

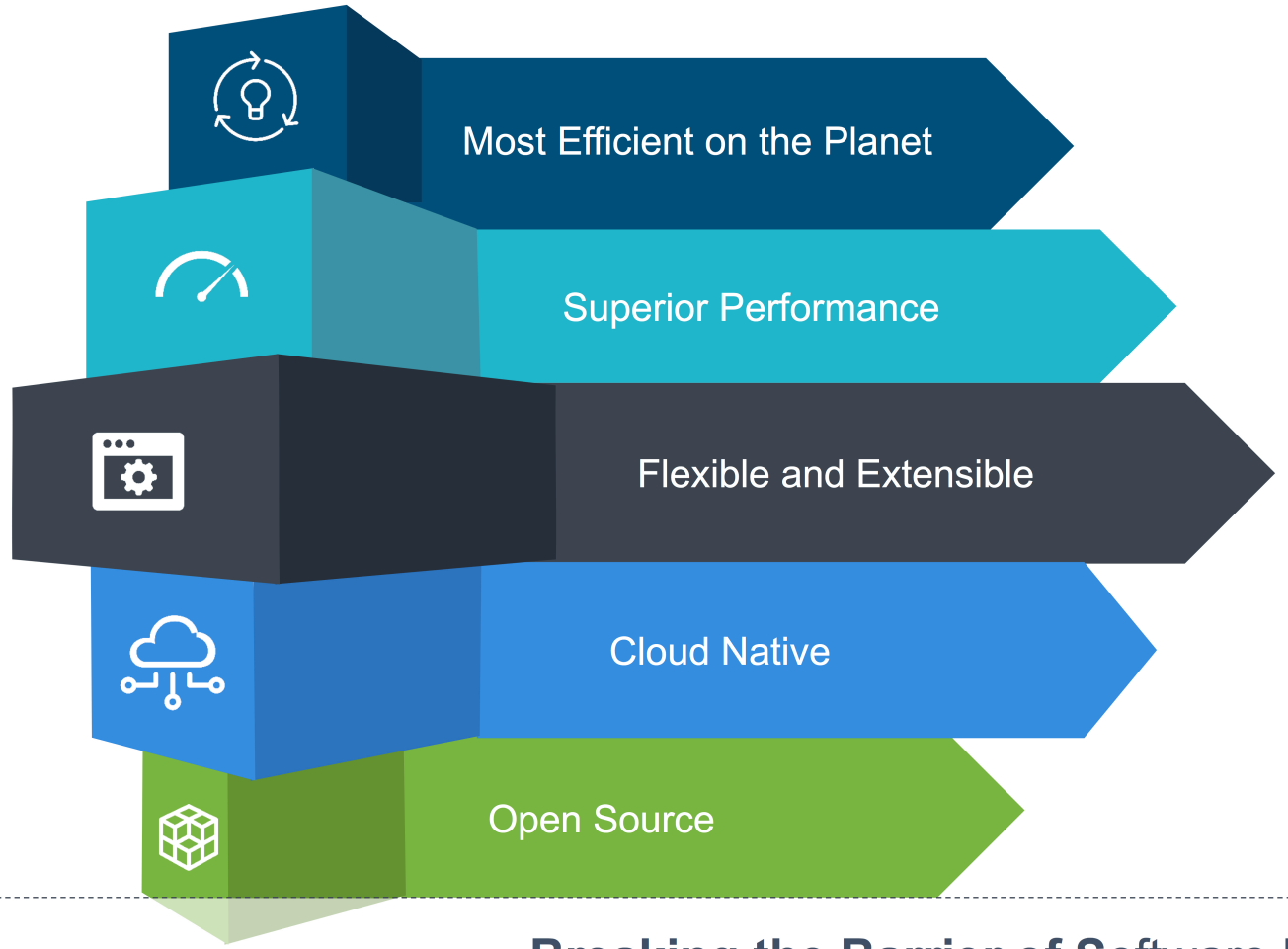
VPP Host Stack

TCP and Session Layers

Florin Coras, Dave Barach,
Keith Burns, Dave Wallace

VPP - A Universal Terabit Network Platform

For Native Cloud Network Services



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

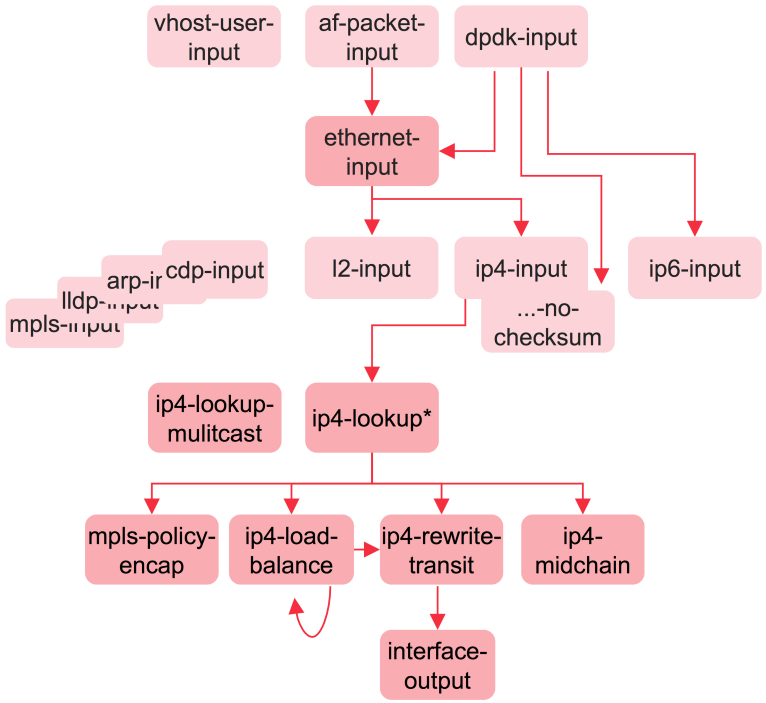
LINUX FOUNDATION
Open source collaborative project in Linux Foundation

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

VPP – How does it work?

Compute Optimized SW Network Platform

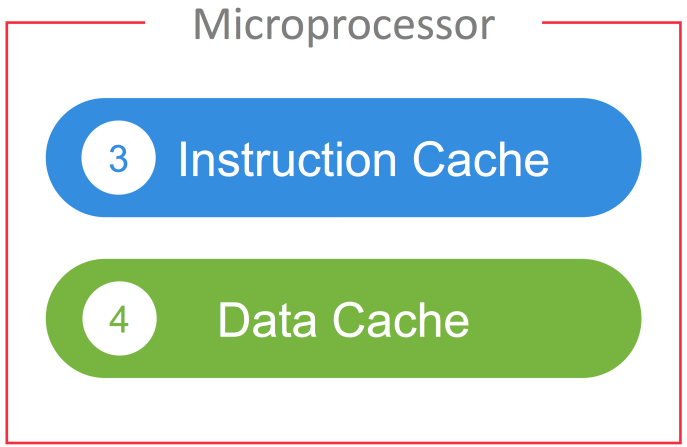
1 Packet processing is decomposed into a directed graph of nodes …



2 … packets move through graph nodes in vector …

Packet 0
Packet 1
Packet 2
Packet 3
Packet 4
Packet 5
Packet 6
Packet 7
Packet 8
Packet 9
Packet 10

3 … graph nodes are optimized to fit inside the instruction cache …

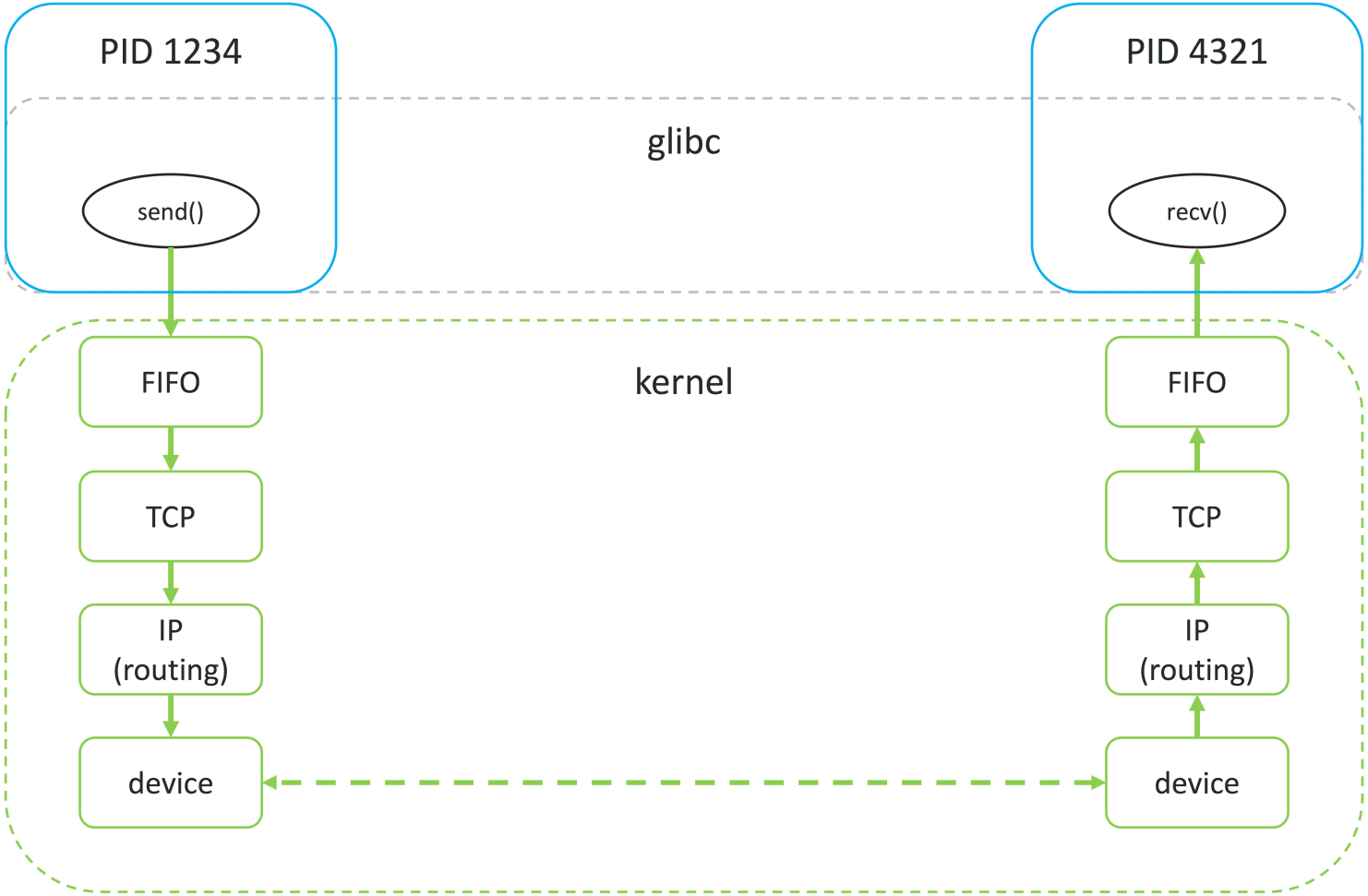


4 … packets are pre-fetched into the data cache.

