intel
experience what's inside™

FOSDEM '17
Brussels 4 & 5 February 2017
www.fosdem.org »

# TLDK OVERVIEW

## TRANSPORT LAYER DEVELOPMENT KIT

Ray Kinsella
February 2017
Email : ray.kinsella [at] intel.com
IRC: mortderire

Contributions from Keith Wiles & Konstantin Ananyev

# Legal Disclaimer

**General Disclaimer:**

© Copyright 2017 Intel Corporation. All rights reserved. Intel, the Intel logo, Intel Inside, the Intel Inside logo, Intel. Experience What's Inside are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

**Technology Disclaimer:**

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

**Performance Disclaimers:**

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings.  Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

# Why TCP Performance matters?

## In the Cellular Network[1].

- TCP is 95.7% of downlink traffic, UDP is 4.1%.
  - HTTP/HTTPS account for 77.6% of all downstream bytes.
- Most TCP flows are small. On average 60.6% of the flows are smaller than 4 KB and only 9.5% of flows are larger than 32 KB.

## In the Wired Network [2].

- HTTP traffic dominates making up 60% of all traffic by bytes.
- Non-P2P lines pre-dominantly use HTTP, for which it contributes 72% of their traffic volume.

**TCP connection establishment rate is as important as throughput.**

# Evaluating TCP Stacks

| | |
|---|---|
| Performance | Connections setup/teardown rate<br>Throughput<br>RR Latency |
| Optimization | Core locality<br>Eliminating Mode Switching<br>Lock reduction<br>... |
| RFC Compliance | RFC 793 (TCP v4)<br>RFC 1948 (Defending Against Sequence Number Attacks)<br>RFC 2018 (TCP Selective Acknowledgment Options)<br>RFC 5681 (TCP Congestion Control)<br>... |
| API Compatibility | BSD Socks API |
| Maintenance | Total cost of ownership |

# What is TLDK (Transport Layer Development Kit)?

- TLDK is a high performance L4 protocol library with termination support for applications built using DPDK.
  - Support for UDP &TCP in client & server modes.
  - Support for common TCP options; MSS, timestamp, wnd scaling, s/ack.
  - Support for common TCP features; ddos protection, delayed ack, congestion control.
  - Support for common TCP HW offloads; TSO
  - Code examples demonstrating a number of use cases.
- TLDK implements core DPDK design concepts, such as
  - Bulk packet processing
  - Non-blocking API
  - No-mode switching
  - Cache optimization
  - Memory locality etc.

# What is TLDK (deeper dive)

- TDLK provides a socket-like API with familiar semantics (where possible).
- TLDK is application driven, protocols are driven by the application needing the data. Instead of data rx/tx events driving the application.

| Core TLDK Concepts | |
|---|---|
| Context (**tle_ctx**) | represents an 'independent copy of the stack‘, each owns set of <**tle_*_stream**> and <**dev**>. |
| Device (**tle_dev**) | abstracts the underlying device that RX/TX packets, understands the available transport protocols and hardware offloads. |
| Stream (**tle_*_stream**) | • represents an L4(UDP/TCP, etc.) endpoint address, and is an analogy to socket entity.<br>• belongs to particular <**tle_ctx**> but is visible globally across all threads (is multi-process safe). |

