Exercises: RPC

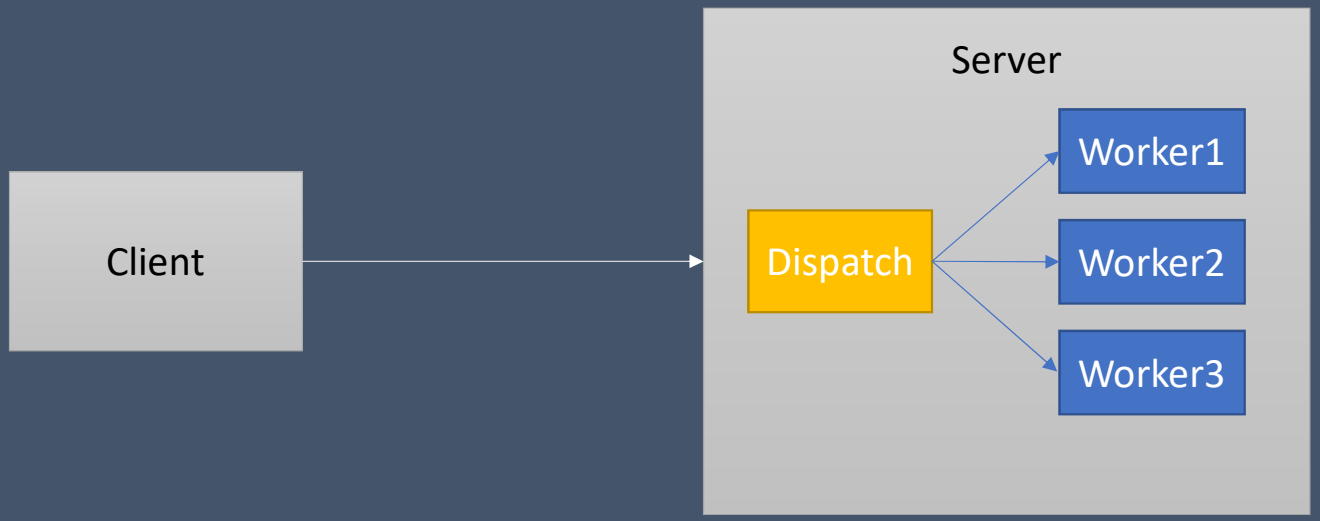
- Today's goal: implement our own RPC server
- Communication: Zero MQ sockets
- Serialization: pickle
- Test-driven approach:
  - Unit tests given
  - Write an RPC server, that satifies the unit tests
- DS_Examples/myrpc/myrpc_client_test.py

Exercises: RPC

1. Write the corresponding RPC server using Zero MQ sockets that satifies all the unit tests

Exercises: Multithreading

- Multithreaded servers



Exercises: Multithreading

- Server creates a thread-pool

```
import threading

for x in range(n):
     thread = threading.Thread(target = worker_fn, args=(x,y))
     thread.start()
```

- Each thread “waits” for requests listening to a given socket
- Zero MQ sockets are **not** thread-safe
- MT recommendation #1: **do not share state between threads!**

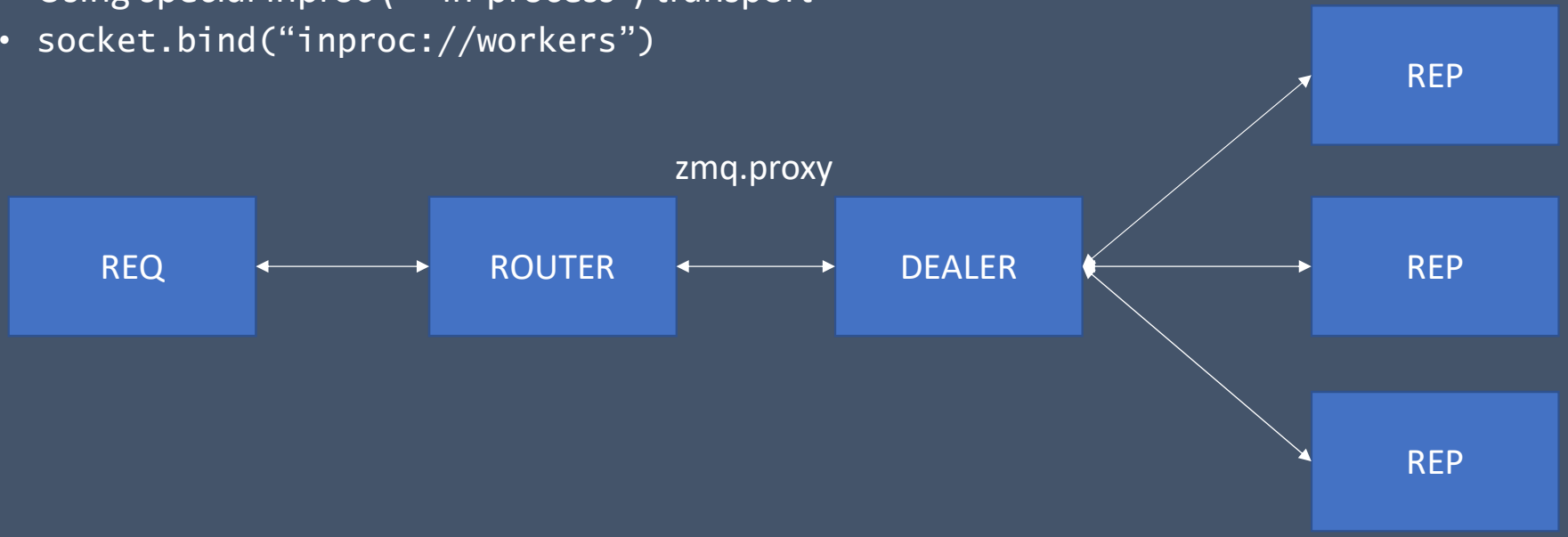
Exercises: Multithreading

- REQ/REP pattern needs to share REP socket between threads

- New pattern needed for multiple workers

Exercises: Multithreading

- DEALER/ROUTER sockets
  - ROUTER: talks to the clients
  - DEALER: talks internally with the worker-threads
    - Using special inproc (“= in-process”) transport
    - `socket.bind(“inproc://workers”)`



Exercises: Multithreading

2. Convert the server into a multithreading server where:
   - The server holds a thread pool of 10 threads
   - Each request is handled by a thread of the thread pool
   - The communication between threads and between client and server is handled by the REQ/ROUTER/DEALER/REP pattern