March 10th 2011, OSG All Hands Meeting, Network Performance

Jason Zurawski – Internet2

BWCTL

Agenda

- Tutorial Agenda:
  - Network Performance Primer - Why Should We Care? (**15 Mins**)
  - Getting the Tools (**10 Mins**)
  - Use of the BWCTL Server and Client (**30 Mins**)
  - Use of the OWAMP Server and Client (**30 Mins**)
  - Use of the NDT Server and Client (**30 Mins**)
  - Diagnostics vs Regular Monitoring (**30 Mins**)
  - Network Performance Exercises (**1 hr 30 Mins**)

BWCTL – What is it?

- BWCTL is:
  - A command line client application
  - A scheduling and policy daemon
  - Wraps the throughput testing tools Iperf and Nuttcp.
- These tests are able to measure:
  - Maximum TCP bandwidth (with various tuning options available)
  - The delay, jitter, and datagram loss of a network when doing a UDP test

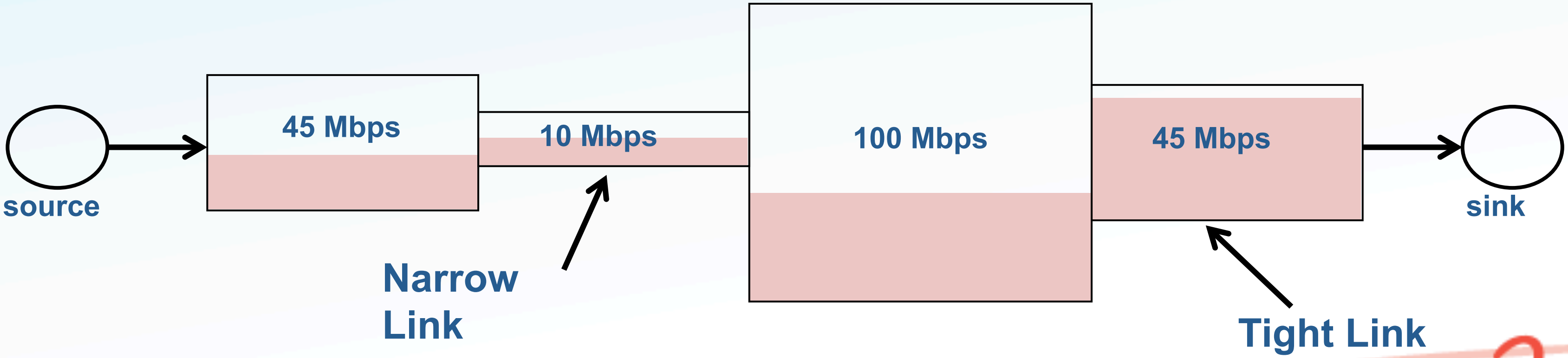
Problem Statement

- Users want to verify available bandwidth/throughput:
  - Between their site and a remote resource
  - Between two remote resources
  - Validate/Verify an SLA

- Methodology:
  - Verify available bandwidth from each endpoint to points in the middle
  - Determine problem area(s)
  - Re-run tests over time – requires access to tool instead of doing a 'one off' test

Throughput? Bandwidth? What?

- The term "throughput" is vague
  - Capacity: link speed
    - Narrow Link: link with the lowest capacity along a path
    - Capacity of the end-to-end path = capacity of the narrow link
  - Utilized bandwidth: current traffic load
  - Available bandwidth: capacity – utilized bandwidth
    - Tight Link: link with the least available bandwidth in a path
  - Achievable bandwidth: includes protocol and host issues



(Shaded portion shows background traffic)

Typical Solution

- Run “iperf” or similar tool on two endpoints and hosts on intermediate paths
  - Roadblocks:
    - Need software on all test systems
    - Need permissions on all systems involved (usually full shell accounts*)
    - Need to coordinate testing with others *
    - Need to run software on both sides with specified test parameters *
- Desirable features for an alternate method
  - ‘Daemon’ to run in the background
  - Protocol to exchange results/errors
  - Works with firewalls
  - Protect resources

- (* BWCTL was designed to help with these)