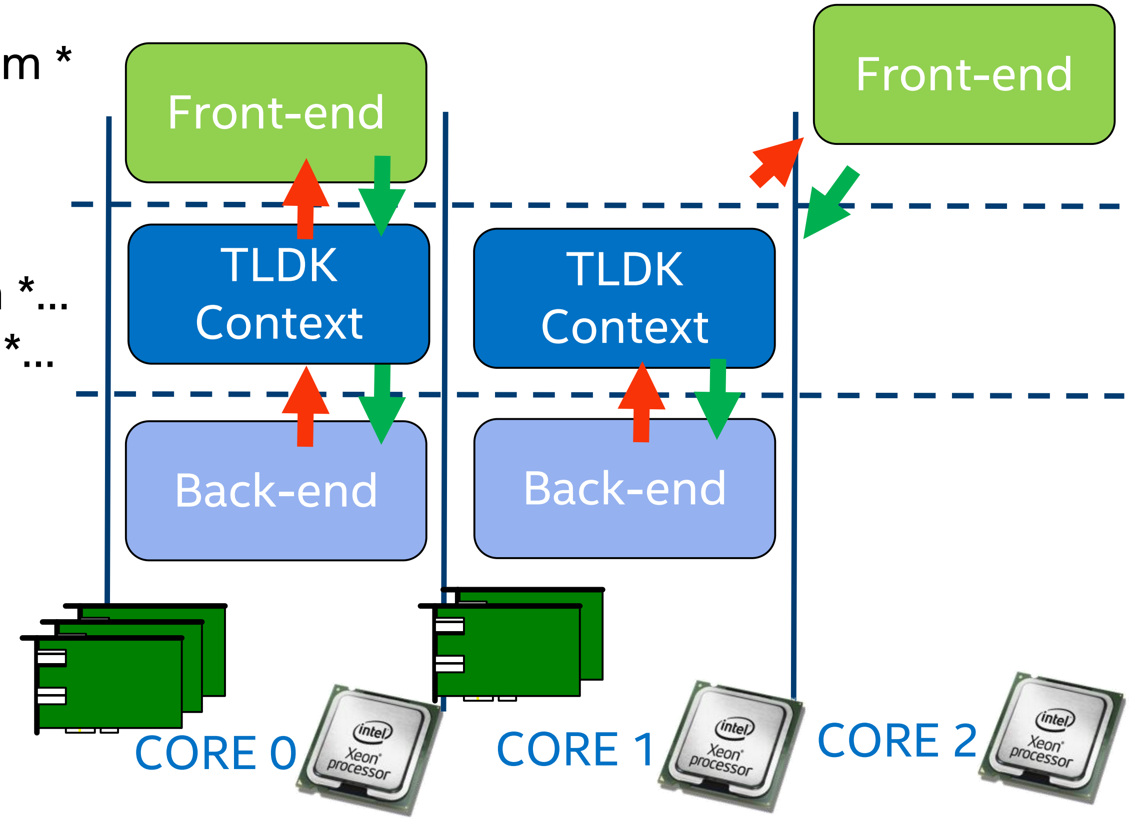
# How to write a TLDK TCP Application?

## TLDK Frontend

- tle_tcp_stream_accept(tle_stream *
- tle_tcp_reject(tle_stream *
- tle_tcp_stream_close_bulk(…
  tle_stream *ts[])
- tle_tcp_stream_send(tle_stream *…
- tle_tcp_stream_recv(tle_stream *…

## TLDK Backend

- tle_ctx_create
- tle_add_dev(tle_context* …
- tle_tcp_tx_bulk(tle_ctx *ctx
  ⇒ *rte_eth_rx_burst*
- tle_tcp_rx_bulk(tle_ctx *ctx
  ⇒rte_eth_tx_burst



# TLDK Performance (non-optimized code)

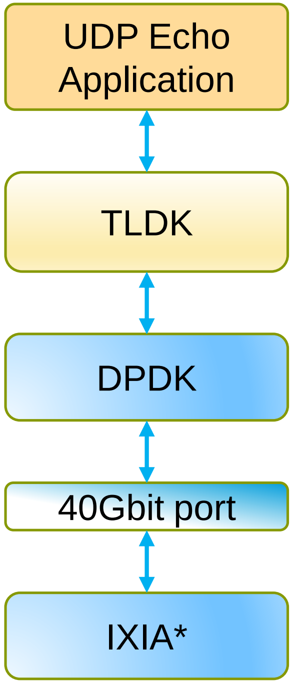
CPU: Intel® Xeon® Processor E5-2699 v3 @ 2.30GHz
64G Ram, Dual socket system, 2x400GB SSD, 2x1TB drives

NIC: Intel® Ethernet Controller XL710 for 40GbE QSFP+
Firmware: 5.04 0x80002505 0.0.0

DPDK: 16.07

Linux: Ubuntu* 15.10 (GNU/Linux 4.2.0-16-generic x86_64)

TLDK: Current release (2016-09-15)

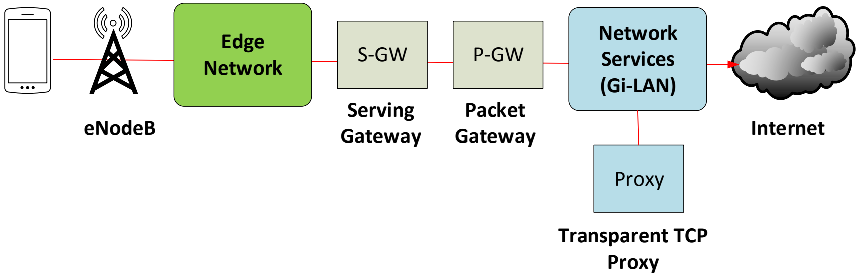



UDP Packet size used is 64 bytes, 5 cores we max out the PCI

| #Physical Cores | #Queues | Frame Rate Mpps |
|---|---|---|
| 1 | 1 | 7.4 |
| 2 | 2 | 14.8 |
| 3 | 3 | 22.2 |
| 4 | 4 | 29.5 |
| 5 | 5 | 36.4 (max for PCI) |

