User-space MPTCP Practices inside Data Center

System Techonology Engineer (STE)

ByteDance

10th March 2025

NetDev 0x19

youzhiqiang@bytedance.com

wangwanchen.0316@bytedance.com

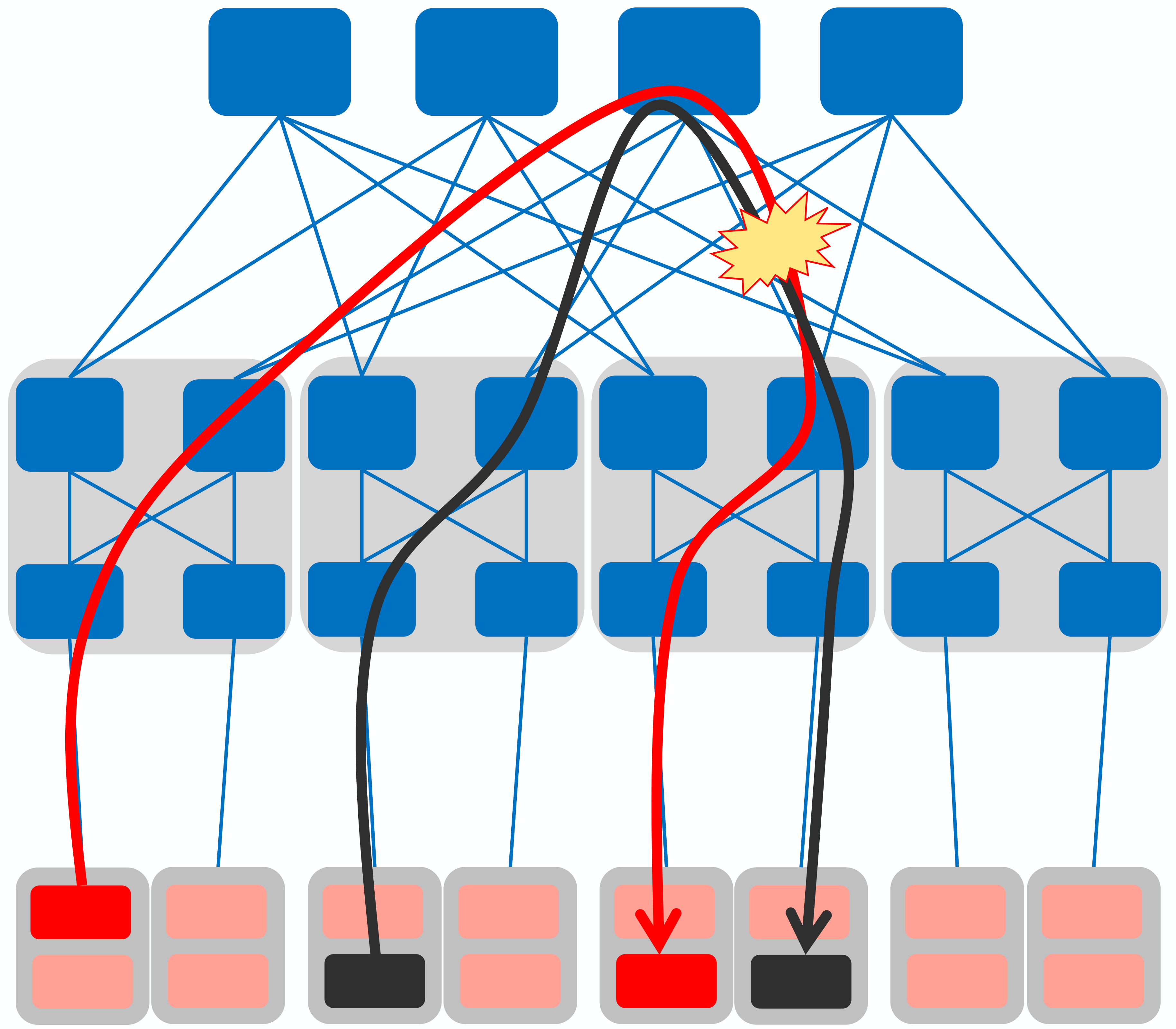
ByteDance 字节跳动

Agenda

- **Background**
  - **Why MPTCP？**
  - **Why DPDK based MPTCP？**
- **Design**
- **Performance Data**
- **Future Work**

Background – Why MPTCP?

- **Avoid service interruption caused by failure of a single network node or a single network path**
  - **MPTCP establishes multiple TCP sub-flows between computing nodes, and can quickly detect a sub-flow failure and switch traffic to others**
- **Maximize utilization of network bandwidth**
  - **A single flow cannot meet the bandwidth requirements of some applications**
  - **MPTCP setups multiple sub-flows by using different source ports and distribute traffic among all sub-flows**
- **Large-scale existing TCP-based applications**
  - **MPTCP can fallback to TCP**
- **Linux kernel support since 5.6**
  - **Easy to scale in data centers**



Background – Why DPDK based MPTCP？

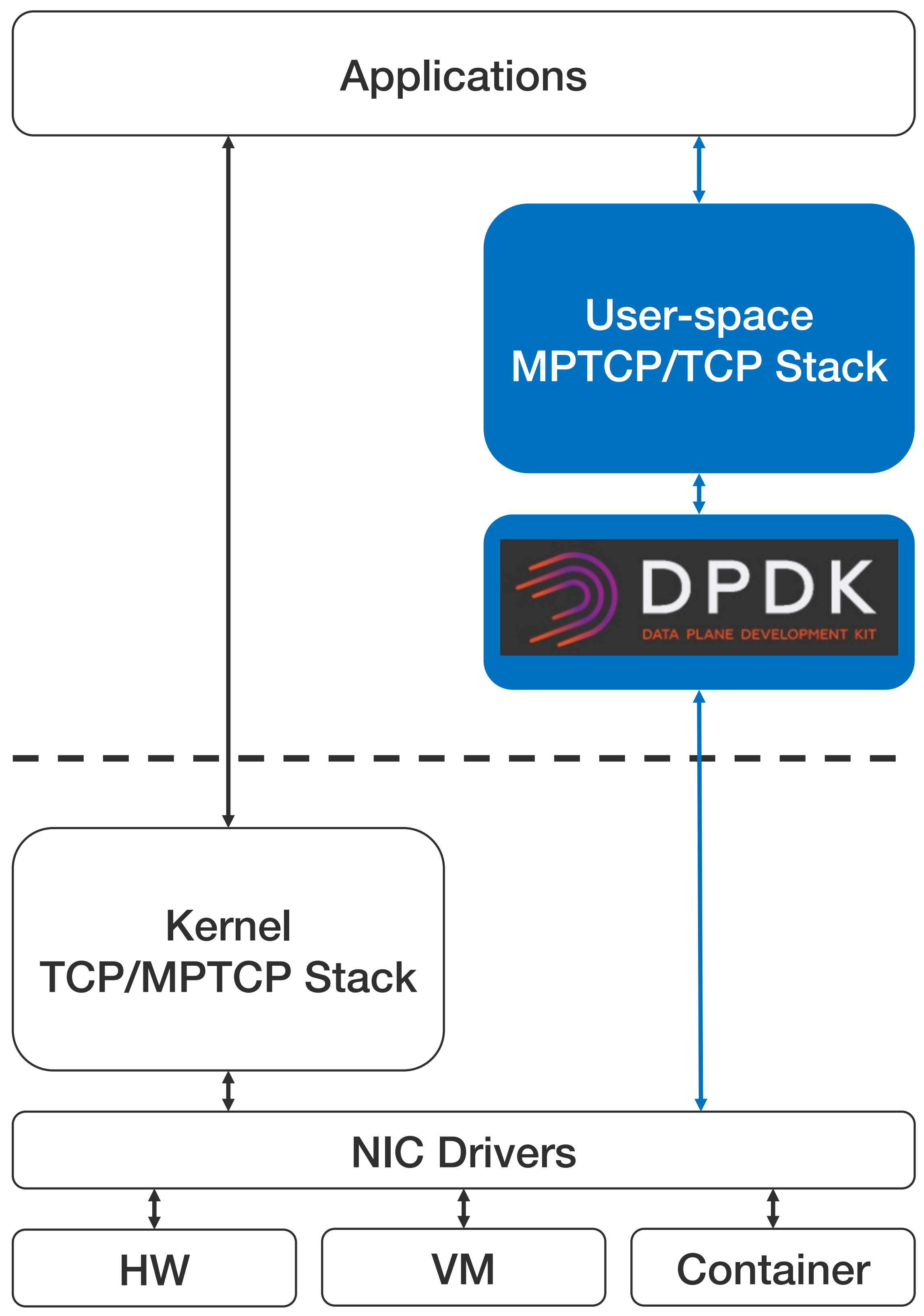
- **High Performance Packet Processing Requirements**
  - **Bottlenecks in kernel-based MPTCP**
  - **DPDK Acceleration:**

