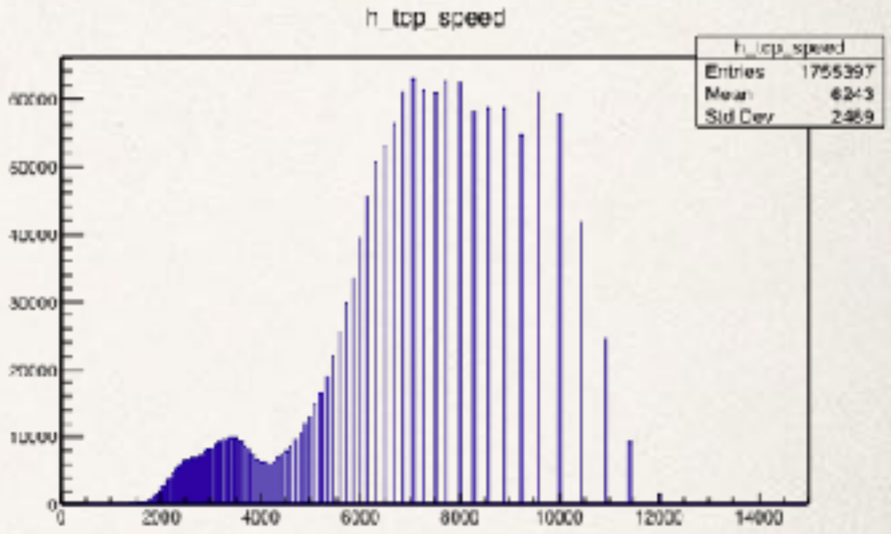
tcp_test_20170619

Date

Outline: further investigations based on the feedback

- mini_daq bug fixes: spike & memcpy bugs
- iperf+UDP for data loss/data collision study
- Use jumbo frame for TCP;
- TCP congestion control algorithm

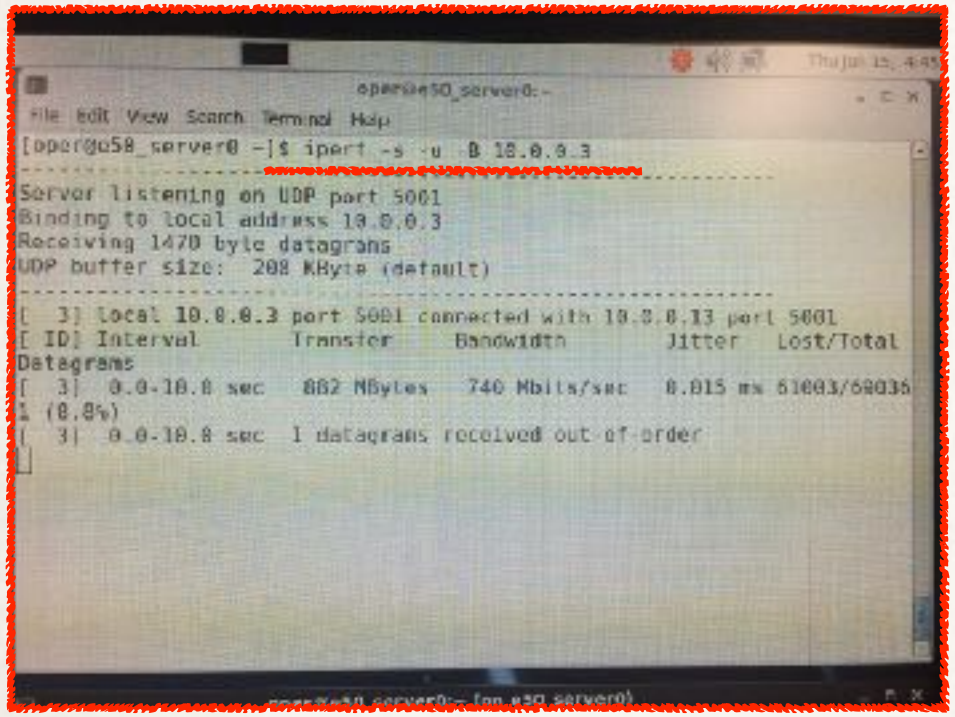
mini_daq bug fixes



- Online analyzer modification:
  - Use nano second time stamp to “cure” the spikes
  - Use recorder_thrd.c to fill TCP speed histogram to avoid local ethernet throughput
  - Handling dynamic data_length

