zmq.rs

A brief history of concurrency in Rust

https://github.com/zeromq/zmq.rs

http://zeromq.org/

rust-zmq

https://github.com/erickt/rust-zmq

libgreen

Optional runtime on top of libuv in Rust. There's also libnative.

green thread

or
micro thread
or
coroutine

for example

```
while True:
    future = socket.recv(1024)
    data = yield from future
    print(data)
    future = socket.send(data)
    status = yield from future
```

native thread

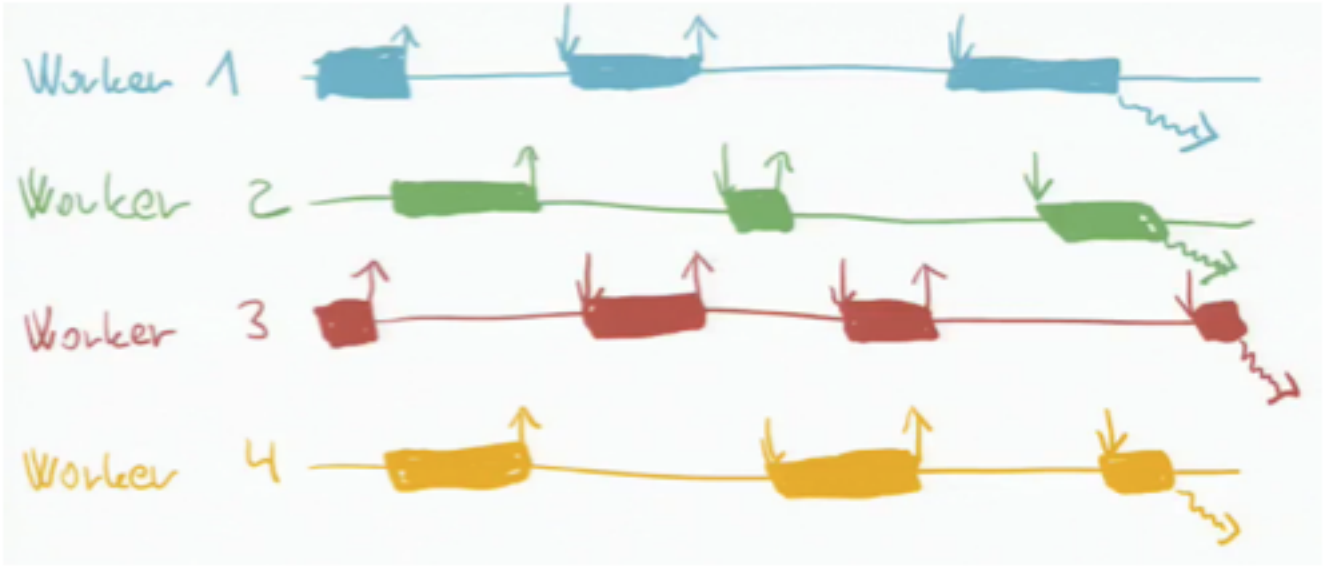
- OS-managed
- preemptive
- thousands
- low latency