    Kernel bypass

    Zero Copy

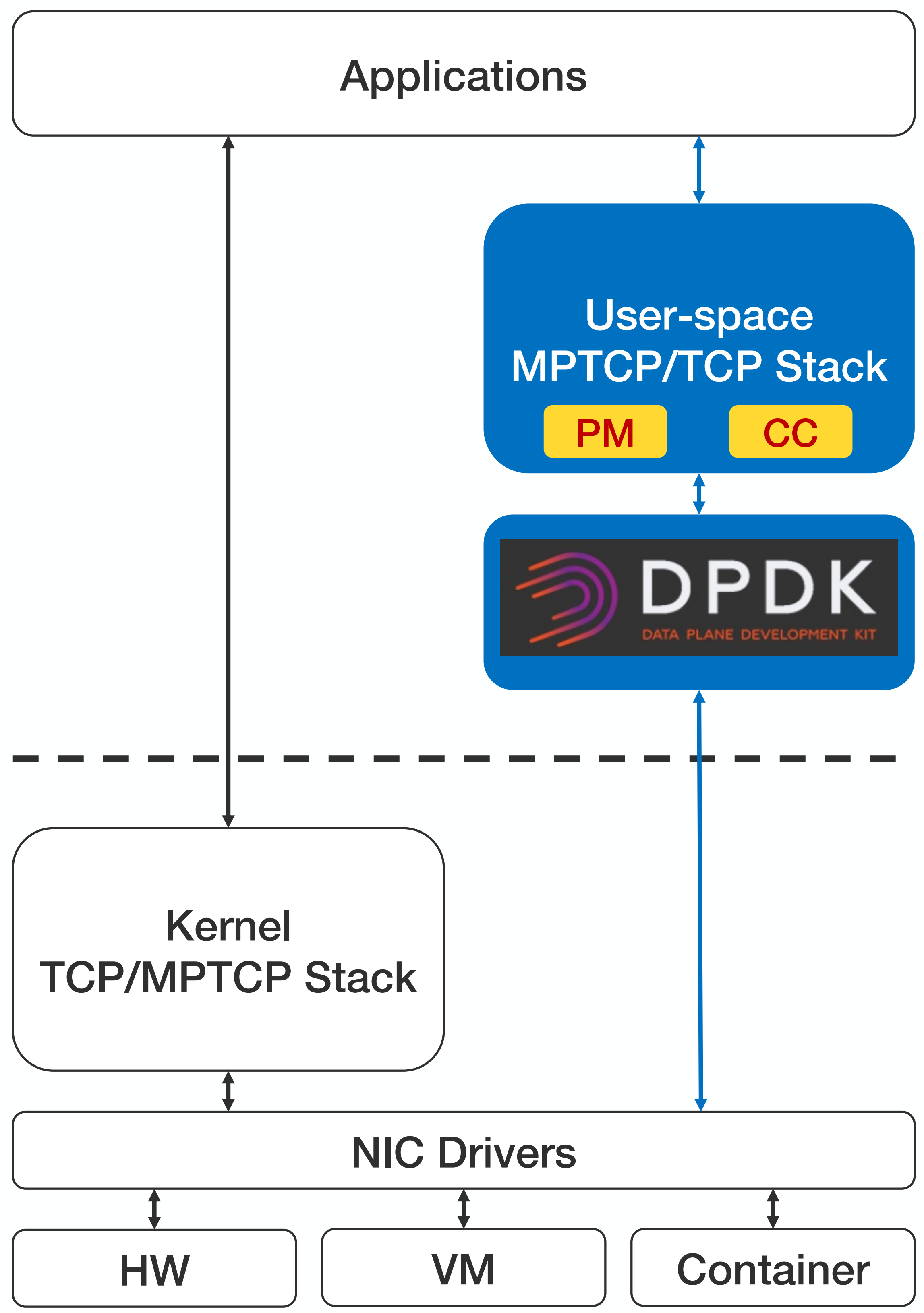
    Batch Processing

    Polling mode



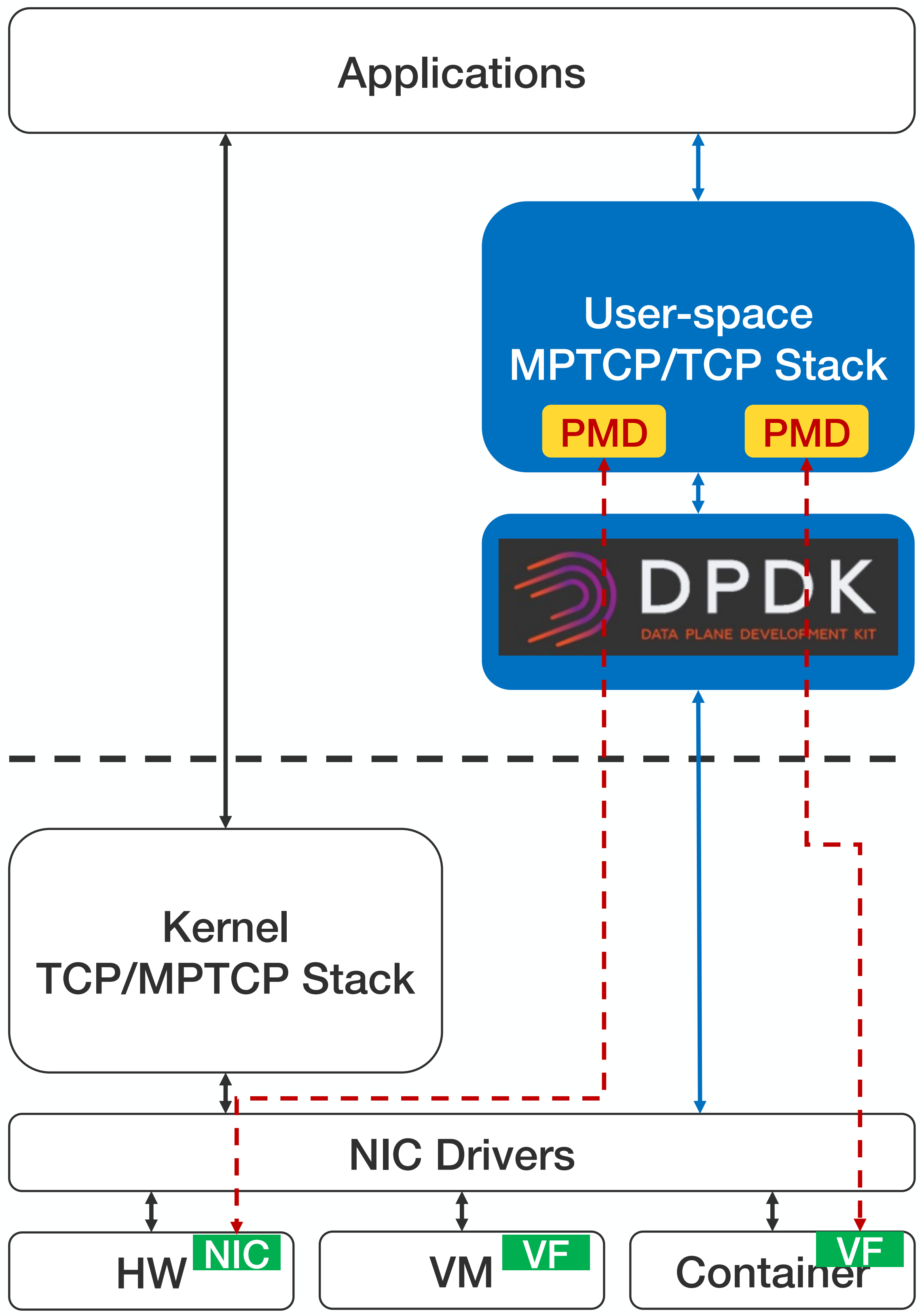
Background – Why DPDK based MPTCP？

- **Optimized Multipath Transmission**
  - **DPDK enables User-space traffic scheduling algorithms to achieve Fine Grained Path Control**
  - **DPDK's fast processing supports more flexible congestion control, path selection and traffic distribution**



