Empower Diverse Open Transport Layer Protocol in the Cloud Networking

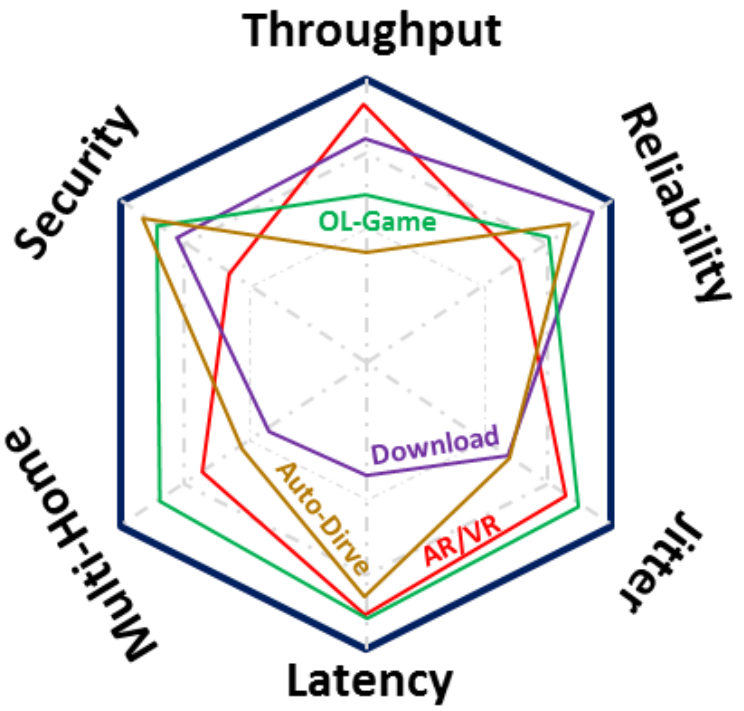
Qing Chang

Challenges - What We Face Today

- **TCP Protocol - a performance bottleneck**
  - ✓ Loss sensitive
  - ✓ Poor in larger bandwidth-delay product network
  - ✓ No delivery latency SLA guarantee
  - ✓ Difficult to tune performance
- **Operating System Kernel Stack**
  - ✓ Low performance
  - ✓ Monolithic in design
  - ✓ Hard to customize
  - ✓ Long protocol/algorithm release cycle

Challenges - 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 - Some Answers

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

Challenges

- **One-size-fits-all protocol is not feasible**

DMM: Re-design the Protocol Stack

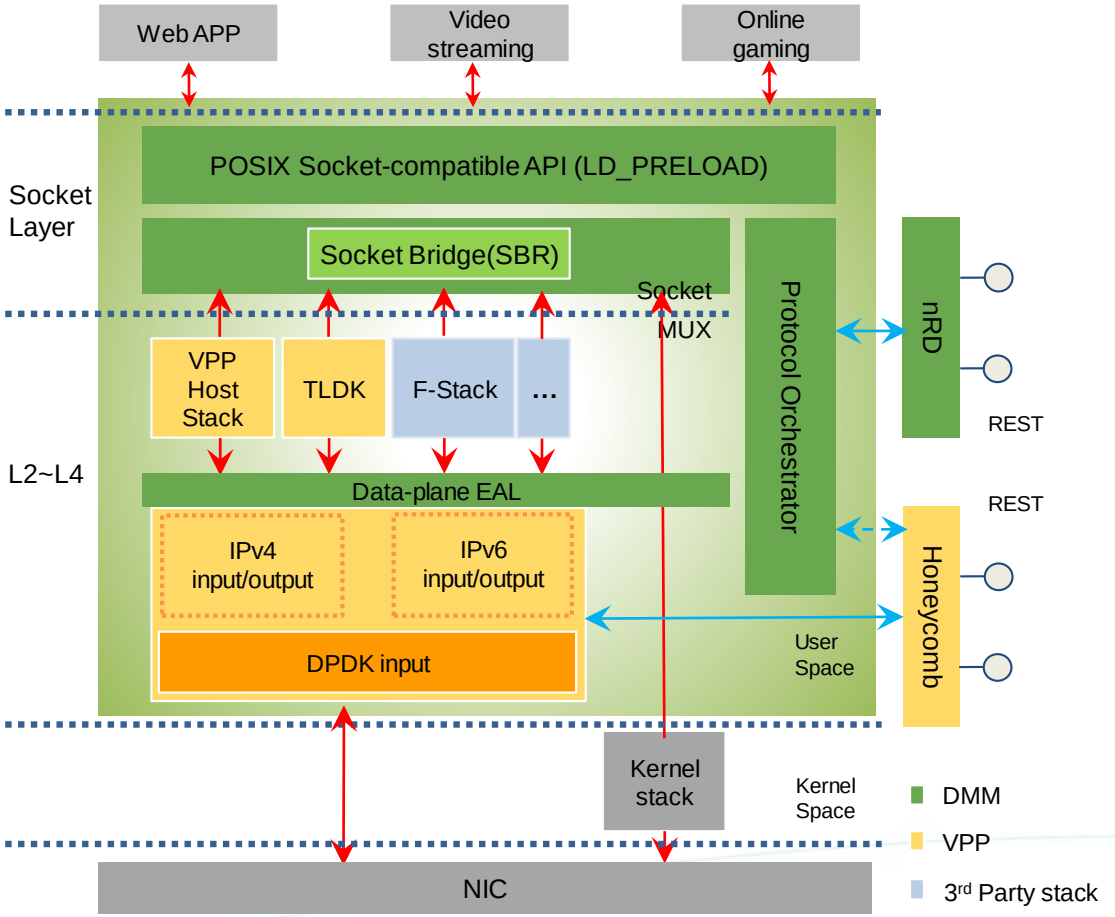
DMM (Dual modes Multi-protocols Multi-instances) is a network stack framework which enables:

- ✓ **Dual mode:** Kernel space and User space
- ✓ **Multi-protocols:** Simple new protocol adoptions and integrations
- ✓ **Multi-instances:** Enable “Protocol Routing”

DMM Architecture

Key Techniques

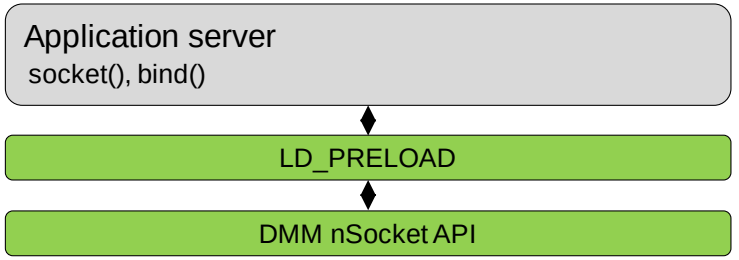
- ✓ Distributed and Centralized nRD deployment (LRD & CRD) provide end-to-end protocol orchestration
- ✓ Stack-transparent “Protocol Routing” (Stack orchestrator)
- ✓ POSIX compatible socket APIs
- ✓ Flexible socket API redirection and mapping (SBR)
- ✓ Flexible APIs for integration of third party stacks (EAL)
- ✓ Multiple stack instances support
- ✓ Multiple I/O engines support

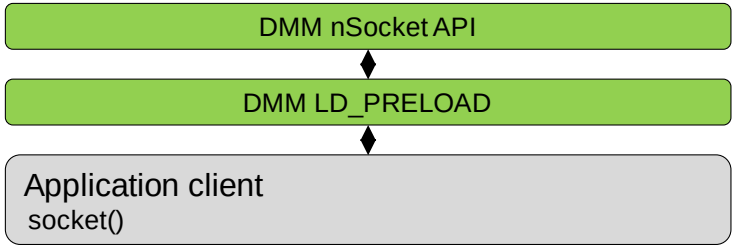


DMM: Protocol Routing Workflow

1 Application server and client calls socket interface.

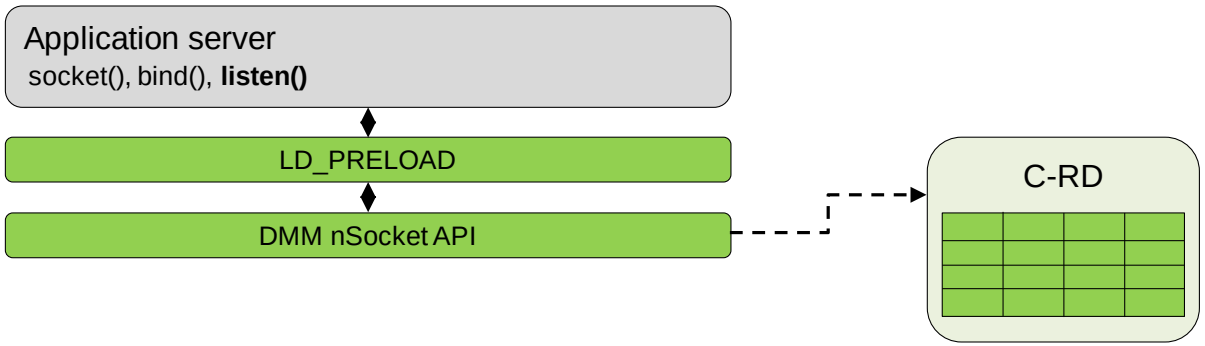
DMM: Protocol Routing Workflow



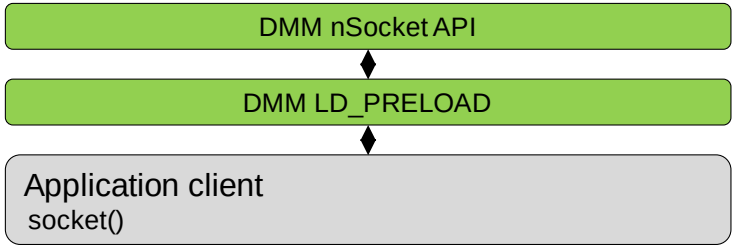


1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.

