01_HW_04

March 9, 2022

1 Distributed Systems (LTAT.06.007)

1.0.1 Seminar 4: Berkeley algorithm in python

Student : ChengHan Chung

1.1 Task-1: You should implement "clock-sync.py" to achieve the Berkeley algorithm.

1. Still show the list of processes running while also show the process acting as coordinator.

```
utlab@DESKTOP-SO1IIEA:04_Seminar_4_Berkeley_Algorithm$ python3 clock-sync.py input.txt
[WARNING] Duplicate procees id 5 discarded
Commands: exit, list, clock, kill
list
1, A_0
3, B_0
4, D_0
7, E_0 (Coordinator)
5, F_0
```

[ ]:
```python
def list(processes):
    # utility method to list proceeses
    for p in processes:
        if p.label == "C":
            print("%d, %s_%d %s" %
                  (p.id, p.name, p.elections, "(Coordinator)"), end="\n")
        else:
            print("%d, %s_%d" % (p.id, p.name, p.elections), end="\n")
```

2. Define a clock-update function to show the current clock time of each process. (Clocks are synchronized based on coordinator clock; we won't take the time average of each process as stated by Berkeley)

```
clock
A_0, 11:00:00
B_0, 13:33:00
D_0, 17:30:00
E_0, 23:03:14
F_0, 03:00:00
clock
A_0, 23:03:14
B_0, 23:03:14
D_0, 23:03:14
E_0, 23:03:15
F_0, 23:03:14
```

[ ]:
```python
def synchronize_time(processes):
    cordinator = 0
    for p in processes:
        if p.label == "C":
            break
        cordinator += 1
    for p in processes:
        if p.label != "C":
            p.time = processes[cordinator].time
    return processes
```

3. Define a kill function to remove a process except coordinator.

```
kill 1
list
3, B_0
4, D_0
7, E_0 (Coordinator)
5, F_0
clock
B_0, 23:03:15
D_0, 23:03:15
E_0, 23:06:06
F_0, 23:03:15
```

[ ]:
```python
def kill(kill_p, processes):
    i = 0
    for p in processes:
        if int(p.id) == kill_p and p.label != "C":
            processes.pop(i)
            break
        i += 1
    return processes
```

1.2 Task-2: In Seminar 3, we already learn to implement Exponential back-off function with multi-threads by using RPyC ThreadedServer in python. For this task:

1. Integrate the function of Task-1 into RPyC server side.
2. Pass arguments ("input.txt" and different commands) from client to server.

Client Side

```
utlab@DESKTOP-SO1IIEA:04_Seminar_4_Berkeley_Algorithm$ python3 rpyc_client.py localhost
Input the file name: input.txt
Input the command: list
Input the command: clock
Input the command: exit
```

Server Side

```
utlab@DESKTOP-SO1IIEA:04_Seminar_4_Berkeley_Algorithm$ python3 rpyc_server.py

connected on 2022-03-09 12:20:57.069251
Receive file name from client: input.txt
[WARNING] Duplicate procees id 5 discarded
Commands: exit, list, clock, kill
Receive command from client: list
1, A_0
3, B_0
4, D_0
7, E_0, (Coordinator)
5, F_0
Receive command from client: clock
A_0, 11:00:00
B_0, 13:33:00
D_0, 17:30:00
E_0, 23:00:00
F_0, 03:00:00
Receive command from client: exit
Program exited

disconneced on 2022-03-09 12:20:57.069251
```