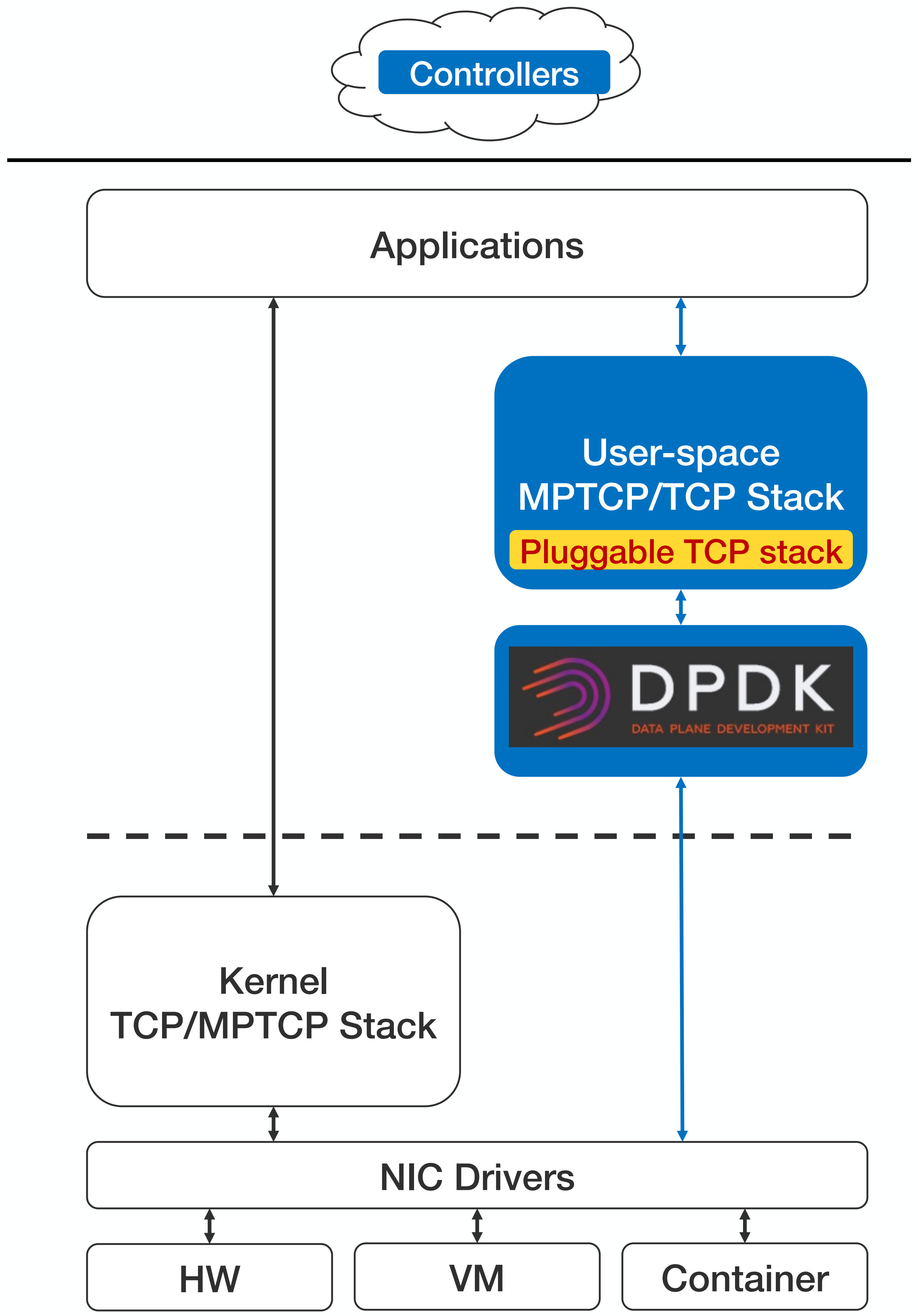
Background – Why DPDK based MPTCP？

- **Resource Utilization and Scalability**
  - **DPDK inherently supports parallel processing across CPU cores**
  - **Virtualization and Container Acceleration based on SR-IOV**



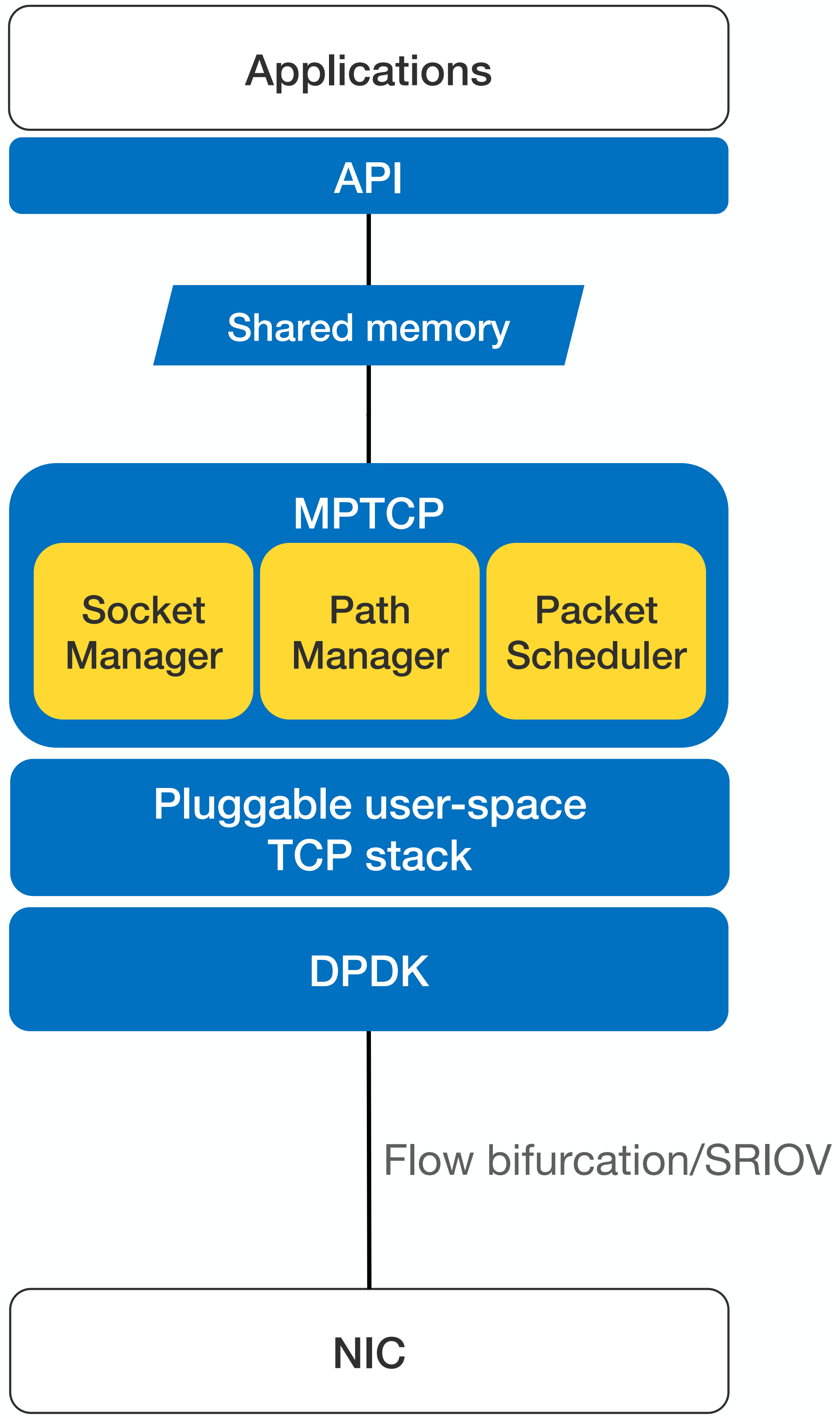
Background – Why DPDK based MPTCP？

- **Flexibility and Customization**
  - **DPDK's user-space implementation allows protocol logic modifications without kernel complexity**
  - **Can interoperate with SDN controllers for dynamic path adjustments**
  - **Can integrate with NFV to enable intelligent traffic steering in service chains**



