DPDK & Layer 4 Packet Processing

TLDK – Transport Layer Development Kit

M Jay

Presentation March 1 2017

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].

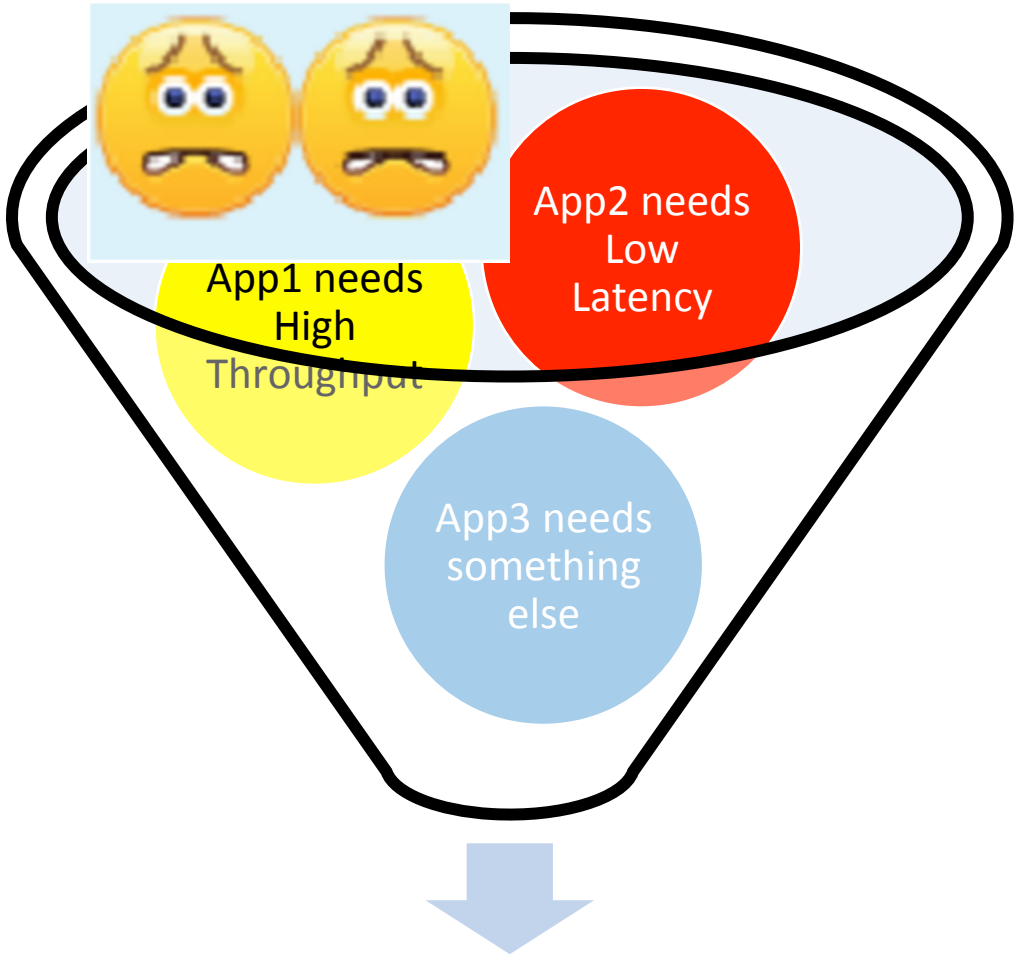
Performance Disclaimers (include only the relevant ones):

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.

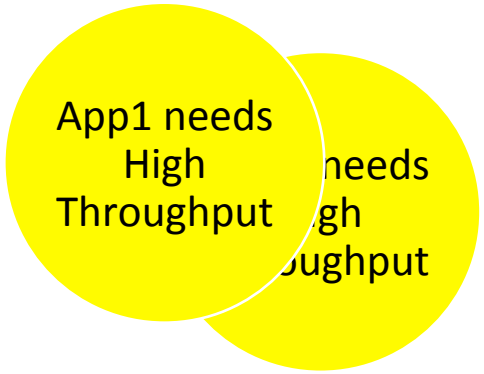
What are the problems with traditional TCP/UDP stacks?

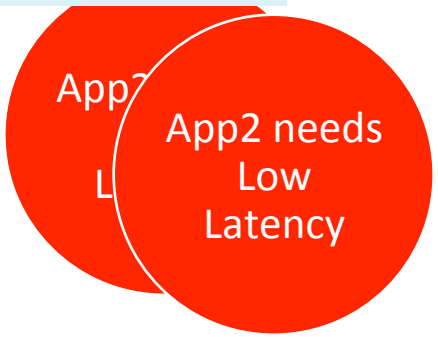
One size “Kernel Stack” does not Fit All App needs

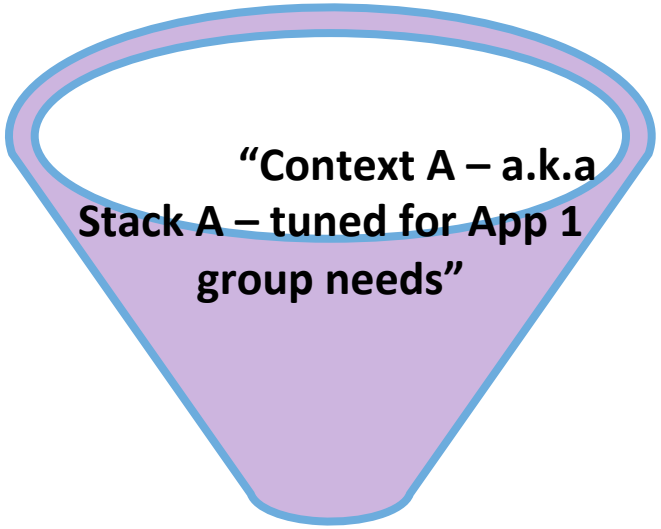


“one size does not fit all”
Single Kernel Stack

What if you can have multiple “user level” stacks?



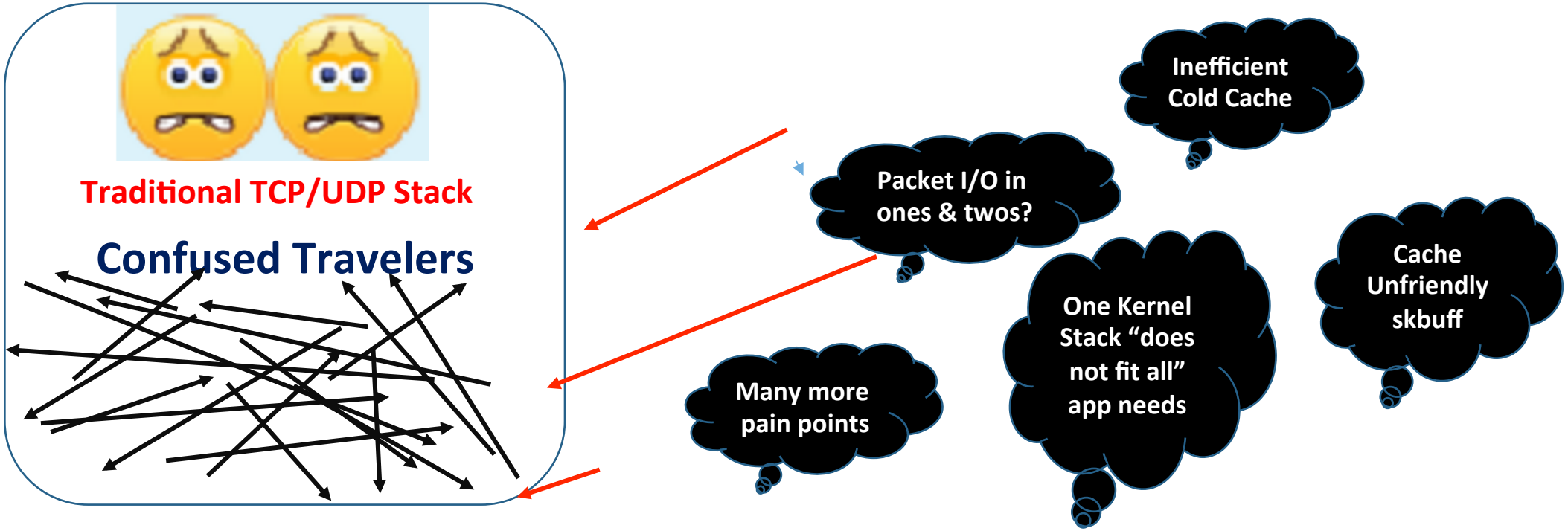


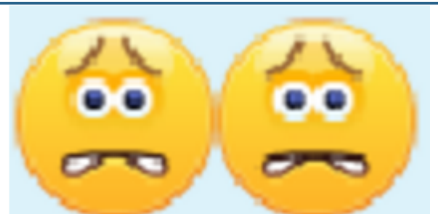


“Context B –
a.k.a Stack B –
tuned for App 2 group needs

“Context C –
a.k.a Stack C –
tuned for App 2 group needs

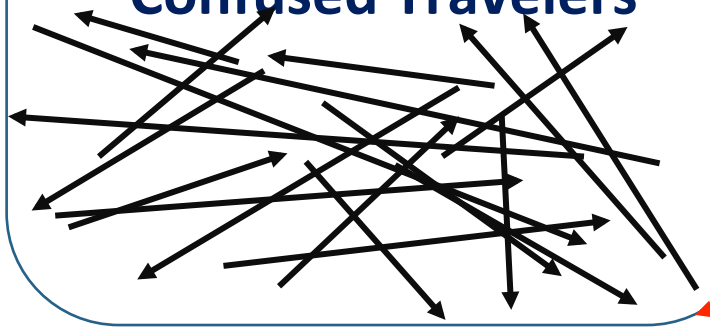
What are the problems with traditional TCP/UDP stacks?





Traditional TCP/UDP Stack

Confused Travelers



Inefficient Cold Cache

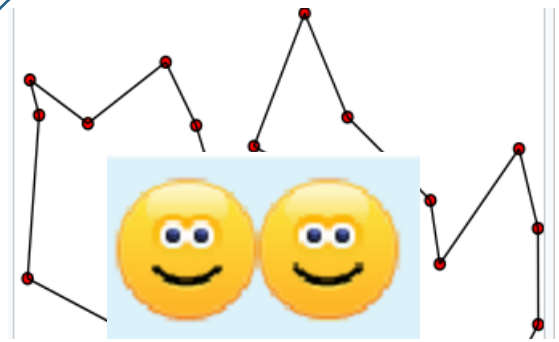
Packet I/O in ones & twos?

One Kernel Stack "does not fit all" app needs

Cache Unfriendly skbuff

Many more pain points

https://en.wikipedia.org/wiki/Travelling_salesman_problem



Optimized TLDK Layer 4 - based on DPDK

User Level TLDK – no constraint of one stack

Warm I Cache & D Cache – Pull packets only when ready to process

Cache Friendly mbuf, Dynamic Ring

Bulk Transfer

TLDK – Any Sample Applications?

You know L2fwd, L3fwd

- What about udpfwd?

