



BWCTL (Bandwidth Test Control)

Jeff Boote (boote@internet2.edu)

Network Performance Workshop

5-Apr-06

Controls throughput or “bandwidth” tests.

What is it?

A resource allocation and scheduling daemon for arbitration of iperf tests

Bwctl controls the throughput tests by adding resource allocation and scheduling policy controls.

Problem Statement

- Users want to verify available bandwidth from their site to another.

Methodology

- Verify available bandwidth from each endpoint to points in the middle to determine problem area.

This is arguably one of the best ways to determine if an application will work because it is doing the throughput test end-2-end exactly like the application would be doing it.

Typical Solution

- Run “iperf” or similar tool on two endpoints and hosts on intermediate paths

If NOC operators had a quarter for every time someone asked them to start an iperf server somewhere...

Typical road blocks

- 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 *
- (* BWCTL was designed to help with these)

Basically, lots of administration. And the worst part is that there is no consistent policy taken for the amount of resources different users are allowed to use.

BWCTL allows you to think about what policy you want and gives you a way to codify and enforce it uniformly.

Applications

- bwctld daemon
- bwctl client

Built upon protocol abstraction library

- Supports one-off applications
- Allows authentication/policy hooks to be incorporated

The policy implementation is compile-time pluggable.

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
- If no server is available on the localhost, client handles test endpoint
- *Mostly* the same command line options as iperf (some options limited or not implemented.)

The command like options are as similar to iperf as possible. Probably too much so. ;)

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
- Resource “Broker”
- Runs tests
- Both “sides” of test get results

This is where the policy is implemented.

Bwctld is a traditional accept/fork style daemon with the parent process also listening for resource requests from child processes.

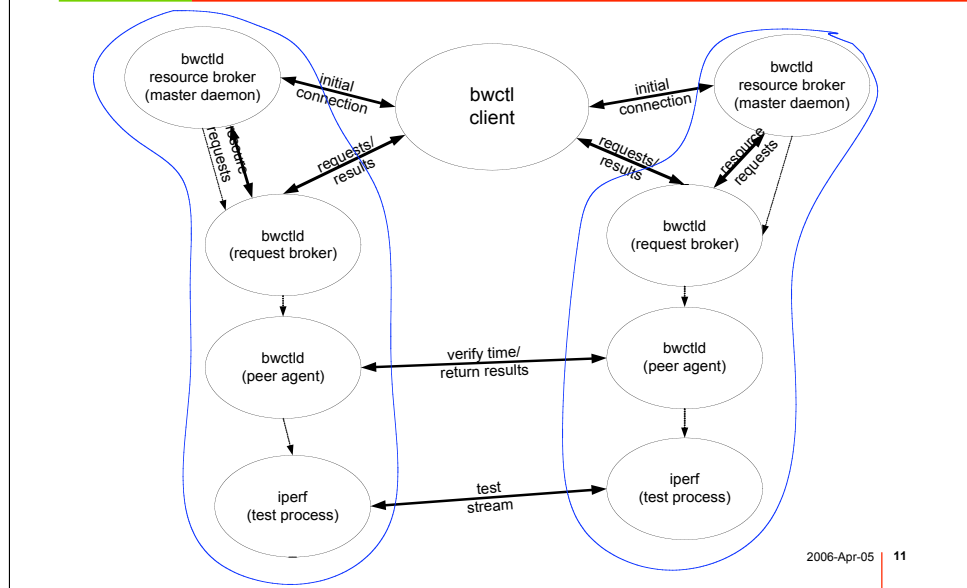
A time slot is simply a time-dependant resource that needs to be allocated just like any other resource. It therefore follows the resource allocation model.

bwctl scheduling currently only allows a single test to happen at a time.

- 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)

The parent bwctld keeps track of current resource utilization needed to implement policy.

