**Requirement for the TestUI:**

1. A text file i.e. configuration.txt containing a set of test cases within which contains a set of test programs. Test case are declared with different attributes, such as how it is run, whether locally or in a distributed set up.

Syntax for configuration.txt

(Meta syntax is more-or-less conventional BNF grammar. { x }\* means 0 or more instances of x. “|” means alternative productions. Tokens are separated by white space (spaces or tabs), but new lines are significant. One or more newline characters are indicated by the terminal <newline>.

Comments can be included, denoted by characters from “#” up to and not including the next newline.

<configuration> := <testcase>\*

<testcase> := test <id> <attributes>\* <newline>

{ <executable> { <parameters> }\* <newline> }\*

<attributes> := local | multiple

<executable> := <id>

Each test case has a unique name <id> and attributes “local” and/or “multiple”. The test is conducted by running all of the executables as specified. “local” means all executables are run on a single host. If “local” is not specified, executables are spread across available hosts. If there are not as many hosts as executables, more than one executable will run on each host, distributing the executables across hosts in a round-robin manner. If there is only one host, all executables will run on the host as if the executable was “local” (even if “local” is not specified).

“multiple” means that each executable is run multiple times. In a “multiple” test case, if there are N processes, each executable receives 2 “extra” command line arguments after any explicitly specified arguments. If N == 3, the extra arguments are “1 3”, “2 3”, and “3 3”. In other words the first extra argument is 1 through N and the second extra argument is N. This allows each instance of the executable to construct unique service names and perhaps to exhibit unique behavior, e.g. instance 1 N could be a server expecting N-1 clients, and instances 2 through N could be clients. The value N is specified manually in the UI. If N is larger than the number of hosts, multiple processes may run on each host.

Each executable is run with the specified command line arguments (perhaps with 2 additional arguments if this is a “multiple” test case. The actual executable on Windows will have a “.exe” extension. The location of executables is by convention. [Maybe there should be, in addition to “test”, commands named “windowspath” that takes a path for windows machines, “linuxpath” for linux, and “osxpath” for OS X (mac). Otherwise, you must document how the system chooses the path, which might be o2/test/debug, o2/debug, o2/test/release, o2/release, etc.)

Eg. Here arraytest is the test case containing the single test program arraytest.

test arraytest

arraytest

And here is a test case that runs client and server with command line arguments “tcp” on two different hosts (if available)

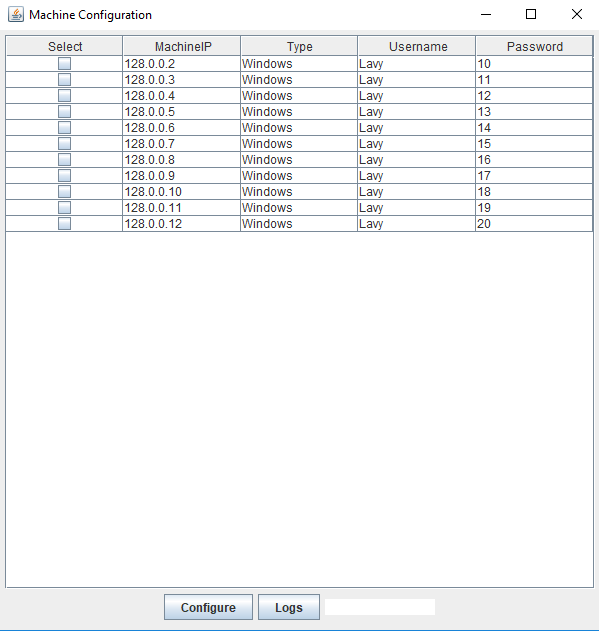
test clientserver

client tcp

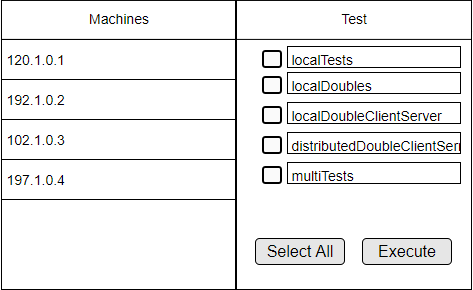
server tcp

2.The Machines selected from the 1st screen (shown below) will be displayed in the first column.