iperf+UDP for data loss/data collision study

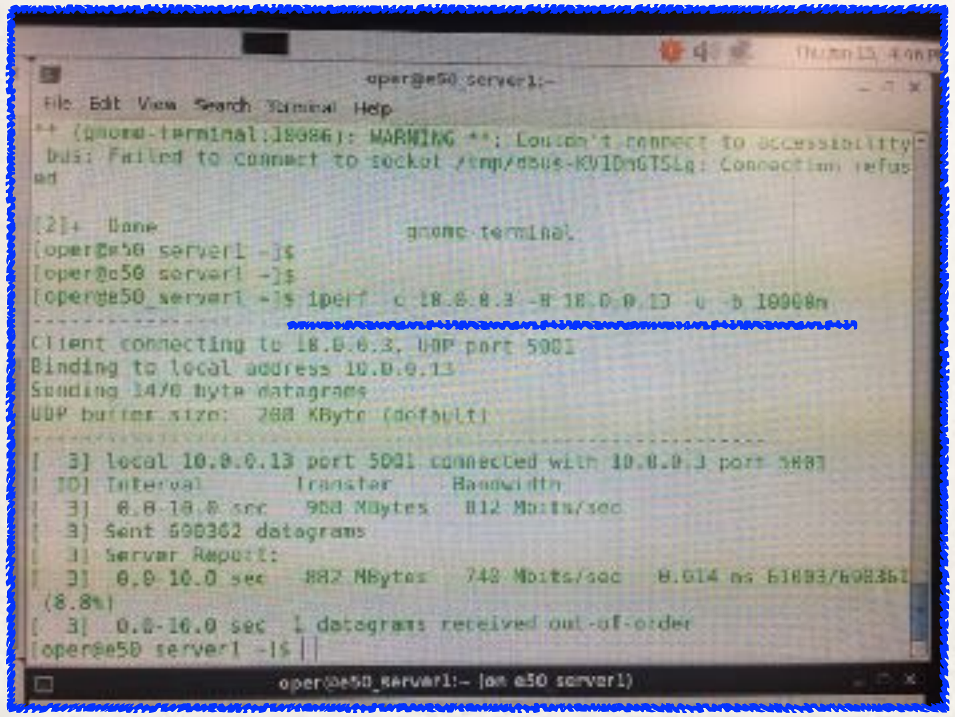
iperf server setup



Thu Jun 15, 4:45
oper@e50_server0:~
File Edit View Search Terminal Help
oper@e50_server0:~ (on e50 server0)

```
[oper@e50_server0 ~]$ iperf -s -u -B 10.0.0.3
------------------------------------------------------------
Server listening on UDP port 5001
Binding to local address 10.0.0.3
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.3 port 5001 connected with 10.0.0.13 port 5001
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total
Datagrams
[  3]  0.0-10.0 sec   882 MBytes   740 Mbits/sec   0.015 ms 61003/69036
1 (8.8%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
```

iperf client setup



Thu Jun 15, 4:06 PM
oper@e50_server1:~
File Edit View Search Terminal Help
oper@e50_server1:~ (on e50 server1)

```
** (gnome-terminal:18086): WARNING **: Couldn't connect to accessibility
 bus: Failed to connect to socket /tmp/dbus-KV1DnGTSLq: Connection refus
ed

[2]+  Done                    gnome-terminal
[oper@e50_server1 ~]$
[oper@e50_server1 ~]$
[oper@e50_server1 ~]$ iperf -c 10.0.0.3 -B 10.0.0.13 -u -b 10000m
------------------------------------------------------------
Client connecting to 10.0.0.3, UDP port 5001
Binding to local address 10.0.0.13
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.13 port 5001 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   968 MBytes   812 Mbits/sec
[  3] Sent 690362 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec   882 MBytes   740 Mbits/sec   0.014 ms 61003/690361
 (8.8%)
[  3]  0.0-10.0 sec  1 datagrams received out-of-order
[oper@e50_server1 ~]$
```

Band width parameter seems to be most important:

1, by setting "-b 10000m", the data loss is ~10% no matter four pair connections or single connection