The Eternal Loop of /tldk/examples/udpfwd/main.c

\tldk\examples\udpfwd\main.c

```
/* launch all slave lcores. */
RTE_LCORE_FOREACH_SLAVE(i) {
   if (prm[i].be.lc != NULL || prm[i].fe.max_streams !=
      rte_eal_remote_launch(lcore_main, prm + i, i);
}

/* launch master lcore. */
i = rte_get_master_lcore();
if (prm[i].be.lc != NULL || prm[i].fe.max_streams != 0)
   lcore_main(prm + i);
```

Eternal Loop of /tldk/examples/udpfwd/main.c

```
static int
lcore_main(void *arg)
{
   int32_t rc;
   uint32_t lcore;
   struct lcore_prm *prm;

   prm = arg;
   lcore = rte_lcore_id();

   RTE_LOG(NOTICE, USER1, "%s(lcore=%u) start\n",
      __func__, lcore);

   rc = 0;

   /* lcore FE init. */
   if (prm->fe.max_streams != 0)
      rc = netfe_lcore_init(&prm->fe);

   /* lcore BE init. */
   if (rc == 0 && prm->be.lc != NULL)
      rc = netbe_lcore_setup(prm->be.lc);

   if (rc != 0)
      sig_handle(SIGQUIT);

   while (force_quit == 0) {
      netfe_lcore();
      netbe_lcore();
   }
```

What Are The Main APIs ?

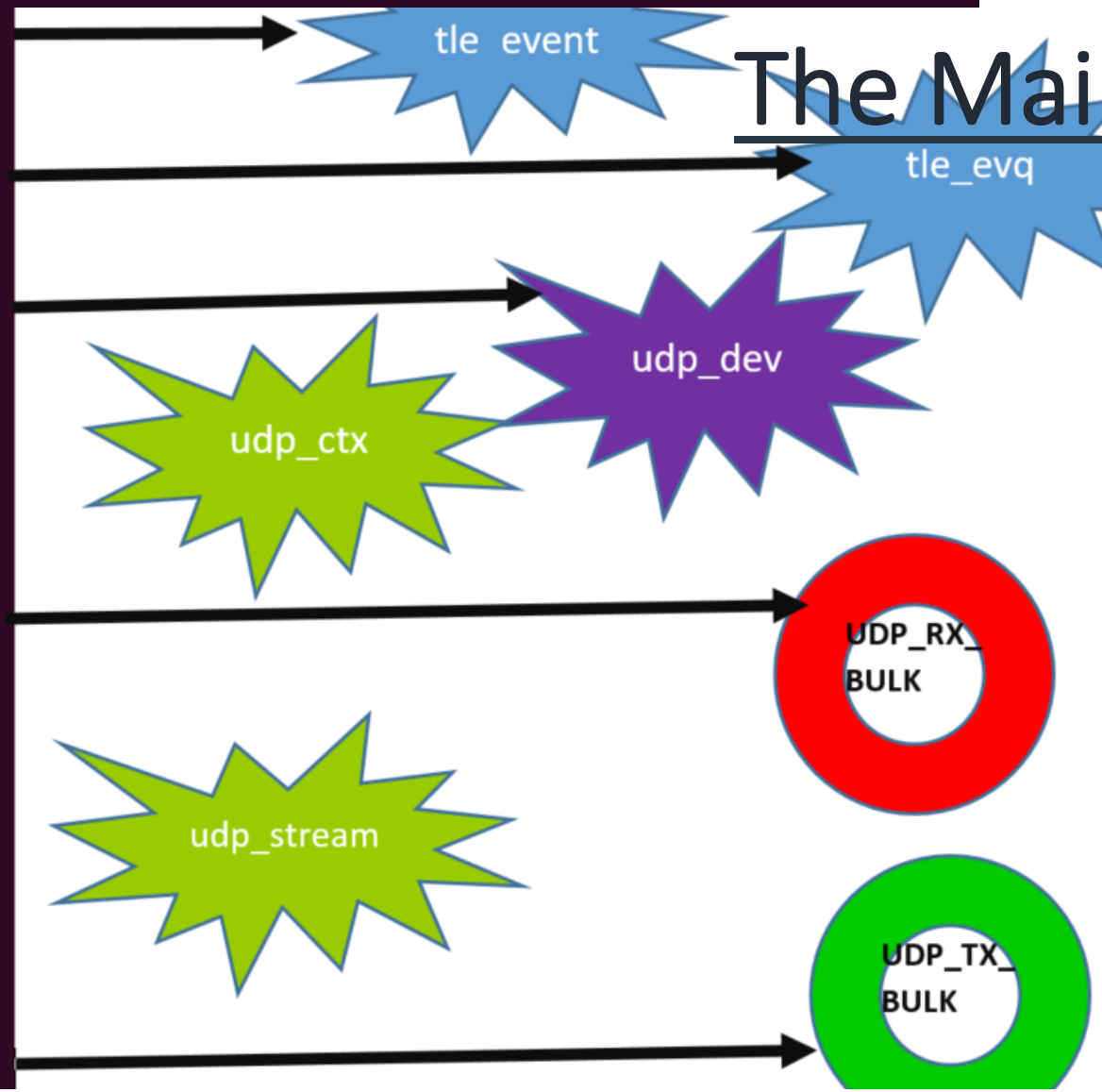
3 Key Modules

1) event.o
2) Udp_ctl.o
3) Udp_rxtx.o

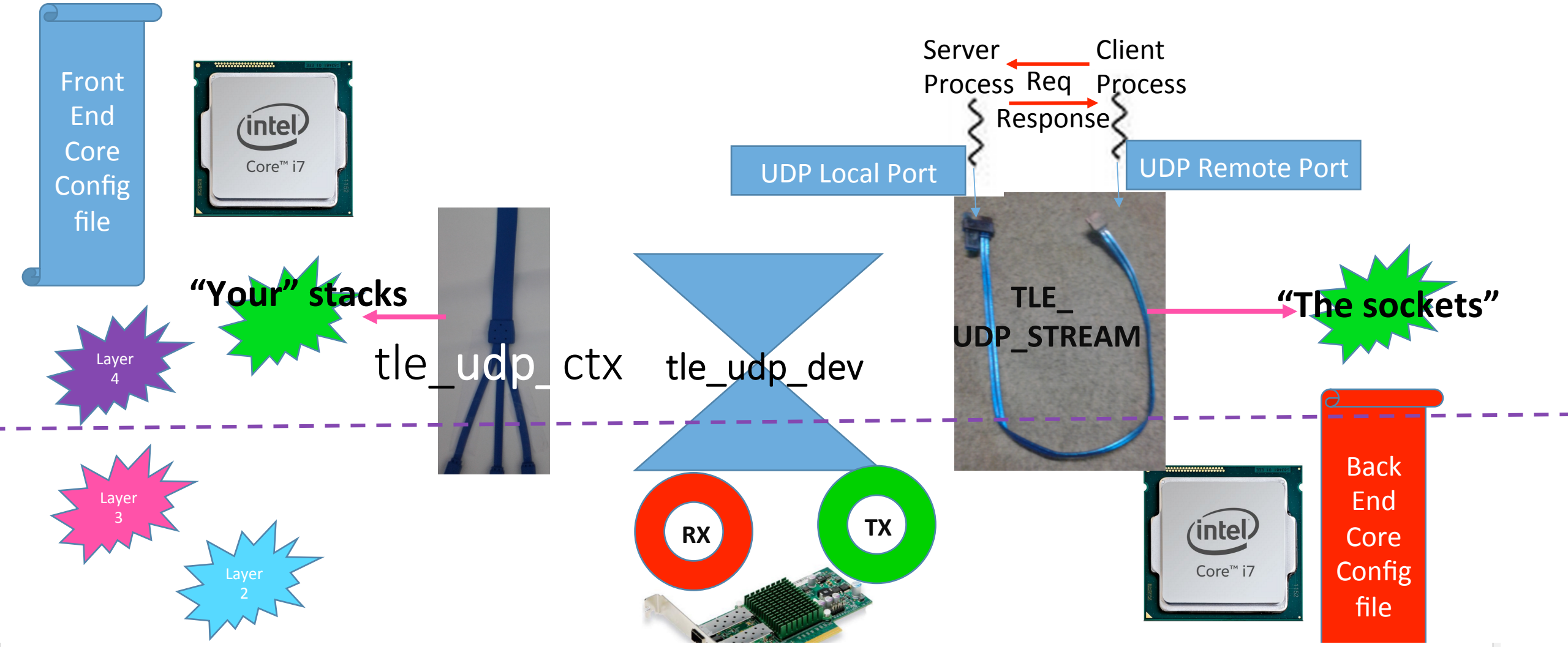
```
0x0000000004302b0      0x3983 main.o
0x0000000000433c33      0x3a3 /home/user/tldk/x86_64-native-linuxapp-gcc/lib/libtle_udp.a(event.o)
0x0000000000433c7c            tle_evq_create
0x0000000000433e22            tle_evq_destroy
0x0000000000433e3d            tle_event_alloc
0x0000000000433f0d            tle_event_free
0x0000000000433fd6     0x5796 /home/user/tldk/x86_64-native-linuxapp-gcc/lib/libtle_udp.a(udp_ctl.o)
0x0000000000436bde            tle_udp_create
0x0000000000436e37            tle_udp_destroy
0x0000000000436ef5            tle_udp_ctx_invalidate
0x0000000000437096            tle_udp_add_dev
0x0000000000437e72            tle_udp_del_dev
0x00000000004388a4            tle_udp_stream_open
0x0000000000438b93            tle_udp_stream_close
0x00000000004396c3            tle_udp_stream_get_param
0x000000000043976c     0x7be1 /home/user/tldk/x86_64-native-linuxapp-gcc/lib/libtle_udp.a(udp_rxtx.o)
0x000000000043ce42            tle_udp_rx_bulk
0x000000000043dd33            tle_udp_tx_bulk
0x000000000043e380            tle_udp_stream_recv
0x0000000000440c3c            tle_udp_stream_send
```

The Main APIs

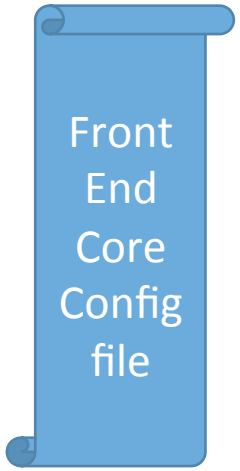
```
/tldk/x86_64-native-linuxapp-gcc/lib/libtle_udp.a(event.o)

tle_event_free

tle_evq_create

tle_evq_destroy

tle_udp_add_dev

tle_udp_create

tle_udp_ctx_invalidate
tle_udp_del_dev
tle_udp_destroy

tle_udp_rx_bulk

tle_udp_stream_close

tle_udp_stream_get_param

tle_udp_stream_open

tle_udp_stream_recv

tle_udp_stream_send

tle_udp_tx_bulk
```