green thread

- user-managed
- cooperative
- hundreds of thousands
- depends

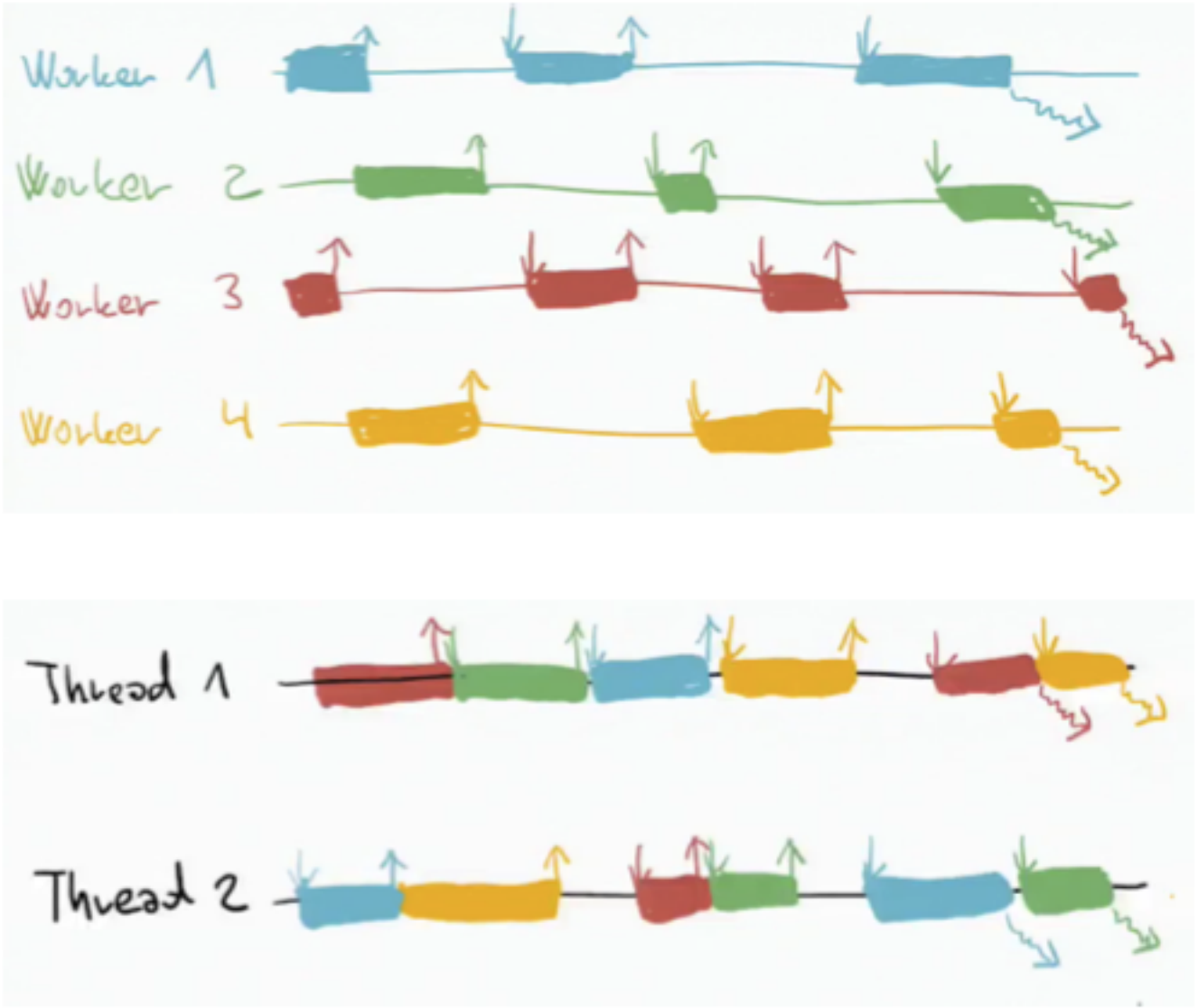
M:N in Rust

4 green threads,



http://stackoverflow.com/questions/22621514

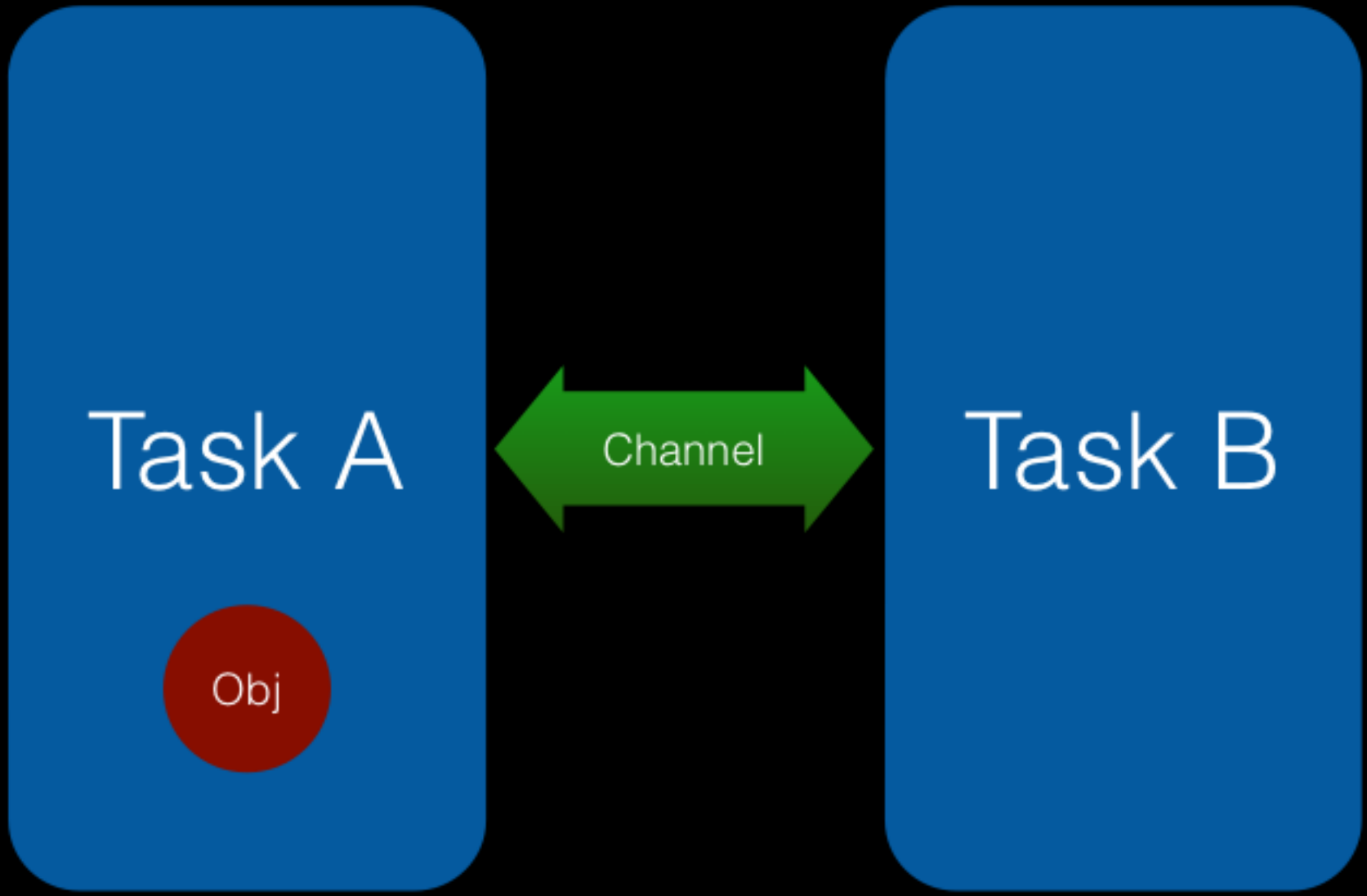
in 2 OS threads



http://stackoverflow.com/questions/22621514

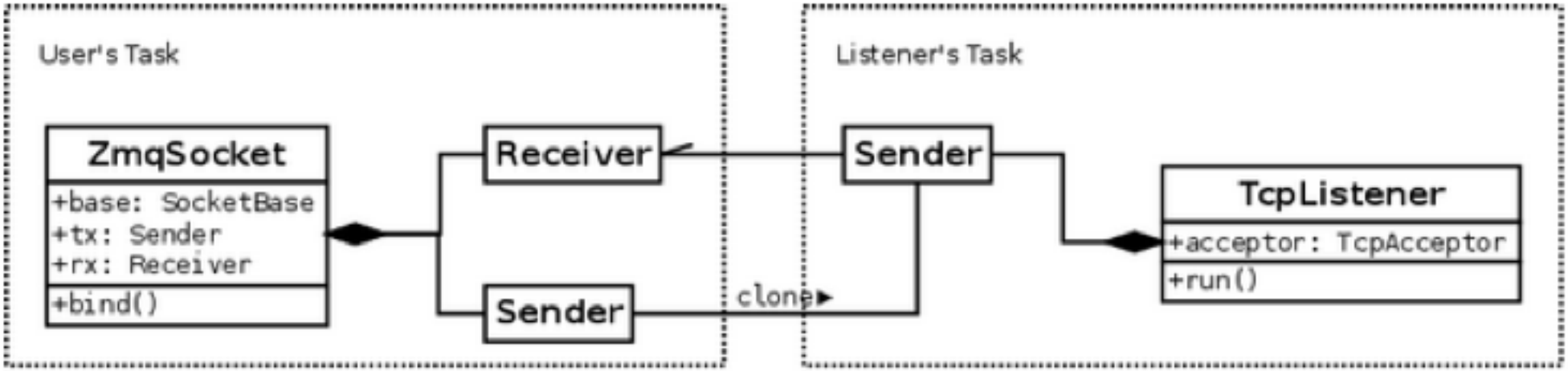
multi-core, async,
memory-safe

Task in Rust

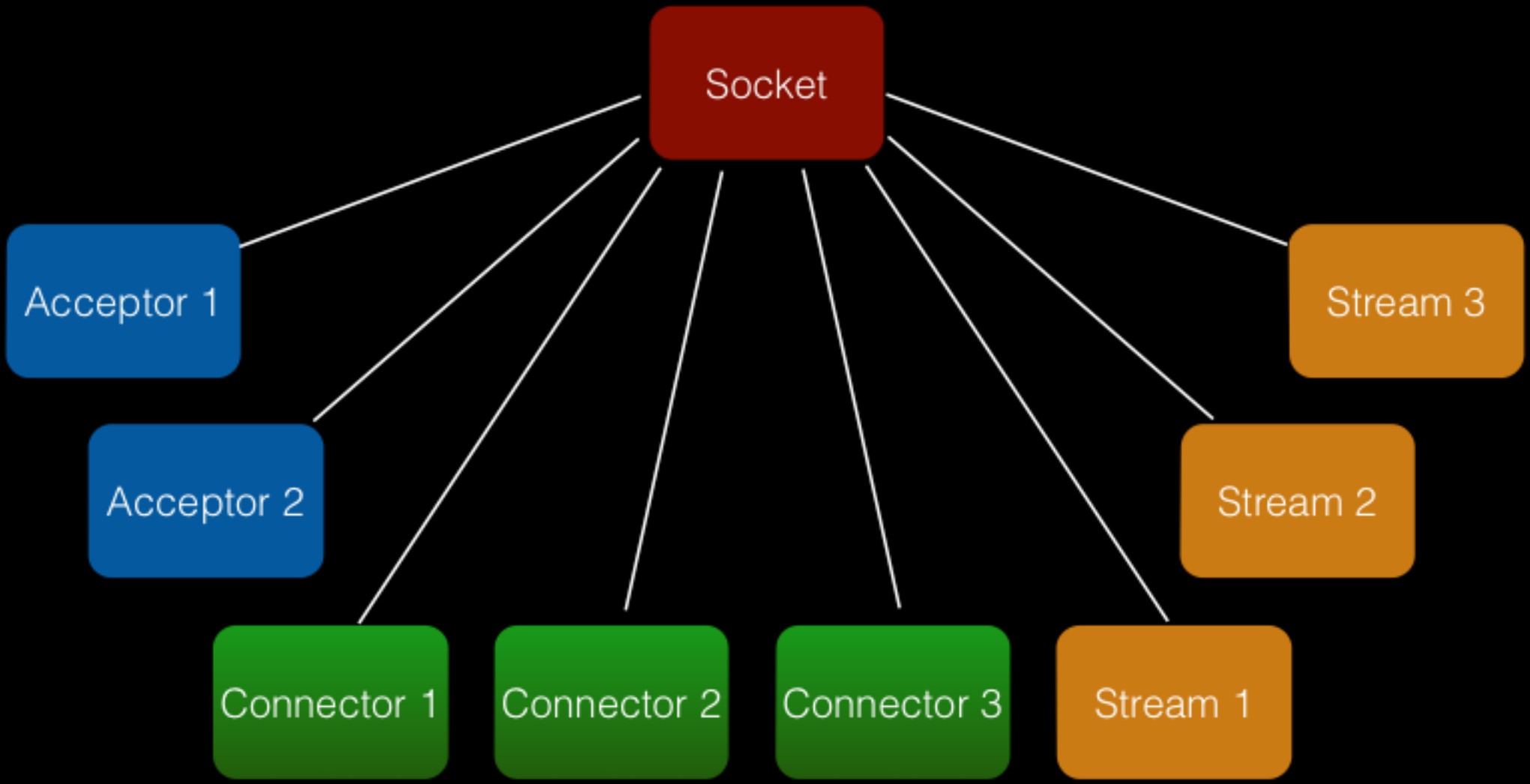


using Tasks

Example: listen TCP