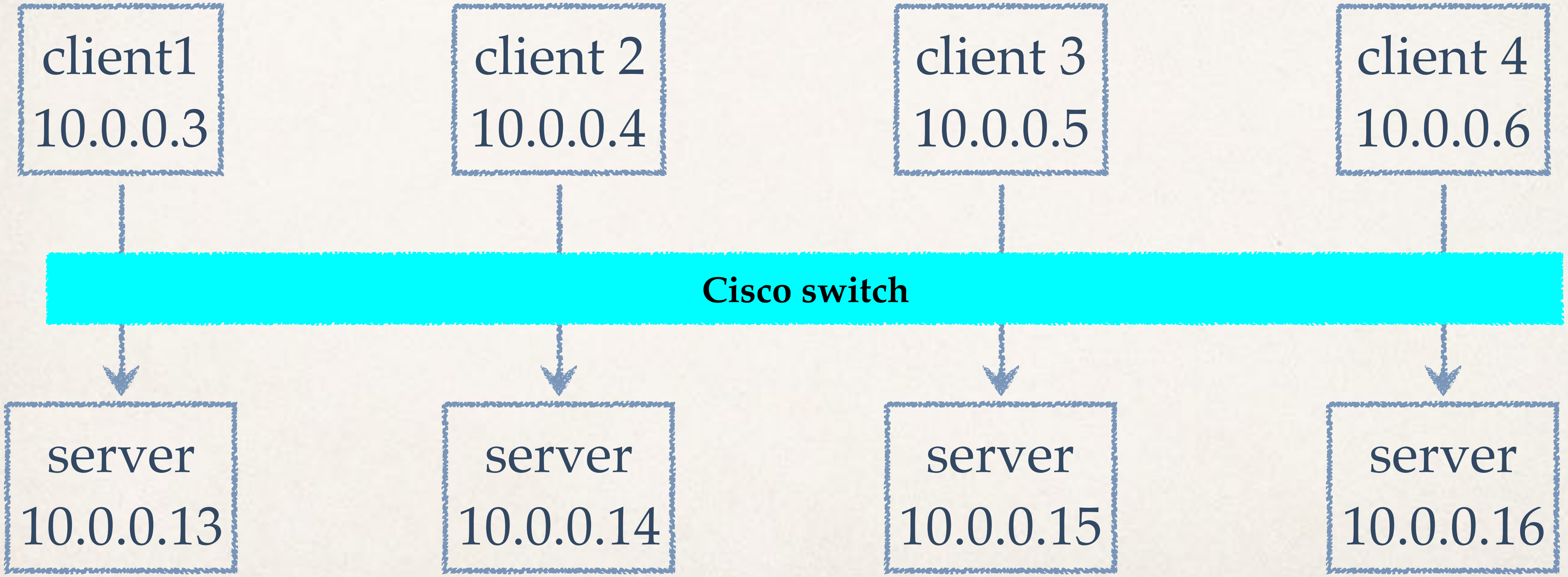
2, by setting "-b 1000m", the data loss is ~3% no matter four pair connections or single connection

conclusion: *the current iperf version doesn't support 10gbps??*

(iperf version 2.0.5)