Front End Core + Back End Core Pulling UDP Stack



What does FE config file configure?




```
lcore=3,
op=fwd,
laddr=0.0.0.0,
lport=11211,
raddr=0.0.0.0,
rport=0,
fwladdr=::,
fwlport=0,
fwraddr=2001:4860:b002::56,
fwrport=11211
```

1) Read [Server]
2) Write [Client]
3) Forward [Gateway]

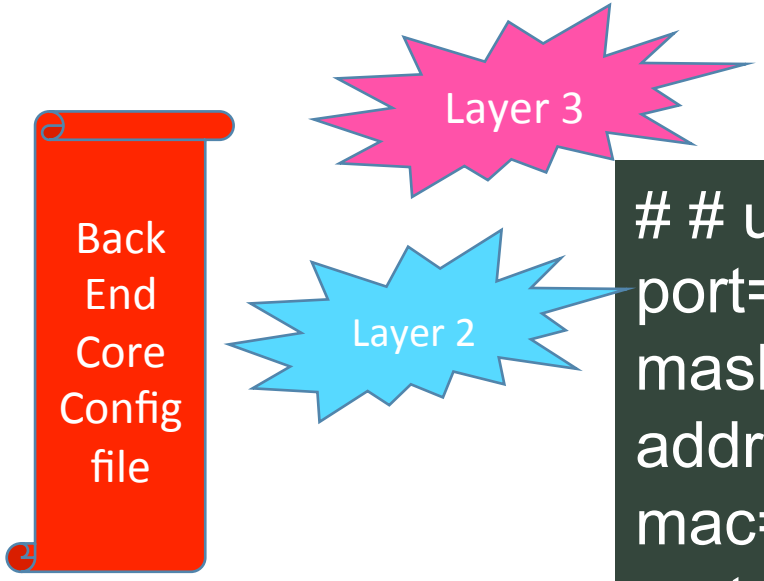
FE config record format:
---

lcore=<uint>,op=<"rx|tx|echo|fwd">,\
laddr=<ip>,lport=<uint16>,raddr=<ip>,rport=<uint16>,\
[txlen=<uint>,fwladdr=<ip>,fwlport=<uint16>,fwraddr=<ip>,fwrport=<uint16>,\
belcore=<uint>]

lcore - EAL lcore to manage that stream(s) in the FE.
op - operation to perform on that stream:
"rx" - do receive only on that stream.
"tx" - do send only on that stream.
"echo" - mimic recvfrom(..., &addr);sendto(..., &addr); on that stream.
"fwd" - forward packets between streams.

laddr - local address for the stream to open.
lport - local port for the stream to open.
raddr - remote address for the stream to open.
rport - remote port for the stream to open.
txlen - data length to send with each packet ("tx" mode only).
fwladdr - local address for the forwarding stream(s) to open ("fwd mode only).
fwlport - local port for the forwarding stream(s) to open ("fwd mode only).
fwraddr - remote address for the forwarding stream(s) to open ("fwd mode only).
fwrport - remote port for the forwarding stream(s) to open ("fwd mode only).
belcore - EAL lcore to manage that stream(s) in the BE.

What does BE config file configure?




```
# # udpfwd BE config file example #
port=0,
masklen=16,
addr=192.168.0.0,
mac=01:de:ad:be:ef:01
port=0,
addr=2001:4860:b002::,
masklen=64,
mac=01:de:ad:be:ef:01
```

In addition, can I do command line config. as well?

Yes ! Command line Options configuring

As an example: udpfwd
--lcores='3,6,8'
-w 01:00.0 -- \ --promisc
--rbufs 0x1000
--sbufs 0x1000
--streams 0x100
\ --fecfg ./fe.cfg
--becfg ./be.cfg
\ port=0,
lcore=6,
lcore=8,
rx_offload=0xf,
tx_offload=0,\
ipv4=192.168.1.233,
ipv6=2001:4860:b002::28

Will create TLDK UDP context on

lcore=6 and lcore=8 (BE lcore) to manage DPDK port 0.

Will assign IPv4 address 192.168.1.233 and
IPv6 address 2001:4860:b002::28 to that port.

The following supported by

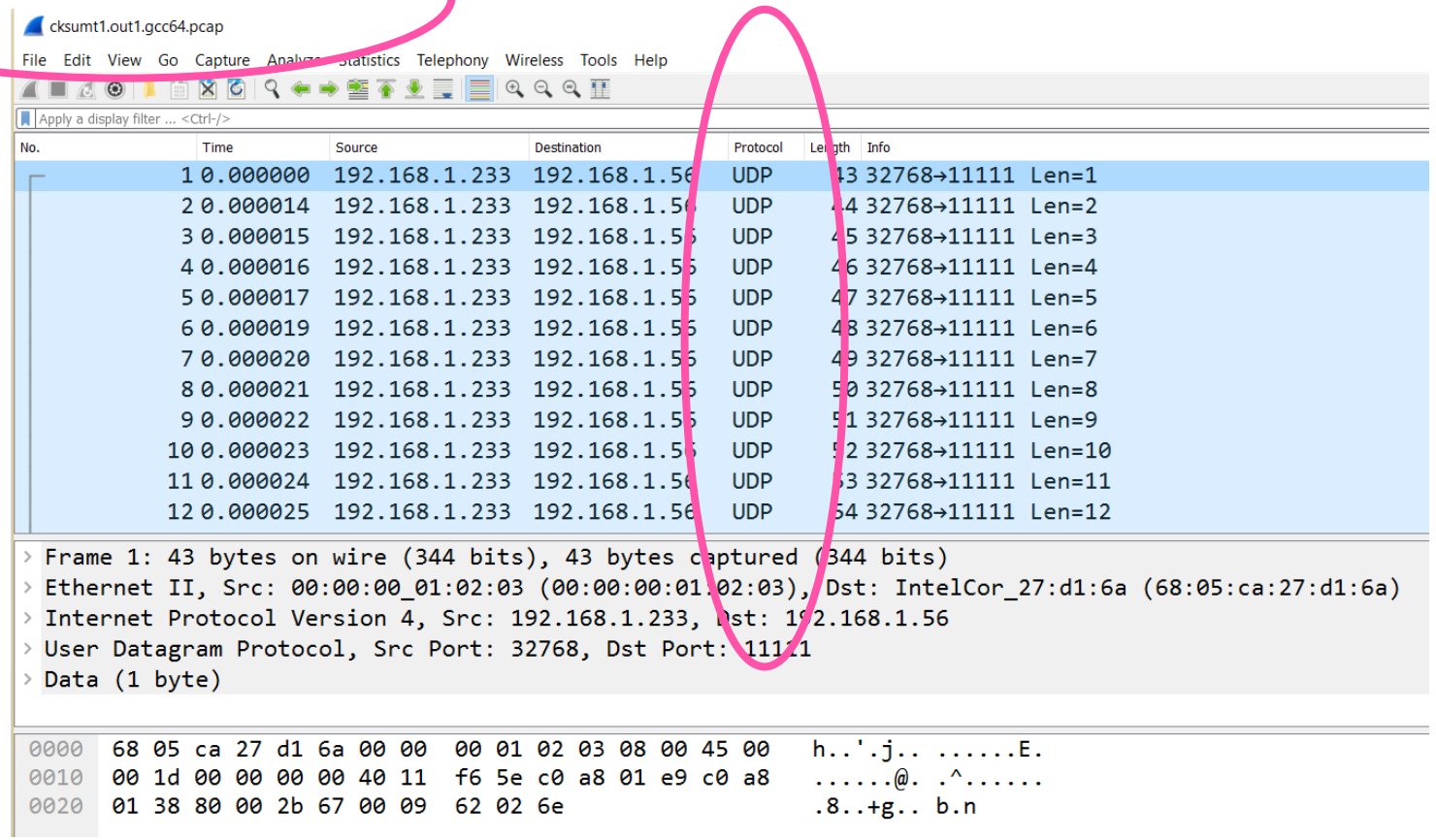
DPDK RX HW offloads:

DEV_RX_OFFLOAD_VLAN_STRIP,
DEV_RX_OFFLOAD_IPV4_CKSUM,
DEV_RX_OFFLOAD_UDP_CKSUM,
DEV_RX_OFFLOAD_
TCP_CKSUM will be enabled on that port. No HW TX offloads will be enabled.

Have you run udpfwd in echo mode?

- Do you have output – we can look at?

