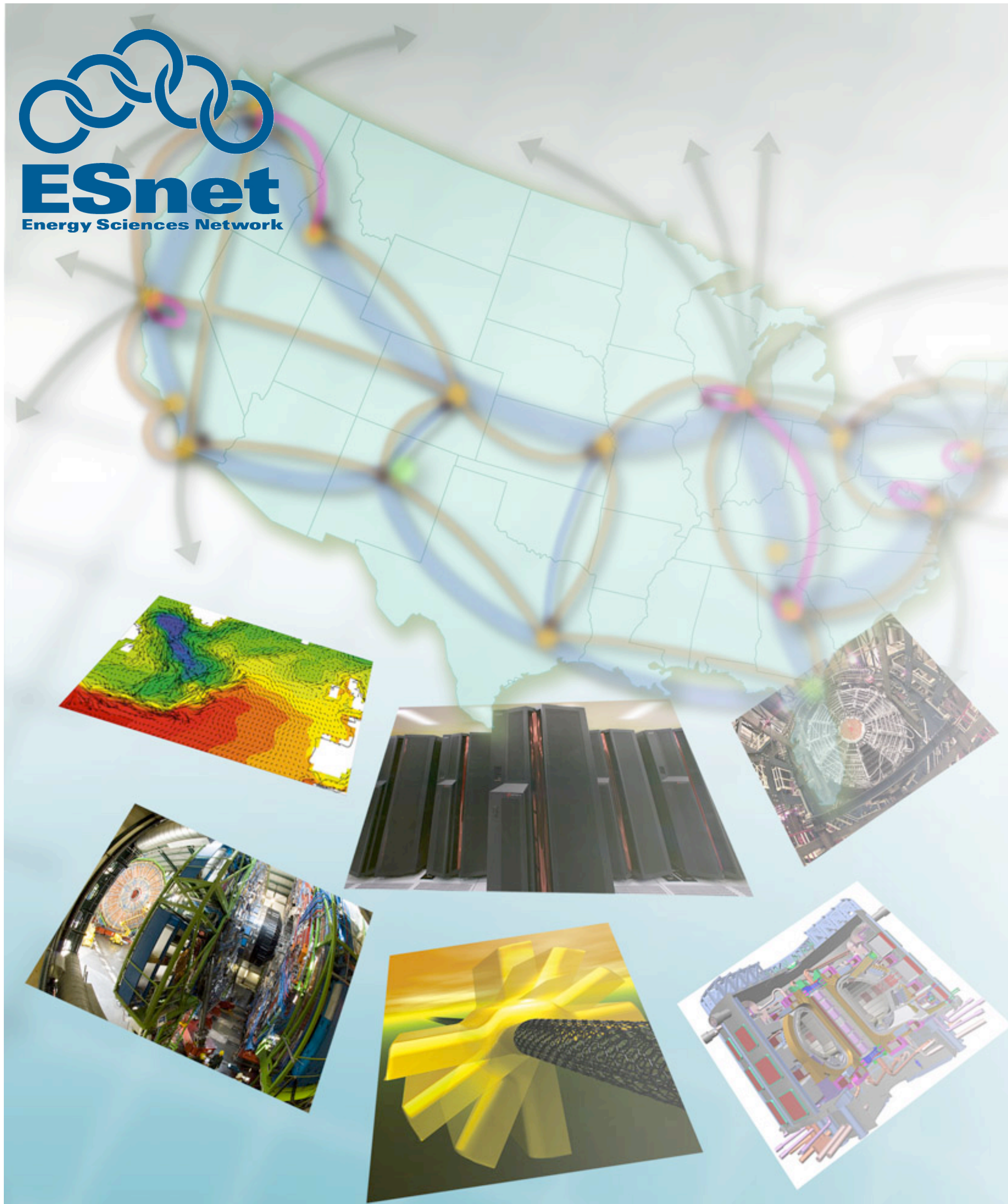


Using Iperf

Jon M. Dugan
<jdugan@es.net>

Energy Sciences Network
Lawrence Berkeley National Laboratory

NANOG 43, Brooklyn, NY
June 1, 2008

Networking for the Future of Science

Outline

- TCP Measurements
- UDP Measurements
- Useful tricks

Iperf’s notion of clients and servers

Client is the sender

Server is the receiver
(discard server)

TCP Measurements

- **Measures TCP Achievable Bandwidth**
  - Measurement includes the end system
  - Sometimes called “memory-to-memory” tests
- **Limits of what we can measure**
  - TCP is a largely a black box
- **Many things can limit TCP throughput**
  - Loss
  - Congestion
  - Buffer Starvation
  - Out of order delivery

Example Iperf TCP Invocation

Server (receiver):

```
$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.1.5 port 5001 connected with 10.0.1.10 port 60830
[  4]  0.0-10.0 sec  1.09 GBytes    933 Mbits/sec
[  4] local 10.0.1.5 port 5001 connected with 10.0.1.10 port 60831
[  4]  0.0-10.0 sec  1.08 GBytes    931 Mbits/sec
```

Client (sender):

```
$ iperf -c 10.0.1.5
------------------------------------------------------------
Client connecting to 10.0.1.5, TCP port 5001
TCP window size:  129 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.10 port 60830 connected with 10.0.1.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.2 sec  1.09 GBytes   913 Mbits/sec
```

Bandwidth Delay Product

- The amount of “in flight” data allowed for a TCP connection
- BDP = bandwidth * round trip time
- Example: 1Gb/s cross country, ~100ms