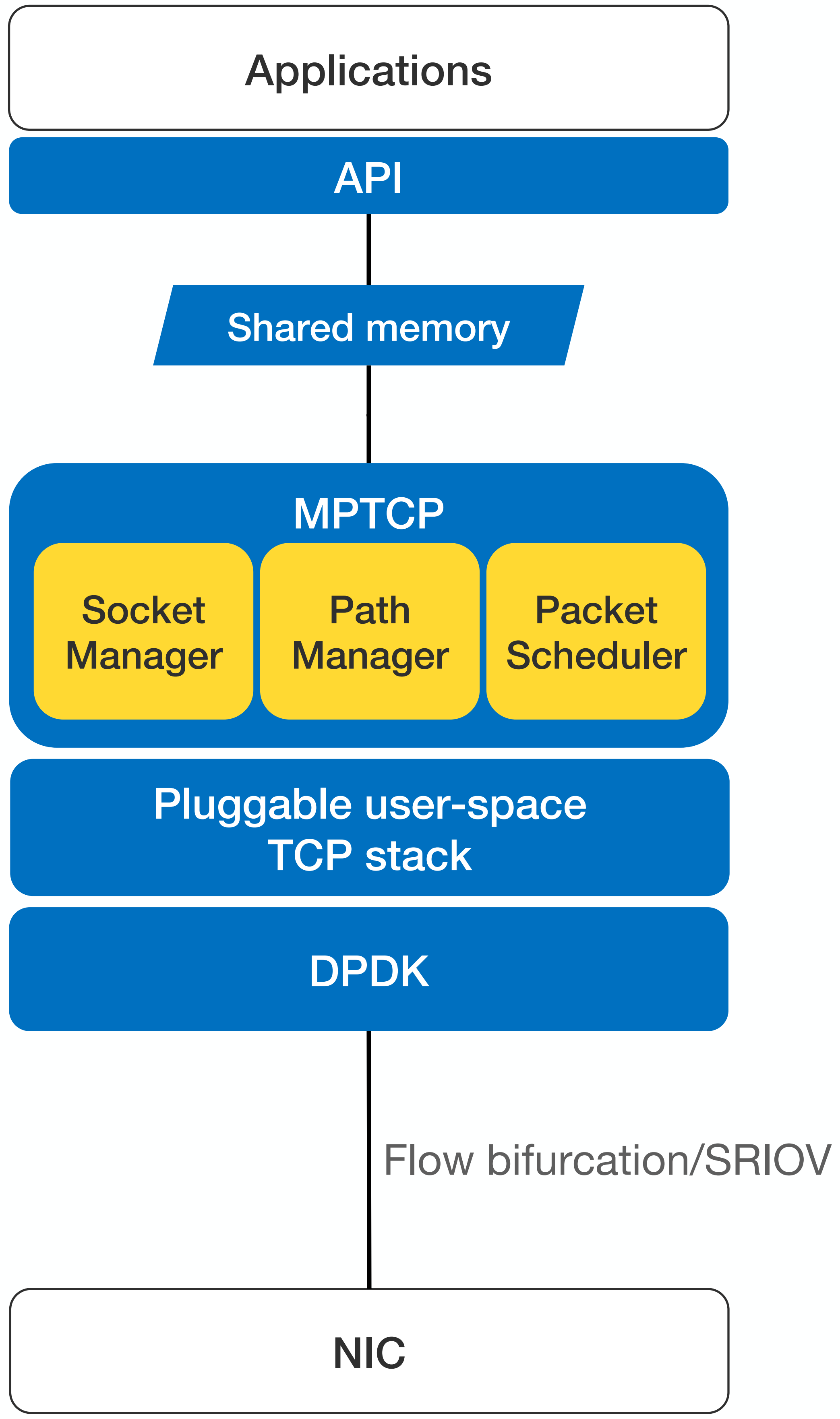
Design - Architecture

- **MPTCP/TCP as a dedicated service**
  - Dedicated process to support multiple applications
  - Communicating with application via shared memory
  - API is integrated into applications as SDK
  - On top of DPDK
  - Use flow bifurcation (NV NIC) or SRIOV (non-NV NIC) to distribute traffic on NIC
  - Composed of 3 modules:
    * Socket Manager
    * Path Manager
    * Packet Scheduler



Design - Architecture

- **Socket Manger**
  - **- Context Management of MPTCP sockets**
  - **- Mapping with TCP sockets**



ByteDance字节跳动

Design - Architecture

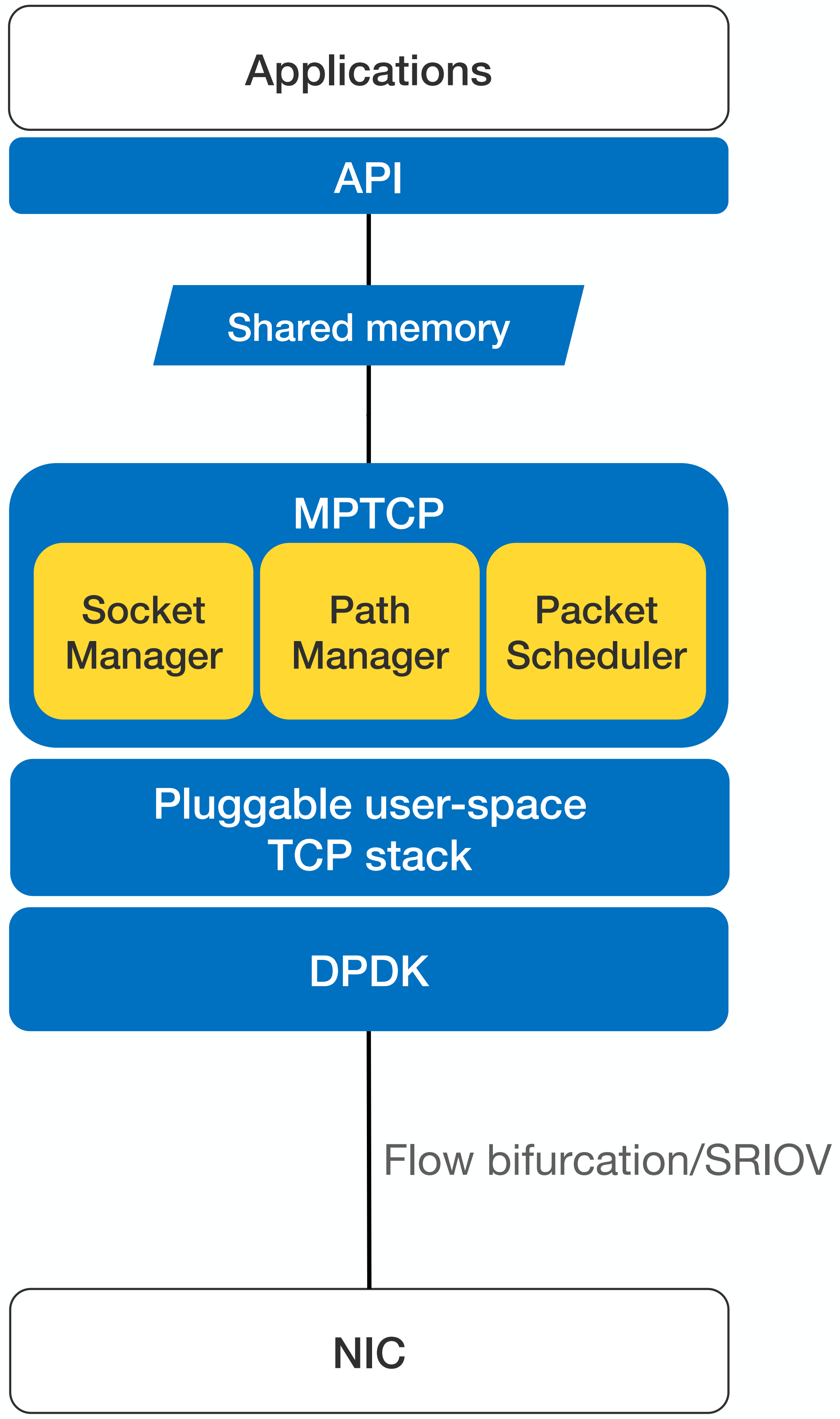
- **Path Manger**

  - Responsible for the life cycle management of sub-flows:

  *** creation**

  *** deletion**

  *** address announcements**



Design - Architecture

- **Packet Scheduler**

  - Responsible for selecting which available *sub-flow(s)* to use to send the next data packet

