TLDK

TRANSPORT LAYER DEVELOPMENT KIT

Keith Wiles PE @ intel.com
2016

Legal Disclaimer

General Disclaimer:

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.

TLDK project

- Transport Layer Development Kit – TLDK

- TLDK is a project housed under the FD.io Linux Foundation group
  - FD.io (pronounced Fido) contains many projects the primary is VPP
  - VPP is the Cisco contributed Vector Packet Processing code base for L2/L3 fast path used in Cisco routers today

- TLDK and other projects in FD.io are open source projects
  - Anyone can contribute to the project without having to pay any fees
  - Need a free Linux Foundation login ID to contribute code
  - The code is free to clone via Git or tarball from:
    - TLDK Wiki page https://wiki.fd.io/view/TLDK

TLDK (Transport Layer Development Kit)

- TLDK is to provide a clean set of ‘C’ libraries to enable network protocol handling at the application layer
- TLDK will provide IPv4/v6 and TCP/UDP protocols along with others as required for normal network operation
- Goal is to provide a very high performance network stack with termination support for applications using VPP and DPDK
- TLDK will provide a set of libraries to allow for applications to build a complete network stack support
  - Including a high performance non-socket type application interface
  - Including a socket layer for applications linked with the application
  - Including a LD_PRELOAD socket layer to run native Linux applications

TLDK (Transport Layer Development Kit)

- TLDK is not a normal network designed stack!
  - TLDK has turned the network stack upside down for better performance
- Network protocols are driven by the application needing the data

- Normal network stack designs drive packet into the protocols, then to the application
  - In TLDK the packets are per-filtered to a given DPDK core/thread first
  - The application then drives the packets into the stack when it needs the data not before
  - The design attempts to keep the CPU cache warm to reduce wasted cycles

- The goal is to move multiple packets thru the stack at a time, using the vector style packet processing
- Multiple packets at a time allows us to amortize packet processing overhead for higher throughput

TLDK Uses case with VPP

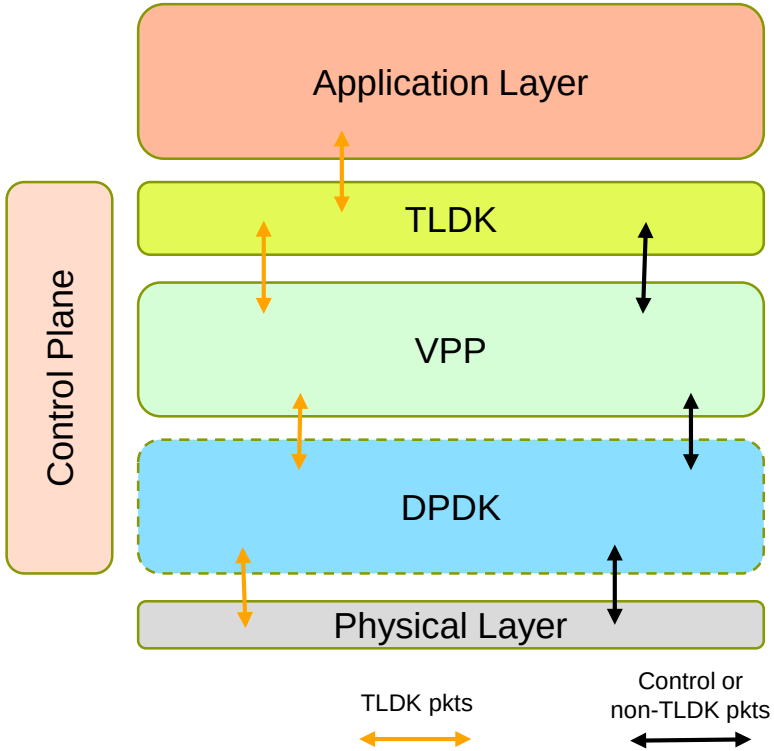
TLDK:
- Handles packet I/O and protocol processing of packets
- Application sets up the UDP/TCP protocol contexts and then calls I/O routines in TLDK to start processing packets

VPP Fastpath:
- Using VPP as the first layer for packet processing before packets are sent to the application layer

DPDK:
- DPDK provides the I/O abstraction to the physical layer for the network devices. The DPDK could be optional here only if some other I/O layer is used.