Implementation

- Applications
  - Daemon (bwctld)
  - Client (bwctl)
- Open Source License & Development
  - Modified BSD (http://www.internet2.edu/membership/ip.html)
  - Mailing lists for developer communication – come join us!
- Protocol Abstraction Library
  - Will support development of new clients
  - Add custom 'hooks' into the policy (e.g. add authentication via OpenID or similar)

Server Functionality (bwctld)

- bwctld on each test host
  - Accepts requests for “iperf” tests including time slot and parameters for test
  - Responds with a tentative reservation or a denied message
  - Reservations by a client must be confirmed with a “start session” message
  - Acts as the “Resource Broker”
  - Runs the test
  - Both “sides” of test get results

Client Functionality (bwctl)

- bwctl client application makes requests to both endpoints of a test
  - Communication can be "open", "authenticated", or "encrypted" (encrypted reserved for future use)
  - Requests include a request for a ***time slot*** as well as a full parameterization of the test
  - "Third party" requests – run a test on two distributed hosts
  - If no server is available on the localhost, client handles test endpoint
  - *Mostly* the same command line options as testers (e.g. iperf, nuttcp – read the help or man pages to be sure…)

TCP Measurements

- Measures TCP Achievable Bandwidth
  - Measurement includes the end system
  - Sometimes called "memory-to-memory" tests
  - Set expectations for well coded application
- Limits of what we can measure
  - TCP ***hides*** details
  - In hiding the details it can obscure what is causing errors
- Many things can limit TCP throughput
  - Loss
  - Congestion
  - Buffer Starvation
  - Out of order delivery

TCP Performance: Window Size

- Use TCP auto tuning if possible
  - Linux 2.6, Mac OS X 10.5, FreeBSD 7.x, and Windows Vista
- The –w option can be used to request a particular buffer size.
  - Use this if your OS doesn't have TCP auto tuning
  - This sets both send and receive buffer size.
  - The OS may need to be tweaked to allow buffers of sufficient size.
  - See http://fasterdata.es.net/fasterdata/host-tuning/ for more details
- Parallel transfers may help as well, the –P option can be used for this
- To get full TCP performance the TCP window needs to be large enough to accommodate the Bandwidth Delay Product

Bandwidth Delay Product Explained

- 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

TCP Performance: Read/Write Buffer Size

- TCP breaks the stream into pieces transparently
- Longer writes often improve performance
  - Let TCP “do it’s thing”
  - Fewer system calls
- How?
  - -l <size> (lower case ell)
  - Example –l 128K
- UDP doesn’t break up writes, don’t exceed Path MTU