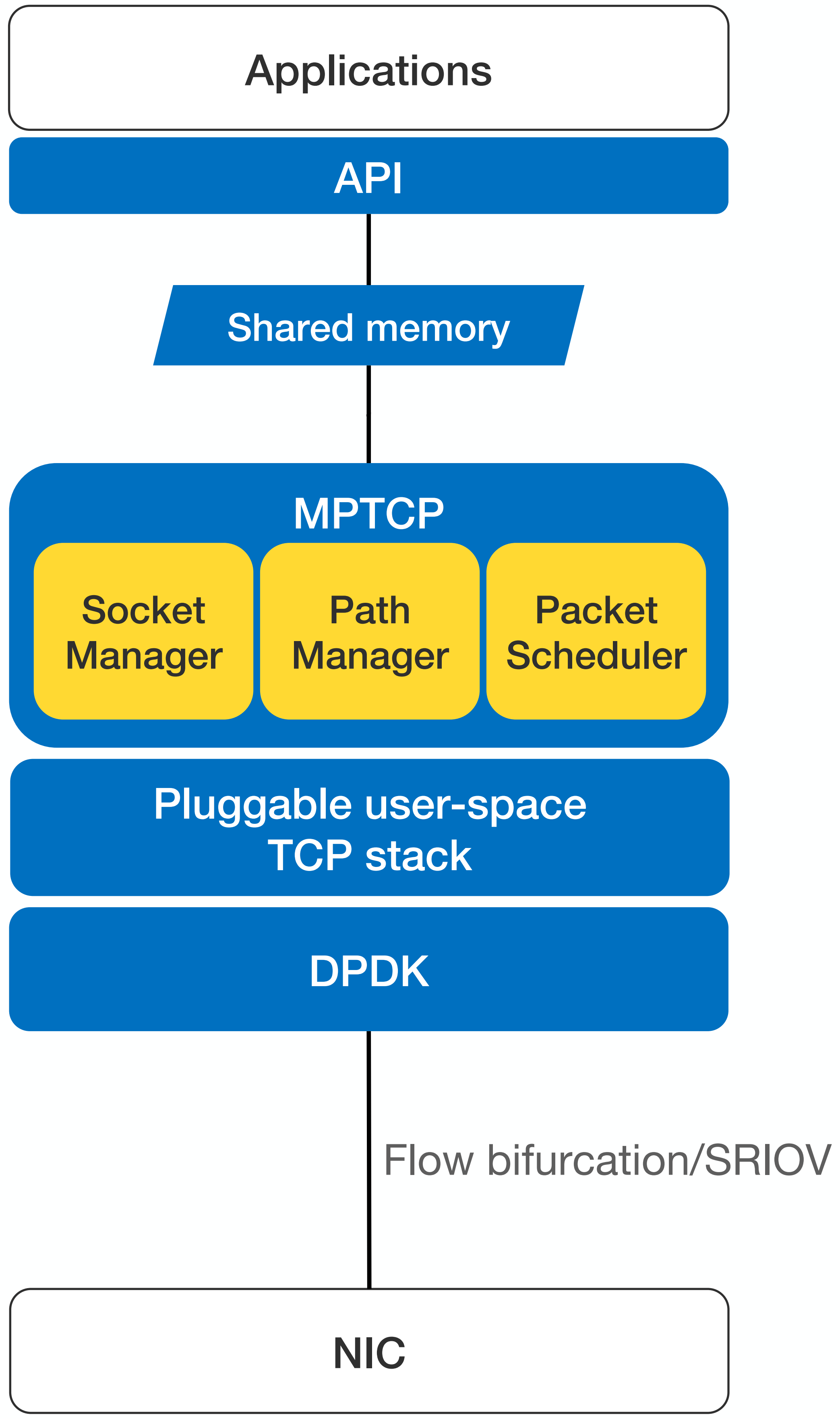
  - Can decide to maximize the use of the available bandwidth

  - Configurable policies:

    * Round robin

    * Pick the path with the lower latency

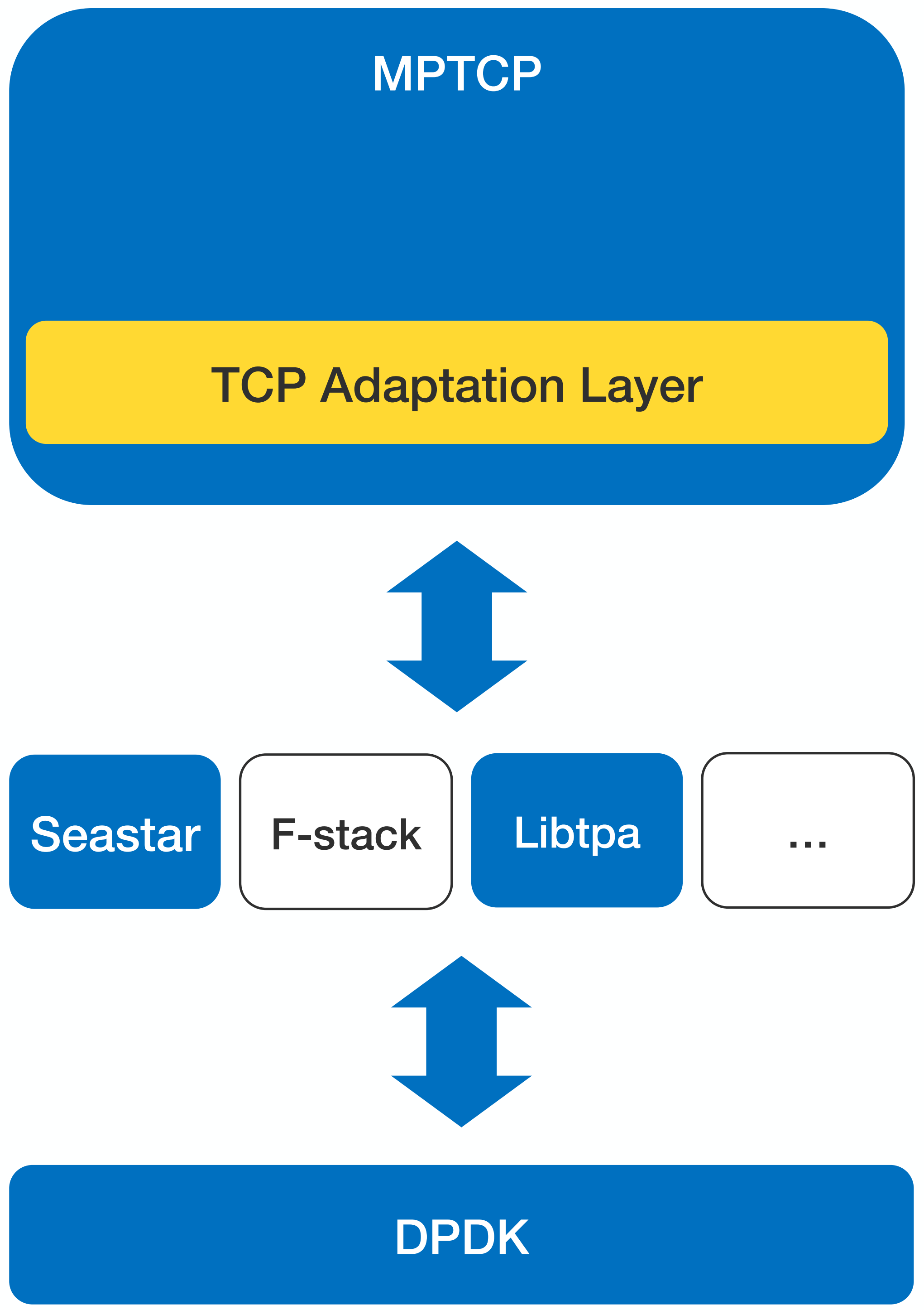
    * Any other policy depending on the configuration



Design – Pluggable user-space TCP stack

- **Decoupling from the underly TCP stack**
  - **Introduce a TCP adaptation layer**
  - **Integrate the underly TCP stack as a library**

- **Can switch the underly stack as needed to upgrade the existing user-space TCP to MPTCP**