Wireshark Output of the file created by udpfwd – echo mode

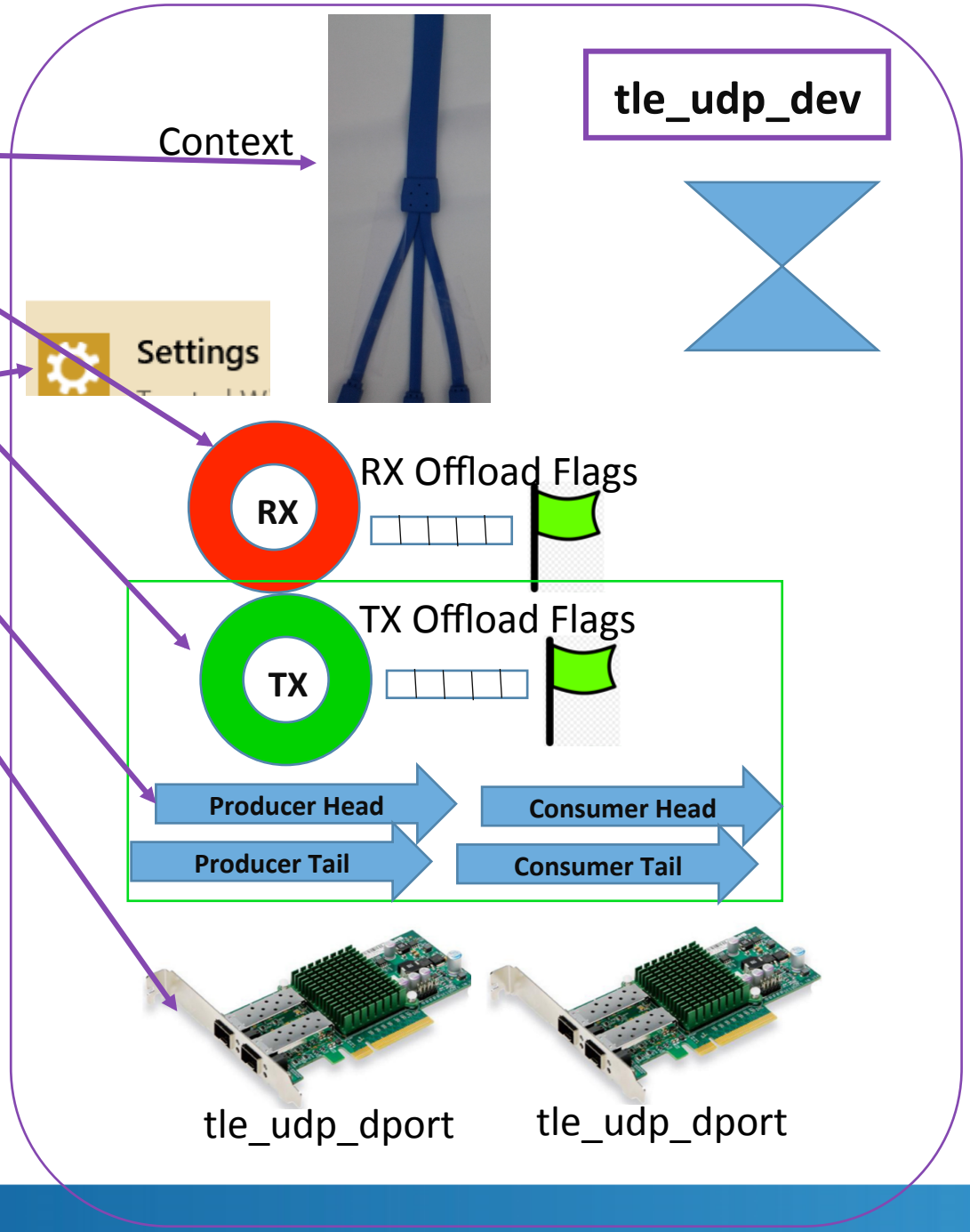


tle_udp_dev

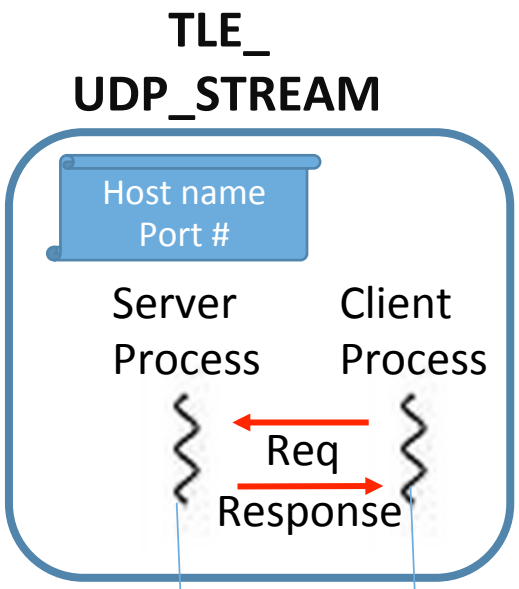
What is tle_udp_dev?

What it contains?

```
(gdb) ptype dev
type = struct tle_udp_dev {
    struct tle_udp_ctx *ctx;
    struct {
        uint64_t ol_flags[2];
    } rx;
    struct {
        uint64_t ol_flags[2];
        rte_atomic32_t packet_id[2];
        struct tle_dring dr;
    } tx;
    struct tle_udp_dev_param prm;
    struct tle_udp_dport *dp[2];
} *
```



What is TLE_UDP_STREAM?



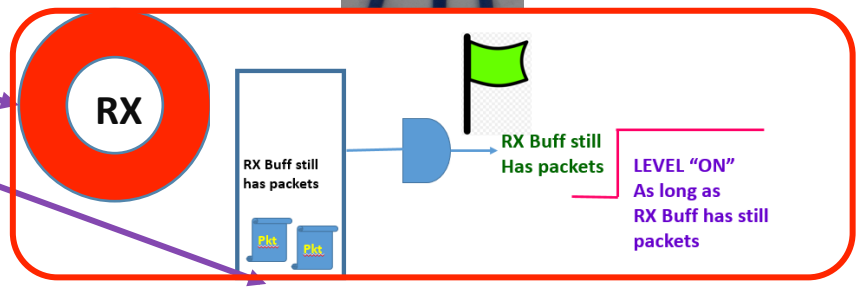
TLE_
UDP_STREAM

What is TLE_UDP_STREAM?

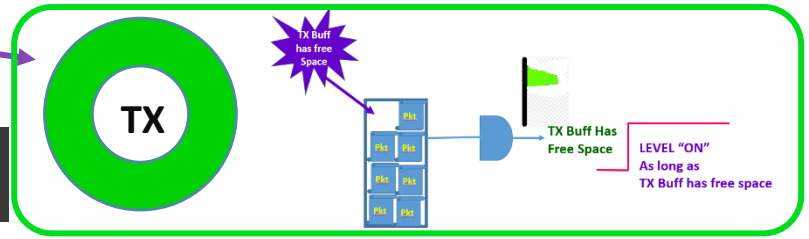
Linked List of
UDP_Stream

```
(gdb) ptype s
type = struct tle_udp_stream {
    struct {
        struct tle_udp_stream *stqe_next;
    } link;
    struct tle_udp_ctx *ctx;
    uint8_t type;
    struct {
        struct rte_ring *q;
        struct tle_event *ev;
        struct tle_udp_stream_cb cb;
        rte_atomic32_t use;
    } rx;
    union udp_ports port;
    union udp_ports pmsk;
    union {
        struct {...} ipv4;
        struct {...} ipv6;
    };
    struct {
        rte_atomic32_t use;
        struct {...} drb;
        struct tle_event *ev;
        struct tle_udp_stream_cb cb;
    } tx;
    struct tle_udp_stream_param prm;
} *
```

Context



DNS – 53 DHCP – 67 …

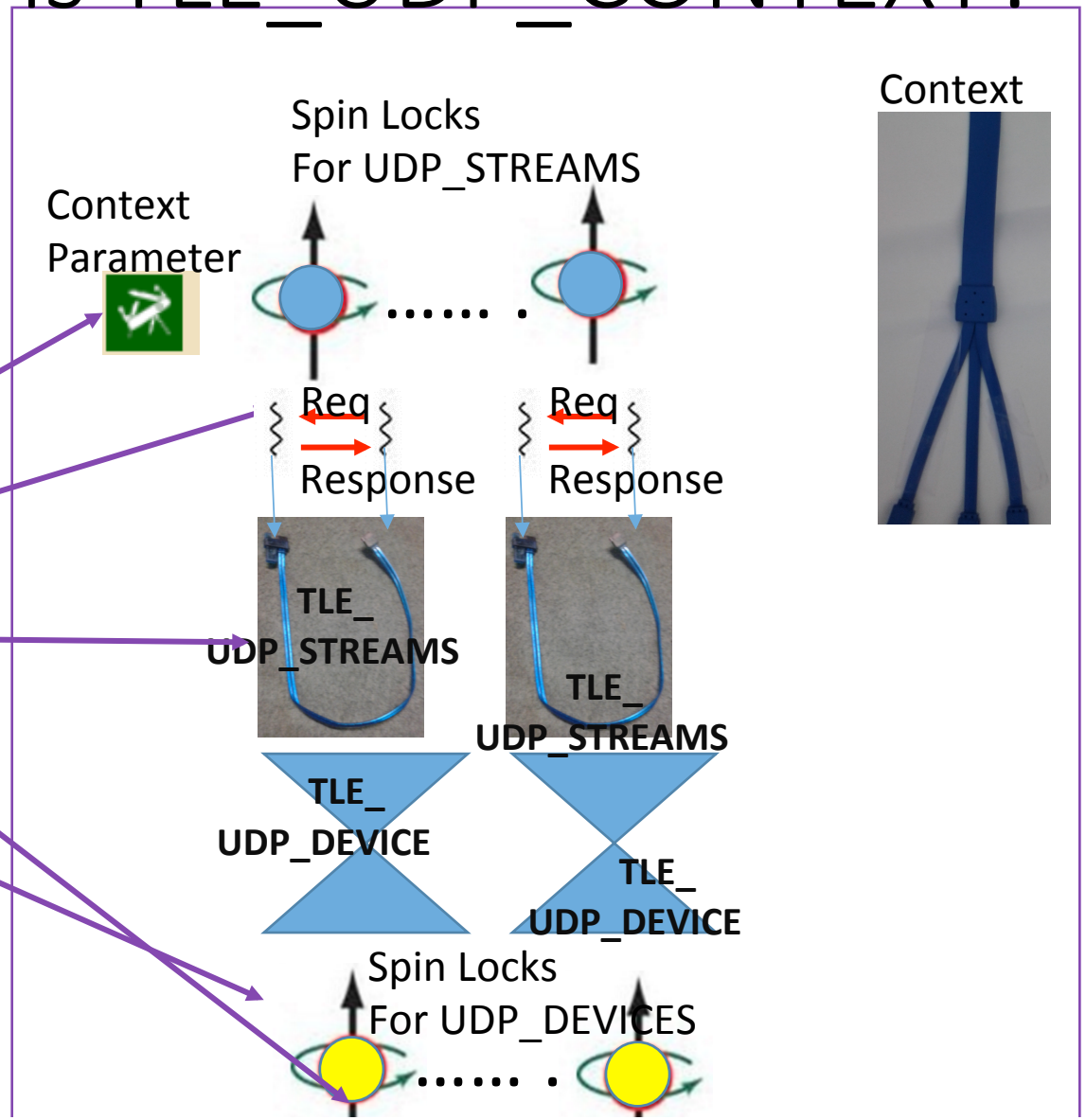


Settings

What is TLE_UDP_CONTEXT?

Context

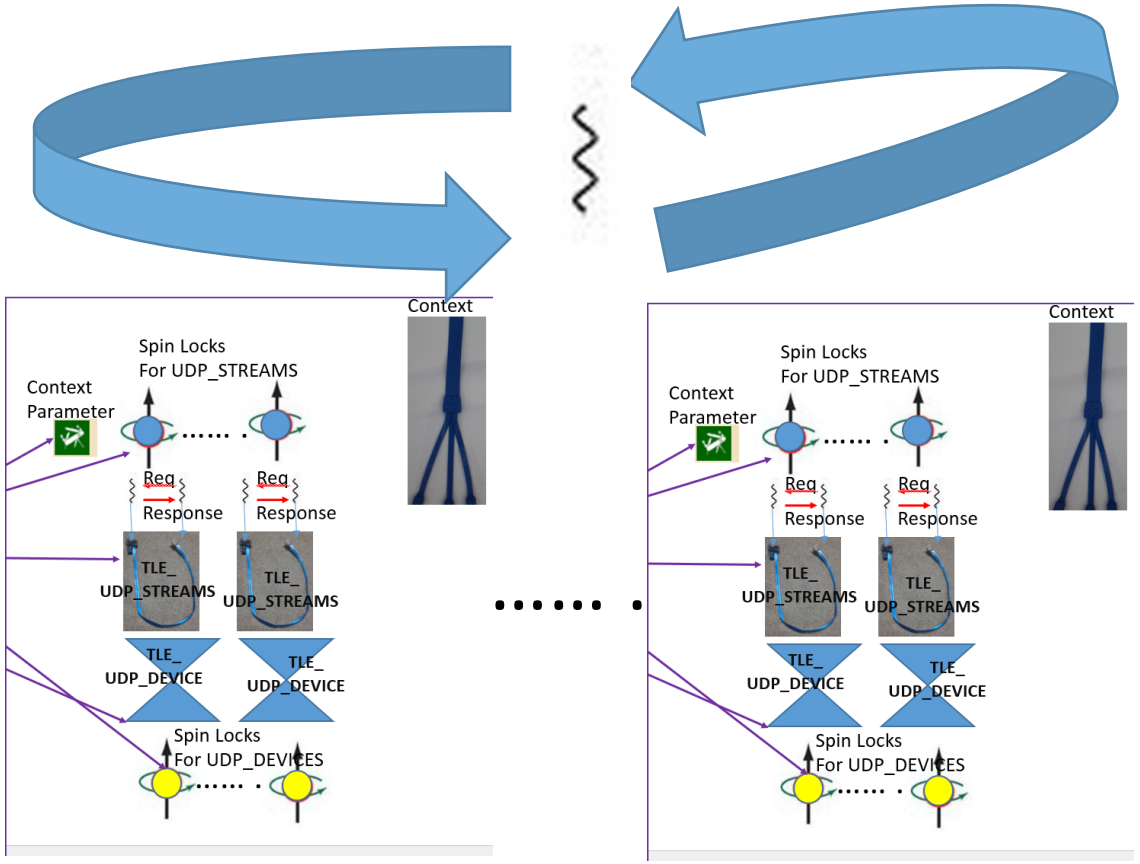
```
(gdb) ptype ctx
type = struct tle_udp_ctx {
    struct tle_udp_ctx_param prm;
    struct {
        rte_spinlock_t lock;
        uint32_t nb_free;
        struct {...} free;
        struct tle_udp_stream *buf;
    } streams;
    rte_spinlock_t dev_lock;
    uint32_t nb_dev;
    struct udp_pbm use[2];
    struct tle_udp_dev dev[32];
} *
```