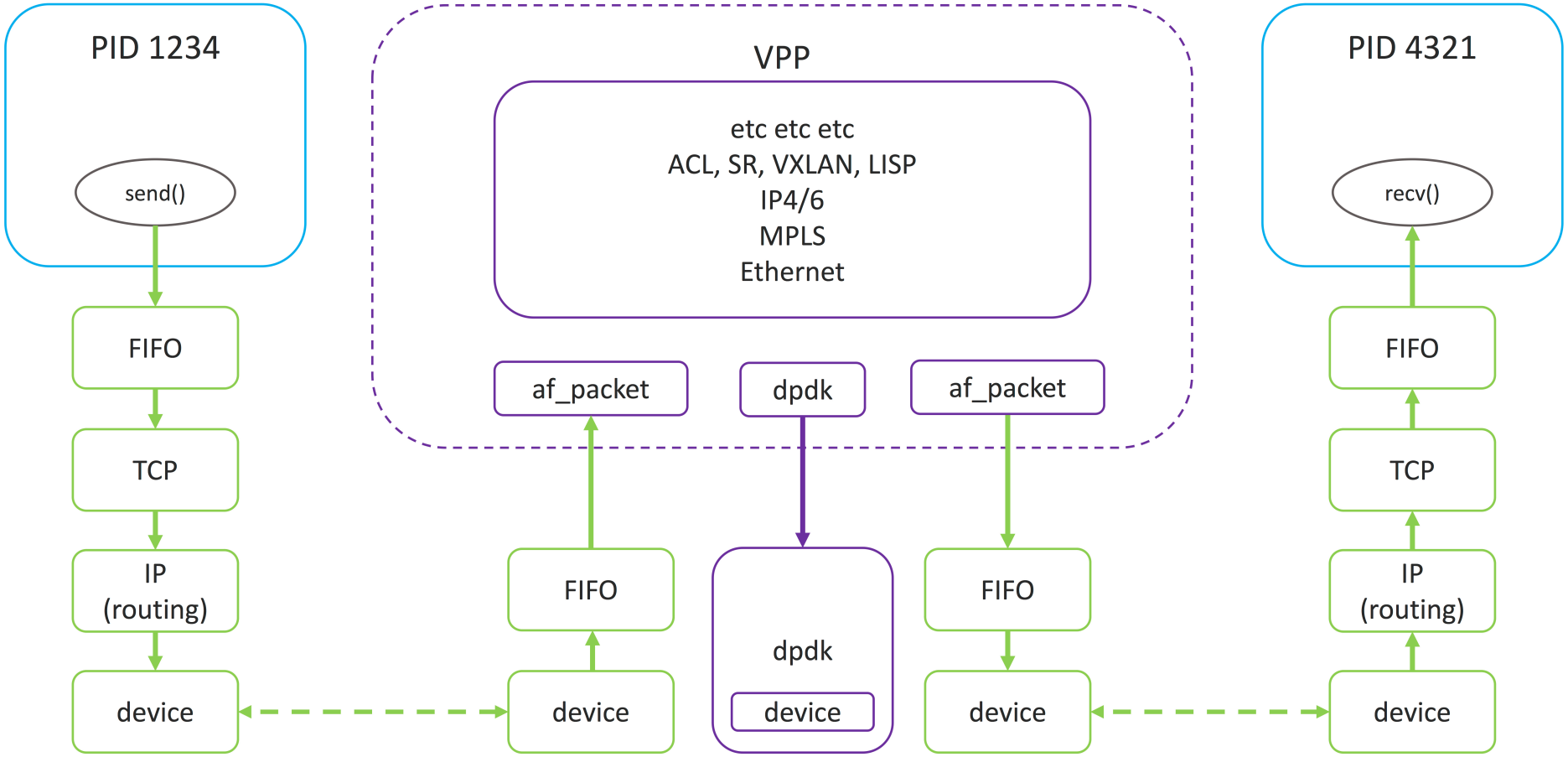
* Each graph node implements a "micro-NF", a "micro-NetworkFunction" processing packets.

Makes use of modern Intel® Xeon® Processor micro-architectures.
Instruction cache & data cache always hot ➔ Minimized memory latency and usage.

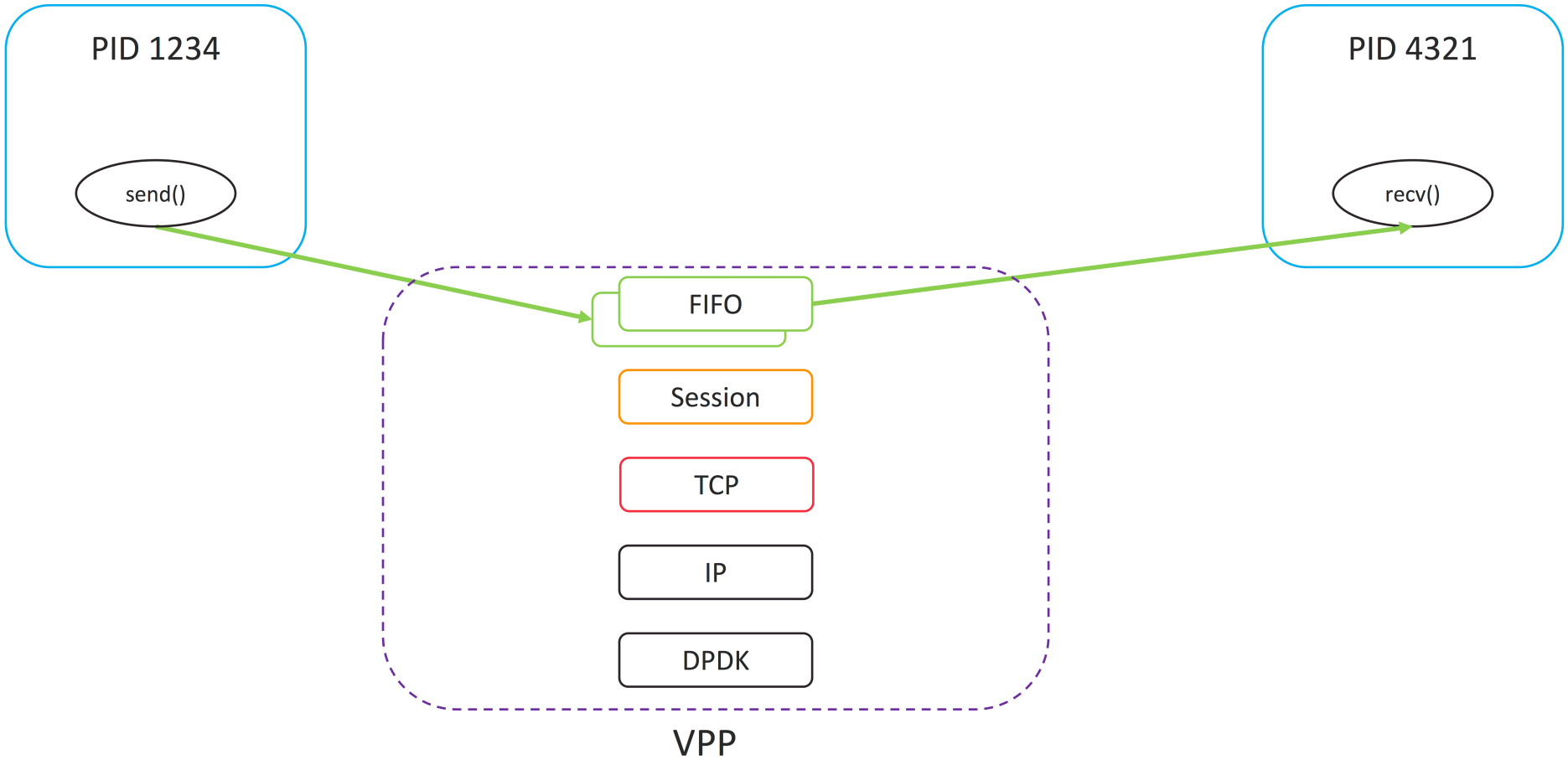
Motivation: Container networking



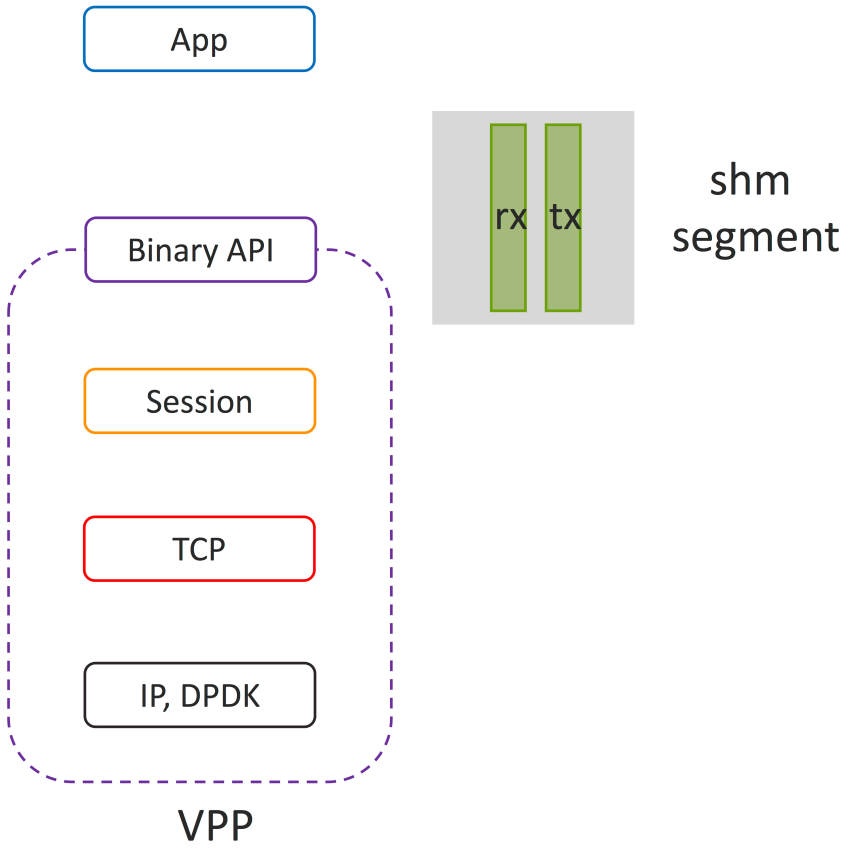
Motivation: Container networking



Why not this?