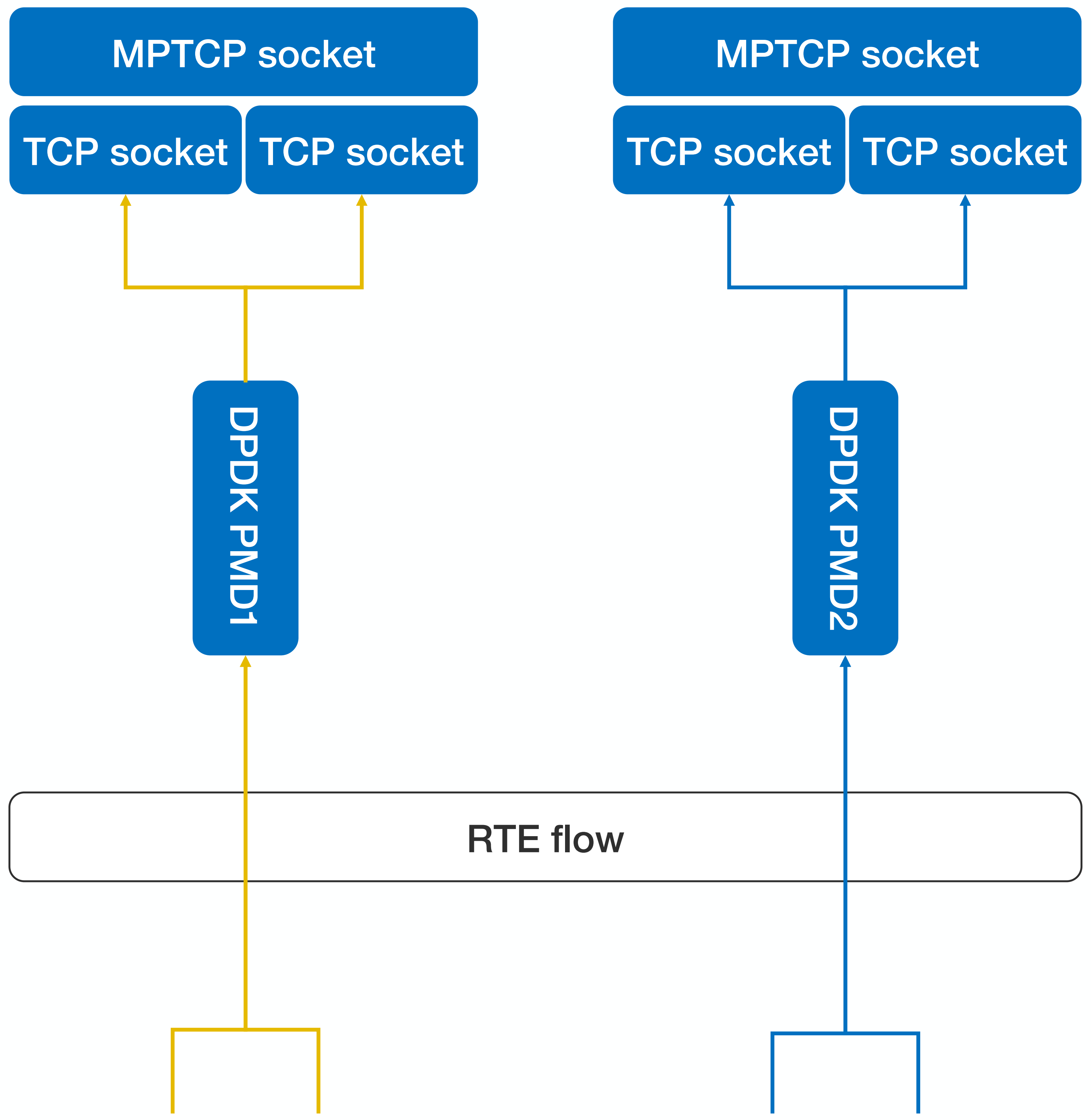


Design – Keep sharing nothing among PMDs

The purpose is to ensure that all sub-flows of the same MPTCP connection are processed in the same DPDK PMD to achieve lock-free forwarding

- **Client**
  - **- new connections uses port ranges rte-flow**
  - **- connection forwarding uses 5 tuple rte-flow**

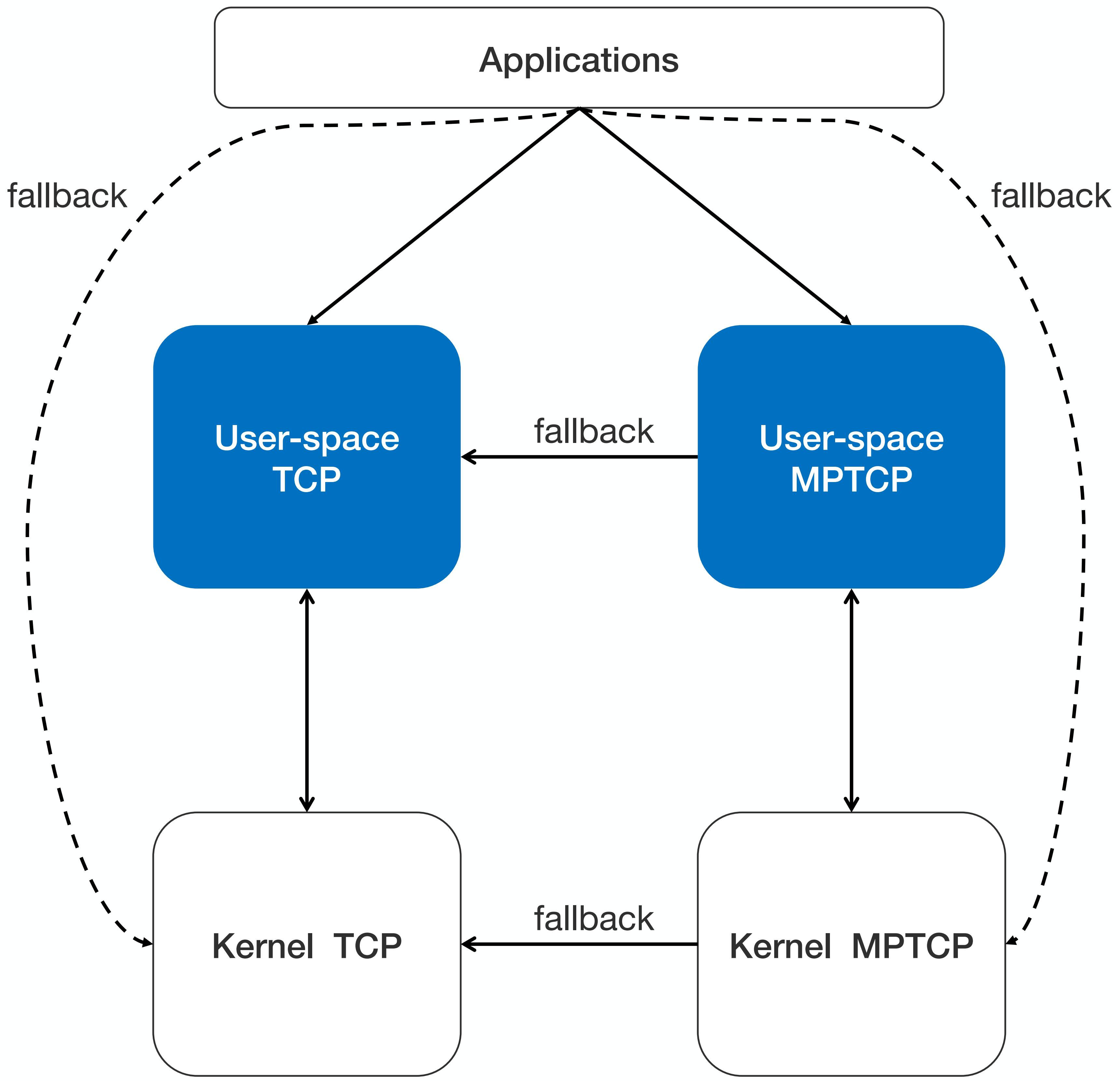
- **Server**
  - **- new connections uses listen ports rte-flow per PMD**
  - **- connection forwarding uses 5 tuple rte-flow**



Design – Compatible with kernel

- **Fallback**
  - **- User-space MPTCP fallbacks to user-space TCP**
  - **- Kernel MPTCP fallbacks to kernel TCP**
  - **- Applications fallbacks to kernel stack in case of user-space stack unavailable**

- **Comply with RFC specs**
  - **- User-space MPTCP is compatible with Kernel MPTCP**
  - **- User-space TCP is compatible with kernel TCP**
  - **- Enable One-sided deployment**