Test configuration

Use the following configuration for test

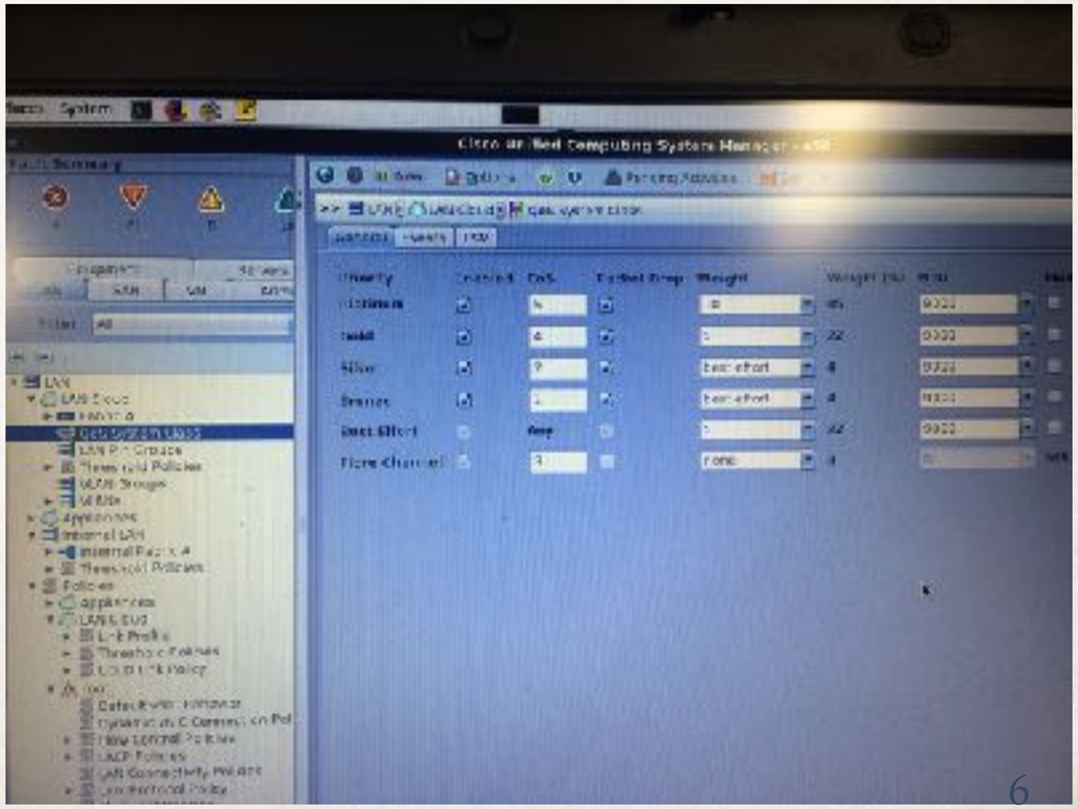


Select Jumbo fram (9000Bytes/frame)

- From E50 server0 and server1, select Jumbo frame with
  [root@e50_server0 oper]# ifconfig ens6f3 mtu 9000

- Confirm the change with [root@e50_server0 oper]# ifconfig ens6f3

- Also configure Cisco UCS 6120 for Jumbo frame



Cisco Unified Computing System Manager - e50

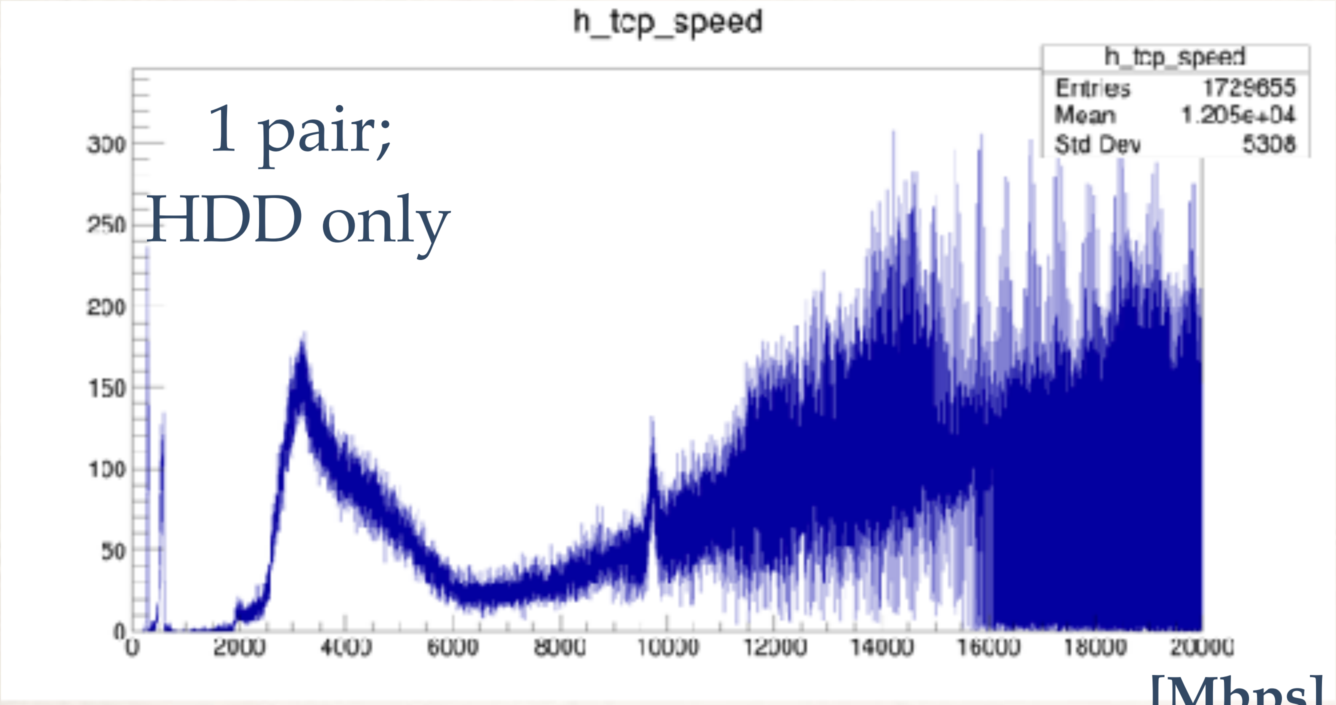
>> LAN > LAN Cloud > QoS System Class

Priority	Enabled	CoS	Packet Drop	Weight	Weight (%)	MTU
Platinum	☑	5	☑	10	45	9000
Gold	☑	4	☑	5	22	9000
Silver	☑	2	☑	best-effort	4	9000
Bronze	☑	1	☑	best-effort	4	9000
Best Effort	□	Any	□	5	22	9000
Fibre Channel	□	3	□	none	4	fc