How are these machines identified? Is there another configuration file, or do you enter the machines into the application? I think it would be better to use a configuration file. Then you could carefully keep the configuration file in a read-only directory that only the user can read, giving a little better protection for passwords. It would be even better, I suppose, to use ssh keys and keep private keys in the usual local key store, avoiding passwords altogether, but I’m OK to just create “guest” accounts for testing so that disclosed passwords do not offer many privileges or secrets to anyone that might obtain them.



3.The configuration file will be parsed and all the test cases will be displayed in the second column.



4.Button "Select All" checkboxes all the test cases.

5.Button "Execute" executes all the test cases accordingly.

**Configuration File – Sample:**

# Usage:

# Name the test cases starting phrase as "Test" (not case sensitive)

# The command line arguments to be written after the name of the test program with space separation

# The test program to be given without any extensions (like .c, .exe, and etc.)

Test ClientServer local

o2client 5000 R

o2server S

test status local

statusclient

statusserver

test tcpclientserver

tcpclient

tcpserver

# lo\_benchmk\_client and lo\_benchmk\_server are for performance measurements

# and are not O2 tests

Test ClientServerDist

o2client 500 R a

o2server S a

Test clock

clockmaster

clockslave

test bndl\_lo

lo\_bndlrecv

lo\_bndlsend

test osc\_lo

lo\_oscrecv

lo\_oscsend

test bndl

oscbndlrecv

oscbndlsend

test osc

oscrecvtest

oscsendtest

Test threeprocs

o2client1

o2client2

o2server

Test ClientServerDist

o2client

o2server

test StatusDist

statusclient

statusserver

--------------------OK, hopefully you get the idea. I’m not going to correct all of this –RBD ---------------------------

tcpclient

tcpserver

lo\_benchmk\_client

lo\_benchmk\_server

#this runs all the test programs written in the local system.

Test localTests

arraytest

arraytest a #any number of debug flags can be appended after the test program name separated by space.

bundletest

coercetest

dispatchtest

infotest1

infotest2

longtest

oscanytest

taptest

typestest

o2client

o2server

statusclient

statusserver

tcpclient

tcpserver

lo\_benchmk\_client

lo\_benchmk\_server

clockmaster

clockslave

lo\_bndlrecv

lo\_bndlsend

lo\_oscrecv

lo\_oscsend

oscbndlrecv

oscbndlsend

oscrecvtest

oscsendtest

Test localSingles

arraytest

arraytest a

bundletest

coercetest

dispatchtest

infotest1

infotest2

longtest

oscanytest

taptest

typestest

Test distributedSingles

arraytest

arraytest a

bundletest

coercetest

dispatchtest

infotest1

infotest2

longtest

oscanytest

taptest

typestest

**Output Consolidation Structure:**

With regard to the output consolidation in the dispatch machine, we are planning to adopt the following directory structure.

<test-id>\_<timestamp>/<executable>[-<index>].log

Here, the <test-id> is the <id> following “test” in configuration.txt. <timestamp> is in the format yyyy-mm-dd\_hh:mm:ss. [Note: PLEASE do not put whitespace into filenames – it makes everything harder since you then have to carefully quote every filename in every script you might write to deal with these logs. ~~This directory has two sub-directories – local and distributed.~~ [why should I care to separate local from distributed tests? And why local/distributed but not multiple/non-multiple?] ~~Here, local directory contains the log files of tests that were run locally. The distributed directory contains several sub-directories – one for each machine with the machineIP as the name of the directory, within which the log for tests run on that machine are consolidated.~~

<executable> is the program name from configuration.txt. If this is a “multiple” test, <index> is appended to create, e.g. client-1.log, clien-2.log, etc.

[I am assuming each process generates one “.log” file which is just the standard output of each process. If there is additional data, e.g. if you record start/stop times, run time, memory usage, or other info, then maybe the spec should be that you have directories named <executable>[-<index>], and each directory contains stdout.log, info.txt, etc.