Overview of zmq.rs



Later ...

1:1 became default

M:N remained optional

Issues with the unity

Issues with the unity of 1:1 and M:N

- Cost is huge, to keep a unified interface
- Lost a lot of possibilities and flexibility
- libgreen is not green

libgreen is not green

libgreen -> green-rs

https://github.com/alexcrichton/green-rs

Task -> Thread

concurrency -> ...

doomed!

doomed?

concurrency ->

- 1:1
- mio
- generator
- green-rs
- …more

https://github.com/carllerche/mio

mio in example

mio in example

```rust
#![allow(unstable)]

use mio::{EventLoop, Handler};
use std::time::Duration;

struct MyHandler;

impl Handler<u32, ()> for MyHandler {
    fn timeout(&mut self, event_loop: &mut EventLoop<u32, ()>, timeout: u32) {
        assert_eq!(timeout, 123);
        event_loop.shutdown();
    }
}


let mut event_loop = EventLoop::new().unwrap();
let timeout = event_loop.timeout(123, Duration::milliseconds(300)).unwrap();
let _ = event_loop.run(MyHandler);
```

next steps in zmq.rs

Next steps in Erlang

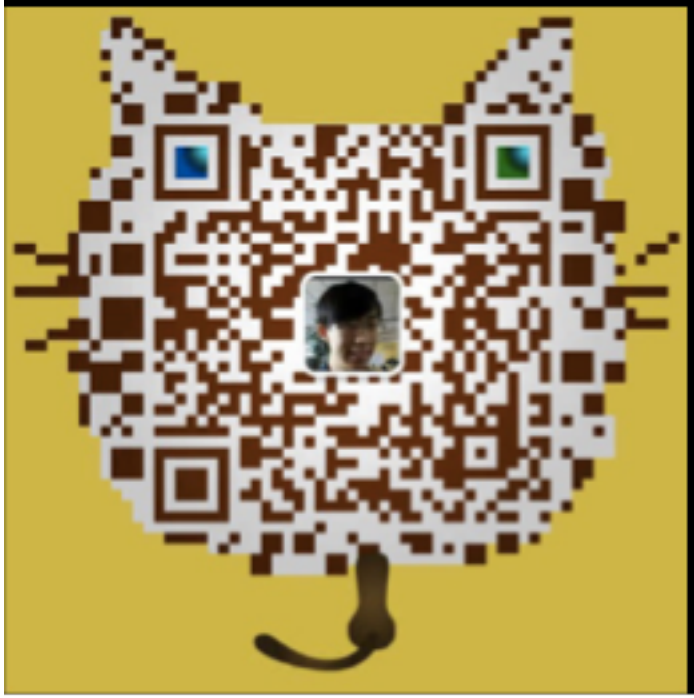
- Rewrite with MIO
- Follow the design of libzmq in C++
  - A few worker threads, each with an event loop
  - Sockets mapped to workers manually

Thank you!

Thank you!

zmq.rs

A brief history of concurrency in Rust



http://about.me/fantix

rust.cc 中文社区 QQ 群：144605258

https://github.com/zeromq/zmq.rs