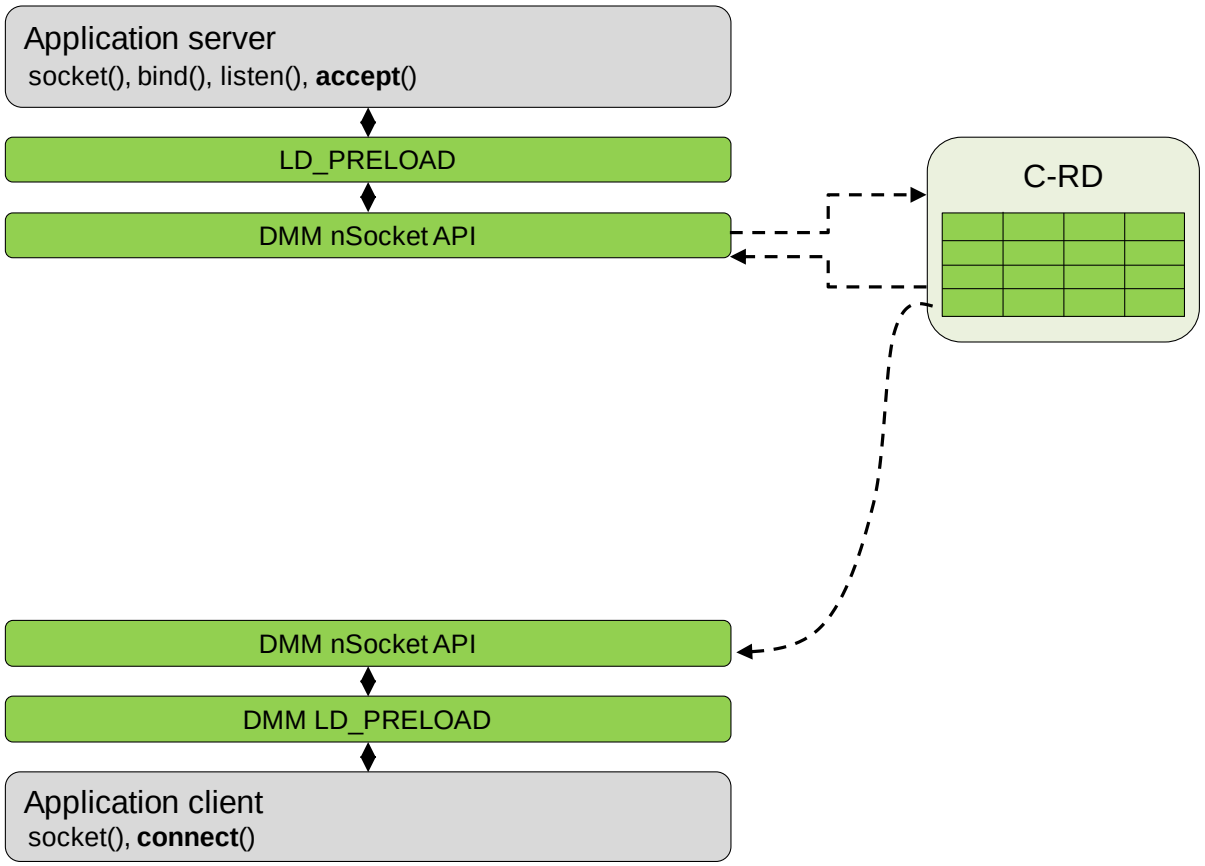
DMM: Protocol Routing Workflow



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 publish capacity policies and preference policies to C-RD.