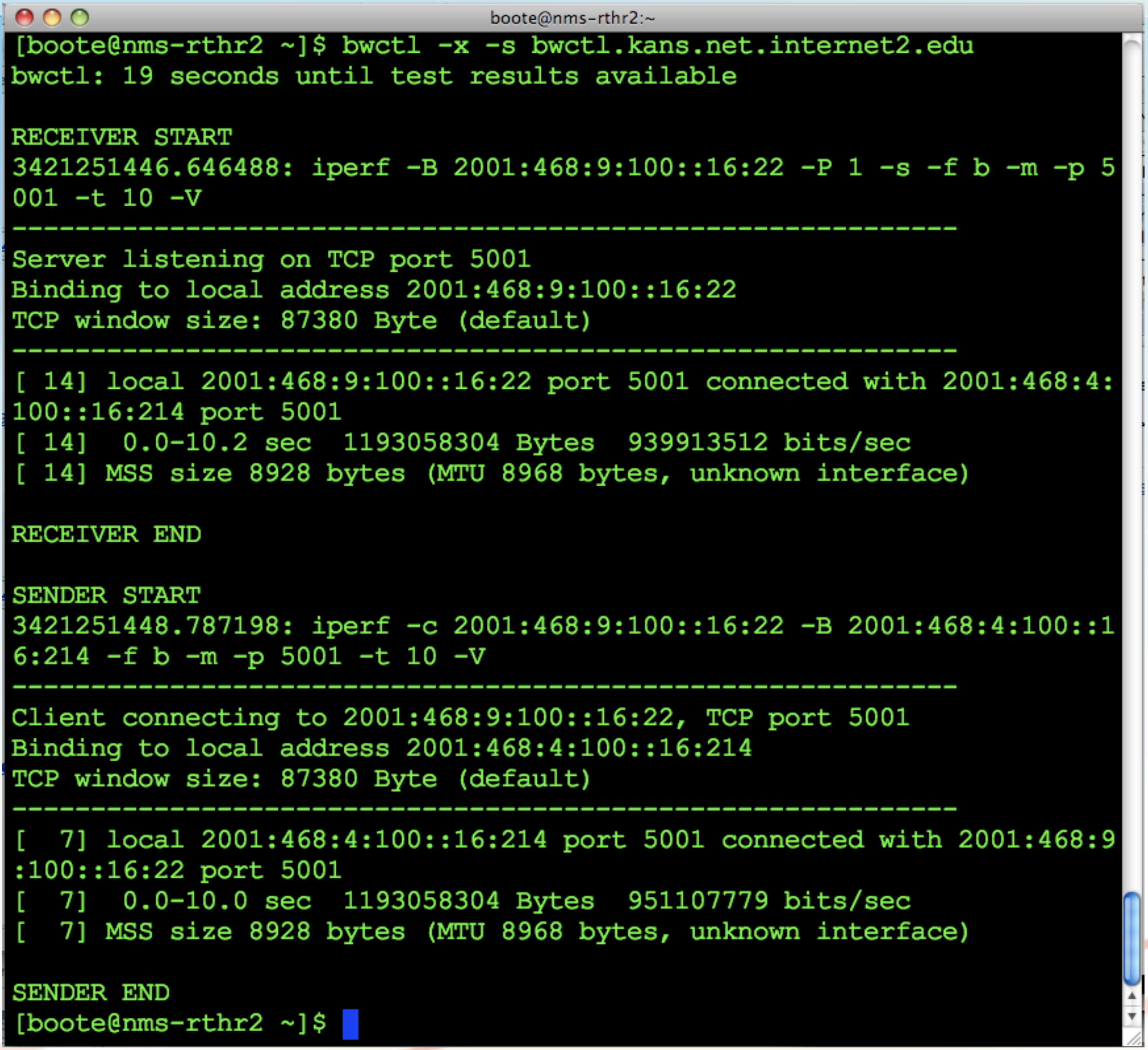
TCP Parallel Streams

- Parallel streams can help in some situations
- TCP attempts to be “fair” and conservative
  - Sensitive to loss, but more streams hedge bet
  - Circumventing fairness mechanism
    - 1 bwctl stream vs. n background: bwctl gets 1/(n+1)
    - X bwctl streams vs. n background: bwctl gets x/(n+x)
    - Example: 2 background, 1 bwctl stream: 1/3 = 33%
    - Example: 2 background, 8 bwctl streams: 8/10 = 80%
- How?
  - The –P option sets the number of streams/threads to use
  - There is a point of diminishing returns