# Some possible use-cases for TLDK

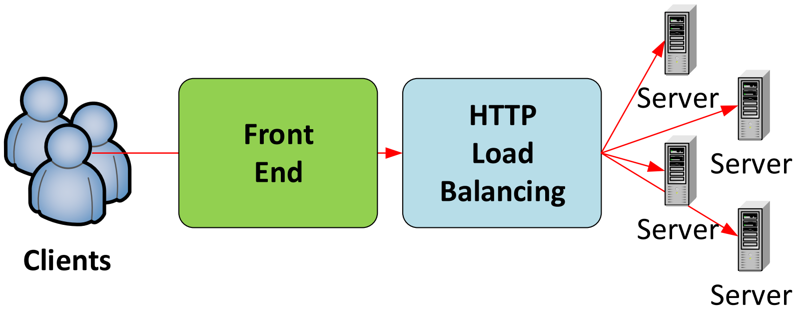
## TCP Transparent Proxy




- Common middle box in the cellular network, deployed in the Gi-LAN
- Observes & passes-through the 3-way hand-shake and setups a shadow flow.
- Sends ACK's on behalf of Client, caches TCP packets close to the network.
- Benefits are latency splitting, reduction in retransmission and buffer-bloat.

## Reverse Proxy & Load-balancer

- Common middle box in the data-centre, deployed between the Front-end (IDS/FW) and webservers.
- Typically supports features; server fault detection, session persistence.
- Typically supports distribution algorithms; round-robin, ip-hash and least-connected.
- Benefits are optimizing resource utilization, maximizing throughput, reducing latency, and ensuring fault-tolerant configurations.



# Summary

- Network operators, data centres are optimizing TCP workloads to reduce impedance mismatch, improve overall utilization, UX …
- Userspace TCP/UCP stack's are growing in popularity for reasons of performance and ease of upgrade.
- TLDK is a grounds up based on core DPDK design concepts designed to achieve ***Network Node*** performance requirements.

ML: **tldk-dev@lists.fd.io**

E-mail: ray.kinsella [at] intel.com

Wiki: **wiki.fd.io/view/TLDK**

Call to action – ***we need feedback and contributions from fellow travellers to design, contribute code, optimizations, sample use cases etc!***

experience
what's inside™

# References


1. Comparison of Caching Strategies in Modern Cellular Backhaul Networks. S Woo et al., 2013
2. On Dominant Characteristics of Residential Broadband Internet Traffic, Maier et al., 2009

# Why TCP Performance matters?

## In the Cellular Network†

- TCP is 95.7% of downlink traffic, UDP is 4.1%.
  - HTTP/HTTPS account for 77.6% of all downstream bytes.
- Most TCP flows are small. On average 60.6% of the flows are smaller than 4 KB and only 9.5% of flows are larger than 32 KB.
- Median throughput is 21.1 Kpbs, the 90th %-tile is only 182.4 Kpbs. Most flows are short lived and never leave the TCP slow-start phase)

† Source: Comparison of Caching Strategies in Modern Cellular Backhaul Networks. S Woo et al., 2013

**TCP connection establishment rate is as important as throughput.**

# Why TCP Performance matters?

## In the Wired Network †

- HTTP traffic dominates making up 60% of all traffic by bytes.
- Roughly 3% of DSL-lines use P2P protocols and their traffic accounts for 30% of all volume.
- Non-P2P lines pre-dominantly use HTTP, for which it contributes 72% of their traffic volume.

† Source: On Dominant Characteristics of Residential Broadband Internet Traffic, Maier et al., 2009

TCP performance is an important part of UX

# Userspace TCP Implementations†

| | |
|---|---|
| NetBSD Based | Rump Kernel - github.com/rumpkernel |
| FreeBSD Based | Libplebnet - github.com/opendp<br>Libuinet - github.com/pkelsey/libuinet<br>DPDK_ANS - github.com/opendp/dpdk-ans |
| Linux Based | Linux Kernel Library - github.com/lkl/linux |
| Non-OS origin | VPP TCP - git.fd.io/vpp<br>OpenFastPath - www.openfastpath.org (is this based on FreedBSD).<br>mTCP - shader.kaist.edu/mtcp<br>Seastar - www.seastar-project.org |

† a non-comprehensive list.

# TLDK Design

| | |
|---|---|
| Performance | Connections setup/teardown rate<br>Throughput<br>RR Latency |
| Optimization | Core locality<br>Eliminating Mode Switching<br>Lock reduction<br>... |
| RFC Compliance | RFC 793 (TCP v4)<br>RFC 1948 (Defending Against Sequence Number Attacks)<br>RFC 2018 (TCP Selective Acknowledgment Options)<br>RFC 5681 (TCP Congestion Control)<br>... |

Prioritize TCP performance, target TCP aggregation points in the network.