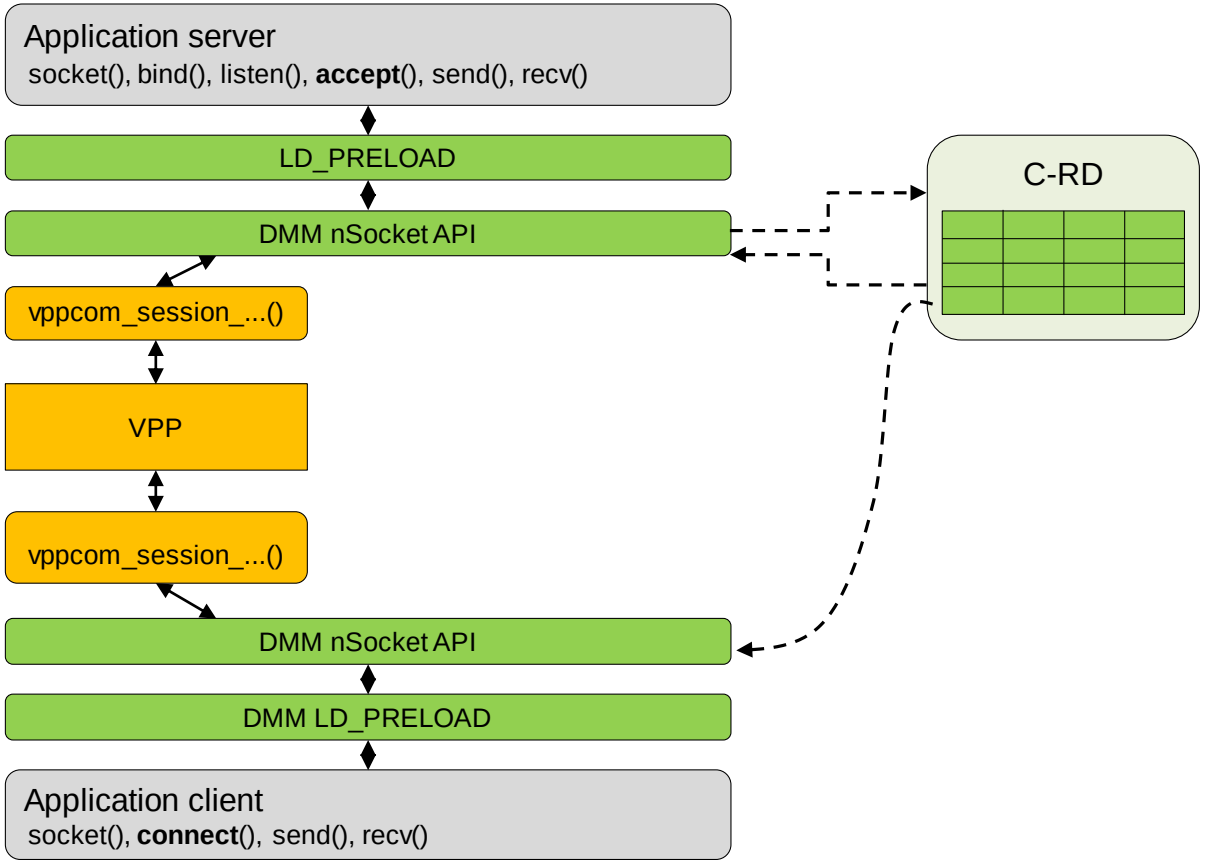


DMM: Protocol Routing Workflow



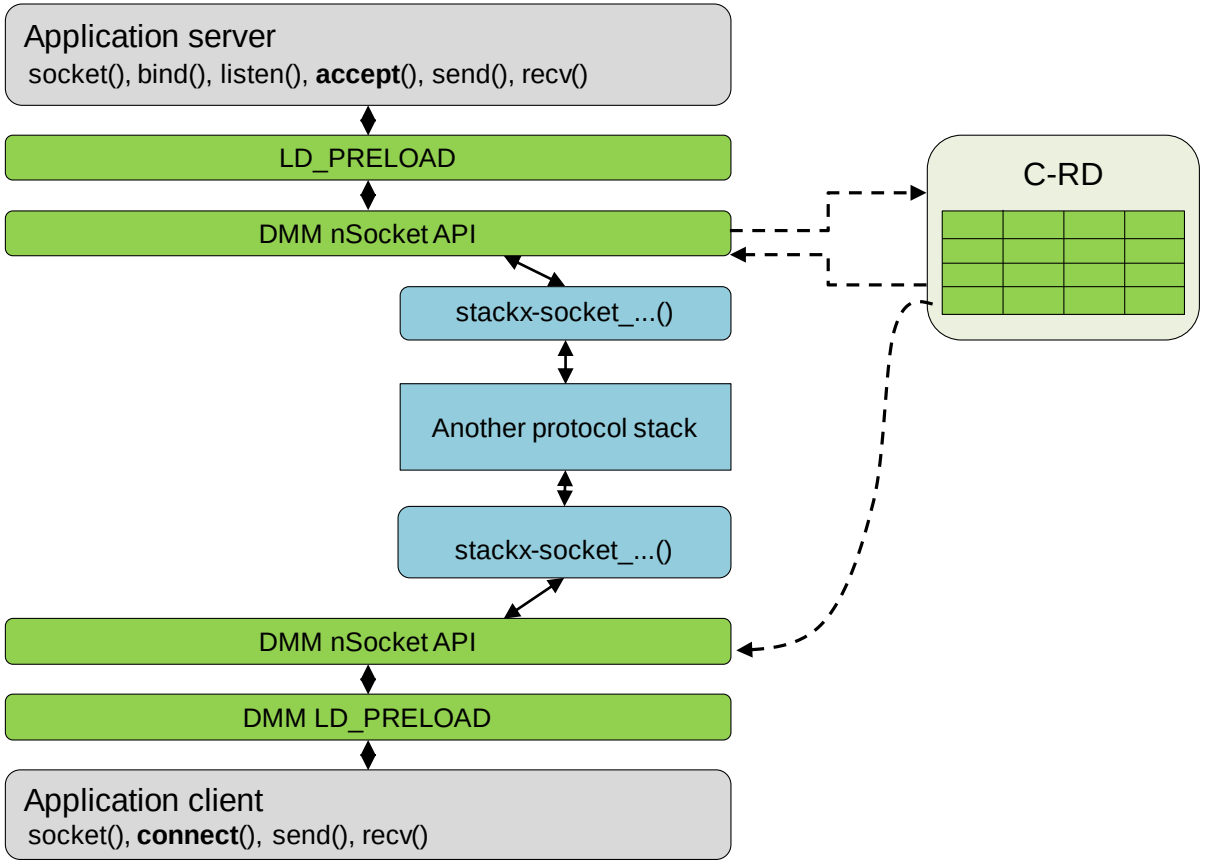
1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call `listen()` triggers L-RD to publish capacity policies and preference policies to C-RD.
4. Server call `accept()` and client call `connect()` trigger L-RD to retrieve and resolve protocol stack mapping.

DMM: Protocol Routing Workflow



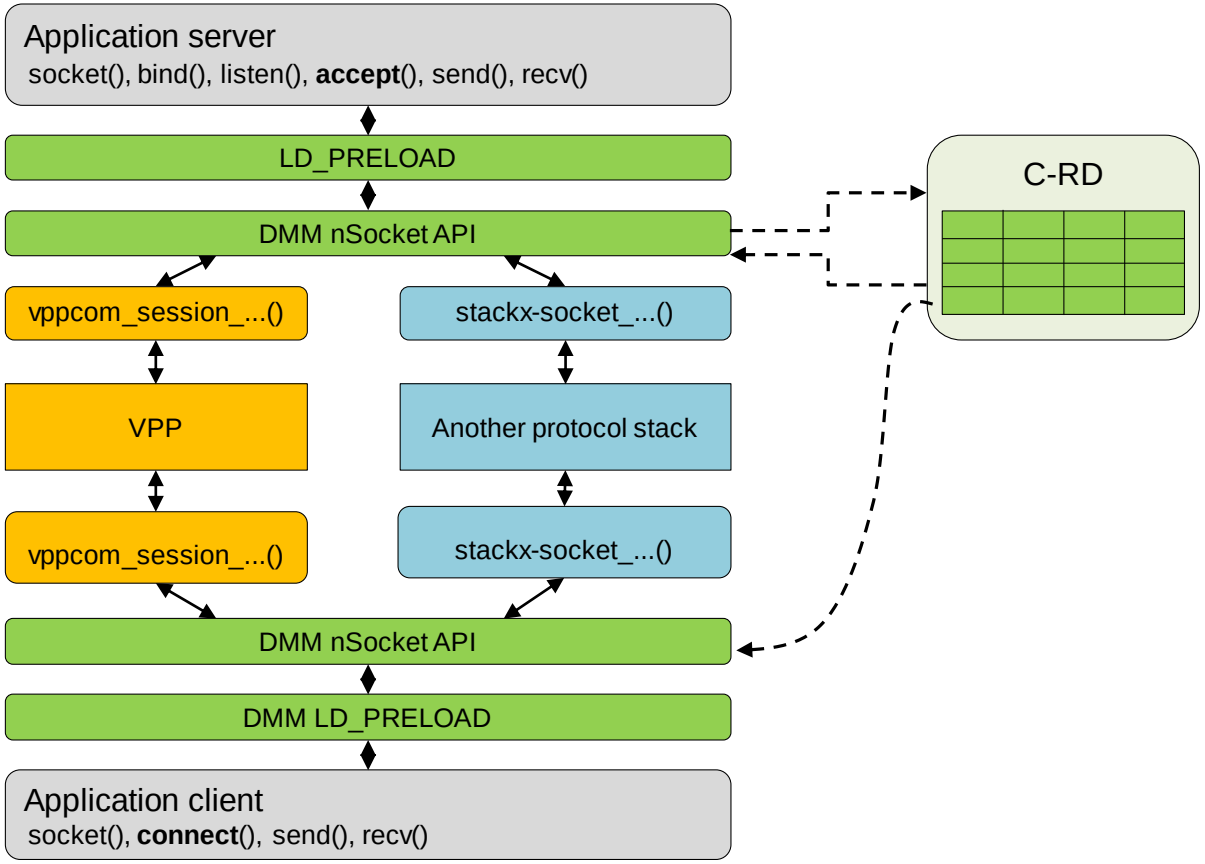
1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call `listen()` triggers L-RD to publish capacity policies and preference policies to C-RD.
4. Server call `accept()` and client call `connect()` trigger L-RD to retrieve and resolve protocol stack mapping.
5. According to the mapping, the socket is instantiated to one protocol stack