UDP Measurements

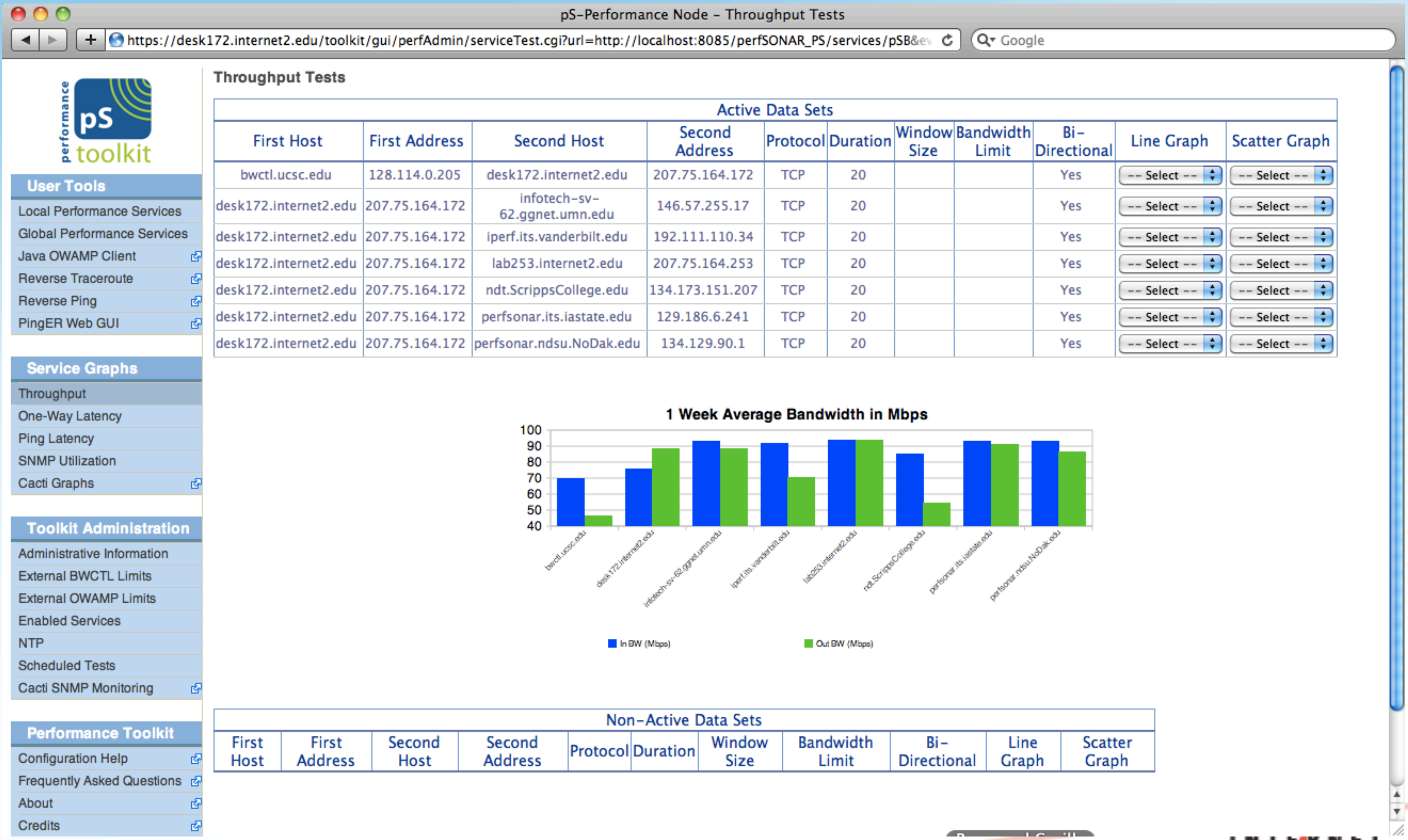
- UDP provides greater transparency
- We can directly measure some things TCP hides
  - Loss
  - Jitter
  - Out of order delivery
- Use -b to specify target bandwidth
  - Default is 1M
  - Two sets of multipliers
    - k, m, g multipliers are 1000, 1000^2,1000^3
    - K, M, G multipliers are 1024, 1024^2,1024^3
  - Eg, -b 1m is 1,000,000 bits per second

Example



```
[boote@nms-rthr2 ~]$ bwctl -x -s bwctl.kans.net.internet2.edu
bwctl: 19 seconds until test results available

RECEIVER START
3421251446.646488: iperf -B 2001:468:9:100::16:22 -P 1 -s -f b -m -p 5
001 -t 10 -V
------------------------------------------------------------
Server listening on TCP port 5001
Binding to local address 2001:468:9:100::16:22
TCP window size: 87380 Byte (default)
------------------------------------------------------------
[ 14] local 2001:468:9:100::16:22 port 5001 connected with 2001:468:4:
100::16:214 port 5001
[ 14]  0.0-10.2 sec  1193058304 Bytes  939913512 bits/sec
[ 14] MSS size 8928 bytes (MTU 8968 bytes, unknown interface)

RECEIVER END

SENDER START
3421251448.787198: iperf -c 2001:468:9:100::16:22 -B 2001:468:4:100::1
6:214 -f b -m -p 5001 -t 10 -V
------------------------------------------------------------
Client connecting to 2001:468:9:100::16:22, TCP port 5001
Binding to local address 2001:468:4:100::16:214
TCP window size: 87380 Byte (default)
------------------------------------------------------------
[  7] local 2001:468:4:100::16:214 port 5001 connected with 2001:468:9
:100::16:22 port 5001
[  7]  0.0-10.0 sec  1193058304 Bytes  951107779 bits/sec
[  7] MSS size 8928 bytes (MTU 8968 bytes, unknown interface)

SENDER END
[boote@nms-rthr2 ~]$
```

