基于 dpdk 的模拟链路级 DDoS 攻击实验

Author：57119108 吴桐
日期：2021.7.14

实验内容及实验目的

分布式拒绝服务攻击（Distributed Denial of Service，简称 DDoS）是指处于不同位置的多个攻击者同时向一个或数个目标发动攻击，或者一个攻击者控制了位于不同位置的多台机器并利用这些机器对受害者同时实施攻击。由于攻击的发出点是分布在不同地方的，这类攻击称为分布式拒绝服务攻击，其中的攻击者可以有多个。

本实验利用 10Gbps 网卡前端的服务器来模拟僵尸网络所产生的流量。利用 Cisco TRex 进行发包来尝试逼近链路带宽的理论值，以达到模拟 DDos 攻击时网络链路没有多余带宽来为正常用户使用的情况。

根据以太网帧的结构分析，帧的大小介于 64bit 和 1518bit，本实验采用这两个极限值作为网络包的大小来进行流量监测。dpdk 是绕过 Linux 内核的网络发包环境，实验将采用控制变量法，对不同大小的报文用不同的核数进行流量监测，分析不同核数时的 CPU 利用率以及吞吐量。

实验环境

工作站：DELL 工作站
网卡：INTEL X710-DA4 10Gbps

实验步骤及运行结果

在实地操作前，实验所需环境已基本配置完善。我们可以使用命令查看配置（图 1-3）。

```
seuguest@Dell-Precision-7920-Tower:~/trex$ cd ./v2.89
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ sudo ./dpdk_nic_bind.py -s
[sudo] seuguest 的密码:

Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
===================================
0000:00:1f.6 'Ethernet Connection (3) I219-LM' if=eth0 drv=e1000e unused=igb_uio
 *Active*
0000:02:00.0 'I210 Gigabit Network Connection' if=eth1 drv=igb unused=igb_uio
0000:a6:00.0 'Ethernet Controller X710 for 10GbE SFP+' if=nic0 drv=i40e unused=i
gb_uio
0000:a6:00.1 'Ethernet Controller X710 for 10GbE SFP+' if=nic1 drv=i40e unused=i
gb_uio
0000:a6:00.2 'Ethernet Controller X710 for 10GbE SFP+' if=nic2 drv=i40e unused=i
gb_uio

Other network devices
=====================
0000:a6:00.3 'Ethernet Controller X710 for 10GbE SFP+' unused=i40e,igb_uio
```

图 1

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ sudo ./dpdk_setup_ports.py -i
By default, IP based configuration file will be created. Do you want to use MAC based config? (y/N)y
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| ID | NUMA |   PCI   |        MAC        |                  Name                  | Driver | Linux IF |  Active  |
+====+======+=========+===================+========================================+========+==========+==========+
| 0  | 0    | 00:1f.6 | 6c:2b:59:f2:e3:f8 | Ethernet Connection (3) I219-LM        | e1000e | eth0     | *Active* |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| 1  | 0    | 02:00.0 | 6c:2b:59:f2:e2:5c | I210 Gigabit Network Connection        | igb    | eth1     |          |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| 2  | 1    | a6:00.0 | 3c:fd:fe:a6:6f:f0 | Ethernet Controller X710 for 10GbE SFP+ | i40e   | nic0     |          |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| 3  | 1    | a6:00.1 | 3c:fd:fe:a6:6f:f1 | Ethernet Controller X710 for 10GbE SFP+ | i40e   | nic1     |          |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| 4  | 1    | a6:00.2 | 3c:fd:fe:a6:6f:f2 | Ethernet Controller X710 for 10GbE SFP+ | i40e   | nic2     |          |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
| 5  | 1    | a6:00.3 |                   | Ethernet Controller X710 for 10GbE SFP+ |        |          |          |
+----+------+---------+-------------------+----------------------------------------+--------+----------+----------+
Please choose an even number of interfaces from the list above, either by ID, PCI or Linux IF
Stateful will use order of interfaces: Client1 Server1 Client2 Server2 etc. for flows.
Stateless can be in any order.
For performance, try to choose each pair of interfaces to be on the same NUMA.
Enter list of interfaces separated by space (for example: 1 3) : 2 3

For interface 2, assuming loopback to its dual interface 3.
Destination MAC is 3c:fd:fe:a6:6f:f1. Change it to MAC of DUT? (y/N).N
For interface 3, assuming loopback to its dual interface 2.
Destination MAC is 3c:fd:fe:a6:6f:f0. Change it to MAC of DUT? (y/N).N
Print preview of generated config? (Y/n)y
### Config file generated by dpdk_setup_ports.py ###

- version: 2
  interfaces: ['a6:00.0', 'a6:00.1']
  port_info:
      - dest_mac: 3c:fd:fe:a6:6f:f1 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  3c:fd:fe:a6:6f:f0
      - dest_mac: 3c:fd:fe:a6:6f:f0 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  3c:fd:fe:a6:6f:f1

  platform:
      master_thread_id: 0
      latency_thread_id: 1
      dual_if:
        - socket: 1
          threads: [10,11,12,13,14,15,16,17,18,19,30,31,32,33,34,35,36,37,38,39]