BWCTL: 3-party Interaction



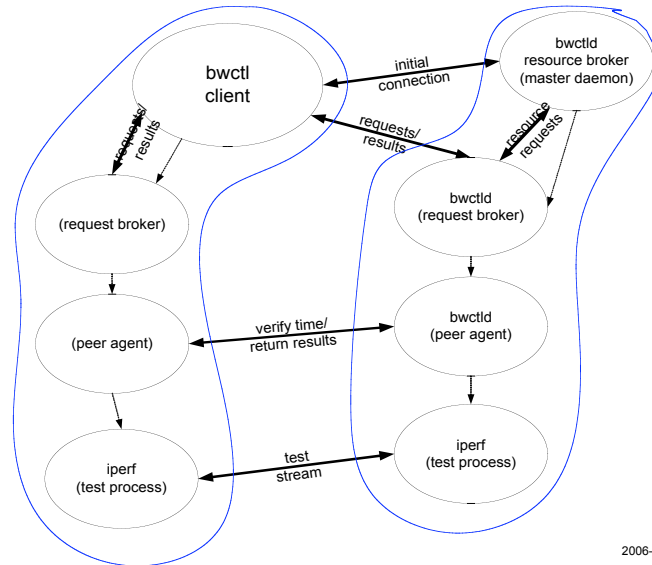
2006-Apr-05 | 11

The master daemon on each server maintains the full schedule for that host.

A request broker is forked off for each new client request.

The peer agent is responsible for verifying the time offset between the servers, it is used as a protection against DoS attacks, and is also used to share the results of each test with the test peer.

The client makes a request for an iperf test indicating the amount of time (duration) it wants, the earliest start time, and the latest time it is willing to accept. It makes this request to each server in sequence. Each server allocates the earliest time slot it is willing to give. The client then updates the request before asking for a matching timeslot from the other server. Either a matching timeslot will be open, or the scheduling algorithm will reach beyond the latest time the client or one of the servers is willing to schedule and the request will be denied.



This is essentially identical to the previous slide except the client implements the functionality of one of the servers.



Iperf is the “tester”

- Well known – widely used
- Problems of integration
 - Iperf server initialization (port number allocation)
 - Iperf error conditions
 - End of session
 - No indication of partial progress (How full was the send buffer when the session was killed?)

2006-Apr-05 | 13

These problems will likely show up in your test results just as they do when you normally run iperf. The difference is that when you run iperf by hand and give up - typing cntrl-C you are not surprised when you don't have results from the test.

Specific difficulties for scheduling (UDP)

Iperf doesn't always send at requested rate

Iperf sender hangs intermittently

End of session is difficult to detect, which is problematic for a “scheduled” timeslot

Iperf sometimes takes large amounts of time to finish

Specific difficulties for scheduling (TCP)

Large pipe to small pipe

Launch a large window

Test waits until completion

Terminate test to remain within schedule

No data to represent real problem

Full mesh presents difficulties for window size selection (and other path specific characteristics)

bwctl uses the peer to peer server connection to deduce a “reasonable” window

If possible, path specific parameters should be dynamically configured

- Iperf version 2.0 and 2.0.2
- 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
- End hosts must be tuned!
http://www.psc.edu/networking/perf_tune.html
<http://www.didc.lbl.gov/TCP-tuning/buffers.html>

NTP may be a surprising requirement - but to schedule the tests without wasting too much time between tests requires a reasonable estimate for the accuracy of the local clock.



Supported Systems

- FreeBSD 4.x, 5.x
- Linux 2.4, 2.6
- (Most recent versions of UNIX should work)

2006-Apr-05 | 15

SunOS is likely in the future.

- Highly dependent upon the network tests
 - Any system that can support an iperf test of a given intensity will be able to handle the additional burden of BWCTL
- To support 990 Mbps TCP flows on Abilene we use:
 - Intel SCB2 motherboard
 - 2 x 1.266 GHz PIII, 512 KB L2 cache, 133 MHz FSB
 - 2 x 512 MB ECC registered RAM (one/slot to enable interleaving)
 - 2 x Seagate 18 GB SCSI (ST318406LC)
 - SysConnect Gigabit Ethernet SK-9843 SX

See <http://abilene.ucaid.edu/observatory/> for more information on the systems we use on Abilene.

- 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

DoS source:

A compromised bwctld server could be used to send packets toward others. The implementation ensures that sessions can not be directed to random hosts in unauthenticated mode. (Only toward the BWCTL-control client.)

DoS target:

Packets directed toward an bwctld server can/will affect the results of the valid test traffic.

Resource Consumption:

bwctld has policy controls to allocate resources to appropriate users.



Policy Recommendations

- Restrictive for UDP
- More liberal for TCP tests
- More liberal still for “peers”
- Protect AES keys!

2006-Apr-05 | 18

On Abilene we attempt to be as open as we can, until we can't anymore.



Availability

- Currently available

<http://e2epi.internet2.edu/bwctl/>

Mail lists:

- bwctl-users@internet2.edu

- bwctl-announce@internet2.edu

<https://mail.internet2.edu/www/lists/engineering>

<http://e2epi.internet2.edu/bwctl/>



The End.