BWCTL GUIs



pS-Performance Node - Throughput Tests
https://desk172.internet2.edu/toolkit/gui/perfAdmin/serviceTest.cgi?url=http://localhost:8085/perfSONAR_PS/services/pSB&ev
Google
performance pS toolkit

User Tools

- Local Performance Services
- Global Performance Services
- Java OWAMP Client
- Reverse Traceroute
- Reverse Ping
- PingER Web GUI

Service Graphs

- Throughput
- One-Way Latency
- Ping Latency
- SNMP Utilization
- Cacti Graphs

Toolkit Administration

- Administrative Information
- External BWCTL Limits
- External OWAMP Limits
- Enabled Services
- NTP
- Scheduled Tests
- Cacti SNMP Monitoring

Performance Toolkit

- Configuration Help
- Frequently Asked Questions
- About
- Credits

Throughput Tests

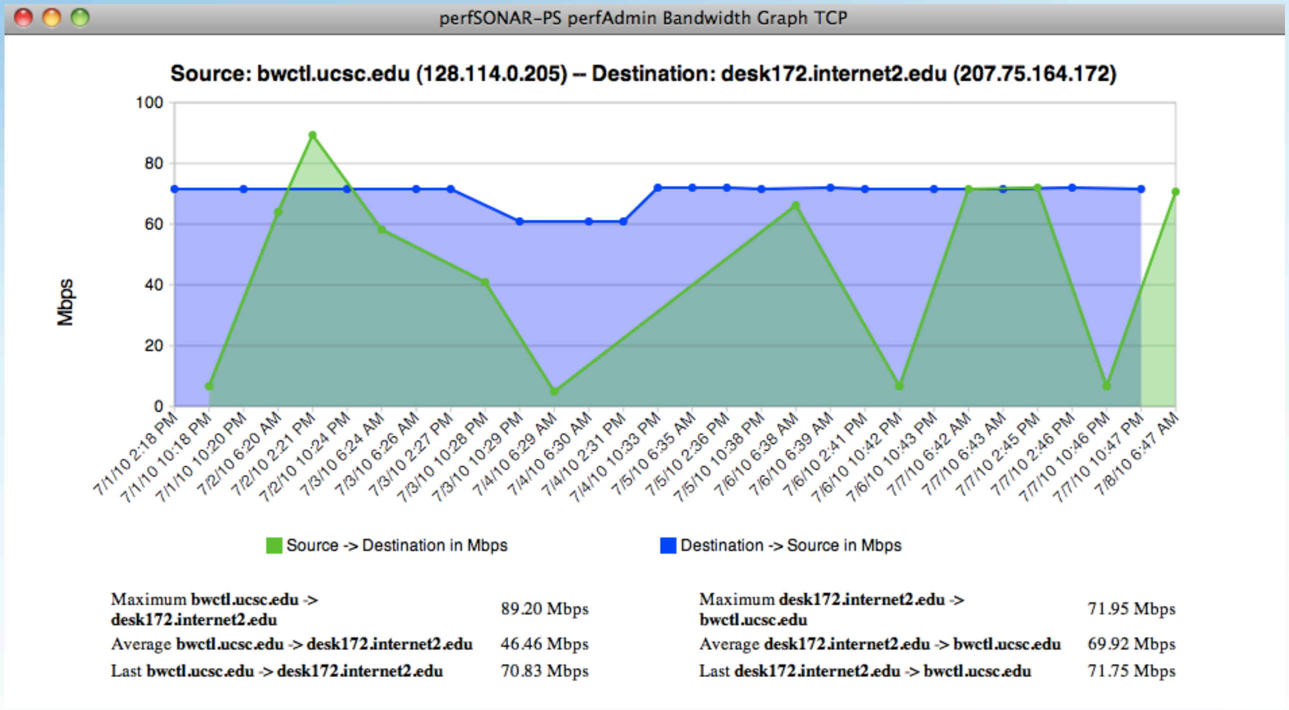
Active Data Sets

First Host	First Address	Second Host	Second Address	Protocol	Duration	Window Size	Bandwidth Limit	Bi-Directional	Line Graph	Scatter Graph
bwctl.ucsc.edu	128.114.0.205	desk172.internet2.edu	207.75.164.172	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	infotech-sv-62.ggnet.umn.edu	146.57.255.17	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	iperf.its.vanderbilt.edu	192.111.110.34	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	lab253.internet2.edu	207.75.164.253	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	ndt.ScrippsCollege.edu	134.173.151.207	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	perfsonar.its.iastate.edu	129.186.6.241	TCP	20			Yes	-- Select --	-- Select --
desk172.internet2.edu	207.75.164.172	perfsonar.ndsu.NoDak.edu	134.129.90.1	TCP	20			Yes	-- Select --	-- Select --

Non-Active Data Sets

First Host	First Address	Second Host	Second Address	Protocol	Duration	Window Size	Bandwidth Limit	Bi-Directional	Line Graph	Scatter Graph

