A simple guide to run Abstract applications on Emulab

Get an account on Emulab

Create an experiment

There are several phases when running experiments on Emulab. On Emulab, you **create** experiments, which you **swap in**, and then **run**. Once you're done, or once the maximum time (~5 days) expires, you **swap out** the experiment, and swap it in later on. If you think you will never run this particular experiment again, you **delete** the experiment. Also, while experiment is swapped in, you can **modify** it, to add/remove machines, or change topology. This may not succeed every time, though.

In Emulab terminology, *experiment* refers to the machines you use, their interconnects, and the OS you run on the machines. You may run many different executables in the same experiment, and use only a subset of the total machines allocated to you.

The procedure to create an experiment contains of a several steps. First, login to *users.emulab.net*, and create a file called *~/myexperiment.ns*. Copy the following code in that file:

```
# This is a simple ns script. Comments start with #.
set ns [new Simulator]
source tb_compat.tcl

# Number of machines to use
set maxreplicas 4
set maxclients 40

# Type of machines to use
tb-make-hard-vtype replicas {pc3000}
tb-make-hard-vtype clients {pc3000}
tb-make-soft-vtype controller {pc3000 d710 pc2000 pc2400c2}

set lanstr0 ""
set lanstr1 ""
```

```
# What OS to run (ubuntu8.04 is our image, preset for Abstract)
set nodeG [$ns node]
tb-set-node-os nodeG ubuntu8.04

tb-set-hardware $nodeG controller

for {set i 0} {$i < $maxreplicas} {incr i} {
       set replica($i) [$ns node]
       append lanstr0 "$replica($i) "
       append lanstr1 "$replica($i) "
       tb-set-node-os $replica($i) ubuntu8.04
       tb-set-hardware $replica($i) replicas
}

append lanstr1 "$nodeG "

for {set i 0} {$i < $maxclients} {incr i} {
       set client($i) [$ns node]
       append lanstr1 "$client($i) "
       tb-set-node-os $client($i) ubuntu8.04
       tb-set-hardware $client($i) clients
}

# Put all the nodes in a lan
# FastEthernet (100Mbps), no latencies (0ms)
set lan0 [$ns make-lan "$lanstr0" 100Mb 0ms]
set lan1 [$ns make-lan "$lanstr1" 100Mb 0ms]

$ns rtproto Static

# Go!
$ns run
```

Parts in red denote user-configurable values, whose meaning is self evident. This code is quite useful, because it generates the topology with two LANs, one for inter-replica communication, and one with all replicas and clients. Moreover, machines are named in a predictable way, that scripts in the Abstract framework expect.

For the next step, go to *http://www.emulab.net*, login and in the Emulab control panel, choose *Experimentation->Begin an Experiment*. Fill in the required values. For the "Your NS File", type:

```
/users/YOURUSERNAME/myexperiment.ns
```

Replace `YOURUSERNAME` with your actual username on Emulab. Click on *Submit*. Emulab will

notify you via email if the experiment successfully swapped in. If there is an error, you will also be notified about it, and given a detailed description what went wrong. If the reason is insufficient number of resources, you will need to wait.

*You should login to the machine named **nodeG**, and run your experiments from that machine. In the above configuration, this machine is connected to all other machines.*

Checkout the code

The code for Abstract is hosted at LPD, EPFL. If you are a part of this group (this is the group "*abstracts*" on Emulab), and you do not need a version control system, the steps are simple. Otherwise, you will first need to get an account to access this repository, and you should contact *Nikola Knezevic (nikola.knezevic@epfl.ch)* in order to get credentials.

Access if part of the "abstracts" group on Emulab (simple)

If you are a member of the "abstracts" group, then you can make a local copy of the repository, while still maintaining the proper history and setup. To do so, issue the following commands in your home directory:

```
git clone -l --no-hardlinks  /proj/abstracts/BFT
cd BFT
perl -i -ple 'next unless /knezevic/; m,([^/]*).git$,;
     $_="\turl =   file:///proj/abstracts/BFT/src/protocols/$1\n";' \
     .gitmodules
git submodule init
git submodule update
```

Now you are ready to configure and build Abstract, and run the experiments, and you have the source code in the BFT directory in your $HOME.