Save the config to file? (Y/n)y
Default filename is /etc/trex_cfg.yaml
Press ENTER to confirm or enter new file:
File /etc/trex_cfg.yaml already exist, overwrite? (y/N)y
Saved to /etc/trex_cfg.yaml.
```

图 2

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ cat /etc/trex_cfg.yaml
### Config file generated by dpdk_setup_ports.py ###

- version: 2
  interfaces: ['a6:00.0', 'a6:00.1']
  port_info:
      - dest_mac: 3c:fd:fe:a6:6f:f1 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  3c:fd:fe:a6:6f:f0
      - dest_mac: 3c:fd:fe:a6:6f:f0 # MAC OF LOOPBACK TO IT'S DUAL INTERFACE
        src_mac:  3c:fd:fe:a6:6f:f1

  platform:
      master_thread_id: 0
      latency_thread_id: 1
      dual_if:
        - socket: 1
          threads: [10,11,12,13,14,15,16,17,18,19,30,31,32,33,34,35,36,37,38,39]
```

图 3

进行 64Byte 最小报文的发包测试

先将 nic2 和 nic3 解绑（图 4），否则后面 nic0 和 nic1 做 dpdk 驱动绑定的时候，会提示冲突。

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ sudo ./dpdk_nic_bind.py -u 0000:a6:00.2
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ sudo ./dpdk_nic_bind.py -u 0000:a6:00.3
```

图 4

在 dell 服务器上执行如下命令：

sudo ./t-rex-64 -f cap2/imix_64_fast.yaml -m 20 -l 1000

其中，参数 m 是发包重放次数，此处是 20 倍，l 是网络抖动检测。

显示结果如图 5 所示。

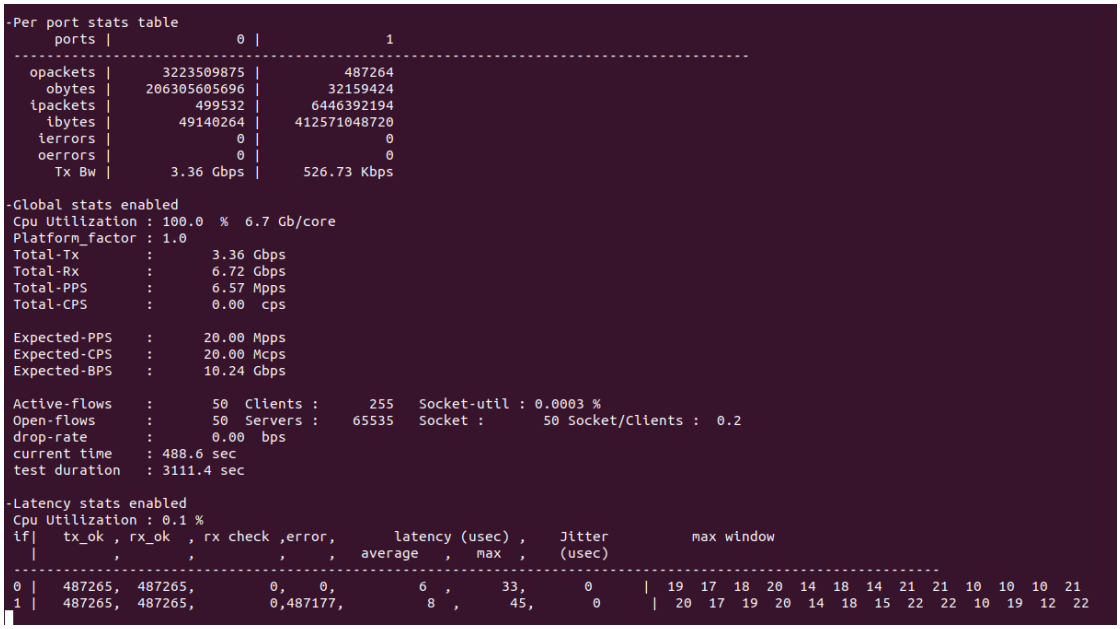


图 5

进行 1518Byte 最大报文的发包实验

在 dell 服务器上执行如下命令。变更-c 的参数，观察 CPU 利用率以及吞吐率。

sudo ./t-rex-64 -f cap2/imix_1518.yaml -m 120 -l 1000 -c 2

其中，参数 c 可以调整服务器上用来收发报文的核数，cap2/imix_1518.yaml 为包大小 1518B 的配置文件。

显示结果如图 6 所示。

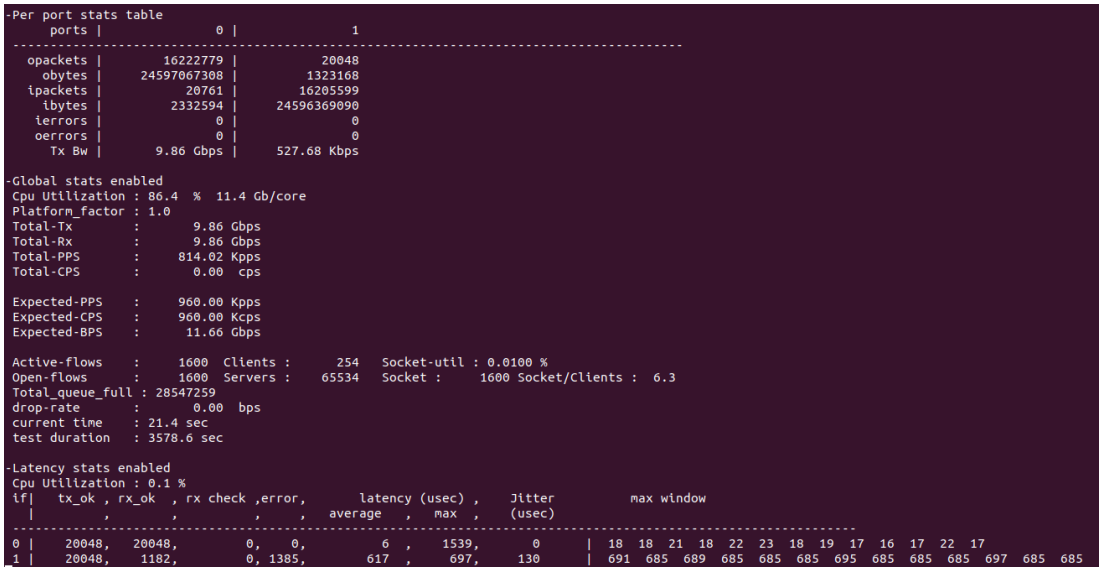


图 6

使用 iperf3 进行 Linux 内核发包

iperf 作为测量网络带宽的常用工具，由 Linux 内核驱动，分为客户端和服务器端两个部分，在 dell 服务器上我们开启客户端，在另一台浪潮服务器上开启服务器端，数据包由客户端发往服务器端。

在浪潮服务器上 ping 通 192.168.100.2，并开启 iperf3 服务器端：