BWCTL GUIs

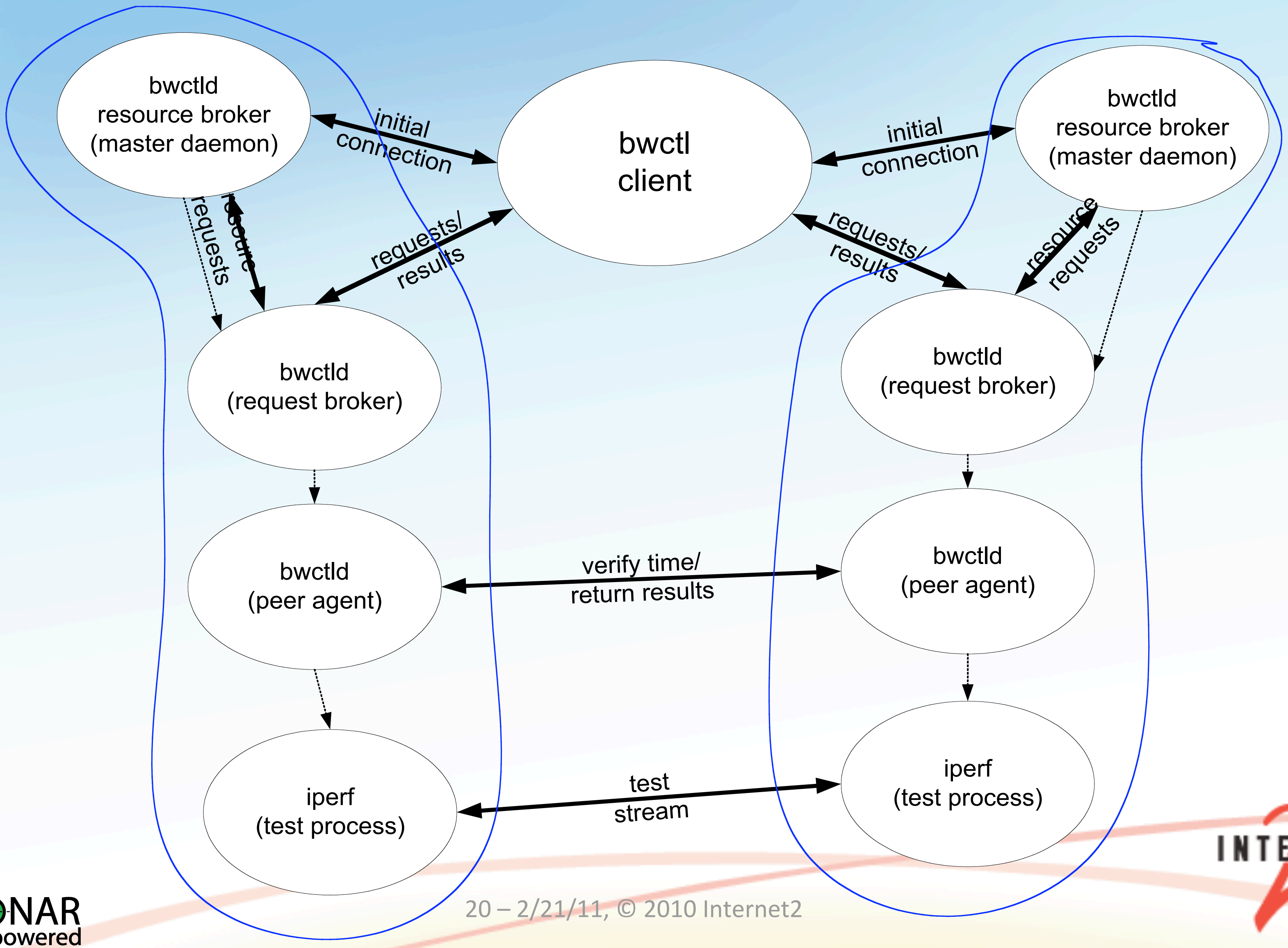


Maximum **bwctl.ucsc.edu -> desk172.internet2.edu**	89.20 Mbps	Maximum **desk172.internet2.edu -> bwctl.ucsc.edu**	71.95 Mbps
Average **bwctl.ucsc.edu -> desk172.internet2.edu**	46.46 Mbps	Average **desk172.internet2.edu -> bwctl.ucsc.edu**	69.92 Mbps
Last **bwctl.ucsc.edu -> desk172.internet2.edu**	70.83 Mbps	Last **desk172.internet2.edu -> bwctl.ucsc.edu**	71.75 Mbps

Resource Allocation

- Each connection is “classified” (authentication)
- Each classification is hierarchical and has an associated set of hierarchical limits:
  - Connection policy (allow_open_mode)
