3. Übung, SS 2013
Verteilte Systeme/ Distributed Systems
Ausgegeben am 25.5.2013
Vorstellung 6.6.2013

Katinka Wolter
Benjamin Güldenring
Inst. für Informatik
Freie Universität Berlin

Assignment 1. Stateful/stateless Processes

Is a server that maintains a TCP/IP connection to a client stateful or stateless?

Solution: Assuming the server maintains no other information on that client, one could justifiably argue that the server is stateless. The issue is that not the server, but the transport layer at the server maintains state on the client. What the local operating systems keep track of is, in principle, of no concern to the server.

Assignment 2. Migration

Consider a process P that requires access to file F which is locally available on the machine where P is currently running. When P moves to another machine, it still requires access to F. If the file-to-machine binding is fixed, how could the systemwide reference to F be implemented?

Solution: The simplest solution is to create a separate process Q that handles remote requests for F. Process P is offered the same interface to F as before, for example in the form of a proxy. Effectively, process Q operates as a file server.

Assignment 3. Processes and Threads

Describe a simple scheme in which there are as many lightweight processes as there are runnable threads.

Solution: Start with only a single LWP and let it select a runnable thread. When a runnable thread has been found, the LWP creates another LWP to look for a next thread to execute. If no runnable thread is found, the LWP destroys itself.

Assignment 4. Peersim internals

It is sometimes useful or necessary to dig into the source code of Peersim to understand how it works. If not already done make yourself familiar with the classdoc documentation of Peersim at `http://peersim.sourceforge.net/doc/index.html`.

4.1 Controls and Observers

Recall that an observer object is just another name for a class implementing the `Control` interface.

- Have a look at the cycle driven gossip averaging example from the last practice sheet. Find the class in Peersims source code responsible for the calculation and printing of the statistics. **Hint:** It will be a `Control` class specified in the configuration file.
- Now have a look at the event driven gossip averaging example again. Which class is responsible here? Is there a difference to the one used in the cycle driven example and if so, what is it?

4.2 Bonus Question: Simulation Engines

Find the responsible classes where the actual simulation is performed and try to understand them. Which classes and methods are involved? **Hints:** Cycle driven is abbreviated `CD` and event driven `ED`; The class `FullNextCycle` implements the `Control` interface.

Assignment 5. Peersim Load Balancer

Read Section 3 of the Tutorial at `http://peersim.sourceforge.net/tutorial1/tutorial1.pdf` and have a look at the source code at `src/example/loadbalance`.

- What are the chief differences between the `BasicBalance` and the `AvgBalance` protocol?
- Name all the implemented controls, describe their purpose and their execution order.

5.1 Evaluation

Take the configuration file from the end of this sheet for the following evaluation and always use gnuplot graphs to answer the questions.

- Use the output of the `LBObserver` control to compare how quick both protocols are in load balancing. Which column of the statistics do you use?
- Using the output of the `QuotaObserver` control compare how efficient (in terms of load transfer) the protocols are. Which column of the statistics can be used for interpretation and how could the amount of transferred load in each cycle be calculated?

Config file for Assignment 5

```
random.seed 1234567890
simulation.cycles 200
network.size 250

protocol.ncast example.newscast.SimpleNewscast
protocol.ncast.cache 20
# comment the next 3...
protocol.bal example.loadbalance.BasicBalance
protocol.bal.linkable ncast
protocol.bal.quota 100
# and uncomment the following 3 lines to change the protocols
#protocol.bal example.loadbalance.AvgBalance
#protocol.bal.linkable ncast
#protocol.bal.quota 100

control.lbo example.loadbalance.LBObserver
control.lbo.protocol bal
control.qo example.loadbalance.QuotaObserver
control.qo.protocol bal
control.rq example.loadbalance.ResetQuota
control.rq.protocol bal

init.rnd WireKOut
init.rnd.protocol ncast
init.rnd.k 20
init.peak example.aggregation.PeakDistributionInitializer
init.peak.value 10000
init.peak.protocol bal

include.init rnd peak
include.protocol ncast bal
include.control lbo qo rq
```

Hints

When executing the simulation in Assignment 5 of you get results like

```
...
CDSimulator: cycle 196 done
control.lbo: 197 40.0 40.0 40.0 0 0.0
control.qo: 197 100.0 100.0 250 100.0 0.0 250 250
...
```

On Unix like operating systems you can write the complete output to one file `experiment.data` and filter the results using `grep`

```
$ grep control.lbo experiment.data > experiment_lbo.data
$ grep control.lbo experiment.data > experiment_qo.data
```

and on windows you can use `findstr` in the very same way:

```
C:\...> findstr control.lbo experiment.data > experiment_lbo.data
C:\...> findstr control.lbo experiment.data > experiment_qo.data
```