```
wxg@inspur2:~$ iperf3 -s -B 192.168.100.5
```

图 7

以上命令中-s 表示 iperf3 开启服务器端，-B 表示绑定服务器 ip 地址，浪潮服务器上的 nic0 地址设置为 192.168.100.5，和 dell 服务器的网口属于同一个网段内，可以用 ifconfig 命令进行确认。

在 dell 服务器中，我们首先使用 ping 命令查看是否能够连接到浪潮服务器。

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
64 bytes from 192.168.100.5: icmp_seq=1 ttl=64 time=0.282 ms
64 bytes from 192.168.100.5: icmp_seq=2 ttl=64 time=0.120 ms
64 bytes from 192.168.100.5: icmp_seq=3 ttl=64 time=0.120 ms
64 bytes from 192.168.100.5: icmp_seq=4 ttl=64 time=0.110 ms
64 bytes from 192.168.100.5: icmp_seq=5 ttl=64 time=0.101 ms
64 bytes from 192.168.100.5: icmp_seq=6 ttl=64 time=0.099 ms
64 bytes from 192.168.100.5: icmp_seq=7 ttl=64 time=0.110 ms
64 bytes from 192.168.100.5: icmp_seq=8 ttl=64 time=0.104 ms
64 bytes from 192.168.100.5: icmp_seq=9 ttl=64 time=0.102 ms
64 bytes from 192.168.100.5: icmp_seq=10 ttl=64 time=0.103 ms
64 bytes from 192.168.100.5: icmp_seq=11 ttl=64 time=0.106 ms
64 bytes from 192.168.100.5: icmp_seq=12 ttl=64 time=0.106 ms
64 bytes from 192.168.100.5: icmp_seq=13 ttl=64 time=0.104 ms
64 bytes from 192.168.100.5: icmp_seq=14 ttl=64 time=0.106 ms
64 bytes from 192.168.100.5: icmp_seq=15 ttl=64 time=0.115 ms
^Z
[1]+  已停止               ping 192.168.100.5
```

图 8

确认连通以后，在 dell 服务器中开启 iperf3 客户端命令。命令中-c 表示开启客户端，目标地址为 192.168.100.2，-M 表示设置 MTU 值，-b 表示设置运行带宽，默认情况下带宽为 1M。