  1,000,000,000 b/s * .1 s = 100,000,000 bits

  100,000,000 / 8 = 12,500,000 bytes

  12,500,000 bytes / (1024*1024) ~ 12MB
- To get full TCP performance the TCP window needs to be large enough to accommodate the Bandwidth Delay Product

UDP Measurements

- **UDP provides greater transparency**
- **We can directly measure some additional things:**
  - Loss
  - Jitter
  - Out of order delivery

Example Iperf UDP Invocation

Server (receiver):

```
$ iperf -u -s
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:   107 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.5 port 5001 connected with 10.0.1.10 port 65299
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.008 ms    0/  893 (0%)
```

Client (sender):

```
$ iperf -u -c 10.0.1.5 -b 1M
------------------------------------------------------------
Client connecting to 10.0.1.5, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 9.00 KByte (default)
------------------------------------------------------------
[  3] local 10.0.1.10 port 65300 connected with 10.0.1.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec  0.003 ms    0/  893 (0%)
[  3] Sent 893 datagrams
```

Adjusting Iperf for performance

- **The –w option for Iperf can be used to request a particular buffer size. This sets both send and receive buffer size.**
  - The OS may need to be tweaked to allow buffers of sufficient size.
  - See http://dsd.lbl.gov/TCP-tuning/ and http://www.psc.edu/networking/perf_tune.html
- **Parallel transfers may help as well, the –P option can be used for this**

Useful Iperf Invocations

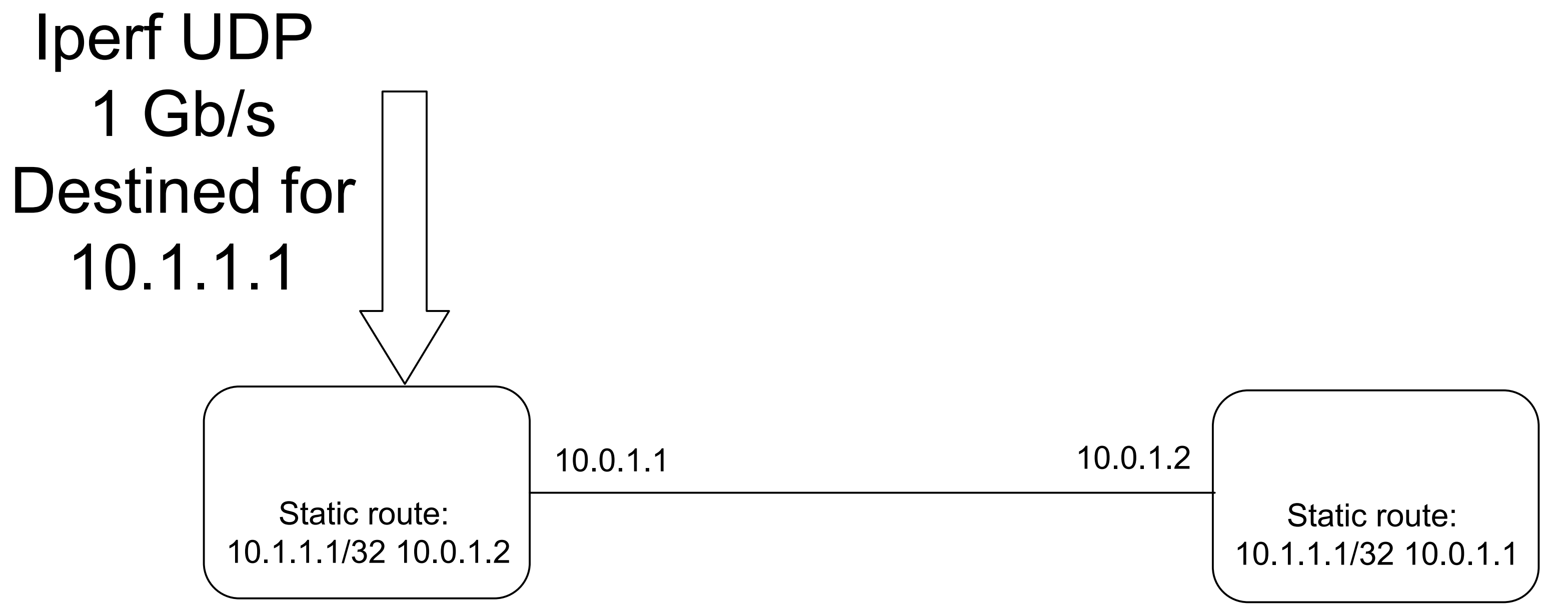
- **UDP and TCP:**
  - -i *n* report status every *n* seconds
  - -d do bidirectional test simultaneously
  - -r do bidirectional test one after another

Using Iperf to generate high rate streams

- UDP doesn't require a receiver
- If you have good counters on your switches & routers those can be used to measure
- Turns out UDP reception can be very resource intensive resulting in drops at the NIC at high rates (8-9 Gb/s)

Never do this

- **Need to generate 10 Gb/s but only have a 1 Gb/s host?**



Use the –T option to Iperf to control
the number of times the traffic loops
Can also use firewall filters to discard a certain TTL range.
Other filters may be prudent as well.

Iperf Development

- **Primarily in maintenance mode**
  - Accepting and apply patches
  - Fixing bugs and documentation as time allows

- **Future Directions**
  - libiperf

More Information

http://iperf.sourceforge.net

iperf-users@lists.sourceforge.net

You can reach me at:

Jon Dugan <jdugan@es.net>