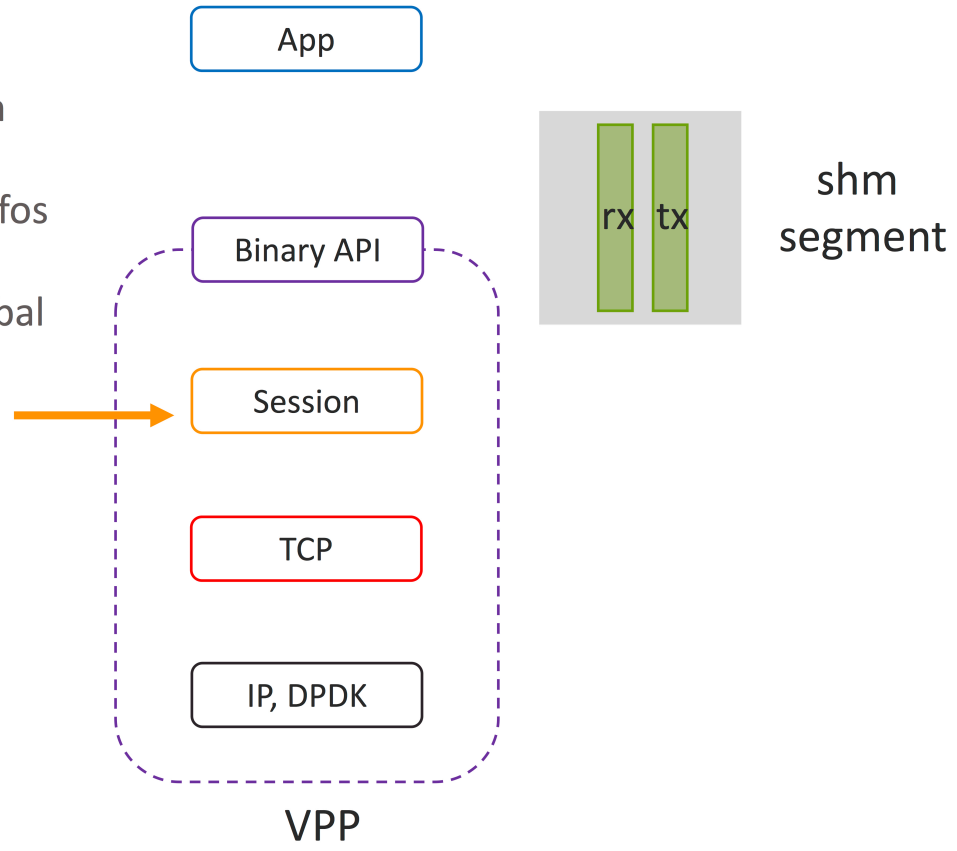


VPP Host Stack

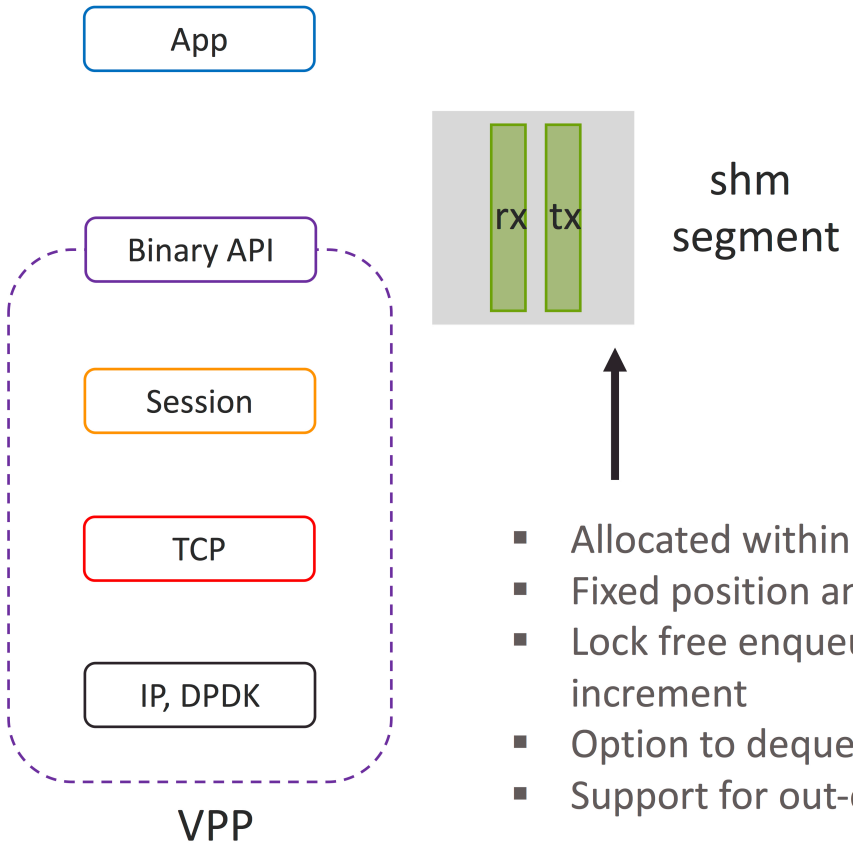


VPP Host Stack: Session Layer

- Maintains per app state and conveys to/from session events
- Allocates and manages sessions/segments/fifos
- Isolates network resources via namespacing
- Session lookup tables (5-tuple) and local/global session rule tables (filters)
- Support for pluggable transport protocols
- Binary/native C API for external/builtin applications



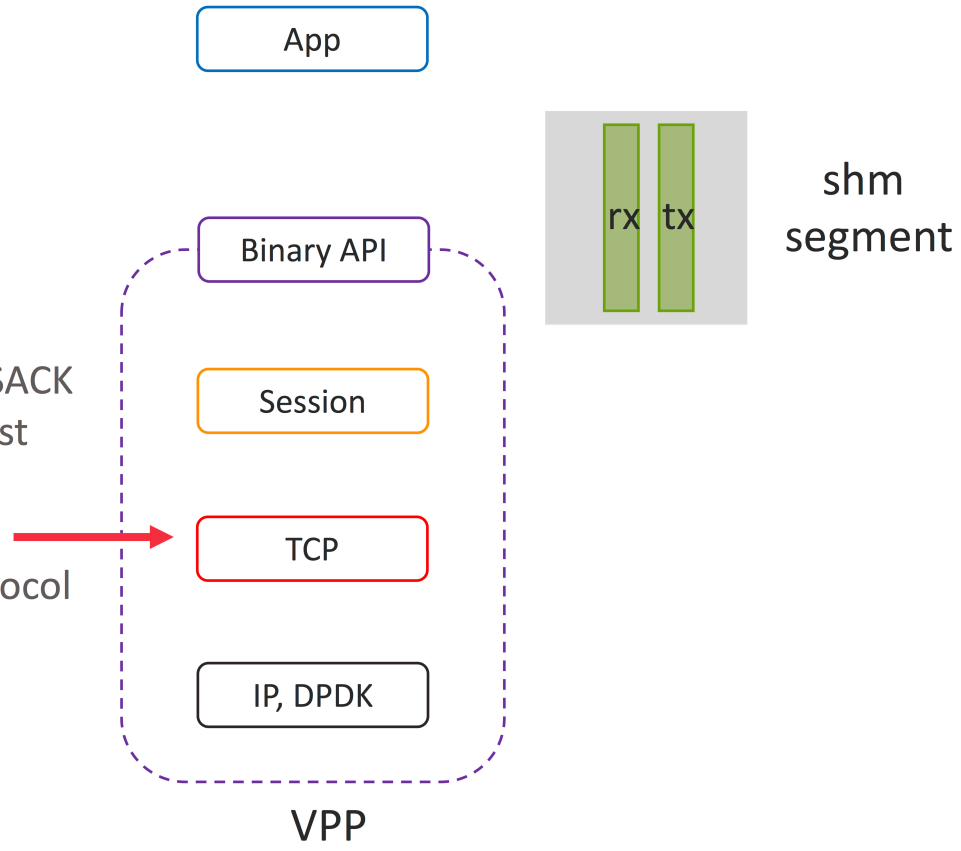
VPP Host Stack: SVM FIFOs



- Allocated within shared memory segments
- Fixed position and size
- Lock free enqueue/dequeue but atomic size increment
- Option to dequeue/peek data
- Support for out-of-order data enqueues