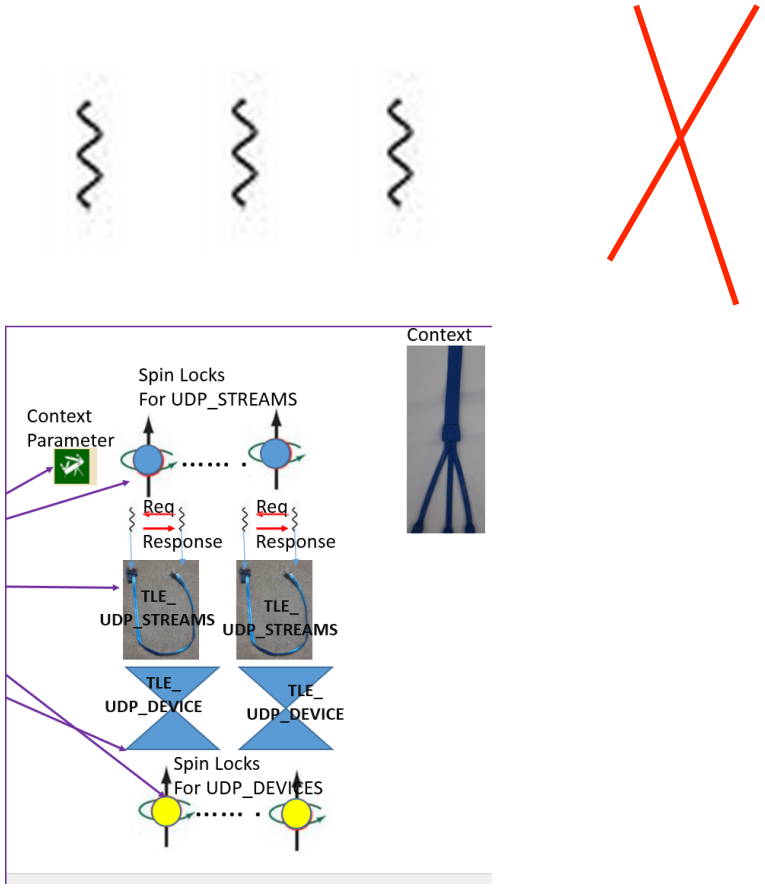
Context

Developer's Responsibility – Scope

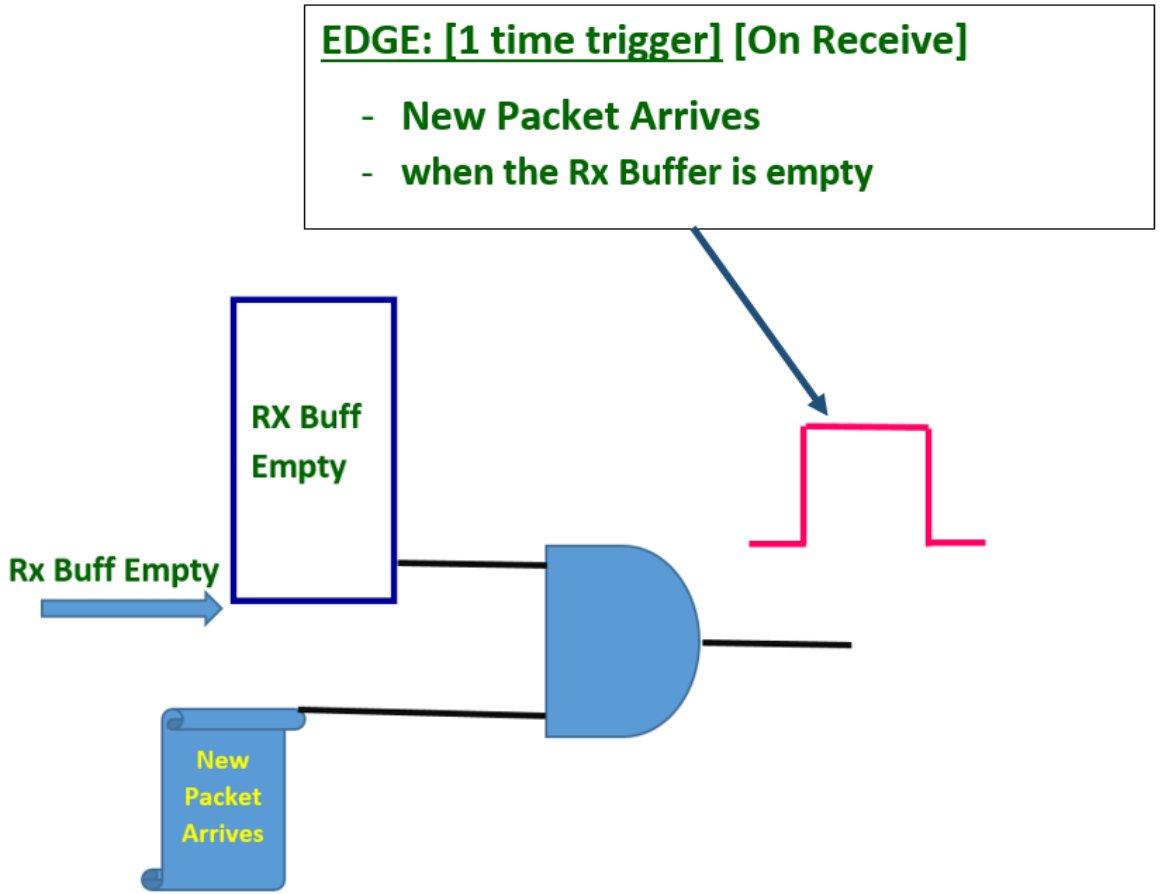
One Thread Driving
Multiple Contexts Allowed As Is



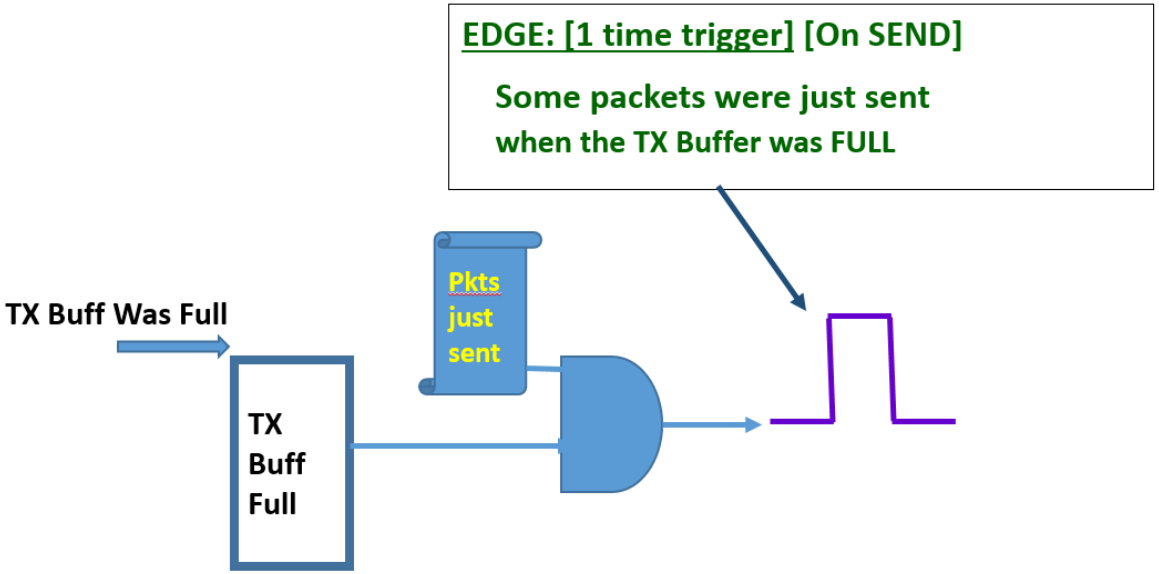
Multiple Threads Driving Single Context
NOT Allowed As Is. You need to provide
Explicit synchronization