Performance Data

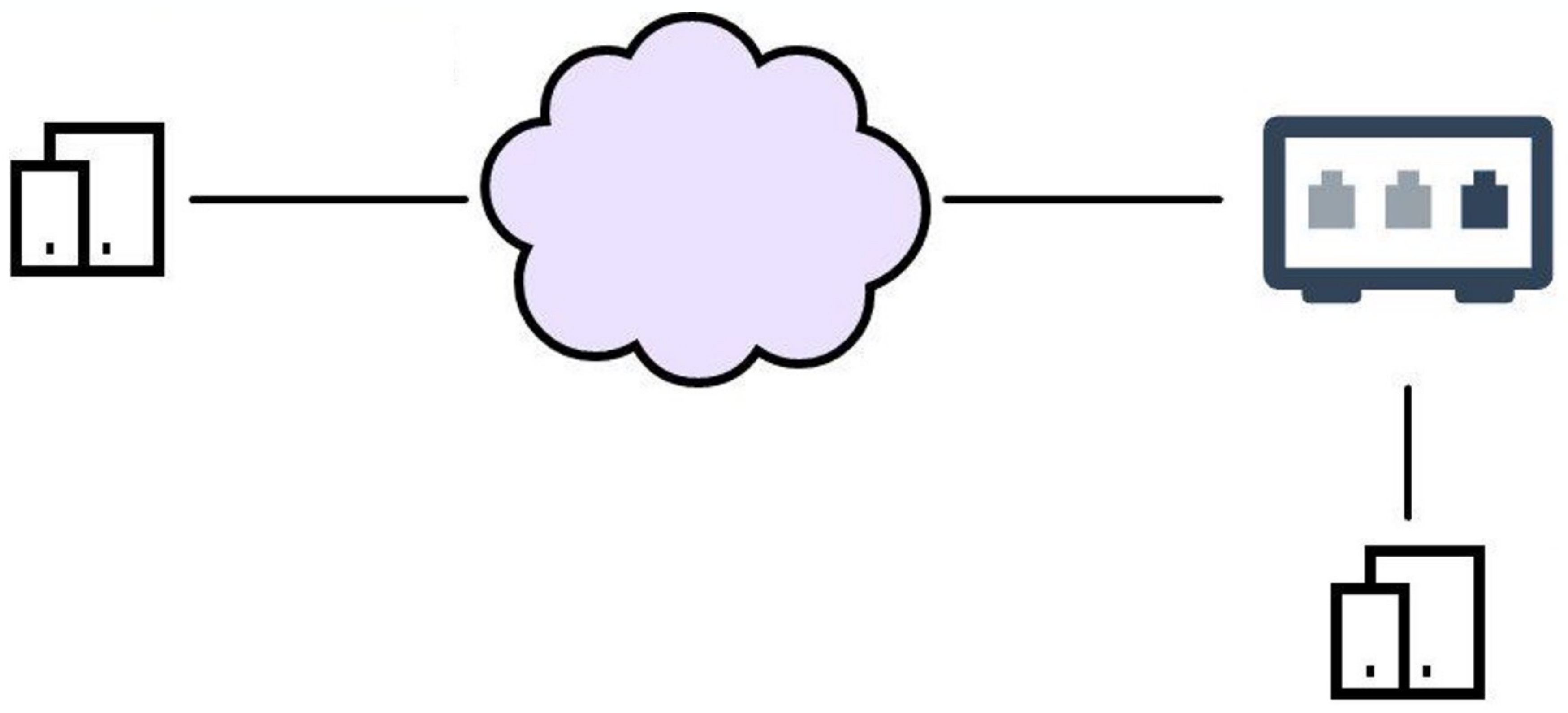
- **Test Environment Setup**

  - Two compute nodes inside different Data Center

  - Average network latencies between the two nodes are about 10ms

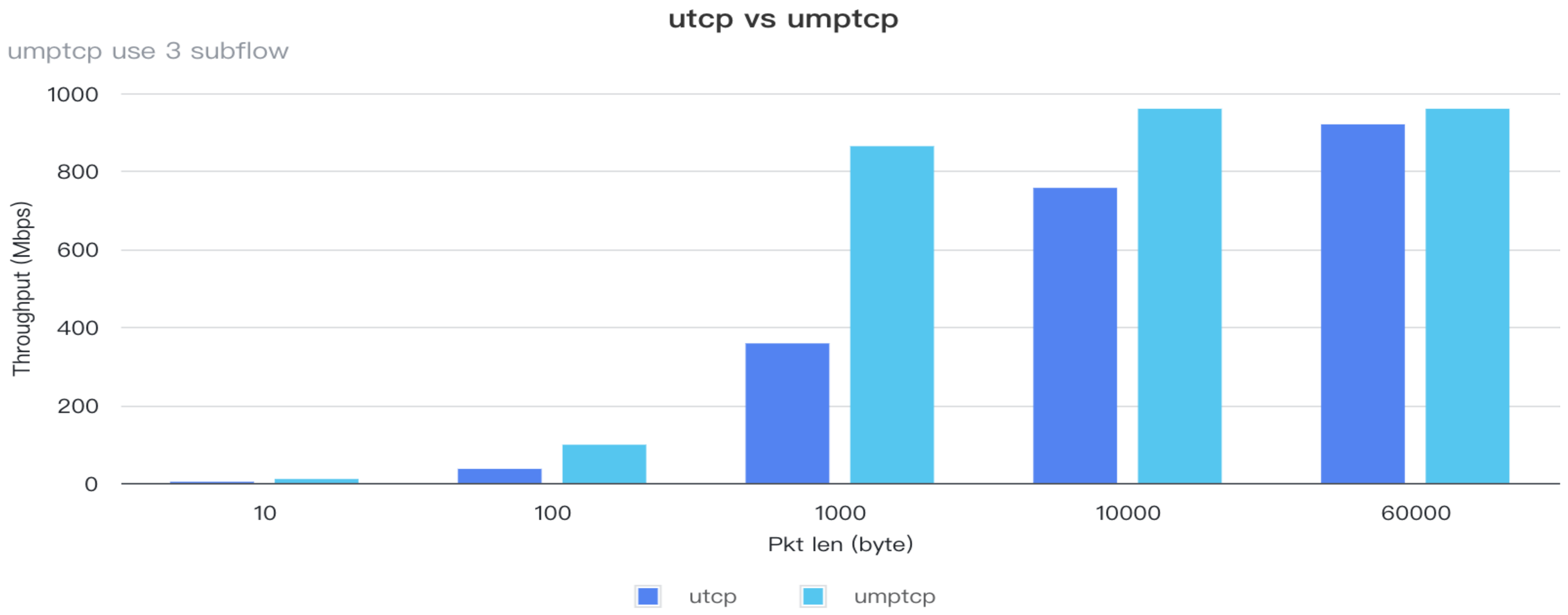
  - MPTCP connections creates 3+ sub-flows

  - The following charts are drawn based on the average fitting of 10 times test data



