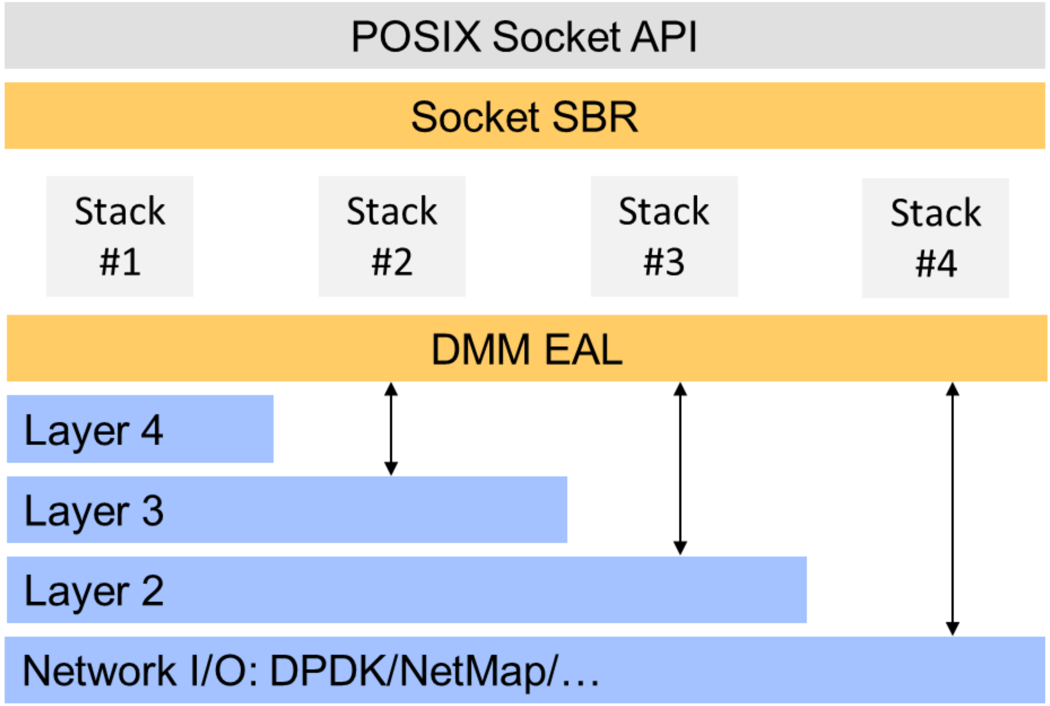
✓ Flexible Pkt I/O NIC/L2/L3/L4 (EAL)

✓ Simplified API (SBR)

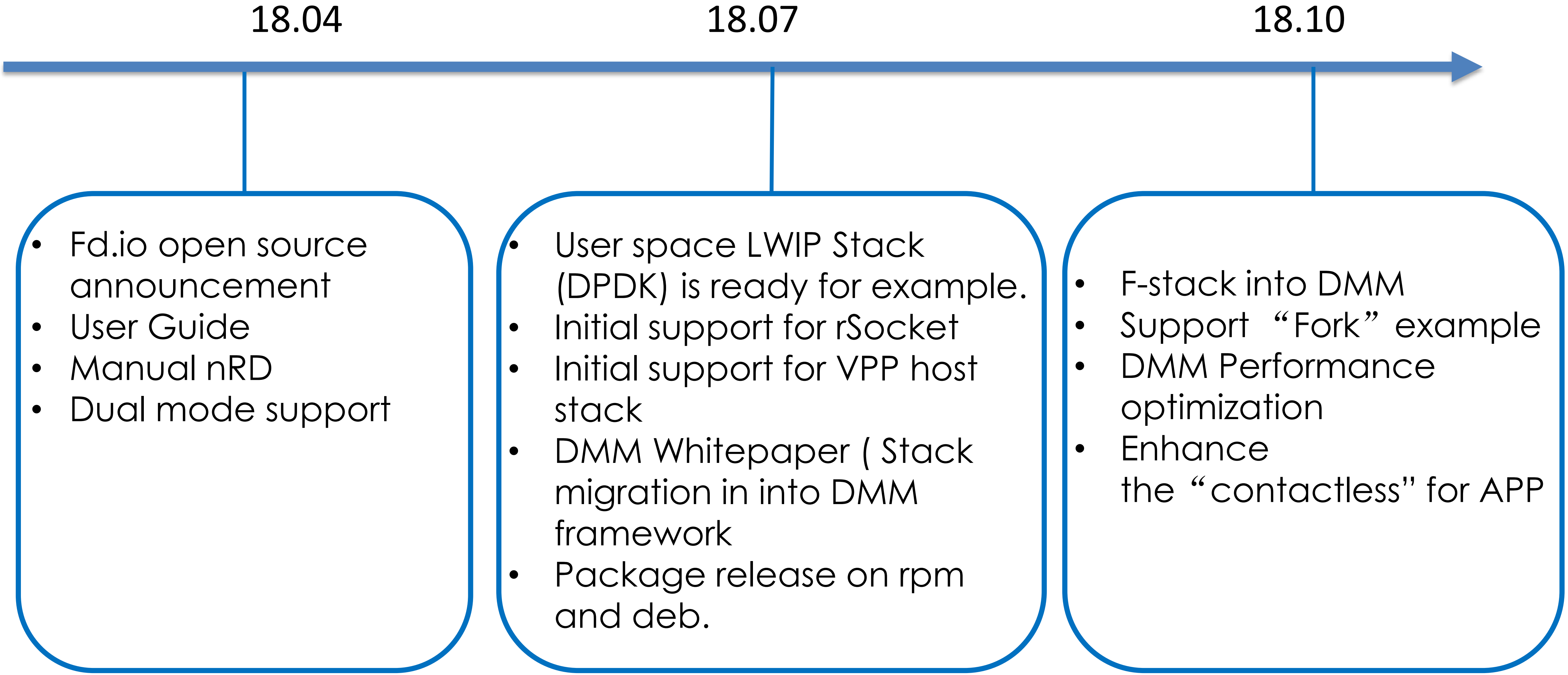
- **Accelerate innovation of new protocol stacks**

✓ Modular and reusable function blocks w/ high perf.

✓ Integration w/ both kernel and user space (VPP)



DMM project roadmap



18.04

- Fd.io open source announcement
- User Guide
- Manual nRD
- Dual mode support

18.07

- User space LWIP Stack (DPDK) is ready for example.
- Initial support for rSocket
- Initial support for VPP host stack
- DMM Whitepaper (Stack migration in into DMM framework
- Package release on rpm and deb.

18.10

- F-stack into DMM
- Support “Fork” example
- DMM Performance optimization
- Enhance the “contactless” for APP

Welcome to join in us

- FD.io DMM Web Site
- https://wiki.fd.io/view/DMM
- Code
- https://git.fd.io/dmm
- Contact us
- Mail list: dmm-dev@lists.fd.io
- IRC: #fdio-dmm

HUAWEI

Thank You.

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.