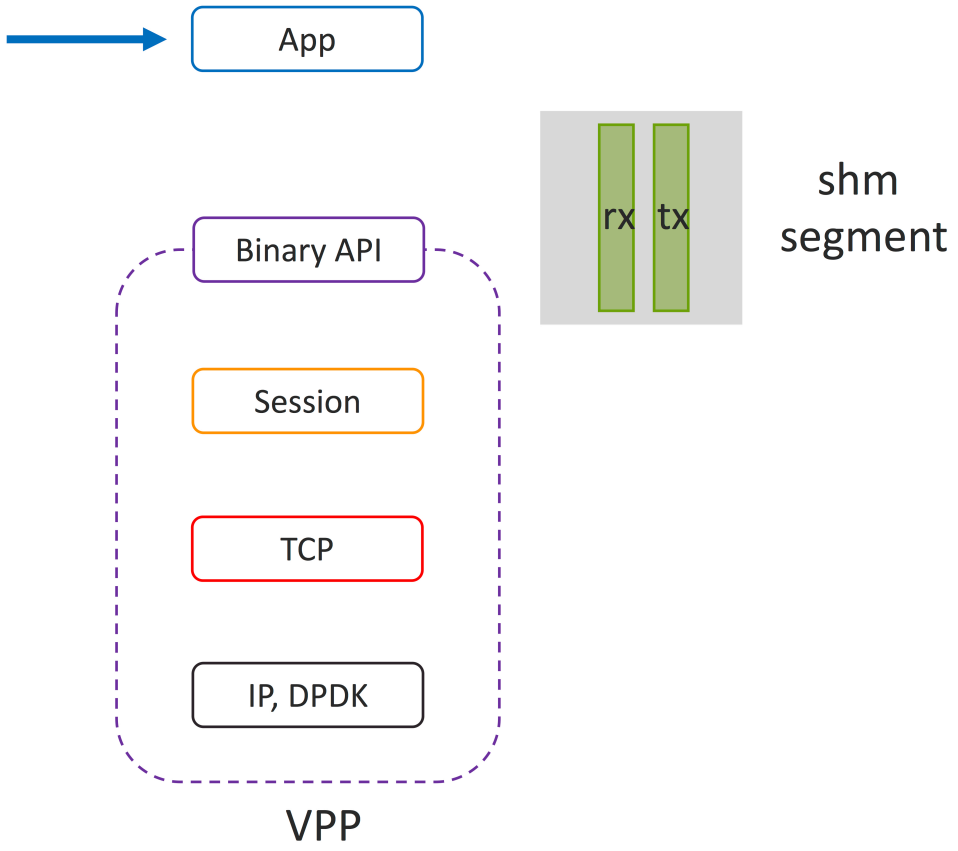
VPP Host Stack: TCP

- Clean-slate implementation
- “Complete” state machine implementation
- Connection management and flow control (window management)
- Timers and retransmission, fast retransmit, SACK
- NewReno congestion control, SACK based fast recovery
- Checksum offloading
- Linux compatibility tested with IWL TCP protocol tester



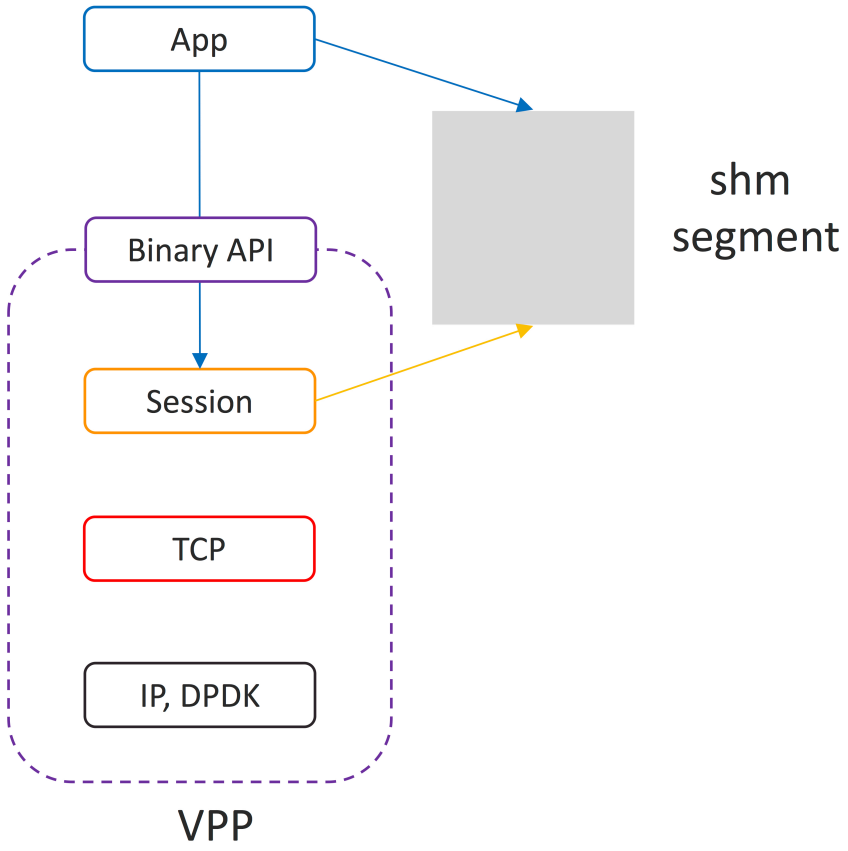
VPP Host Stack: Comms Library (VCL)

- Comms library (VCL) apps can link against
- LD_PRELOAD library for legacy apps
- epoll

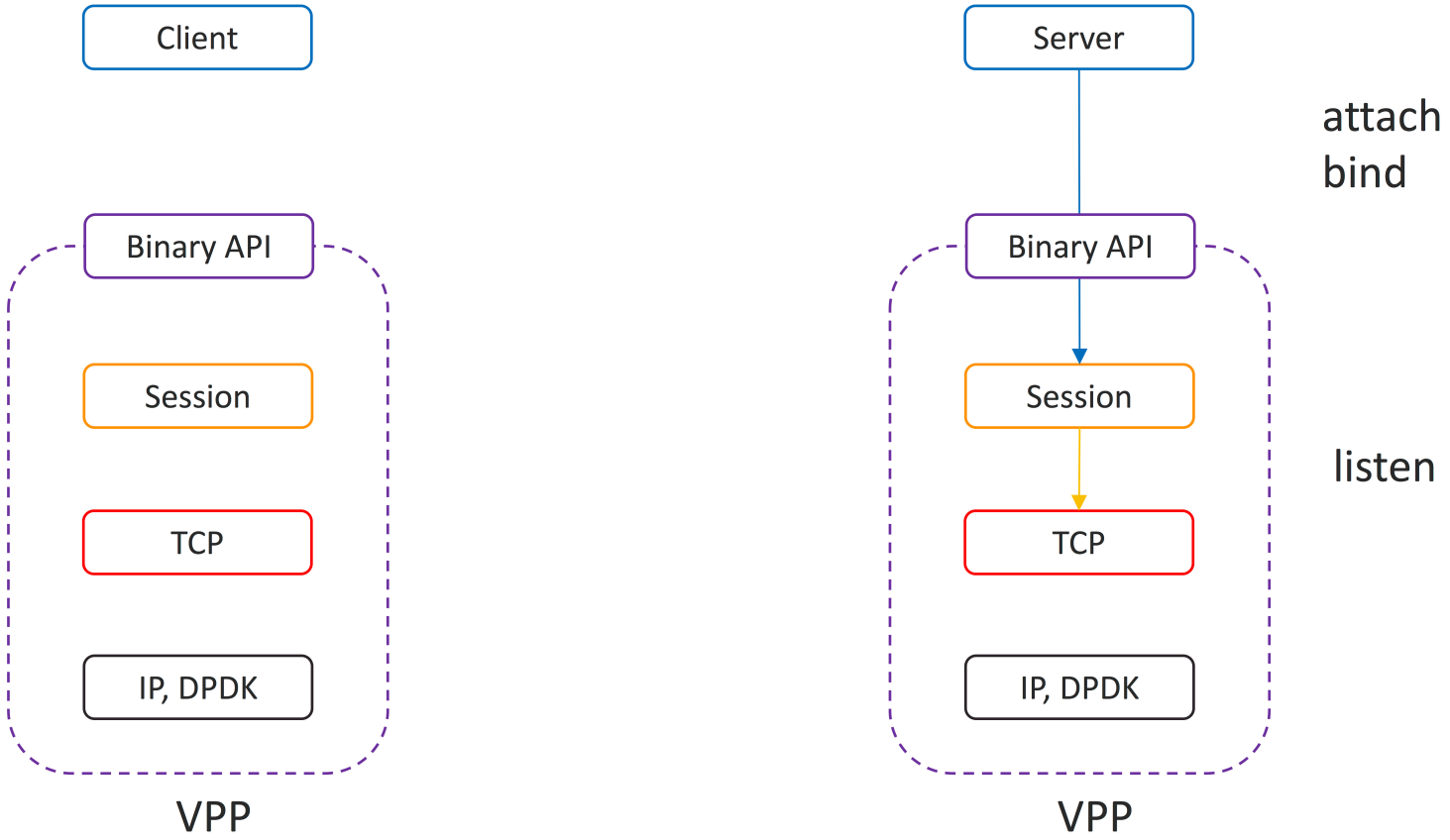


Application Attachment

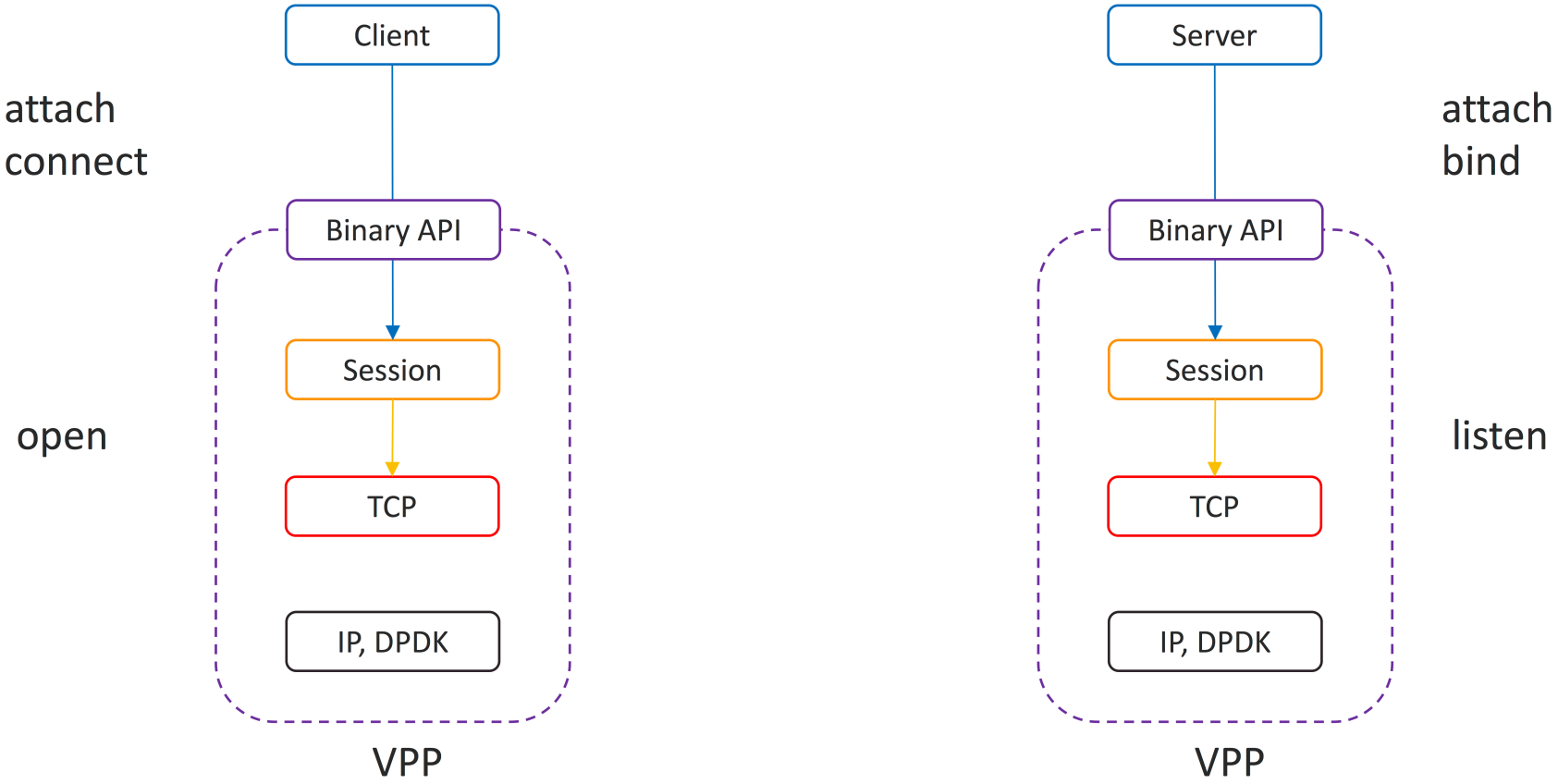
attach
bind (server)
connect (client)