我们可以在 dell 服务器上看到客户端显示的测量数据：

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ iperf3 -c 192.168.100.5 -M 88 -B 192.168.100.2 -b 1000M
Connecting to host 192.168.100.5, port 5201
[  5] local 192.168.100.2 port 40365 connected to 192.168.100.5 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  81.8 MBytes   686 Mbits/sec  2382   28.4 KBytes
[  5]   1.00-2.00   sec  86.2 MBytes   724 Mbits/sec  3233   26.3 KBytes
[  5]   2.00-3.00   sec  87.5 MBytes   734 Mbits/sec  1321   37.5 KBytes
[  5]   3.00-4.00   sec  84.5 MBytes   709 Mbits/sec  2443   27.4 KBytes
[  5]   4.00-5.00   sec  79.6 MBytes   668 Mbits/sec  2630   34.2 KBytes
[  5]   5.00-6.00   sec   106 MBytes   892 Mbits/sec  4348   39.9 KBytes
[  5]   6.00-7.00   sec   120 MBytes  1.01 Gbits/sec  3929   73.1 KBytes
[  5]   7.00-8.00   sec  88.9 MBytes   746 Mbits/sec  2809   30.5 KBytes
[  5]   8.00-9.00   sec  94.8 MBytes   795 Mbits/sec  2448   34.1 KBytes
[  5]   9.00-10.00  sec  99.2 MBytes   832 Mbits/sec  3130   40.6 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec   929 MBytes   779 Mbits/sec  28673             sender
[  5]   0.00-10.04  sec   927 MBytes   774 Mbits/sec                  receiver

iperf Done.
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ iperf3 -c 192.168.100.5 -M 1000 -B 192.168.100.2 -b 10000M
Connecting to host 192.168.100.5, port 5201
[  5] local 192.168.100.2 port 33593 connected to 192.168.100.5 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  1.03 GBytes  8.81 Gbits/sec  225    456 KBytes
[  5]   1.00-2.00   sec  1.05 GBytes  9.04 Gbits/sec    0    456 KBytes
[  5]   2.00-3.00   sec  1.04 GBytes  8.93 Gbits/sec   24    451 KBytes
[  5]   3.00-4.00   sec  1.04 GBytes  8.97 Gbits/sec    0    508 KBytes
[  5]   4.00-5.00   sec  1.03 GBytes  8.82 Gbits/sec    0    539 KBytes
[  5]   5.00-6.00   sec  1.05 GBytes  9.04 Gbits/sec    0    552 KBytes
[  5]   6.00-7.00   sec  1.05 GBytes  9.06 Gbits/sec    0    552 KBytes
[  5]   7.00-8.00   sec  1.04 GBytes  8.91 Gbits/sec    6    437 KBytes
[  5]   8.00-9.00   sec  1.05 GBytes  9.05 Gbits/sec    0    446 KBytes
[  5]   9.00-10.00  sec  1.06 GBytes  9.07 Gbits/sec    0    465 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  10.4 GBytes  8.97 Gbits/sec  255             sender
[  5]   0.00-10.04  sec  10.4 GBytes  8.93 Gbits/sec                  receiver

iperf Done.
```

图 9

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ iperf3 -c 192.168.100.5 -M 500 -B 192.168.100.2 -b 10000M
Connecting to host 192.168.100.5, port 5201
[  5] local 192.168.100.2 port 46445 connected to 192.168.100.5 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   895 MBytes  7.51 Gbits/sec  1504    246 KBytes
[  5]   1.00-2.00   sec   927 MBytes  7.78 Gbits/sec  449     429 KBytes
[  5]   2.00-3.00   sec  1000 MBytes  8.39 Gbits/sec    0     429 KBytes
[  5]   3.00-4.00   sec  1000 MBytes  8.38 Gbits/sec    0     429 KBytes
[  5]   4.00-5.00   sec  1000 MBytes  8.38 Gbits/sec    0     429 KBytes
[  5]   5.00-6.00   sec   983 MBytes  8.25 Gbits/sec  303     377 KBytes
[  5]   6.00-7.00   sec   903 MBytes  7.57 Gbits/sec  1339    422 KBytes
[  5]   7.00-8.00   sec  1000 MBytes  8.39 Gbits/sec    0     433 KBytes
[  5]   8.00-9.00   sec   999 MBytes  8.38 Gbits/sec    0     455 KBytes
[  5]   9.00-10.00  sec   990 MBytes  8.31 Gbits/sec  311     244 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  9.47 GBytes  8.13 Gbits/sec  3906             sender
[  5]   0.00-10.04  sec  9.47 GBytes  8.10 Gbits/sec                  receiver

iperf Done.
```

图 10

```
seuguest@Dell-Precision-7920-Tower:~/trex/v2.89$ iperf3 -c 192.168.100.5 -M 1500 -B 192.168.100.2 -b 10000M
Connecting to host 192.168.100.5, port 5201
[  5] local 192.168.100.2 port 56377 connected to 192.168.100.5 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  1.07 GBytes  9.23 Gbits/sec   14    503 KBytes
[  5]   1.00-2.00   sec  1.09 GBytes  9.39 Gbits/sec    0    525 KBytes
[  5]   2.00-3.00   sec  1.07 GBytes  9.16 Gbits/sec    0    566 KBytes
[  5]   3.00-4.00   sec  1.08 GBytes  9.24 Gbits/sec    0    622 KBytes
[  5]   4.00-5.00   sec  1.09 GBytes  9.39 Gbits/sec    0    622 KBytes
[  5]   5.00-6.00   sec  1.09 GBytes  9.38 Gbits/sec    0    622 KBytes
[  5]   6.00-7.00   sec  1.08 GBytes  9.25 Gbits/sec    0    652 KBytes
[  5]   7.00-8.00   sec  1.09 GBytes  9.39 Gbits/sec    0    652 KBytes
[  5]   8.00-9.00   sec  1.09 GBytes  9.39 Gbits/sec    0    652 KBytes
[  5]   9.00-10.00  sec  1.09 GBytes  9.39 Gbits/sec    0    652 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  10.9 GBytes  9.32 Gbits/sec   14             sender
[  5]   0.00-10.04  sec  10.8 GBytes  9.28 Gbits/sec                  receiver

iperf Done.
```