Access to the source code (more complex)

Once you get the credentials for the Abstract framework repository, issue the following command:

Configure and build Abstract

For the initial setup, type in the following commands, which set *sfslite* (the library used by Abstract framework for networking and cryptography):

```
cd ~/BFT
cd sfslite-1.2
rm -rf install
mkdir -p install
```

```
export SFSHOME=`pwd`
./configure --prefix=$SFSHOME/install &&
make &&
make install
cd ..
rm -f sfs ; ln -s sfslite-1.2/install sfs
rm -f gmp ; ln -s /usr/lib gmp
```

To compile sfslite on a 64 bits machine, you need to export the following flags before running configure (don't forget to unset them once sfslite is compiled):

```
export CPPFLAGS="-m32 -L/usr/lib32"
export CFLAGS="${CPPFLAGS}"
export CXXFLAGS="${CPPFLAGS}"
export AM_CFLAGS="${CPPFLAGS}"
export AM_CPPFLAGS="${CPPFLAGS}"
export AM_CXXFLAGS="${CPPFLAGS}"
export LDFLAGS="${CPPFLAGS}"
```

Now, you are ready to build the Abstract framework protocols. You can remove the unneeded protocols from src/Makefile. Either way, you can set the build options for each protocol in the corresponding Makefile (they are in src/protocols/PROTOCOL/Makefile). Afterwards, this builds you the whole system:

```
cd ~/BFT/src
make clean all
```

To compile the protocols on a 64 bits machine :

- add `-m32 -L/usr/lib32` to the C_OPT_FLAGS variable in `src/Makefile.common` and to the CFLAGS variable in `src/redis/src/Makefile`
- unset the exported variables:
  ```
  unset CPPFLAGS
  unset CFLAGS
  unset CXXFLAGS
  unset AM_CFLAGS
  unset AM_CPPFLAGS
  unset AM_CXXFLAGS
  unset LDFLAGS
  ```
- on Debian, some 32bits library do not have the proper symlink:
  - `$ sudo ln -s /usr/lib32/libsqlite3.so.0 /usr/lib32/libsqlite3.so`
  - `$ sudo ln -s /usr/lib32/libldap-2.4.so.2 /usr/lib32/libldap.so`
  - `$ sudo ln -s /usr/lib32/liblber-2.4.so.2.5.6 /usr/lib32/liblber.so`

Configure the experiment

Experiments related to the Abstract framework are run through a set of shell scripts. These shell scripts start the replica executables on physical machines dedicated for replicas, the collocated client executables on the physical machines dedicated for replicas, and the manager executable that orchestrates the experiment, on its own machine.

First, add these usefull aliases to your `~/.aliases` file:

```
alias k=./kill.sh
alias l=./launch_ring.sh
```

There are parts of configuring the experiment. These encompass building configuration for Abstract framework protocols (node identities and PKI), choosing the experiment to run (which executables), and setting up the configuration of the experiment (message sizes, repetitions, etc...).

Identities and PKI

As started by PBFT, all protocols require a predefined identities, public and private keys. To make things worse, each protocols requires this data in a different form, in different files, that all must be passed to the main executable. These files are located in `~/BFT/config_private` directory.

To ease these pains, there is a tool to automatically generate these config files, based on a some parameters. Keep in mind that this tool assumes that you used the Emulab topology file from the beginning of the text.

This script needs several options. It takes the replication factor (*f*), the number of physical machines for clients (*cli*) and replicas (*rep*), and the total number of clients that will be run (*numclients*). Note that *if you have 10 machines, write down 10, even if you plan to run experiments on only 4, or 7 machines*.

Now, just run the tool, and pass it the number of faults you want to tolerate (f=1,2,3...):

```
perl gen_conf.pl -f 1 -rep 4 -cli 20 -numclients 100 -pki
```

The tool will generate the necessary configuration files. The final switch (*-pki*) instructs the tool to generate the secret keys for nodes. You can do this only *once per Emulab experiment* (after you create an Emulab experiment, not when you are running executables).

Choosing executables

`~/BFT/src/benchmarks` contains many executables you can run and tune. They are created by the `Makefile` in the very same directory. The manager executable is in `thr_manager`, and is the same for all types of experimentation.

Standard microbenchmarks defined by Castro et. al. are implemented in files named `server` and `thr_client`. For the similar experiments that interact with a sqlite3 database, you should use `server_db` and `thr_client_db`. There are also executables for the replication of a Redis store (`server_redis` and `thr_client_redis`), and for the LDAP replication (`server_ldap` and `thr_client_ldap`).

Setting up the configuration

The main configuration file is located at `~/BFT/scripts/ring_options.conf`. In this file, you can set the names of the executables to use, the experiment parameters, etc...

First, change the `LOGIN_NAME` and `MASTER_LOGIN_NAME` to match your credentials on the same machine. Then, set the executables to use through variables `SERVER_EXEC` and `CLIENT_EXEC`. Other parameters, listed at the top of the file, are self-evident in their function.

Running the experiment

You must run experiments from the machine called *nodeG* in your Emulab experiment. The scripts for running the experiment are at `~/BFT/scripts/`.

Once you configured the experiment, you should execute the script called `./launch_ring.sh`, which accepts the number of client machines. The output of the experiment is scattered over several files. Files named like `ring-1s-1-110815.dat` are in `~/times` directory, and contain the full experiment output (from the manager). Then, aggregated statistics are in a file named `~/times/experiments.out`. If you enabled operation timings, the data is in the file called `~/times/timings.out`.