Physical Layer:
- Ports and other devices like crypto, compression, ...

Control Plane:
- Not fully defined yet, but will need support in the future



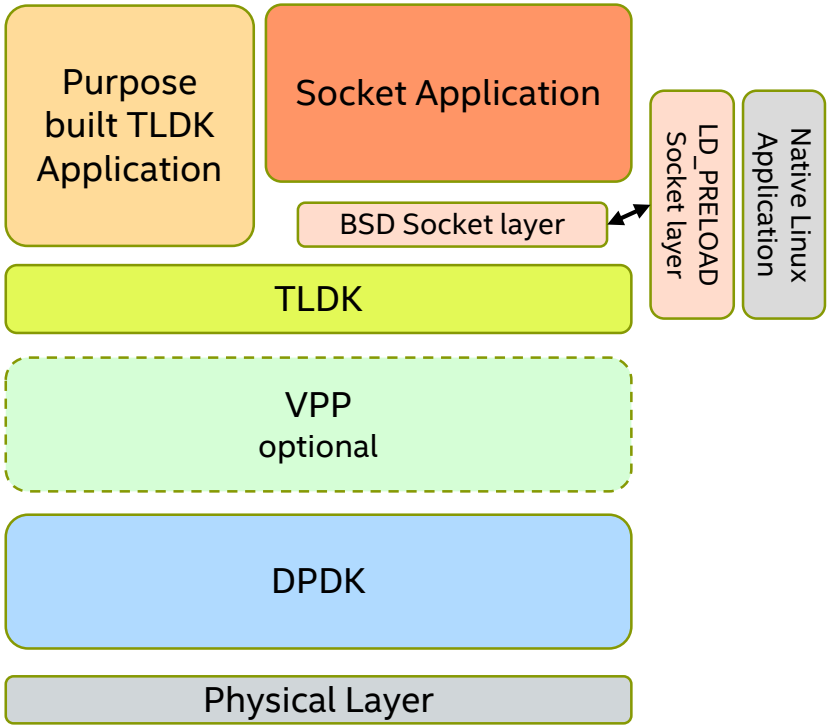
TLDK Application Layer break down

Application Layer:
- The application layer utilizes the TLDK library to process packets for UDP and TCP

Purpose Built Application:
- A purpose built application is one that uses TLDK APIs directly and is built to use these APIs
- Highest performance is expected with this design

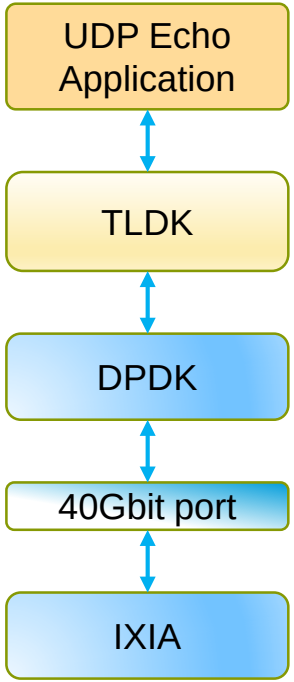
BSD Socket Layer:
- A standard BSD socket layer for applications using sockets in its design
- A lower performance is expected, but allows for current socket type applications to be ported to the system

LD_PRELOAD Socket Layer:
- LD_PRELOAD is used to allow a 'native binary Linux' application to use the accelerated path of VPP/DPDK
- The performance should be a bit better, but does allow these native binary applications to work without any change