图 11

同时，在浪潮服务器上，服务器端也出现了测量数据：

```
wxg@inspur2:~$ iperf3 -s -B 192.168.100.5
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.100.2, port 48851
[  5] local 192.168.100.5 port 5201 connected to 192.168.100.2 port 40365
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  76.7 MBytes   643 Mbits/sec
[  5]   1.00-2.00   sec  87.2 MBytes   730 Mbits/sec
[  5]   2.00-3.00   sec  87.2 MBytes   733 Mbits/sec
[  5]   3.00-4.00   sec  84.2 MBytes   706 Mbits/sec
[  5]   4.00-5.00   sec  78.0 MBytes   655 Mbits/sec
[  5]   5.00-6.00   sec   105 MBytes   885 Mbits/sec
[  5]   6.00-7.00   sec   121 MBytes  1.01 Gbits/sec
[  5]   7.00-8.00   sec  88.8 MBytes   745 Mbits/sec
[  5]   8.00-9.00   sec  96.3 MBytes   806 Mbits/sec
[  5]   9.00-10.00  sec  97.5 MBytes   819 Mbits/sec
[  5]  10.00-10.04  sec  4.59 MBytes   978 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.04  sec   927 MBytes   774 Mbits/sec                  receiver
-----------------------------------------------------------
```

图 12

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.100.2, port 37811
[  5] local 192.168.100.5 port 5201 connected to 192.168.100.2 port 33593
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1004 MBytes  8.42 Gbits/sec
[  5]   1.00-2.00   sec  1.05 GBytes  9.04 Gbits/sec
[  5]   2.00-3.00   sec  1.04 GBytes  8.93 Gbits/sec
[  5]   3.00-4.00   sec  1.04 GBytes  8.97 Gbits/sec
[  5]   4.00-5.00   sec  1.04 GBytes  8.91 Gbits/sec
[  5]   5.00-6.00   sec  1.04 GBytes  8.95 Gbits/sec
[  5]   6.00-7.00   sec  1.06 GBytes  9.07 Gbits/sec
[  5]   7.00-8.00   sec  1.04 GBytes  8.97 Gbits/sec
[  5]   8.00-9.00   sec  1.05 GBytes  8.99 Gbits/sec
[  5]   9.00-10.00  sec  1.06 GBytes  9.07 Gbits/sec
[  5]  10.00-10.04  sec  43.9 MBytes  9.12 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.04  sec  10.4 GBytes  8.93 Gbits/sec                  receiver
-----------------------------------------------------------
```

图 13

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.100.2, port 40789
[  5] local 192.168.100.5 port 5201 connected to 192.168.100.2 port 46445
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   861 MBytes  7.22 Gbits/sec
[  5]   1.00-2.00   sec   919 MBytes  7.71 Gbits/sec
[  5]   2.00-3.00   sec  1000 MBytes  8.38 Gbits/sec
[  5]   3.00-4.00   sec  1000 MBytes  8.38 Gbits/sec
[  5]   4.00-5.00   sec  1000 MBytes  8.38 Gbits/sec
[  5]   5.00-6.00   sec   985 MBytes  8.26 Gbits/sec
[  5]   6.00-7.00   sec   902 MBytes  7.56 Gbits/sec
[  5]   7.00-8.00   sec  1000 MBytes  8.38 Gbits/sec
[  5]   8.00-9.00   sec   999 MBytes  8.38 Gbits/sec
[  5]   9.00-10.00  sec   999 MBytes  8.38 Gbits/sec
[  5]  10.00-10.04  sec  29.9 MBytes  6.22 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.04  sec  9.47 GBytes  8.10 Gbits/sec                  receiver
-----------------------------------------------------------
```

图 14

```
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.100.2, port 50737
[  5] local 192.168.100.5 port 5201 connected to 192.168.100.2 port 56377
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  1.03 GBytes  8.84 Gbits/sec
[  5]   1.00-2.00   sec  1.09 GBytes  9.39 Gbits/sec
[  5]   2.00-3.00   sec  1.07 GBytes  9.16 Gbits/sec
[  5]   3.00-4.00   sec  1.08 GBytes  9.24 Gbits/sec
[  5]   4.00-5.00   sec  1.09 GBytes  9.39 Gbits/sec
[  5]   5.00-6.00   sec  1.09 GBytes  9.38 Gbits/sec
[  5]   6.00-7.00   sec  1.08 GBytes  9.24 Gbits/sec
[  5]   7.00-8.00   sec  1.09 GBytes  9.39 Gbits/sec
[  5]   8.00-9.00   sec  1.09 GBytes  9.39 Gbits/sec
[  5]   9.00-10.00  sec  1.09 GBytes  9.39 Gbits/sec
[  5]  10.00-10.04  sec  44.6 MBytes  9.38 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.04  sec  10.8 GBytes  9.28 Gbits/sec                  receiver
-----------------------------------------------------------
```