  - Bandwidth (allow_tcp,allow_udp,bandwidth)
  - Scheduling (duration,event_horizon,pending)
    - A time slot is simply a time-dependent resource that needs to be allocated just like any other resource. It therefore follows the resource allocation model.

3rd Party Testing



bwctld resource broker (master daemon)

initial connection

bwctl client

initial connection

bwctld resource broker (master daemon)

requests

resource requests

requests/ results

requests/ results

resource requests

bwctld (request broker)

bwctld (request broker)

bwctld (peer agent)

verify time/ return results

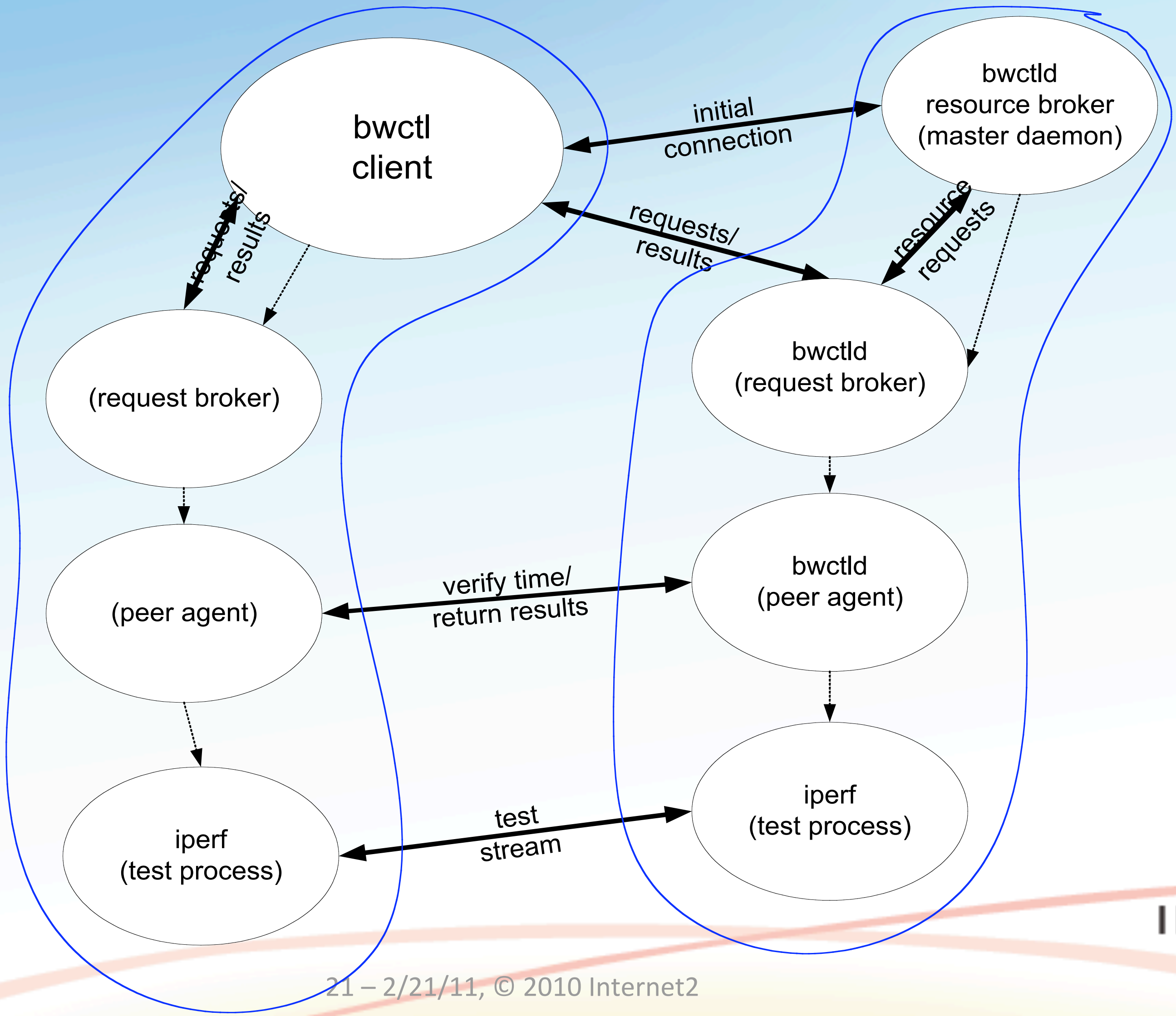
bwctld (peer agent)