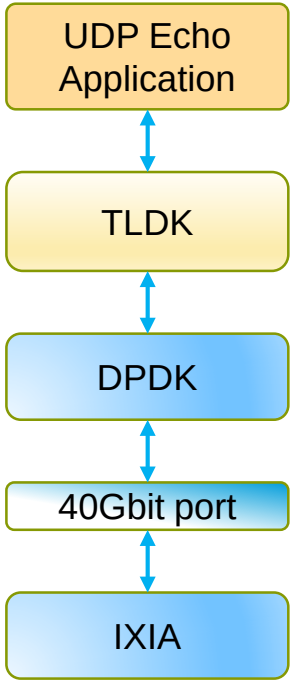
TLDK Developer View

```
struct tle_ctx *tle_ctx_create(struct tle_ctx_param *ctx_param);
struct void tle_ctx_destroy(struct tle_ctx *ctx);
struct tle_dev *tle_add_dev(struct tle_ctx *ctx, struct tle_dev_param *dev);
void tle_del_dev(struct tle_dev *dev);
uint16_t tle_udp_rx_bulk(struct tle_dev *dev, struct rte_mbuf *pkts[],      struct
                         rte_mbuf *rp[], int32_t rc[], uint16_t num);
uint16_t tle_udp_tx_bulk(struct tle_dev *dev, struct rte_mbuf *pkts[], uint16_t num);
struct tle_stream *tle_udp_stream_open(struct tle_ctx *ctx,
                                       struct tle_udp_stream_param *udp_parm);
int tle_udp_stream_close(struct tle_stream *udp);

/* Global variables and structures */
struct tle_ctx *ctx;
struct tle_stream *udp;
struct tle_ctx_param ctx_param;
struct tle_dev_param dev_param;
struct tle_udp_stream_param udp_param;
struct rte_mbuf *pkts[MAX_PKTS], *not_processed_pkts[MAX_PKTS];
int32_t return_codes[MAX_PKTS];
```



TLDK Developer View

```
int main() {
        uint16_t n, r, running = 1;

        ctx = tle_ctx_create(&ctx_param);  /* Fill in the ctx_param structure */

        tle_add_dev(ctx, &dev_param);     /* Fill in the dev_param structure */

        /* fill in udp_param here */
        udp = tle_udp_stream_open(&udp_param);

        while(running) {
                n = tle_udp_rx_bulk(dev, pkts, not_processed_pkts, rc, MAX_PKTS);
                if (n && ((r = tle_udp_tx_bulk(dev, pkts, n)) != n))
                        handle_extra_pkts(dev, pkts, r);     /* Free or resend? */
        }

        tle_udp_stream_close(udp);
}
```

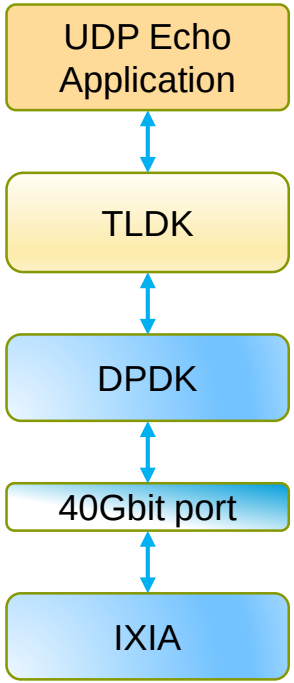


TLDK Performance Numbers (non-optimized code)

CPU: Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
64G Ram, Dual socket system, 2x400GB SSD, 2x1TB drives
NIC: 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)

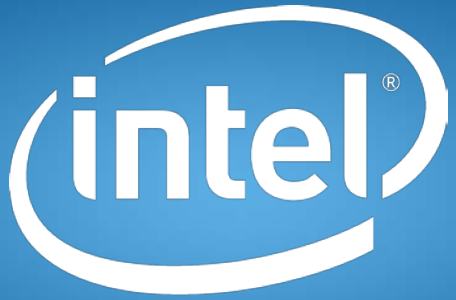


More Information on TLDK

- The project is under the FD.io a Linux Foundation project
  - FD.io Wiki Page (https://wiki.fd.io/view/Main_Page)
  - TLDK is located at: https://wiki.fd.io/view/TLDK
  - Source Code at: git clone https://gerrit.fd.io/r/tldk
- Current code base includes an optimized UDP implementation
- Currently working on TCP implementation
- Each Wednesday10am CST is the TLDK community meeting
  - Meeting info: https://wiki.fd.io/view/TLDK/Meeting
- Additional ideas and contributions welcomed!

TLDK - Status Update

Thank you for attending, any questions?



experience
what's inside™