图 15

实验数据及分析

1、理论分析帧长对于吞吐率和包传输率的关系，并根据实验得到在 10Gbps 线速的链路中的吞吐率（bit per second）和发包速率（packet per second）的曲线图。

计算某帧长下吞吐率和包传输率的公式如下：

$$\mathrm{PPS} = \frac{\text{传输速率 bit per second}}{(\text{帧长 byte} + 20\text{byte}) * 8\text{bit}} \tag{1}$$

$$\mathrm{Throughput} = \mathrm{PPS} * \text{帧长 byte} * 8\text{bit} \tag{2}$$

理论及实验数据如表 1-2 所示，根据表 1-2 绘制出 10Gbps 线速的链路中吞吐率和发包速率的曲线图如图 16-17 所示。

表 1 理论数据

帧长度 Byte	吞吐率 Gbps	发包速率 pps
64	7.62	14.88M
594	9.67	2.04M
1518	9.87	812.74K

表 2 实验数据

帧长度 Byte	吞吐率 Gbps	发包速率 pps
64	6.8	13.27M
594	9.65	2.03M
1518	9.86	814.02K

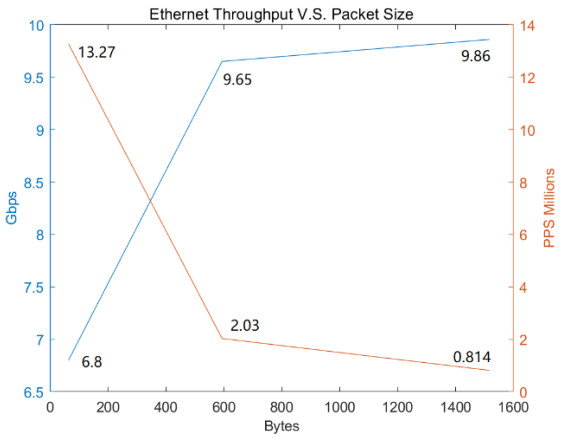


图 16

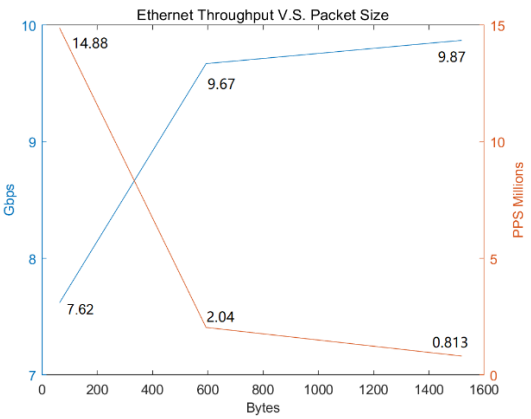
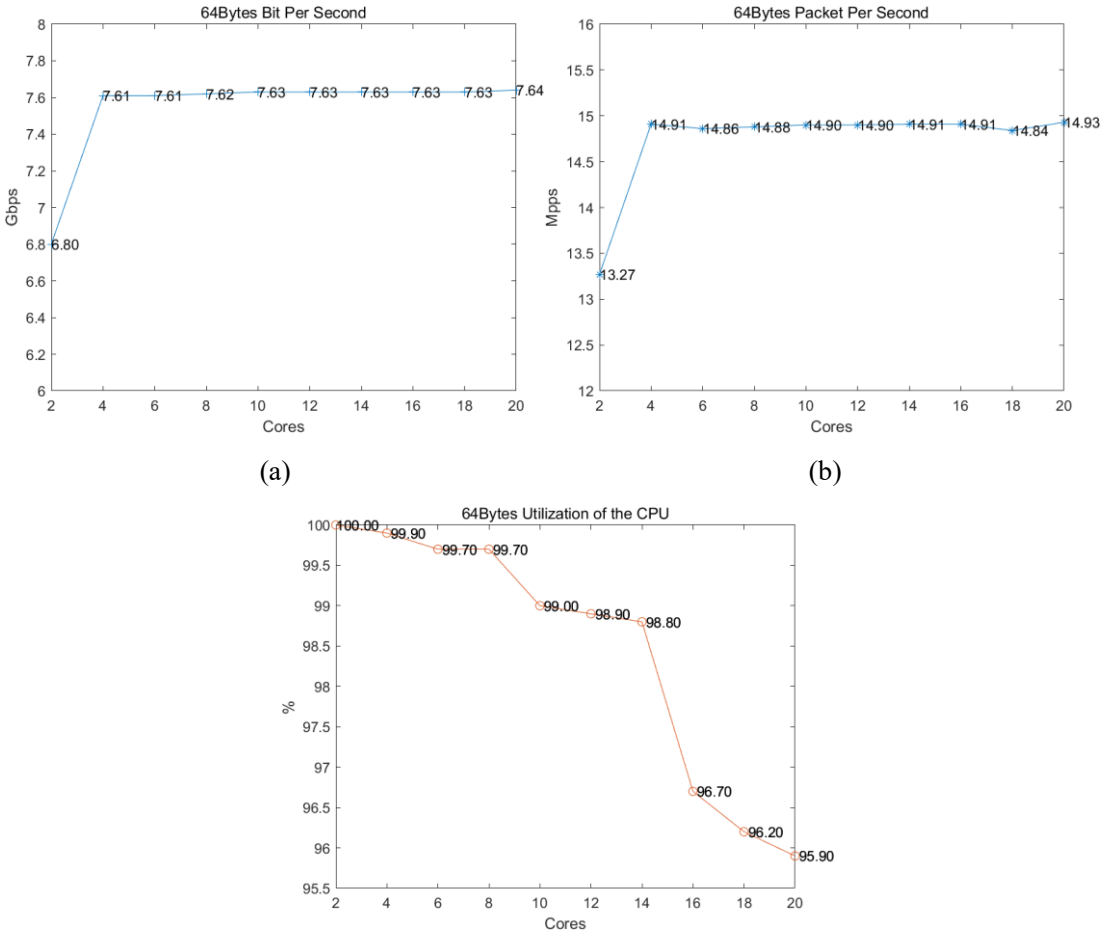