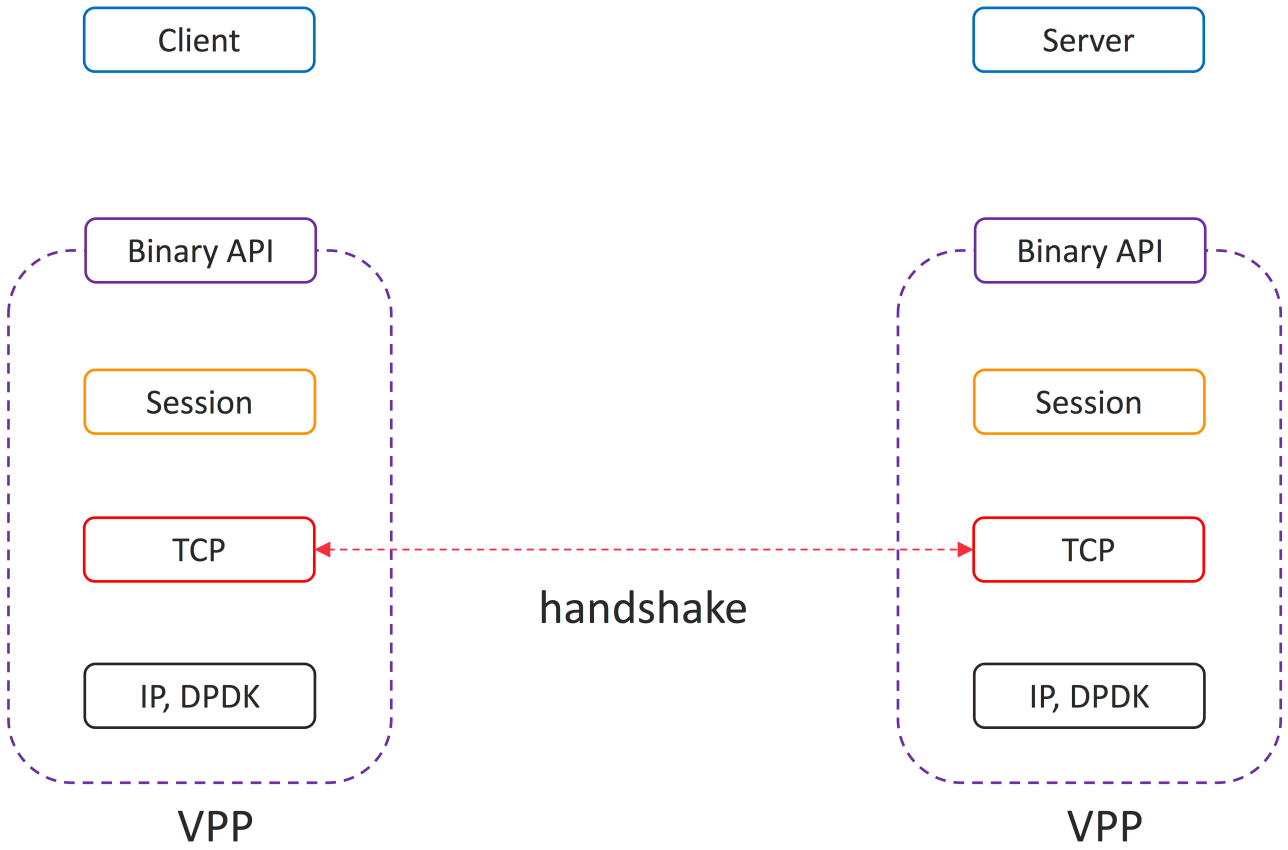
Session Establishment



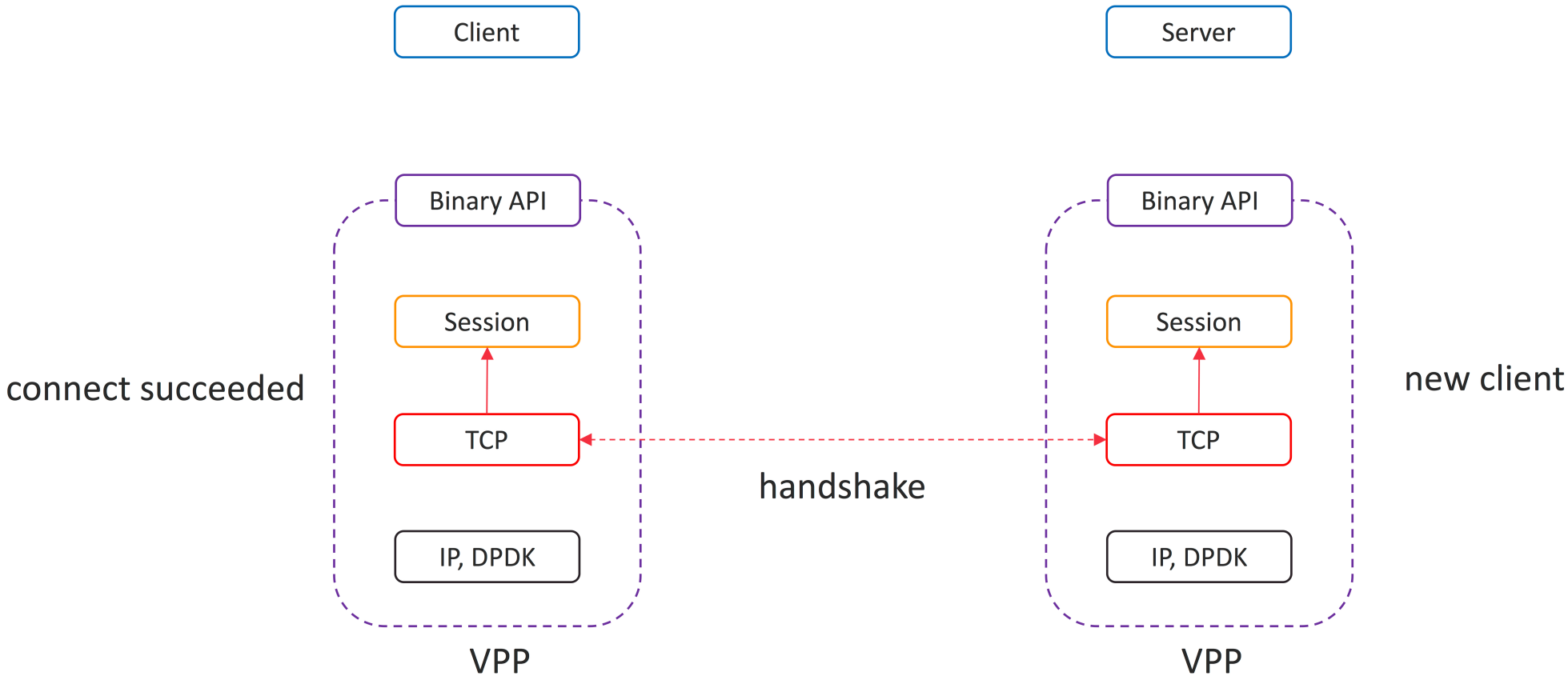
Session Establishment



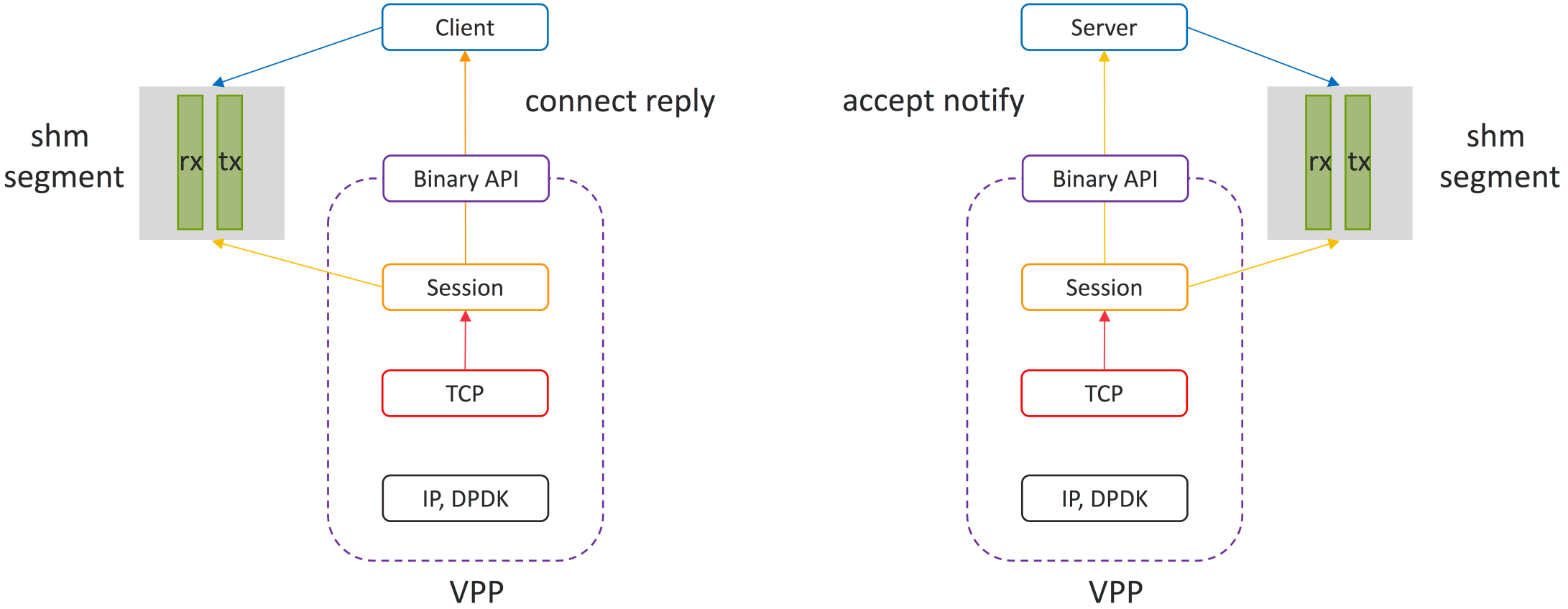
Session Establishment



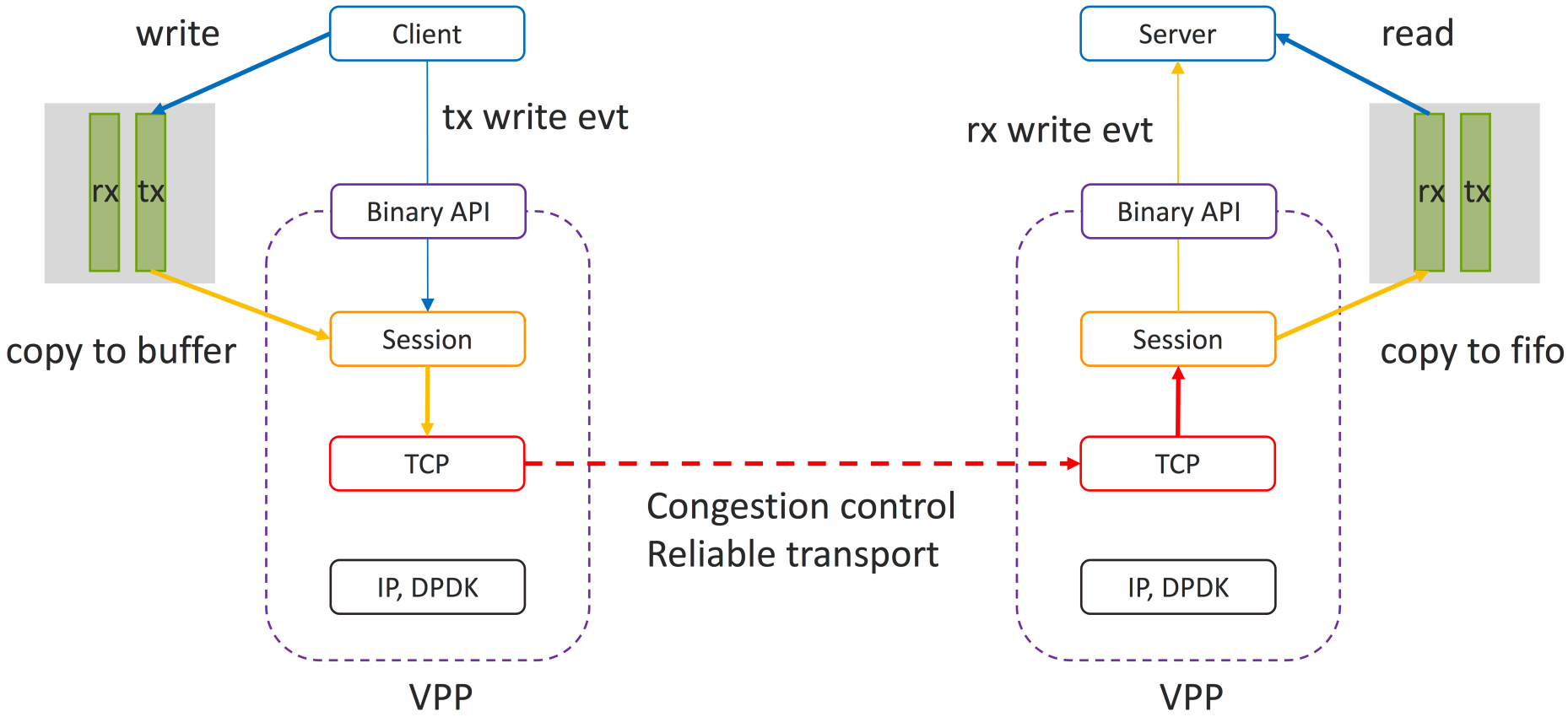
Session Establishment



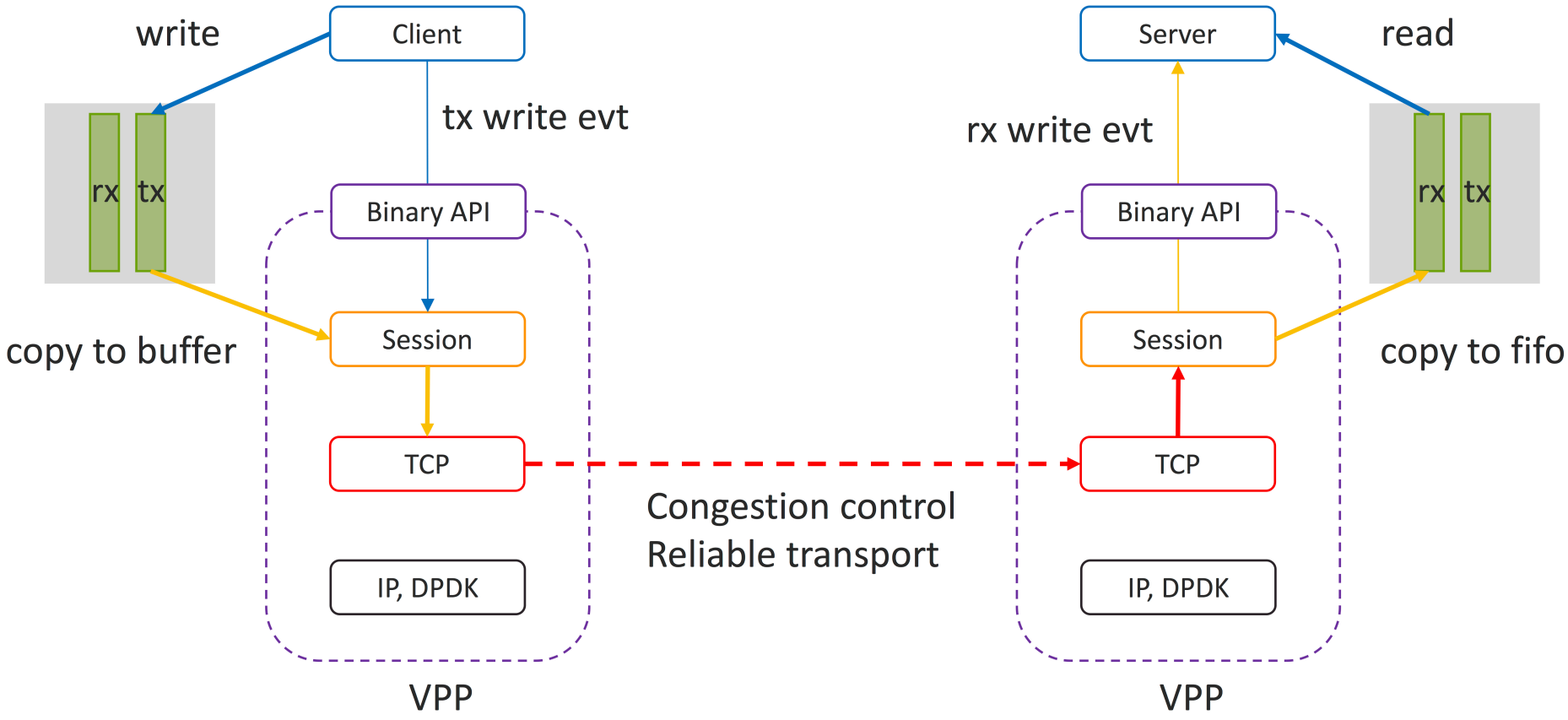
Session Establishment



Data Transfer

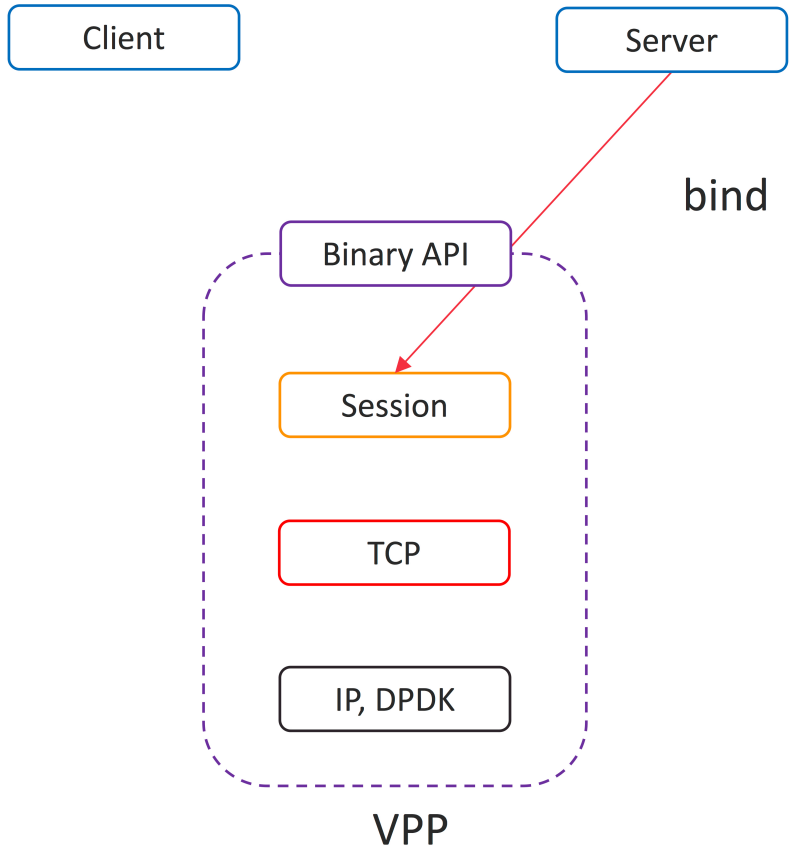


Data Transfer

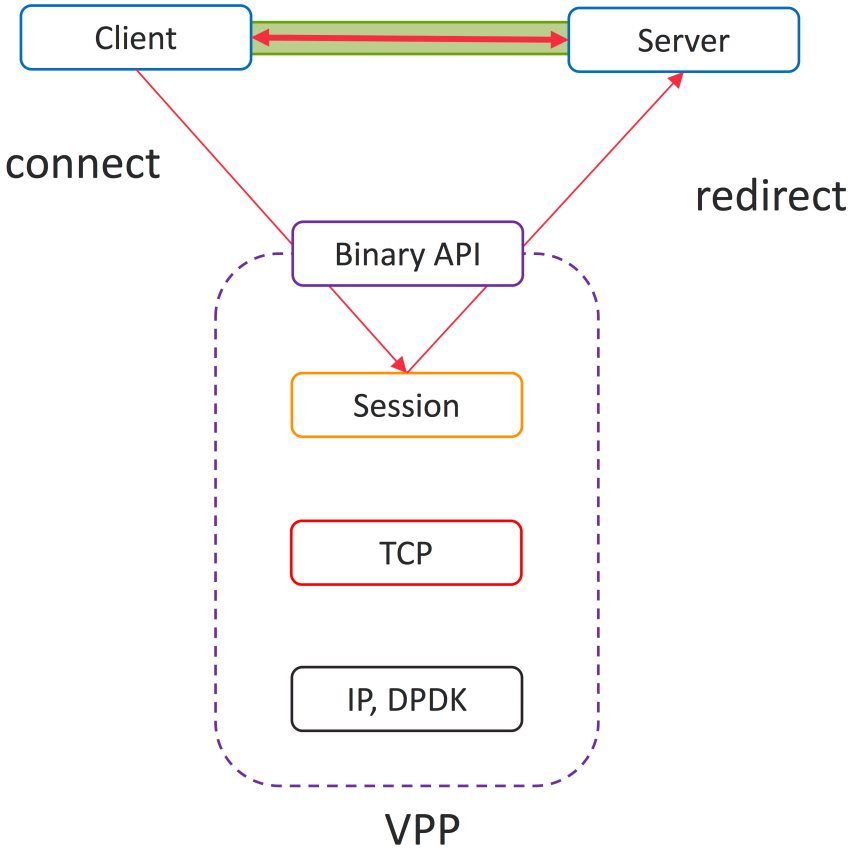