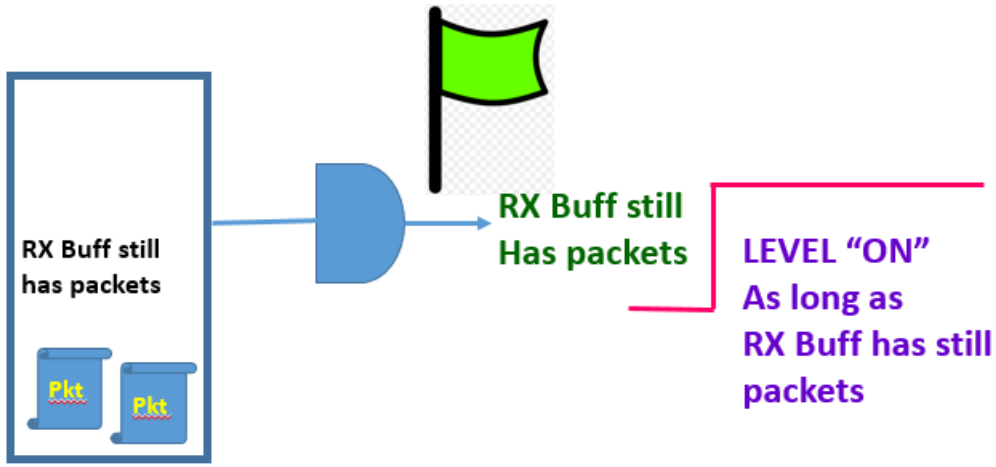
Callback Model for RECEIVE



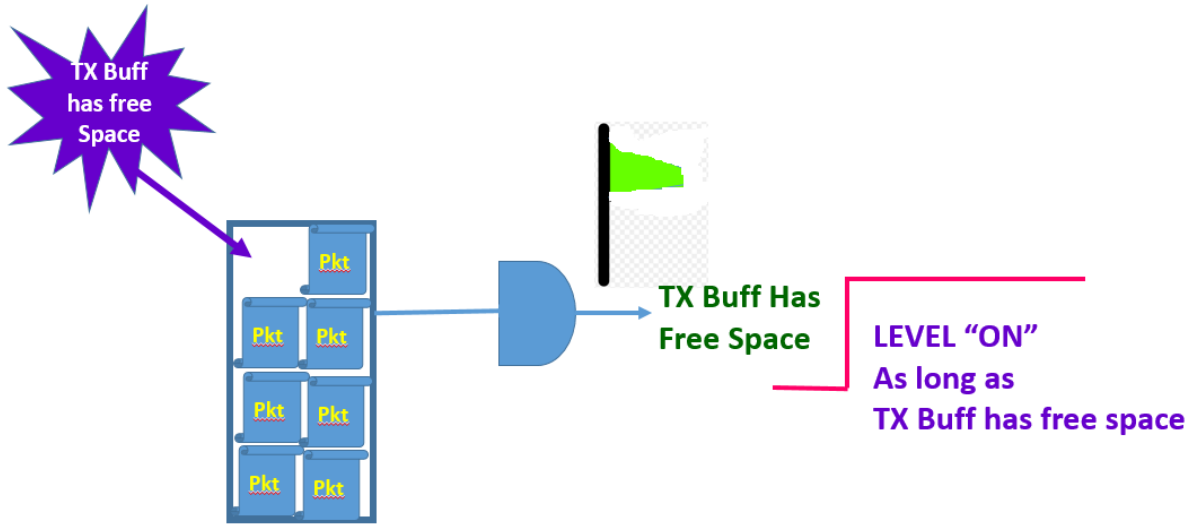
Callback Model for TRANSMIT



Event Model for RX



Event Model for TX



Can I use Callback for RX & Event notification for TX?

- Vice versa?

Back Up

Event/Callback & RX/TX - Combinations

Notification Choice	Event Notify Set for Send	Callback Set for Send	No Notification Set for Send
Event Notify set for Rcv	For both receive & Send, you get LEVEL TRIGGERED notification	You get LEVEL TRIGGERED For recv & EDGE TRIGGERED notification for Send	Only Receive gets notified – LEVEL TRIGGERED. No Notification for Send
Callback set for Rcv	You get EDGE TRIGGERED For recv & LEVEL TRIGGERED notification for Send	For both receive & Send, you get EDGE TRIGGERED notification	Only Receive gets notified – EDGE TRIGGERED. No Notification for Send
No Notification set for Rcv	Only Send gets notified – LEVEL TRIGGERED. No Notification for Recv	Only Send gets notified – EDGE TRIGGERED. No Notification for Recv	No Notification for both send & Receive

Key gdb commands

ptype x	Data Structure Type of stuct x
b <function name>	Set breakpoint
clear <function name>	Clear breakpoint @ function
info locals	
gdb - - arg ./<function><args>	Debugging
run	
continue	

Key virsh & ssh commands

sudo virsh domifaddr <Host Name>		
Sudo virsh list - - all		
ssh user@192.168....		
ip r add default via 192.168… dev ens8	After reboot of VMs, to resolve any connectivity issue	

Key git & Misc commands

git log	Returns 40 bit IDs	
git checkout <40 bit ID>		
sed -i 's/\r$//' filename	Windows control characters issue	

Key Objdump commands

Objdump - - syms <file.o>	To read the symbols	
Objdump –S <file.o>	Disassembly mixed with source + Assembly	Objdump –S <file.o>
Objdump –h <file.o>	To get details about file	Txt, data, bss, stack
Objdump -x <file.o>	For all headers in detail	

Install puTTY

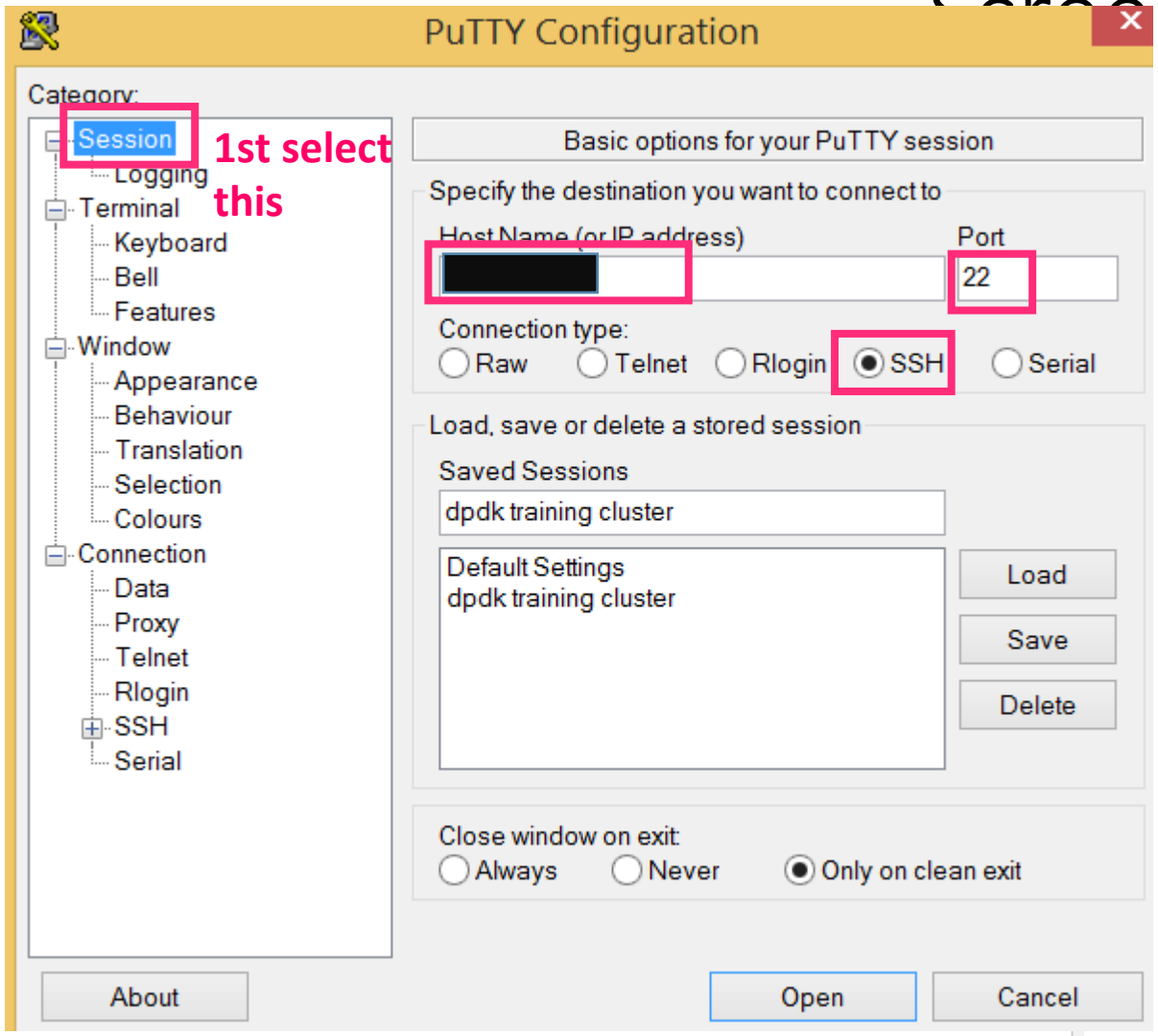
- Click the URL
- http://the.earth.li/~sgtatham/putty/0.60/x86/
- Double Click putty-0.60-installer.exe option
- You have downloaded on your Download folder
- Right Click and Run As Administrator
- You have installed PuTTY