图 17

无论是理论分析还是实验测试都显示，在链路的比特率一定的情况下，吞吐率会随着帧长增加，而发包速率会随着帧长下降。发包速率的变化很好理解，帧长越长，在单位时间内发出的包数就越少；而帧长越长，首部（固定长度）占数据帧的比例就越低，吞吐率就越高。

2、通过实验，构造不同长度的数据包，得到不同的 CPU 核心数情况下，吞吐率和 CPU 的利用率的曲线图。（注：吞吐率包括 bit per second 和 packet per second）

64Byte 数据包在不同 CPU 核心数下的吞吐率和 CPU 利用率曲线图如图 18 所示。

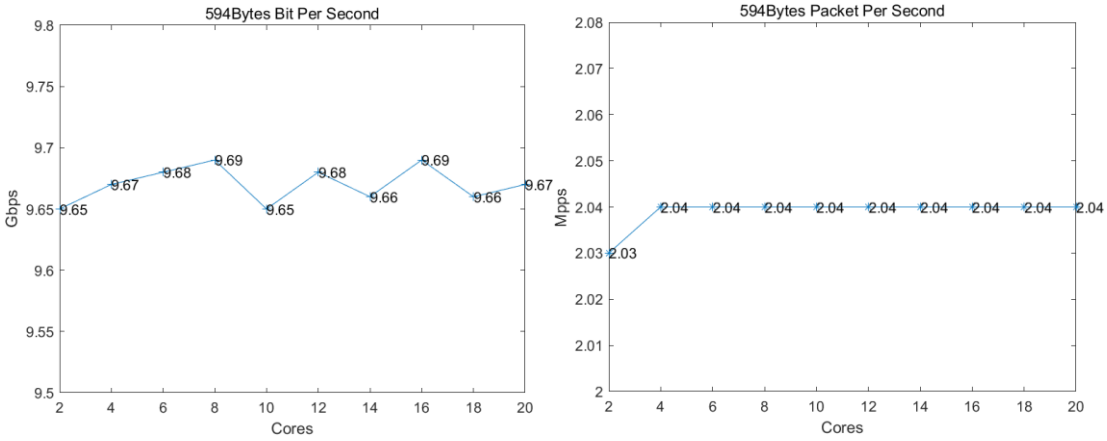


(a) (b)

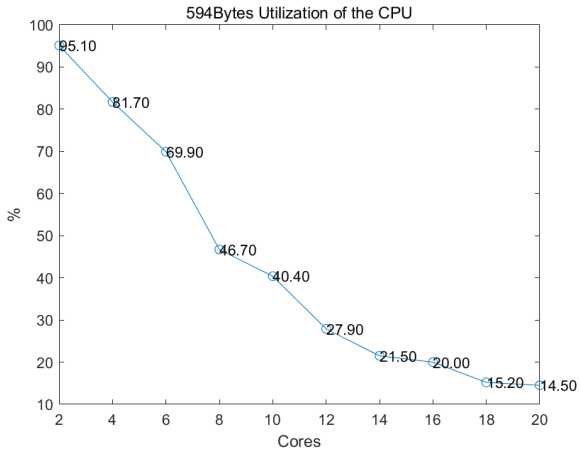
(c)