TCP congestion algorithm

- Check available module: ls /lib/modules/`uname -r`/kernel/net/ipv4/
- Load module: /sbin/modprobe tcp_htcp
- To check the default congestion algorithm: sysctl net.ipv4.tcp_congestion_control
  - results obtained so far are based on default "cubic" algorithm
- To check the control algorithm allowed: sysctl net.ipv4.tcp_allowed_congestion_control
- To set the control algorithm: sysctl -w net.ipv4.tcp_congestion_control=reno

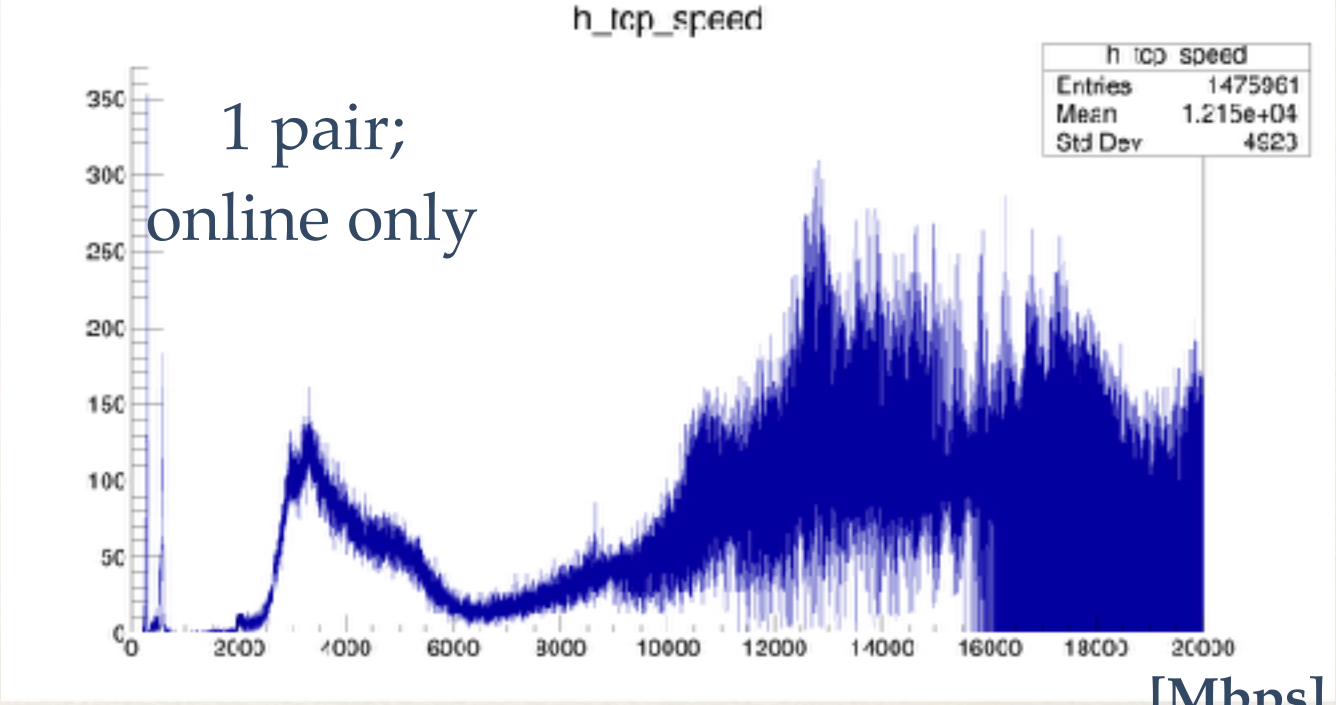
Results



1 pair;
HDD only

[Mbps]

packet = 30kB, 10k buffer, Jumbo frame, congestion control = “highspeed”