Not yet part of CSIT but some rough numbers on a E2690: 200k CPS and 8Gbps/core!

Redirected Connections (Cut-through)

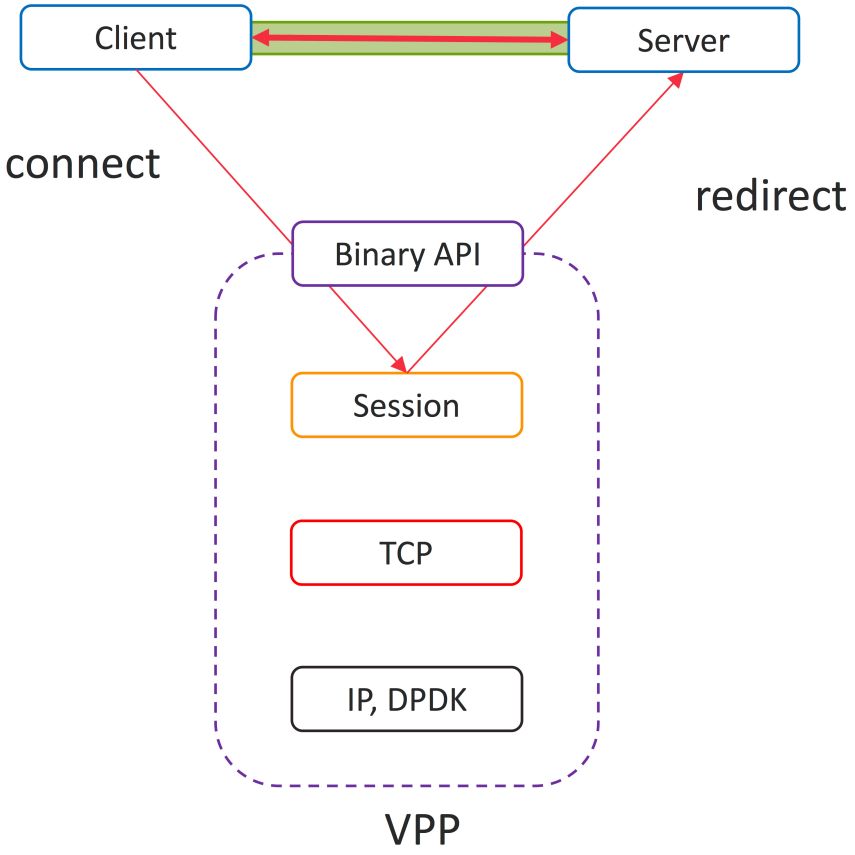


Redirected Connections (Cut-through)



Redirected Connections (Cut-through)

Throughput is memory bandwidth constrained: ~120Gbps!



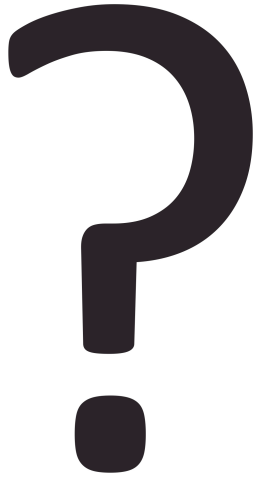
Ongoing work

- Overall integration with k8s
  - Istio/Envoy
- TCP
  - Rx policer/tx pacer