iperf (test process)

test stream

iperf (test process)

Testing with no “Local” Server



Tester Applications

- Iperf is primary “tester”
  - Well known – widely used
- Problems integrating exec’d tool
  - Server initialization (port number allocation)
  - error conditions
  - No indication of partial progress (How full was the send buffer when the session was killed?)
- thrulay/nuttcp are available also

General Requirements

- Iperf version 2.0.x
- NTP (ntpd) synchronized clock on the local system
  - Used for scheduling
  - More important that errors are accurate than the clock itself
- Firewalls:
  - Lots of ports for communication and testing – see the web for specifics
- End hosts must be tuned!
  - http://fasterdata.es.net/fasterdata/host-tuning
  - http://www.psc.edu/networking/perf_tune.html

Supported Systems

- Source Code
  - All modern Unix distributions (Free BSD/Linux)
  - OS X
- Packages
  - Support for CentOS 5.5 (x86)
  - Packages have been shown to operate on similar systems (CentOS, Fedora, RHEL, and x86_64 architecture)

Security Considerations

- DoS source
  - Imagine a large number of compromised BWCTLD servers being used to direct traffic
- DoS target
  - Someone might attempt to affect statistics web pages to see how much impact they can have
- Resource consumption
  - Time slots
  - Network bandwidth

Policy Approaches

- Restrictive for UDP
  - Allow between peers
  - Limit bandwidth, and time of tests
- More liberal for TCP tests
  - Open for all (or peers)
  - Limit length of tests
- Protect AES keys!
  - If being used

Availability

- Currently available
  - http://www.internet2.edu/performance/bwctl
  - http://www.software.internet2.edu
- Mail lists:
  - https://lists.internet2.edu/sympa/info/bwctl-users
    - bwctl-users@internet2.edu
  - https://lists.internet2.edu/sympa/info/bwctl-announce
    - bwctl-announce@internet2.edu

Hands On

- Testing BWCTL:
  - Log on to testbed
  - Test from one host to another:
    - bwctl –f m –t 10 –I 1 –c HOSTNAME
  - Test the other direction:
    - bwctl –f m –t 10 –I 1 –s HOSTNAME
  - Test UDP:
    - bwctl –f m –t 10 –I 1 –u –b 100M –c HOSTNAME
  - Try different hosts. Try longer tests. What happens when we use:
    - -w (Window size, try 128k and 4M)
    - -P (Parallel threads, try 2, try 4)

BWCTL

March 10th 2011, OSG All Hands Meeting, Network Performance

Jason Zurawski – Internet2

For more information, visit http://www.internet2.edu/workshops/npw