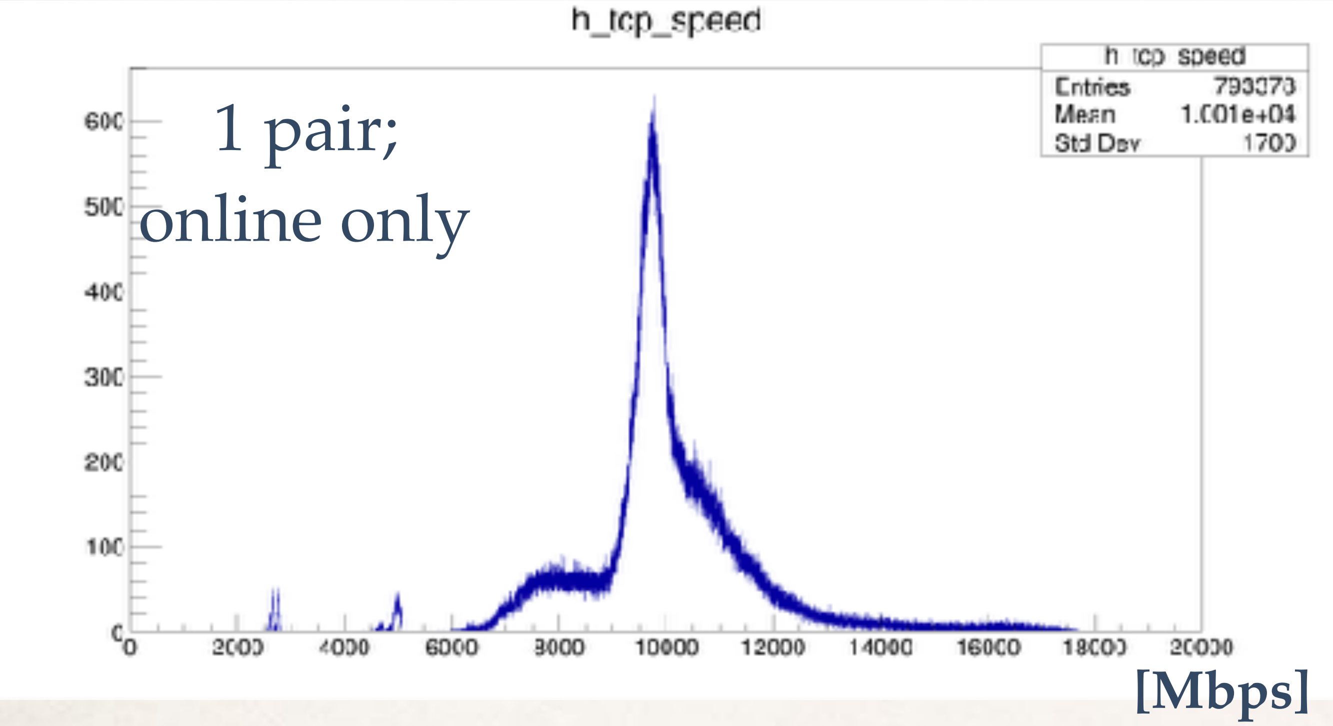


1 pair;
online only

[Mbps]

