01_HW_03

March 2, 2022

1 Distributed Systems (LTAT.06.007)

1.0.1 Seminar 3: Exponential back-off (Client/Server) in python

Student : ChengHan Chung

1.1 1. Server works

1.1.1 1.1 Server works (with reconnection message)

Server side

```
utlab@DESKTOP-SO1IIEA:03_Seminar_3_Exponential_back-off(ClientServer)$ python3 rpyc_server_works.py
Fail, Retrying in 3 seconds...

connected on 2022-03-02 16:15:36.335853
```

Client side

```
utlab@DESKTOP-SO1IIEA:03_Seminar_3_Exponential_back-off(ClientServer)$ python3 retry_works.py localhost
Success: it works!
```

1.1.2 1.2 Server works (without reconnection message)

Server side

```
utlab@DESKTOP-SO1IIEA:03_Seminar_3_Exponential_back-off(ClientServer)$ python3 rpyc_server_works.py

connected on 2022-03-02 16:19:52.436414
```

Client side

```
utlab@DESKTOP-SO1IIEA:03_Seminar_3_Exponential_back-off(ClientServer)$ python3 retry_works.py localhost
Success: it works!
```

1.2 2. Server breaks

Server side

```
utlab@DESKTOP-SO1IIEA:03_Seminar_3_Exponential_back-off(ClientServer)$ python3 rpyc_server_works.py
Fail, Retrying in 3 seconds...
Fail, Retrying in 6 seconds...
Fail, Retrying in 12 seconds...
```

Client side

```
Traceback (most recent call last):
  File "retry_works.py", line 11, in <module>
    conn.root.test_random()
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/netref.py", line 240, in __call__
    return syncreq(_self, consts.HANDLE_CALL, args, kwargs)
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/netref.py", line 63, in syncreq
    return conn.sync_request(handler, proxy, *args)
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/protocol.py", line 473, in sync_request
    return self.async_request(handler, *args, timeout=timeout).value
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/async_.py", line 102, in value
    raise self._obj
Exception: Fail

========= Remote Traceback (1) =========
Traceback (most recent call last):
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/protocol.py", line 324, in _dispatch_request
    res = self._HANDLERS[handler](self, *args)
  File "/home/utlab/.local/lib/python3.8/site-packages/rpyc/core/protocol.py", line 592, in _handle_call
    return obj(*args, **dict(kwargs))
  File "rpyc_server_works.py", line 30, in f_retry
    return f(*args, **kwargs)
  File "rpyc_server_works.py", line 40, in exposed_test_random
    raise Exception("Fail")
Exception: Fail


During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "retry_works.py", line 14, in <module>
    raise Exception("Fail")
Exception: Fail
```

1.3 Server side code

[ ]:
```python
import rpyc
from rpyc.utils.server import ThreadedServer
import datetime
import time
from functools import wraps
import random

date_time = datetime.datetime.now()


class MonitorService(rpyc.Service):
  def retry(ExceptionToCheck, tries=4, delay=3, backoff=2, logger=None):
      def deco_retry(f):
          @wraps(f)
          def f_retry(*args, **kwargs):
              mtries, mdelay = tries, delay
              while mtries > 1:
                  try:
                      return f(*args, **kwargs)
                  #except ExceptionToCheck, e:
                  except ExceptionToCheck as e :
                      msg = "%s, Retrying in %d seconds..." % (str(e), mdelay)
                      if logger:
                          logger.warning(msg)
                      else:
```

```
                        print(msg)
                    time.sleep(mdelay)
                    mtries -= 1
                    mdelay *= backoff
            return f(*args, **kwargs)

        return f_retry

    return deco_retry # true decorator

@retry(Exception, tries=4)
def exposed_test_random(self):
    x = random.random()
    if x < 0.5:
        raise Exception("Fail")
    else:
        print("\nconnected on {}".format(date_time))

if __name__ == '__main__':

  t = ThreadedServer(MonitorService, port=18812)
  t.start()
```

1.4 Client side code

[ ]:
```
import rpyc
import sys

if len(sys.argv) < 2:
   exit("Usage {} SERVER".format(sys.argv[0]))

server = sys.argv[1]

try:
    conn = rpyc.connect(server,18812)
    conn.root.test_random()
    print('Success: it works!')
except Exception as e:
    raise Exception("Fail")
```