图 18

594Byte 数据包在不同 CPU 核心数下的吞吐率和 CPU 利用率曲线图如图 19 所示。



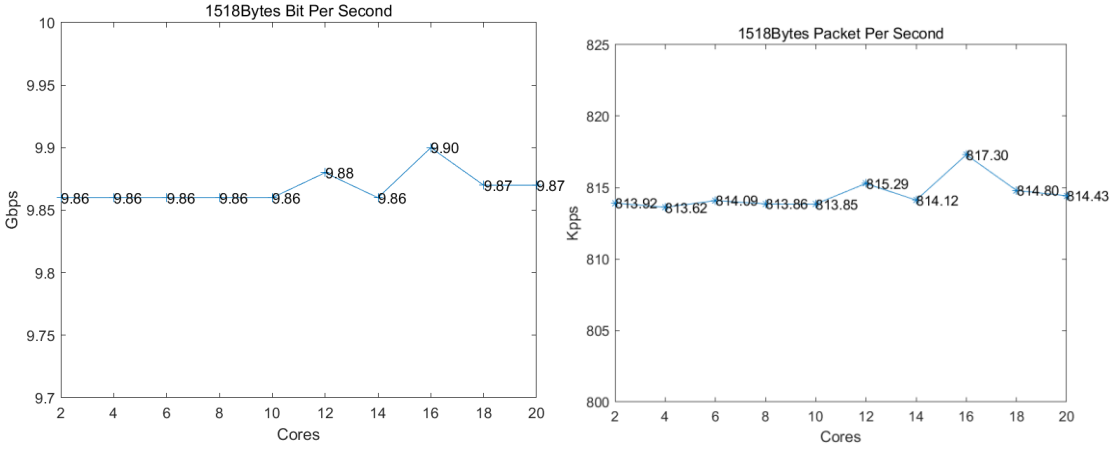
(a) (b)



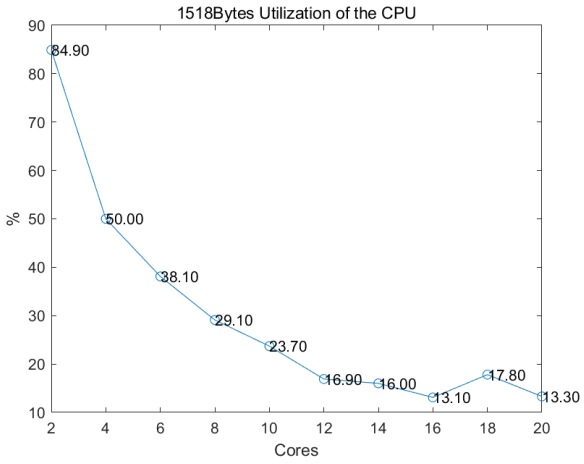
(c)

图 19

1518Byte 数据包在不同 CPU 核心数下的吞吐率和 CPU 利用率曲线图如图 20 所示。



(a) (b)



(c)

图 20

从三组数据中我们可以看出，在链路的比特率一定的情况下，吞吐率会随着核数的增加而上升，但是 CPU 利用率会随着核数的增加而下降。随着 CPU 核数的增加，每个 CPU 核的利用率都会有所下降，而吞吐率会因为参与处理的 CPU 核数增多而上升。

3、通过实验，分别测试 dpdk 和采用 Linux 内核 I/O 的方式下的吞吐量，并画出对比曲线图。

dpdk 和采用 Linux 内核 I/O 的方式下的吞吐量对比图如图 21 所示。

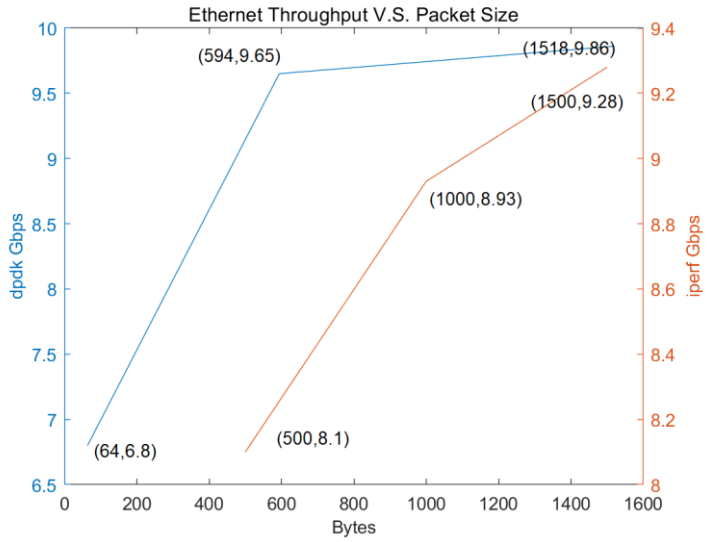


图 21

可见无论是 dpdk 还是采用 Linux 内核 I/O 方式，吞吐量随着帧长的变化趋势都是一样的，然而采用 Linux 内核 I/O 方式的吞吐量要整体低于 dpdk。造成这种现象的原因主要有：①使用 Linux 内核的传统收发报文方式都必须采用硬中断来做通讯，数据必须在内核态和用户态之间切换拷贝带来大量 CPU 消耗；②全局锁竞争；③收发包都有系统调用的开销；④内核工作在多核上，即使采用 Lock Free，也避免不了锁总线、内存屏障带来的性能损耗。

实验总结

本次实实验的具体操作并不困难，感谢助教的指导和帮助。主要难点在于实验数据的处理和分析，我们在测量数据时出现了一些技术问题，导致最后呈现出来的曲线存在轻微的抖动。