DMM: Protocol Routing Workflow



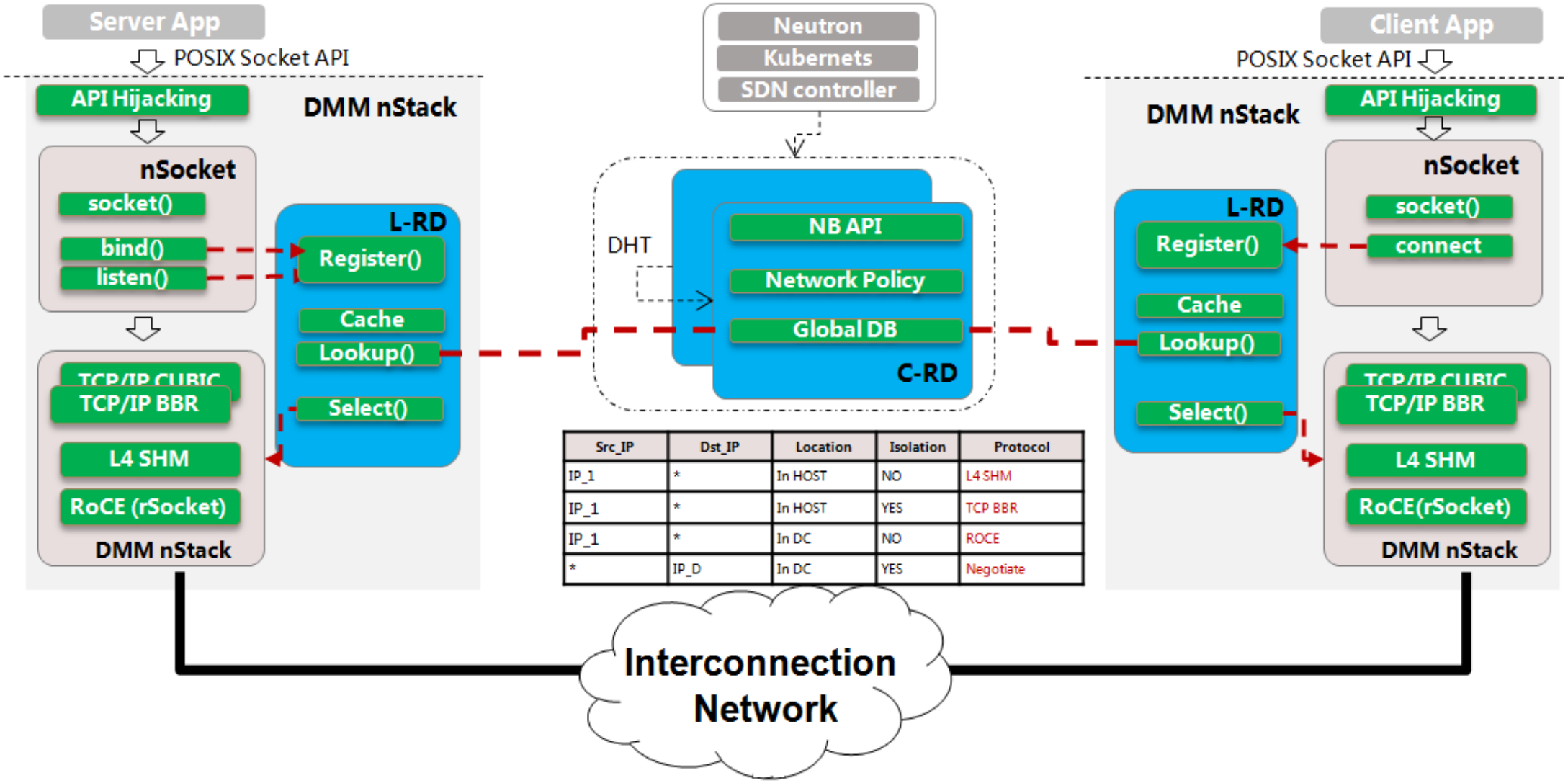
1. Application server and client calls socket interface.
2. Socket APIs are hijacked to DMM nSocket APIs.
3. Server call `listen()` triggers L-RD to publish capacity policies and preference policies to C-RD.
4. Server call `accept()` and client call `connect()` trigger L-RD to retrieve and resolve protocol stack mapping.
5. According to the mapping, the socket is instantiated to one protocol stack or another.

DMM: 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 publish capacity policies and preference policies to C-RD.
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.