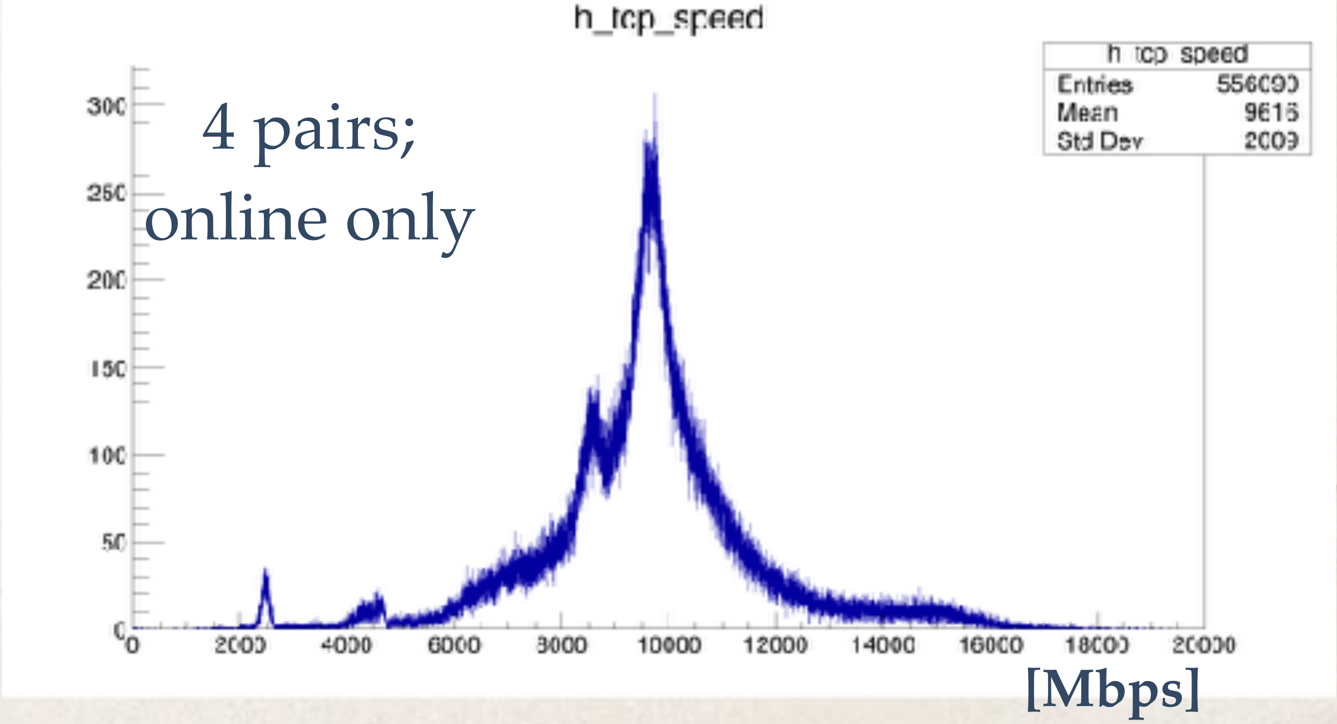
No serious overhead found; use online histogram to evaluate TCP speed

Results



1 pair;
online only

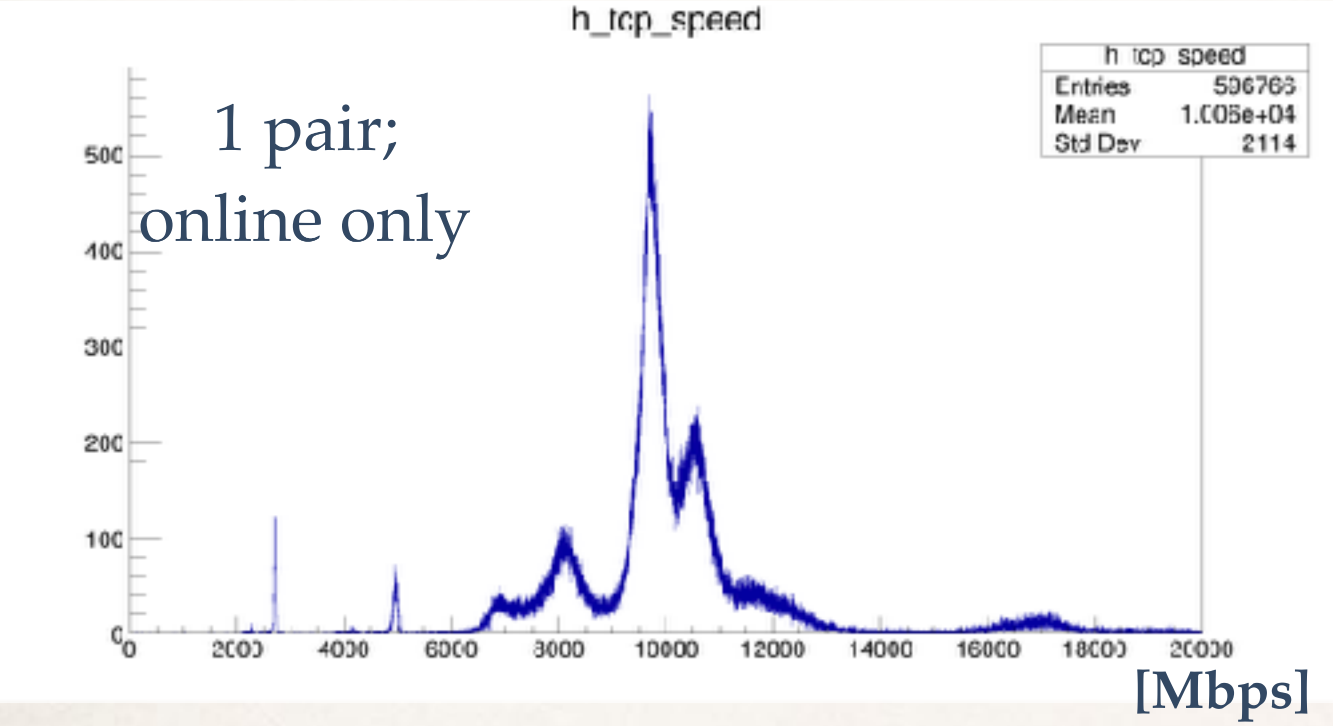
packet = 300kB, 10k buffer, Jumbo frame, congestion control = “highspeed”



4 pairs;
online only

performance converged;
use two Gauss for P.D.F?

