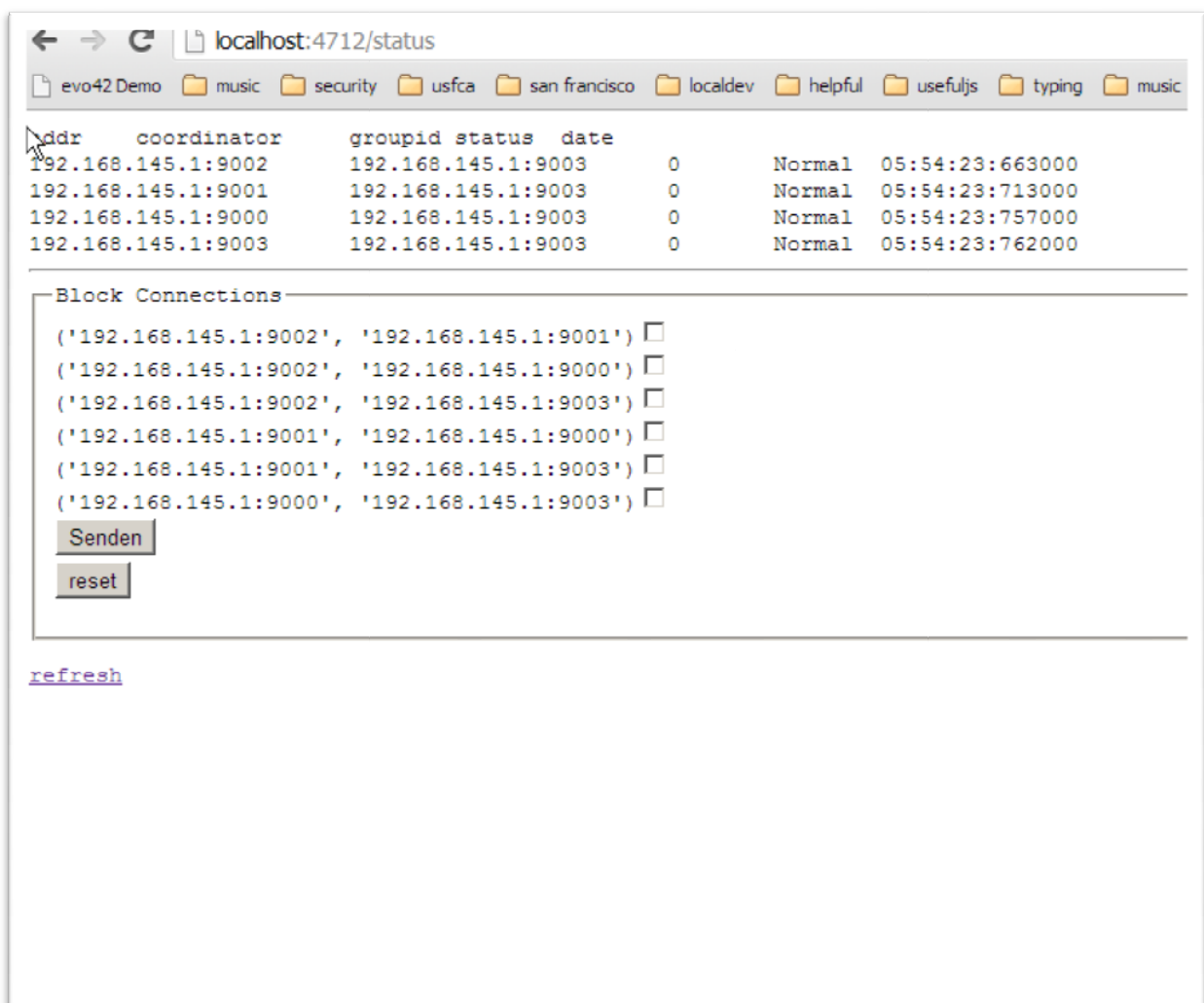


CS636 Leader Election Algorithms

Overview

This project implements three different algorithms for Leader Elections in Python. In every algorithm nodes report to a central log server (see Logserver.py). In addition to keeping a central log of node activities this server has a web interface that allows to keep track of node states. State reported by single nodes is saved to MongoDB which has to be running on the same node as the logs server.