DMM: End to End Network Protocol Orchestration



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

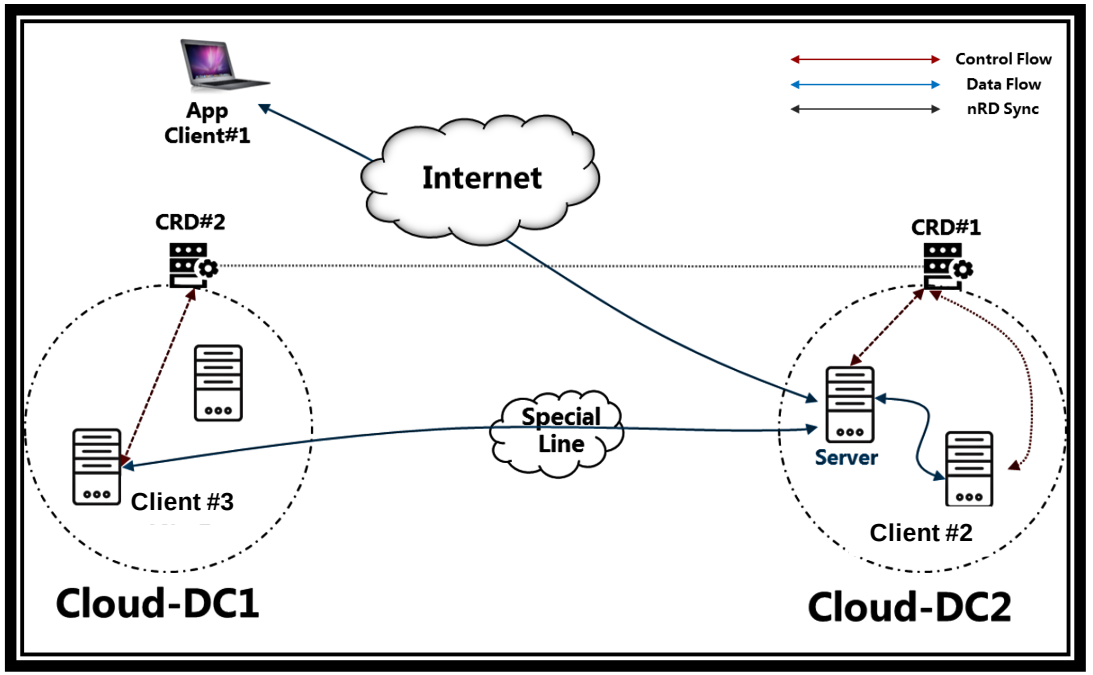
Demo: Protocol Routing for Multi-network Client-Server Application

File Sync Application

- 3 Clients --> Server

Network Setting

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



Comparison scheme

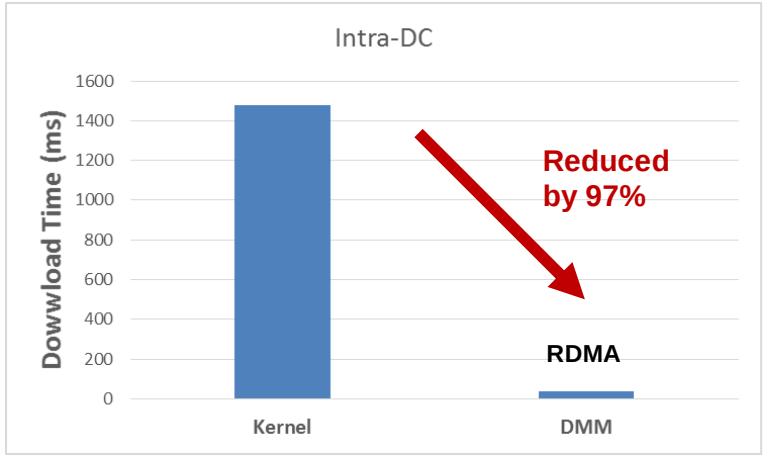
- Default: the kernel TCP/IP stack
- DMM: automatically negotiate appropriate stacks (kernel TCP/IP, customized user-space TCP/IP, RDMA) on different network

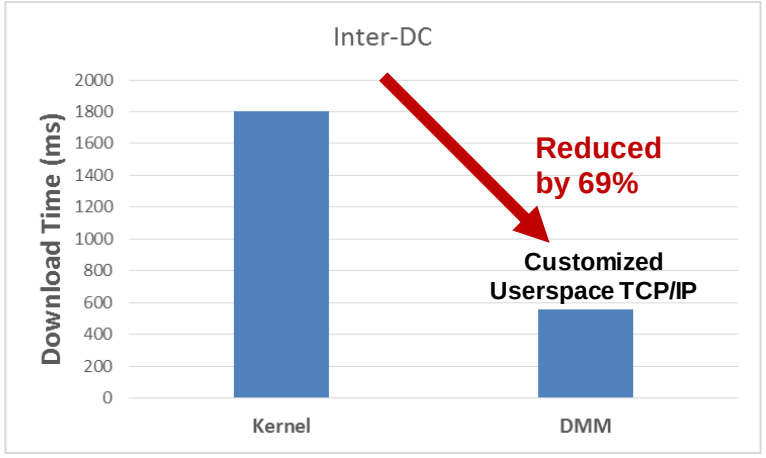
Demo: Protocol Routing for Multi-network Client-Server Application

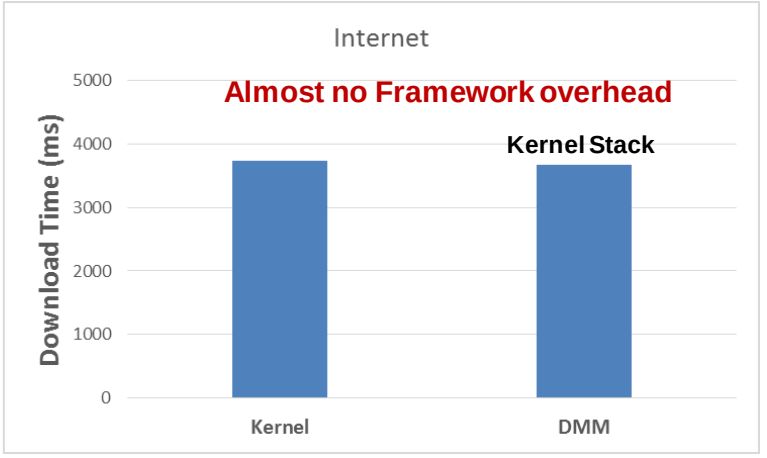
- Insert Demo video

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.