  - TSO
  - New congestion control algorithms
  - PMTU discovery
  - Optimization/hardening/testing
- VCL/LD_PRELOAD
  - Iperf, nginx, wget, curl

Next steps – Get involved

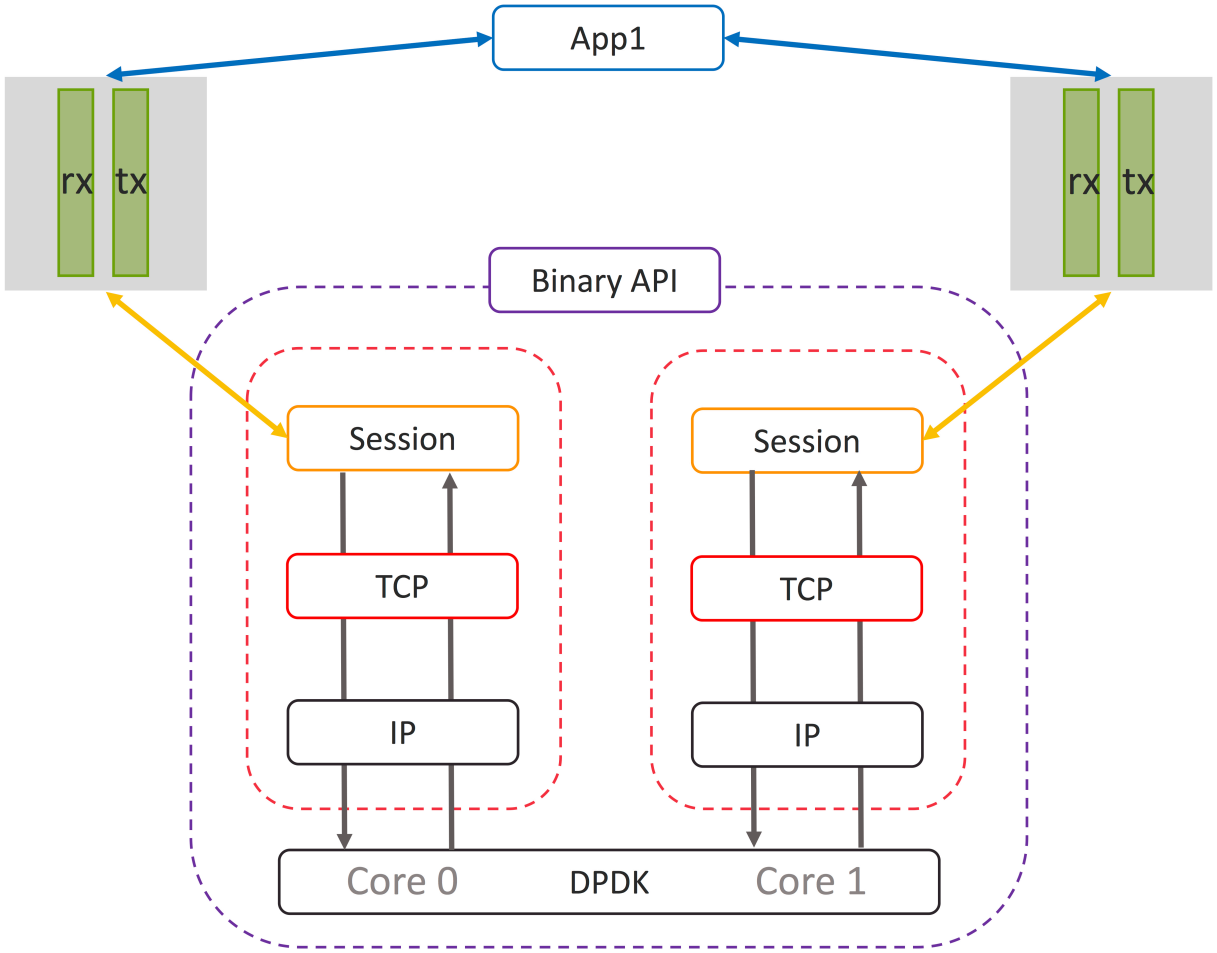
- Get the Code, Build the Code, Run the Code
  - Session layer: src/vnet/session
  - TCP: src/vnet/tcp
  - SVM: src/svm
  - VCL: src/vcl
- Read/Watch the Tutorials
- Read/Watch VPP Tutorials
- Join the Mailing Lists

Thank you!