- **Next Step is To Configure PuTTY**

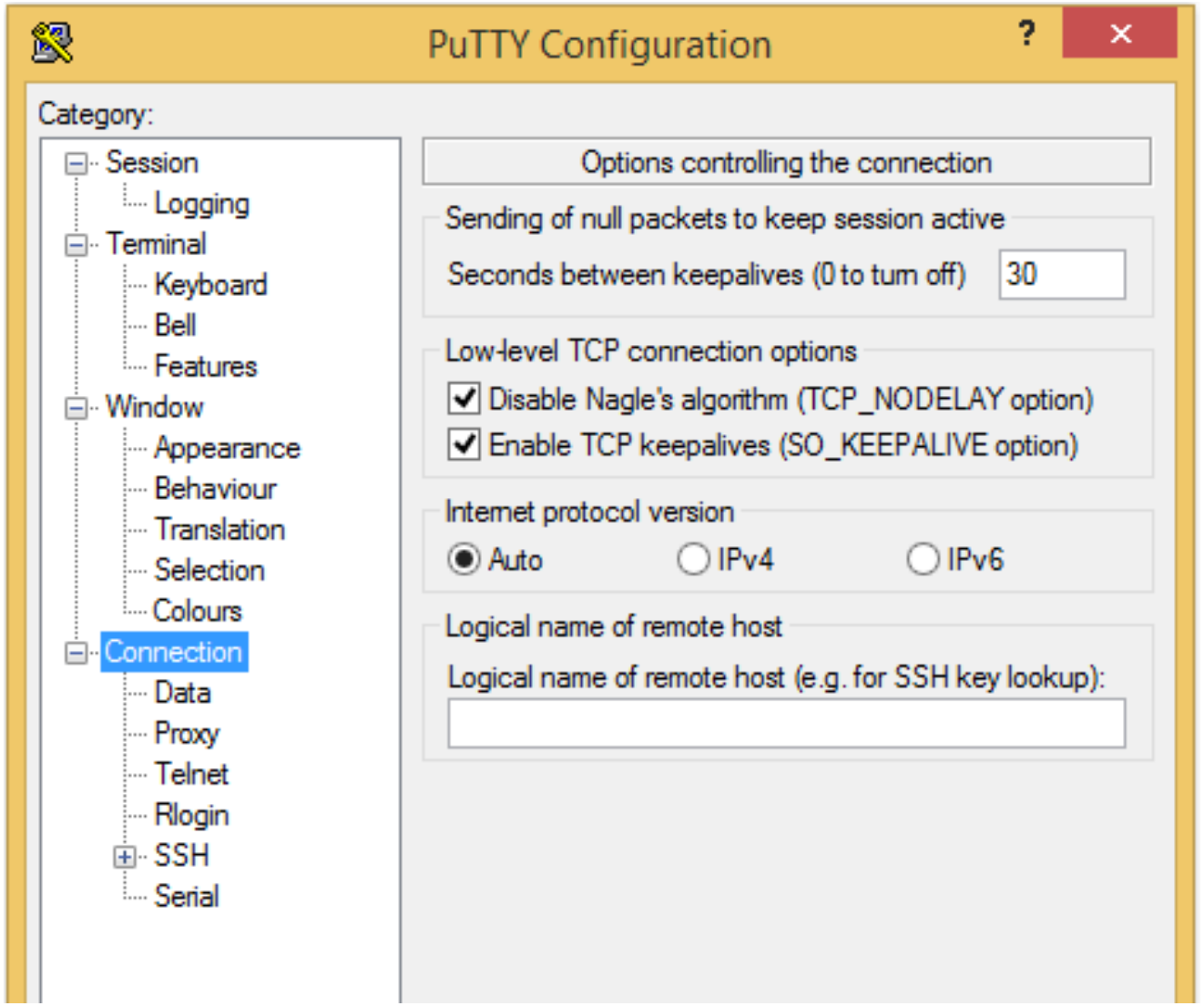
PuTTY Config: → Session

1 / 2

1) Host name
2) Port 22
3) Connection Type SSH



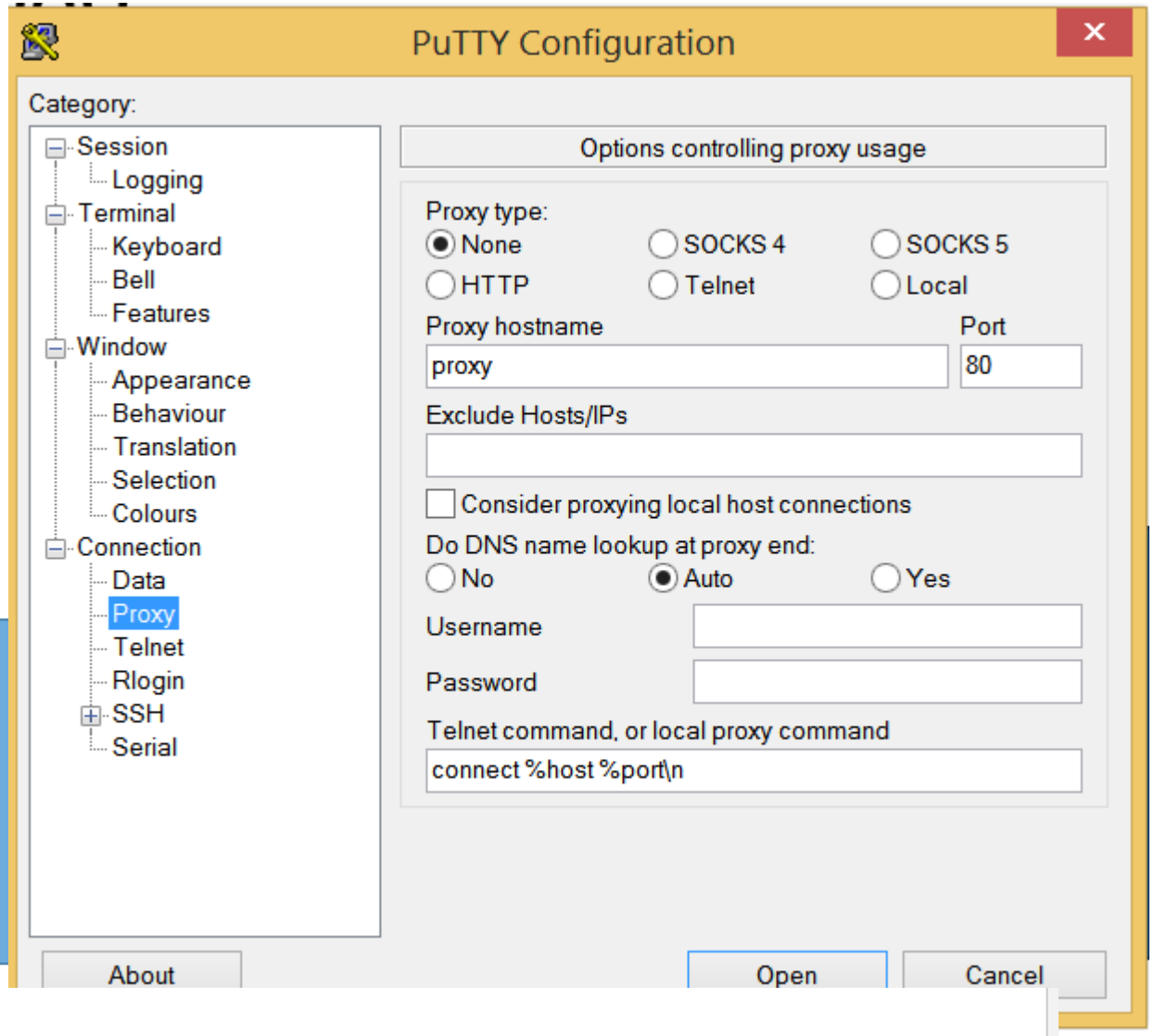
Keep Alive – 30 sec; Enable TCP Keepalives



If you are outside

PuTTY Config: Connection -- > Proxy Screen 2 / 2

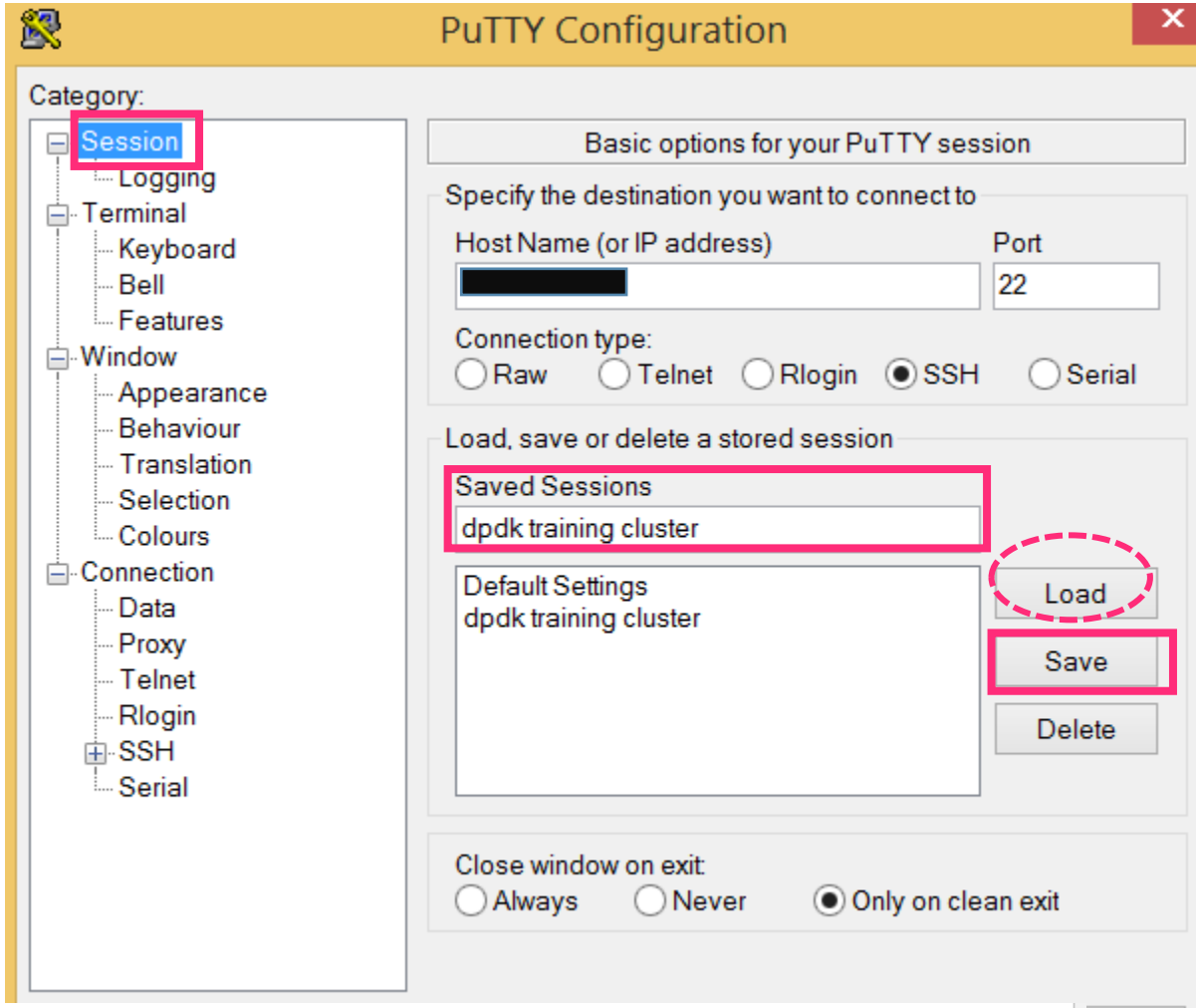
1) **Proxy Type**: None
2) **Proxy host name**: Leave it blank
3) **Do DNS name lookup at proxy end: Auto**



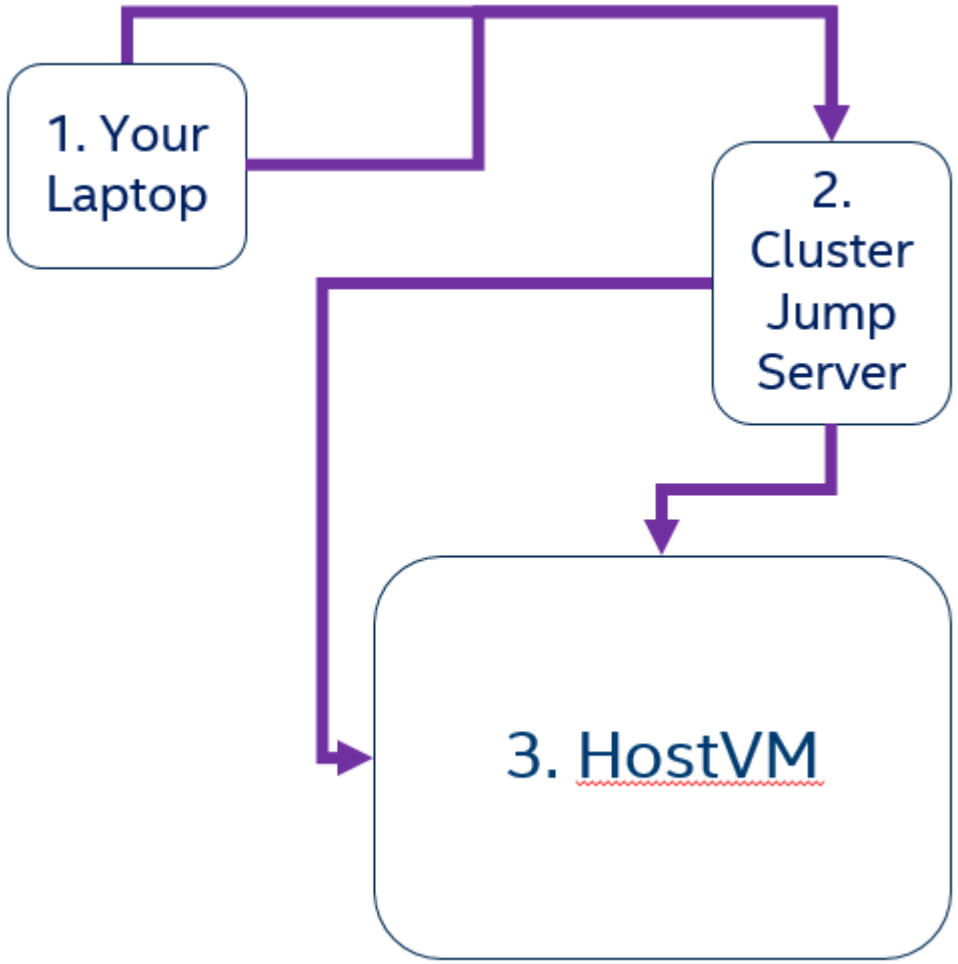
- **Next Step is To Save The Configuration**