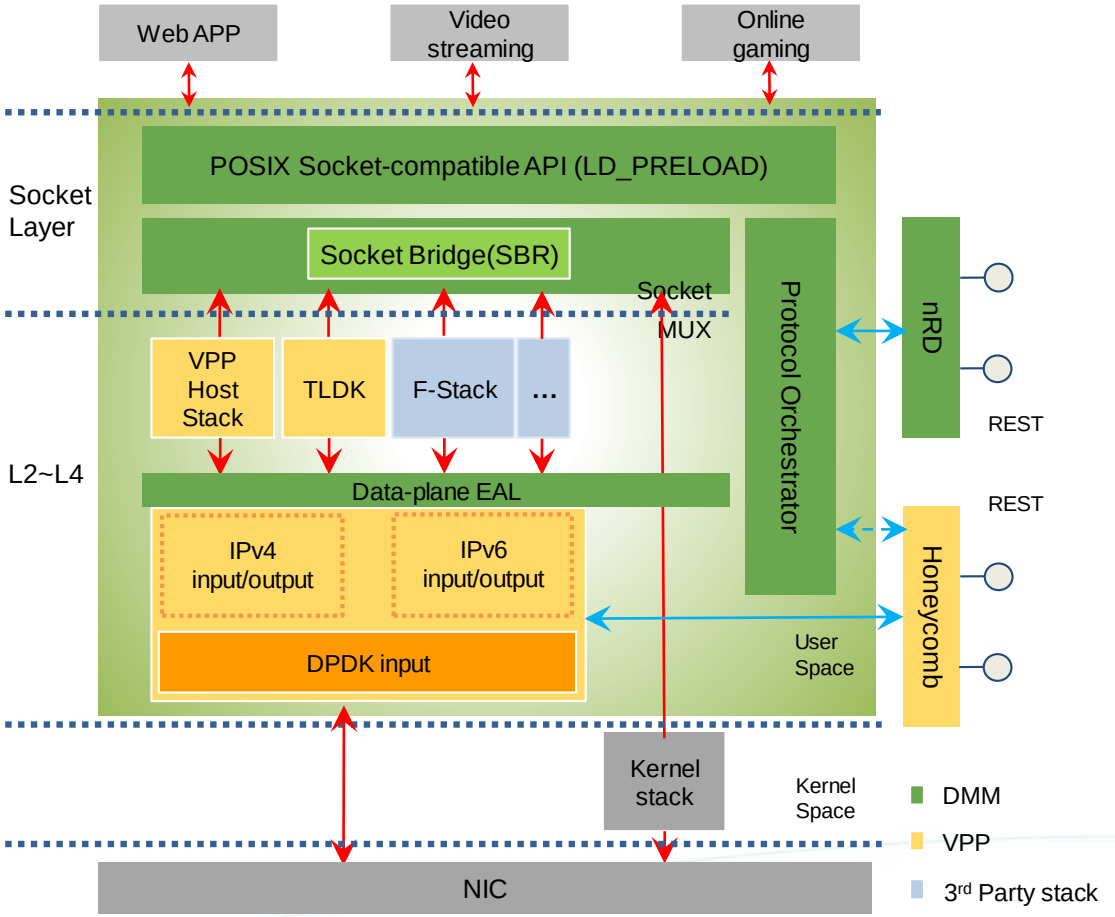






DMM: Key takeaways

- 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



Thank You.

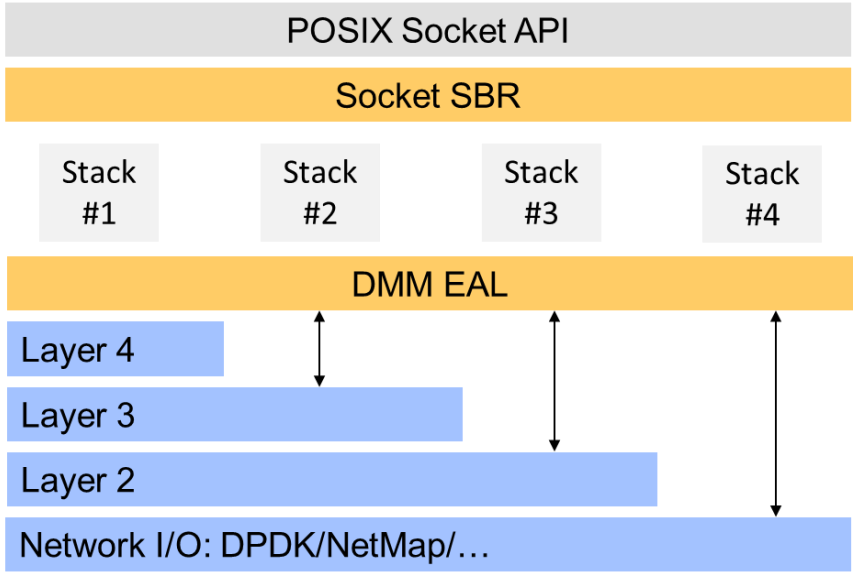
DMM: Benefit To Stack/Protocol Developers

Friendly interfaces to integrate

- ✓ Mechanisms for socket redirect
- ✓ Flexible I/O APIs including NIC/L2/L3/L4 APIs