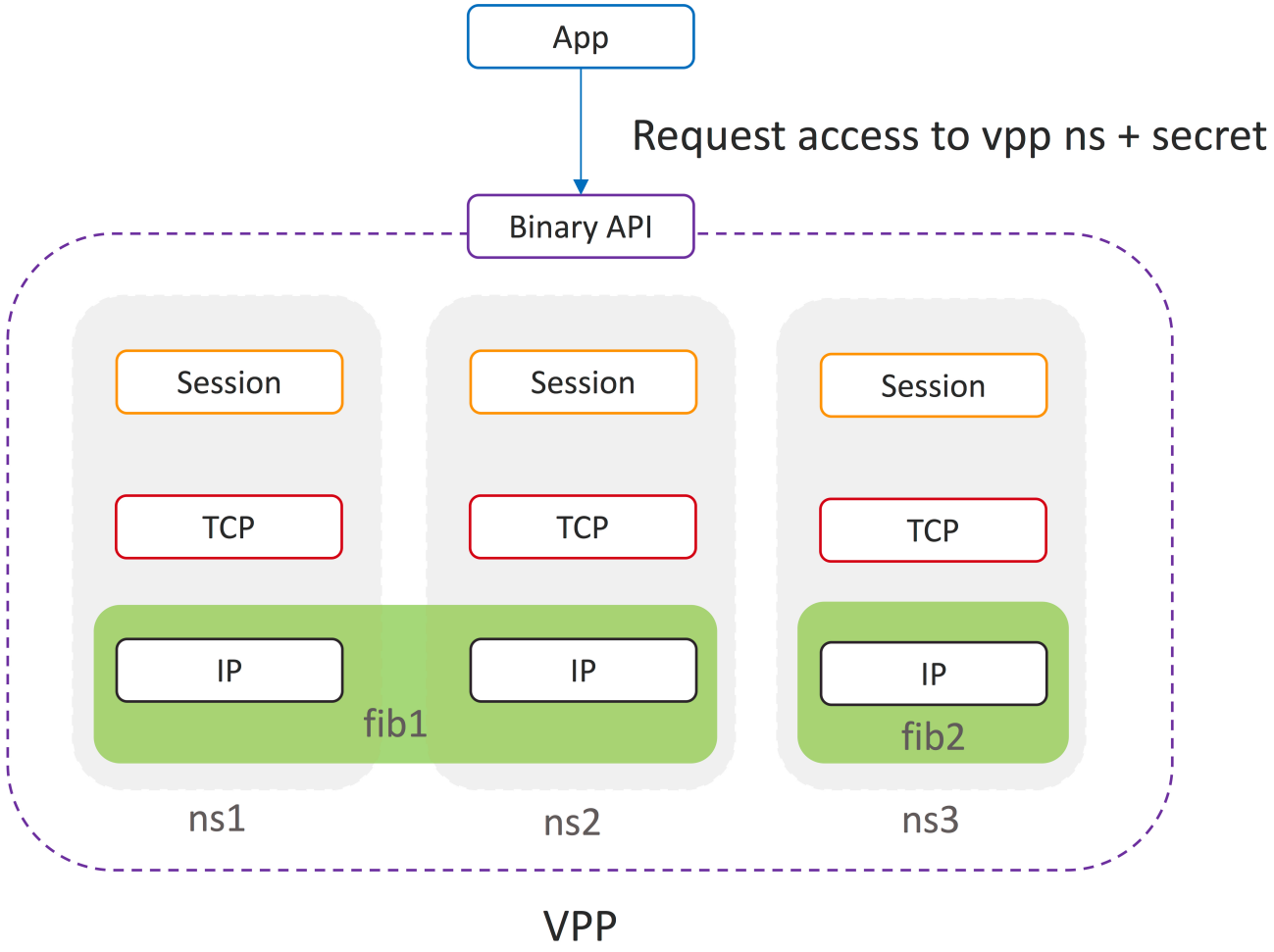


Florin Coras
email: fcoras@cisco.com
irc: florinc

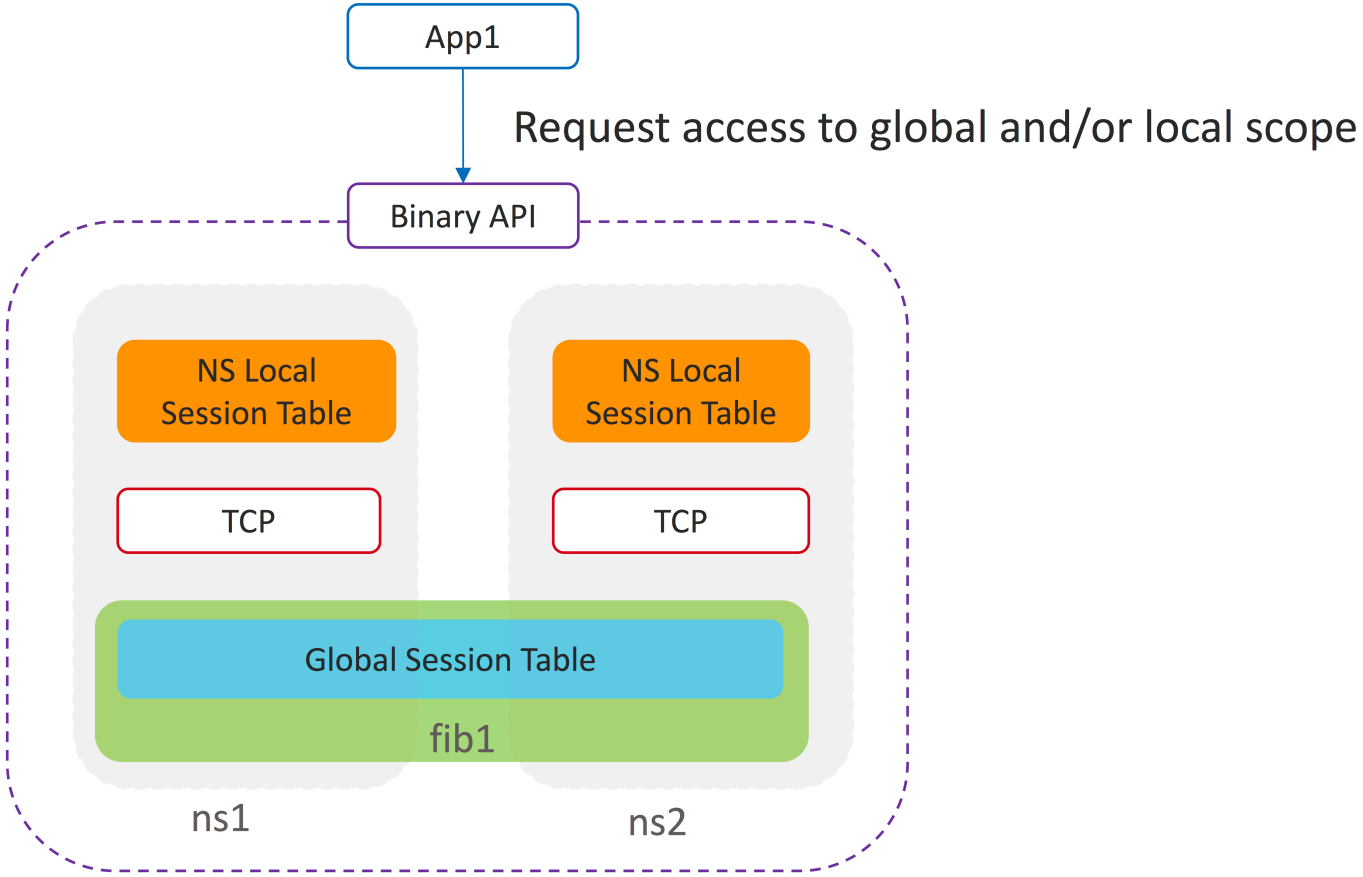
Multi-threading



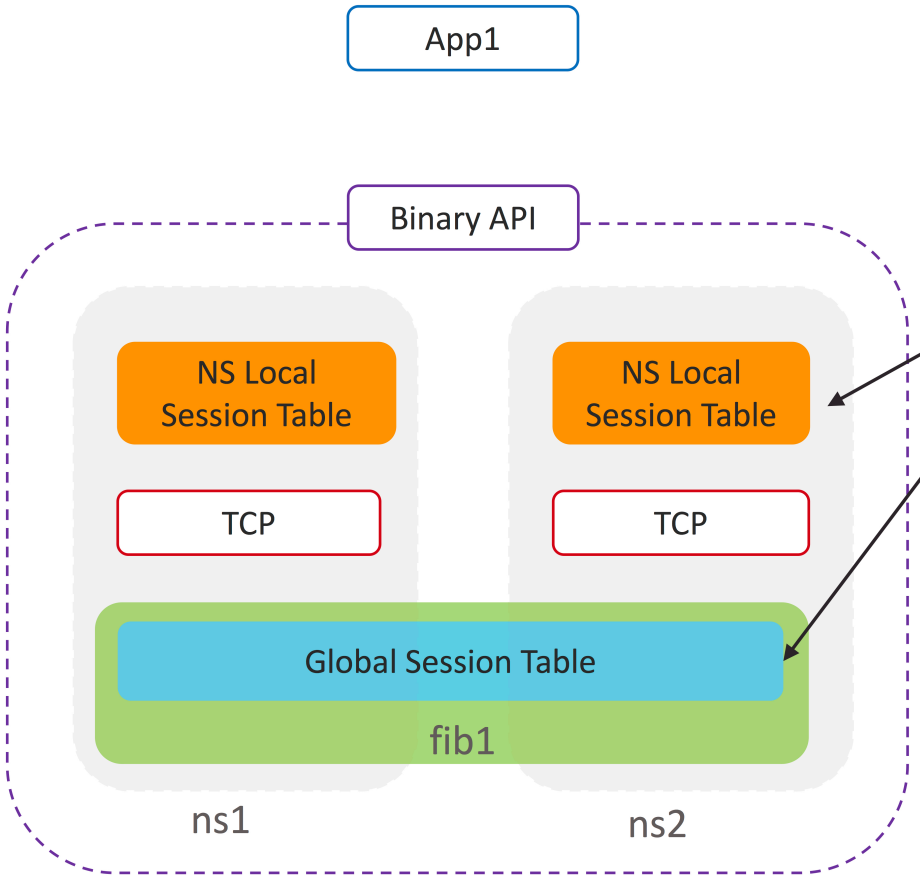
Features: Namespaces



Features: Session Tables



Features: Session Tables



- Both table have “rules table” that can be used for filtering
- Local tables are namespace specific and can be used for egress filtering
- Global tables are fib table specific and can be used for ingress filtering