The invitation algorithm as well as the asynchronous bully algorithm can build groups if the network is partitioned – this partitioning can be simulated by blocking connections between groups over the web interface.



The screenshot shows a web browser window with the address bar displaying `localhost:4712/status`. The browser's file explorer shows a directory structure with folders like `evo42 Demo`, `music`, `security`, `usfca`, `san francisco`, `localdev`, `helpful`, `usefuljs`, `typing`, and `music`.

The main content area displays a table with the following columns: `addr`, `coordinator`, `groupid`, `status`, and `date`. The table contains four rows of data:

addr	coordinator	groupid	status	date
192.168.145.1:9002	192.168.145.1:9003	0	Normal	05:54:23:663000
192.168.145.1:9001	192.168.145.1:9003	0	Normal	05:54:23:713000
192.168.145.1:9000	192.168.145.1:9003	0	Normal	05:54:23:757000
192.168.145.1:9003	192.168.145.1:9003	0	Normal	05:54:23:762000

Below the table, there is a section titled "Block Connections" with a list of node pairs and checkboxes to block connections between them:

- ☐ ('192.168.145.1:9002', '192.168.145.1:9001')
- ☐ ('192.168.145.1:9002', '192.168.145.1:9000')
- ☐ ('192.168.145.1:9002', '192.168.145.1:9003')
- ☐ ('192.168.145.1:9001', '192.168.145.1:9000')
- ☐ ('192.168.145.1:9001', '192.168.145.1:9003')
- ☐ ('192.168.145.1:9000', '192.168.145.1:9003')

Below the list, there are two buttons: "Senden" and "reset".

At the bottom of the page, there is a [refresh](#) link.

Bully Algorithm

The Bully algorithm as described by Hector Garcia Molina is implemented in the file `bully.py`. To run it first start the log server running `Logserver.py` from the command line. Per default it runs on localhost and port 4711 but this can be changed by supplying a port number as the first argument.

Usage `Logserver.py [portnummer]`

After running the log server nodes can be started individually or in groups. All nodes have to be identified by an ip number and a port in a file called `server_config` that has to be found in the same directory as the one the bully process is started from. The `server_config` files simply lists `ip:portnumber` per line where priorities are assigned to nodes according to the position of the line in the file. A sample configuration file can be found in the `leaderelection` directory

To start a single node run `bully.py` provide an ip address and a port as the first argument. These values must match an entry in the `server_config` file.

The `bully.py` file can run on different servers, so if you are using the log server on a different machine you must provide ip address and port of the log server as a second parameter to the program.

Usage `bully.py [ip:port] [logserver_ip:logserverport]`

The `processrunner.py` script can be used to start up all the nodes from the `server_config` file at once. This can be done on different servers, since the process runner only starts nodes with an ip entry that matches the current server.

Usage: `process_runner.py "bully" [logserverip:logserverport]`

After processes have been started you can watch the evolution to a leader over the web interface – to do so direct your browser to

<http://logserverip:logserverport/status>

Invitation Algorithm

The invitation algorithm described by Hector Garcia Molina is implemented in the file `invitation.py`. To run it start the Logserver first, then the script `invitation_proxy.py` and finally the `invitation.py` itself. If you don't specify a port number for the log server it defaults to localhost port 4711, the `invitation_proxy.py` defaults to localhost port 4712.

When you start the log server for the invitation algorithm it has to know where to send its commands for blocking the ips so you have to specify the ip address and the port of the log server.

```
Logserver.py [portnummer] [proxy_ip:proxy_port]
```

```
proxy_invitation.py [proxy_ip:proxy_port]
```

As in the bully algorithm you can start all the nodes either individually or all at once using the process _runner

Usage:

```
process_runner.py "invitation" [logserverip:logserverport] " [proxyip :proxyport]
```

Ansynchronous bully Algorithm

The asynchronous bully algorithm invented by Scott Stoller is implemented in the file `bully_async.py` and works in the same way as the invitation algorithm. The steps are exactly the same as described above, just use `bully_async.py` and `bully_async_proxy.py` instead of the corresponding invitation scripts.