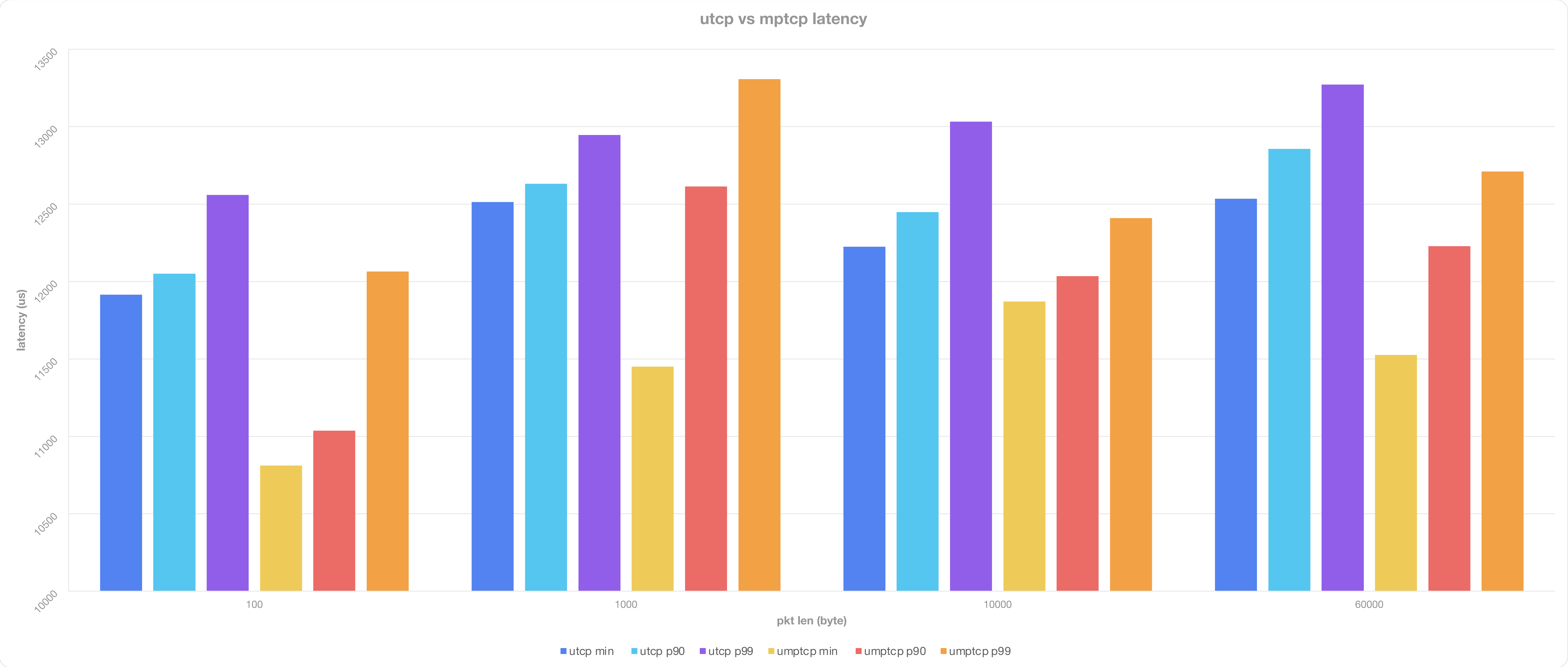
Performance Data

- **User-space TCP vs User-space MPTCP（Forwarding Throughput）**



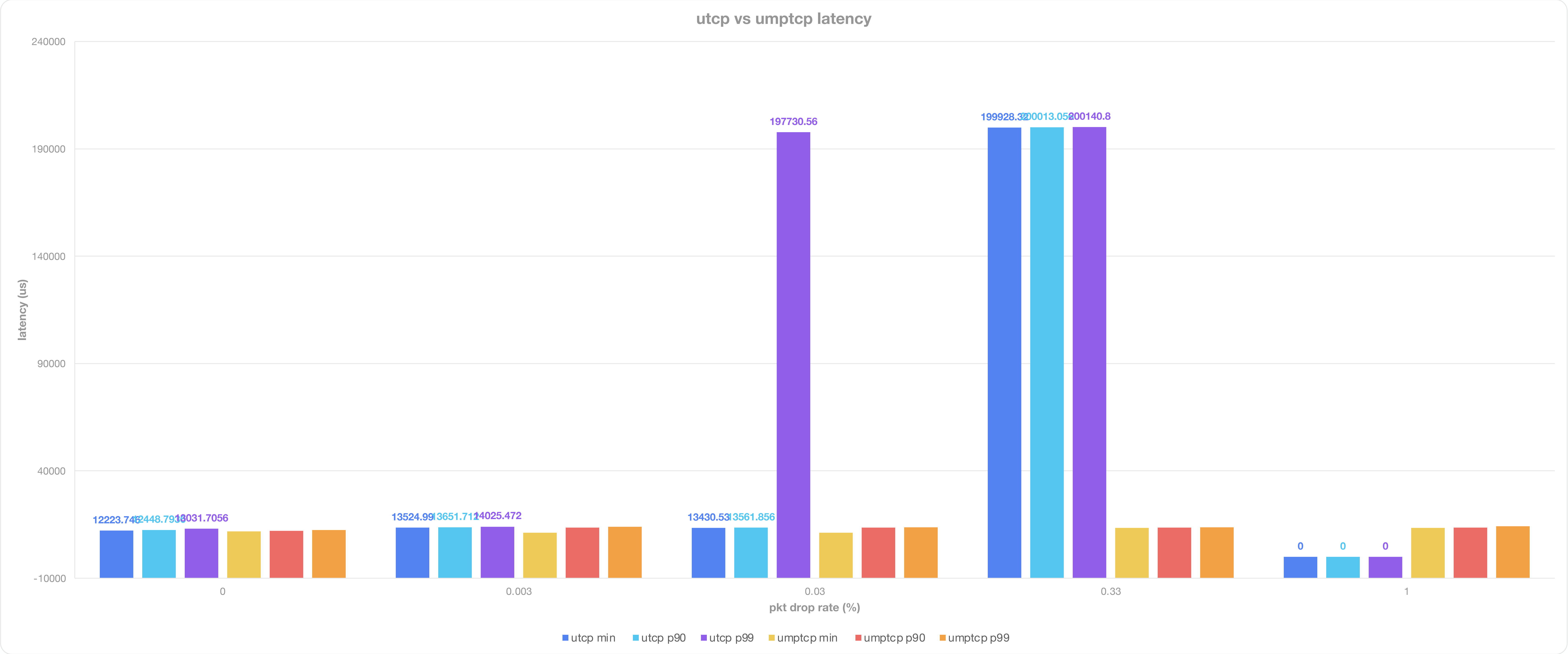
Performance Data

- **User-space TCP vs user-space MPTCP（Forwarding Latency）**



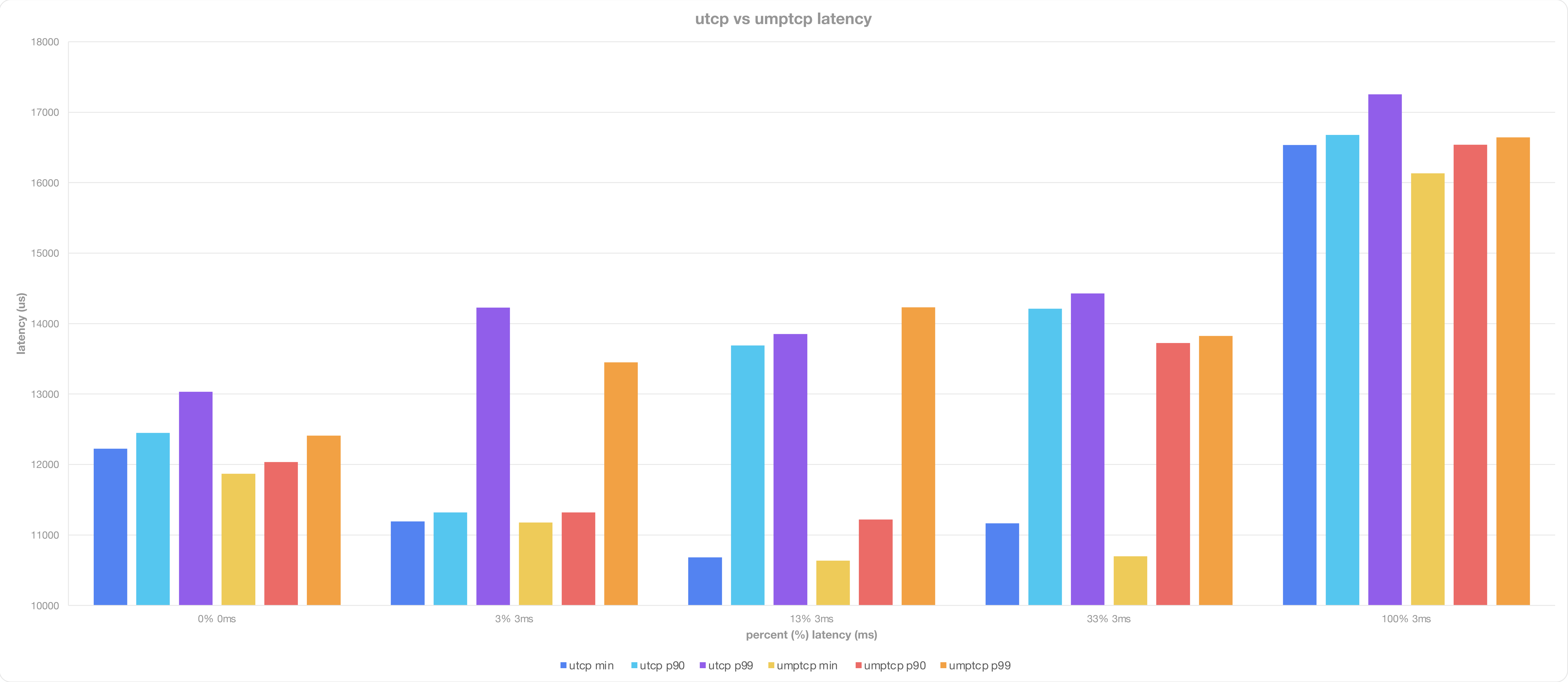
Performance Data

- **User-space TCP vs user-space MPTCP（Latency under Packet Drop）**



Performance Data

- **User-space TCP vs User-space MPTCP（Latency under Packet Delay）**



Future Work

- Performance tunning
- More packet scheduling polices
- Integrate with more user-space TCP stacks

THANKS

ByteDance字节跳动