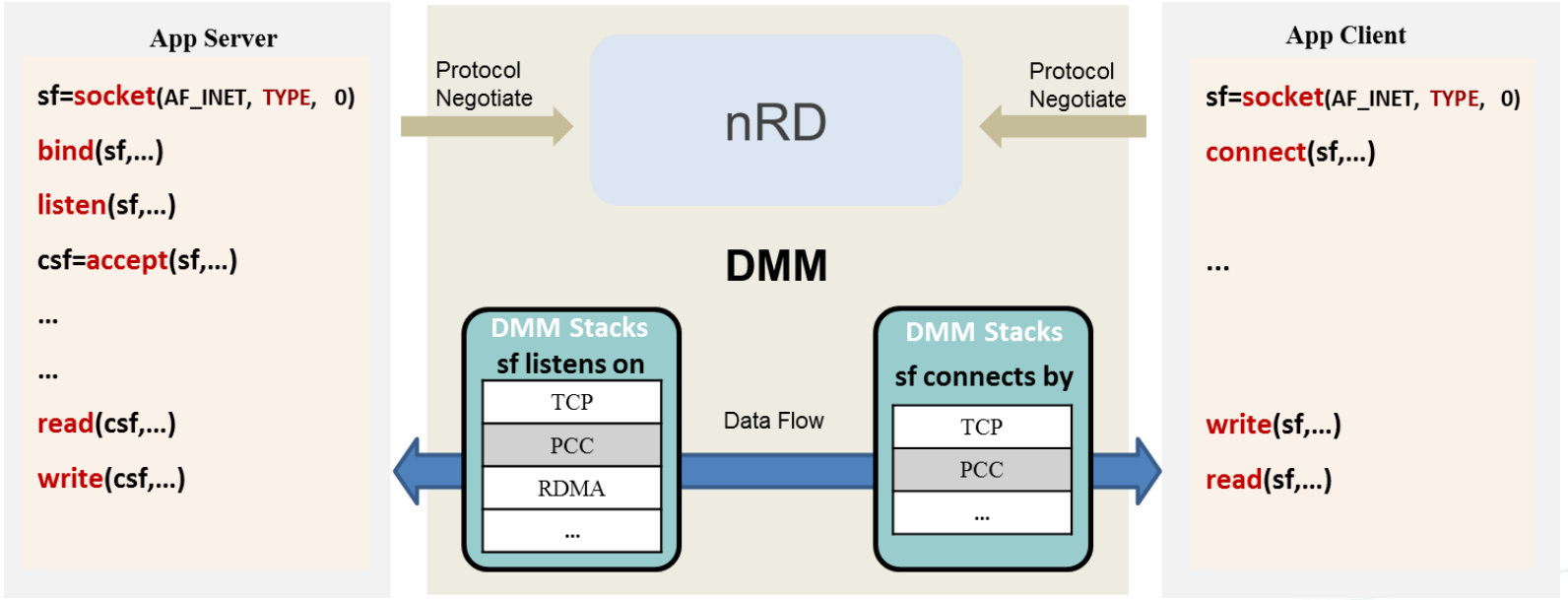
Accelerate innovation of new stacks

- ✓ Concentrate on the core function of the stack
- ✓ Easy to be deployed
- ✓ Easy to be adopted/tested by applications



DMM: Benefit To Application Developers

- ✓ Extended POSIX socket APIs which is backward compatible for legacy applications
- ✓ ‘Protocol Routing’ based on network env, application requirements and host information
- ✓ Rapidly benefit from the new stacks/protocols without any changes on the application code



‘Protocol Routing’ workflow

DMM-Demo-2: Dual mode support for Nginx Server

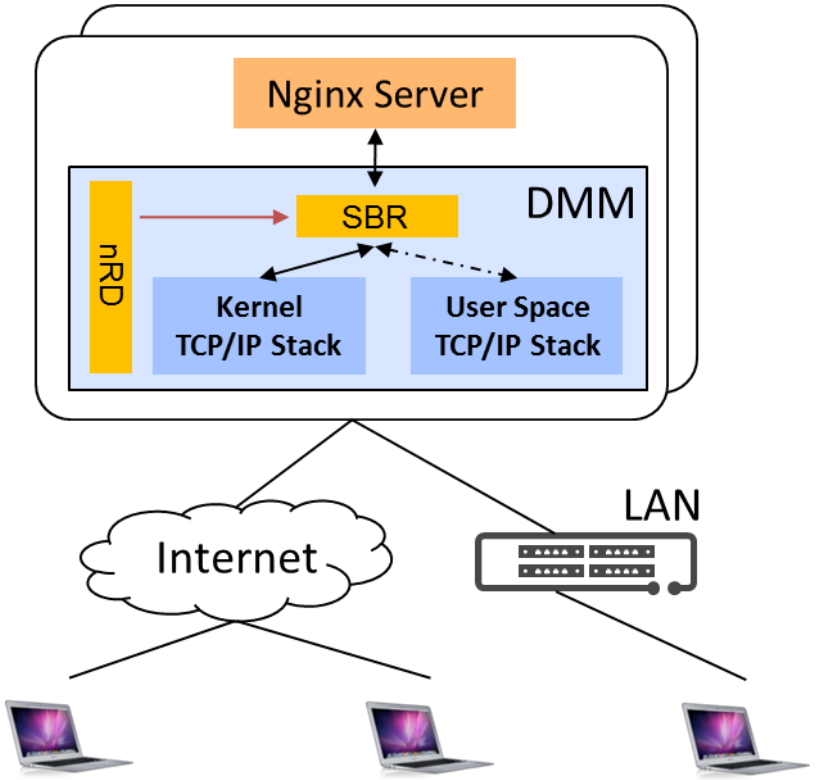
Nginx application

- kernel stack vs user-space stack ?

DMM nRD Policy (Example):

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

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



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