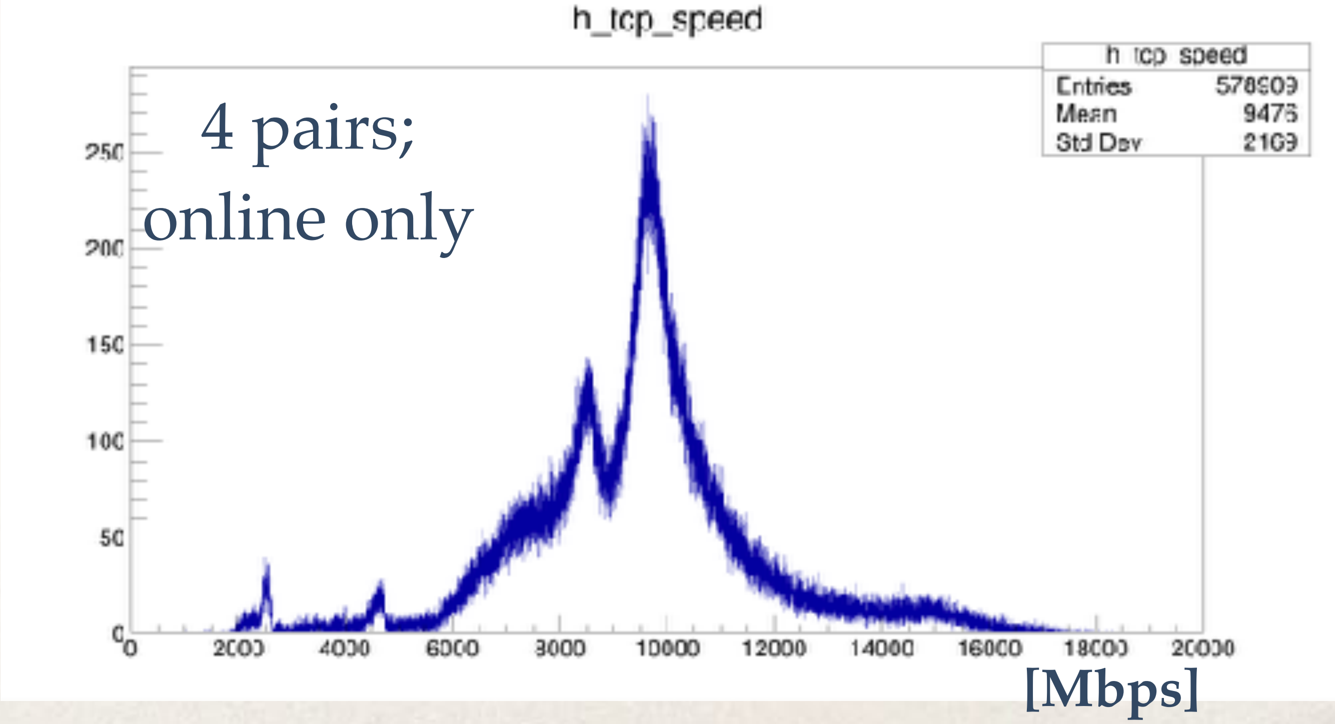
Results



1 pair;
online only

[Mbps]

packet = 300kB, 10k buffer, Jumbo frame, congestion control = "cubic"



4 pairs;
online only

[Mbps]

performance converged;
use two Gauss for P.D.F?

Summary & todo

- Updated TCP speed histograms provide more reliable information
- Jumbo frame slightly improves the performance
- Congestion algorithm seems not very effective
- Packet size is most critical for a good performance: accumulate ~300kB before sending to TCP buffer
- Use two Gauss distribution to represent TCP P.D.F?
    - one for Linux timestamp resolution; another for TCP speed fluctuation??

- Bigger buffer? kernel TCP tuning?
- P.D.F. vs. date rate transmission?