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...

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

PID 1234
  send()
  FIFO
  TCP
  IP (routing)
  device

PID 4321
  recv()
  FIFO
  TCP
  IP (routing)
  device

glibc

kernel

DPDK Summit North America 2017
Motivation: Container networking

PID 1234
- send()
  - FIFO
  - TCP
  - IP (routing)
  - device

PID 4321
- recv()
  - FIFO
  - TCP
  - IP (routing)
  - device

VPP
- etc etc etc
- ACL, SR, VXLAN, LISP
- IP4/6
- MPLS
- Ethernet

af_packet → dpdk → af_packet

dpdk

device → device

DPDK Summit North America 2017
Why not this?

PID 1234

send()

FIFO

Session

TCP

IP

DPDK

VPP

PID 4321

recv()
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 namespace
- 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

Binary API
- Session
- TCP
- IP, DPDK

VPP

App

shm segment

rx tx

DPDK Summit North America 2017
Application Attachment

attach
bind (server)
connect (client)

VPP

Binary API

Session

TCP

IP, DPDK

shm segment

App

DPDK Summit North America 2017
Session Establishment

Client

Binary API

Session

TCP

IP, DPDK

VPP

Server

Binary API

Session

TCP

IP, DPDK

VPP

attach bind

listen

DPDK Summit North America 2017
Session Establishment

Client

Binary API

Session

TCP

IP, DPDK

VPP

Server

Binary API

Session

TCP

IP, DPDK

VPP

attach
connect

open

attach
bind

listen
Session Establishment

Client

Binary API
Session
TCP
IP, DPDK
VPP

handshake

TCP
IP, DPDK
VPP

Server

Binary API
Session
Session Establishment

Client

Server

Binary API

Session

TCP

IP, DPDK

VPP

Binary API

Session

TCP

IP, DPDK

VPP

connect succeeded

handshake

new client
Session Establishment

Client

Binary API

Session

TCP

IP, DPDK

VPP

connect reply

Server

Binary API

Session

TCP

IP, DPDK

VPP

accept notify

shm segment

shm segment
Data Transfer

Client

Binary API

Session

TCP

IP, DPDK

VPP

write

Copy to buffer

rx_tx

tx write evt

Server

Binary API

Session

TCP

IP, DPDK

VPP

read

Copy to fifo

rx_tx

rx write evt

Congestion control

Reliable transport

DPDK Summit North America 2017
Data Transfer

write

Client

tx write evt

Binary API

Session

TCP

VPP

IP, DPDK

copy to buffer

rx tx

Server

rx write evt

Binary API

Session

TCP

VPP

IP, DPDK

copy to fifo

read

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

DPDK Summit North America 2017
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**

DPDK Summit North America 2017
Thank you!

Florin Coras
email: fcoras@cisco.com
irc: florinc
Multi-threading
Features: Namespaces

Request access to vpp ns + secret

VPP

ns1

ns2

ns3

fib1

fib2

App

Binary API
Features: Session Tables

Binary API

Request access to global and/or local scope

App1

NS Local Session Table

NS Local Session Table

TCP

TCP

Global Session Table

fib1

ns1

ns2

DPDK Summit North America 2017
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