1) Name The Configuration. 2) Save It

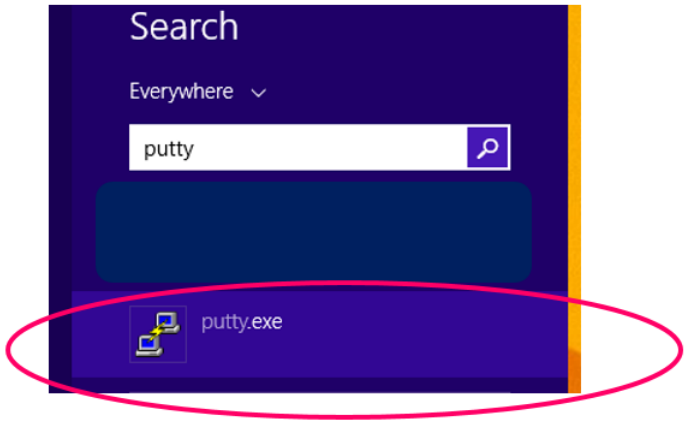
- Click “Session” on the top left corner.
- It will take you to the 1st screen - shown here .
- In Saved Sessons Box, Enter a name, e.g., dpdk training cluster
- Press Save Button.
- From now onwards, you can load the saved session when starting PuTTY

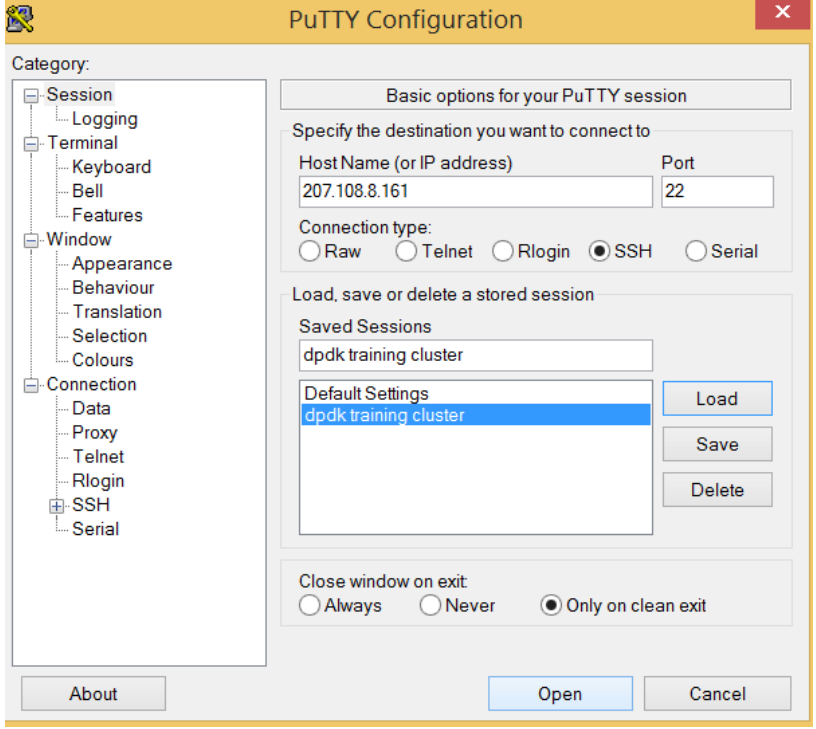


Getting To The HostVM-<m> through Cluster Jump Server



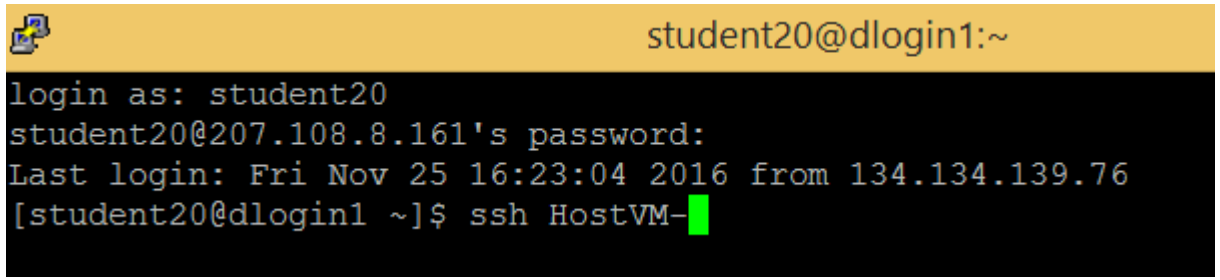
Start PuTTY





- Find PuTTY that you just installed
- Click on icon “putty.exe”. You will get the PuTTY Config Screen
- Select the Saved Session, shown here e.g., dpdk training cluster.
- Press Load Button. Press Open Button.
- You will get ssh session shown in 3rd screen asking username
- Username: student<m> <m> is given to you. For example it may be student19 or student25 or any other student<m>.
- Ask for your specific <m> and use only that. This will avoid overlapping with other teams’ <m>.
- Password is same as username. For example, if your username is student19, then password is student19
- Repeat the above steps so that you have many connections to the jump server

How to login and Connect to HostVM-<m>



- **Login as student<m> password student<m>**
- The next step is to connect to HostVM-<m> assigned to you.
- In case not given, Ask for your own <m>
- Note: We are using 1:1 – same student name and same HostVM- name
- Type in the following command
- ***ssh HostVM-<m>***
- Username: password Password: password
- **sudo su –**
- **cd /home/user/dpdk_hands_on**
- **Vi commands.txt**

Open 2 Windows

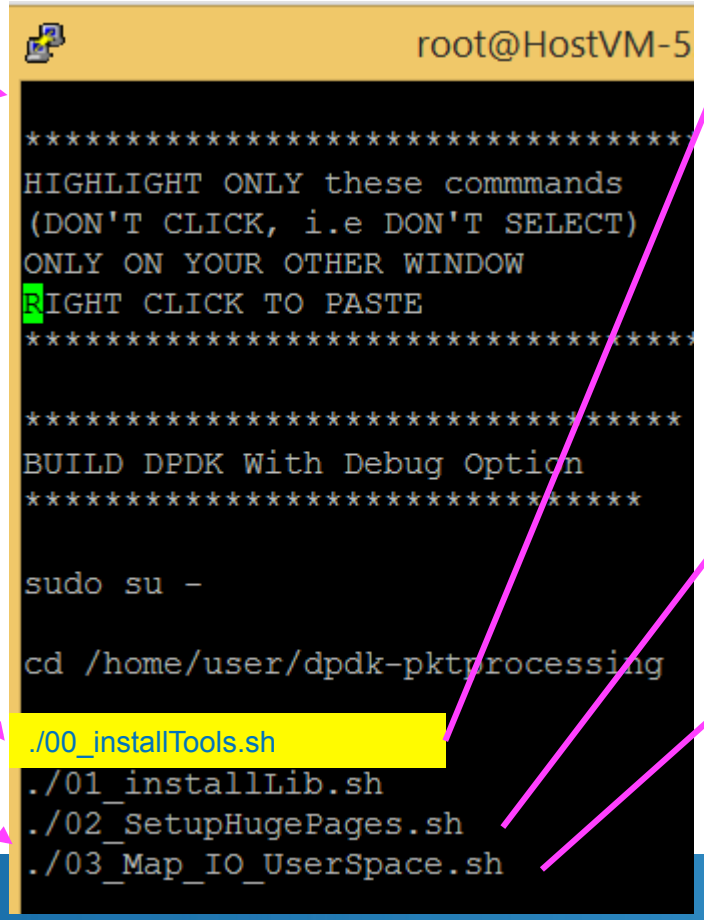
A) FIRST, In the 1st Window, enter these commands

```
ssh HostVM-<M>
Password is password
sudo su -
cd /
vi  Cpy_PasteFromMe.txt
```

B) NEXT, in the 2nd Window, enter these commands

```
ssh HostVM-<M>
Password is password
sudo su –
cd /
```

1



3
Right Click in this Window to Paste

1) Only Highlight from the vi in this Window
This is your "Highlight to Copy From Window"
Don't Click. i.e., Don't Select. Only Highlight

2

Cpy_Paste From Me.txt

```
#Cpy_PasteFromMe.txt
sudo su -
cd /
./00_ReserveHugePage.sh
source 01_SetEnvironmentVariables.sh
cd $RTE_SDK
************************************************
**** Please be patient
*** The following commands will take time to start
************************************************
make config T=$RTE_TARGET
make
make install T=$RTE_TARGET
cd $GTEST_DIR
cmake CMakeLists.txt
make
cd $TLDK_ROOT
make clean
make all O=$RTE_TARGET
```

```
# ********** TEST 1 ************************************
# Shall we take a look at key data structures?
# For that let us use gdb and run Dynamic Ring Buffer Tests
# ************************************************************
# ************************************************************

gdb --arg ./x86_64-native-linuxapp-gcc/app/test_dring

# Set BreakPoints
b drb_dump
b tle_dring_dump
run
#Examine dynamic ring
ptype dr
# Examine dynamic buffer
ptype db
continue
*************************
```

STOP! OBSERVE the tle_dring structure

WHAT DO YOU SEE?

Producer / Consumer pointers with Head and Tail Pointers.

WHAT VALUE YOU SEE for Head and Tail Pointers?

****

continue

continue

****

STOP: Now what value you see for head and tail pointers

****

continue

type continue till you see [Inferior 1 (process <> exited normally]

quit

********** TEST 2 **********

Now run the same dring test without gdb

****

./x86_64-native-linuxapp-gcc/app/test_dring

#**********

#**********TEST 3 **********

Shall we run the Layer 4 UDP Application? Echo mode?

#**********

cd $TLDK_ROOT

./02_run_UDP_Echo_Test.sh

Check that all packet in output pcap file contain valid udp checksum:

How do you “repair” the build problem with gtest?

How To Repair Build error?

- unset GMOCK_DIR
- unset GTEST_DIR

- cd $TLDK_ROOT
- cd make clean
- Make all O=$RTE_TARGET

- cd googletest_feb2017
- cmake CMakeLists.txt
- Make

- export GTEST_DIR=/googletest_feb2017/googletest
- export GMOCK_DIR=/googletest_feb2017/googlemock

- cd $TLDK_ROOT
- make clean
- make all O=$RTE_TARGET