You Aren't a One Man Army: Introducing 0MQ

Wolf Richter

Don't waste time in the future reinventing the wheel.

This is an engineering snafu.

If ZeroMQ didn't exist, it would be necessary to invent it. ZeroMQ simply seems to me a "bare necessity" nowadays.

Gonzalo Diethlem

The more time I spend with ZeroMQ, the less I can think of a reason I'd ever have to open up a raw TCP or UDP socket.

Andrew Cholakian

ZeroMQ: Panacea?

- 30+ Languages: C, C++, Python, Java...
- Transport: inproc, IPC, TCP, multicast
- Patterns: req-rep, pub-sub, push-pull, ...
- Async by design: separate IO thread
- Built for speed: originally for stock trading
- OS-agnosticism: Linux, Windows, OS X
- Vibrant community, active development
- Linux Kernel someday soon?

ZeroMQ, Zero Setup

- Versus: Qpid, OpenAMQ, RabbitMQ, *MQ
- No middleware
- No messaging broker (lose persistance)
- Embedded, linked library
- Messaging fabric becomes part of app

Which brings us back to the science of programming. To fix the world, we needed to do two things.

One, to solve the general problem of "how to connect any code to any code, anywhere".

Two, to wrap that up in the simplest possible building blocks that people could understand and use easily.

Usage: zguide mostly in C

http://zguide.zeromq.org/

Use ZeroMQ 2.1 Stable

ZeroMQ is a new way of thinking about concurrency, multicore systems, distributed systems, and network programming.

It changes your world view.

Not many libraries can do that...

Request-reply

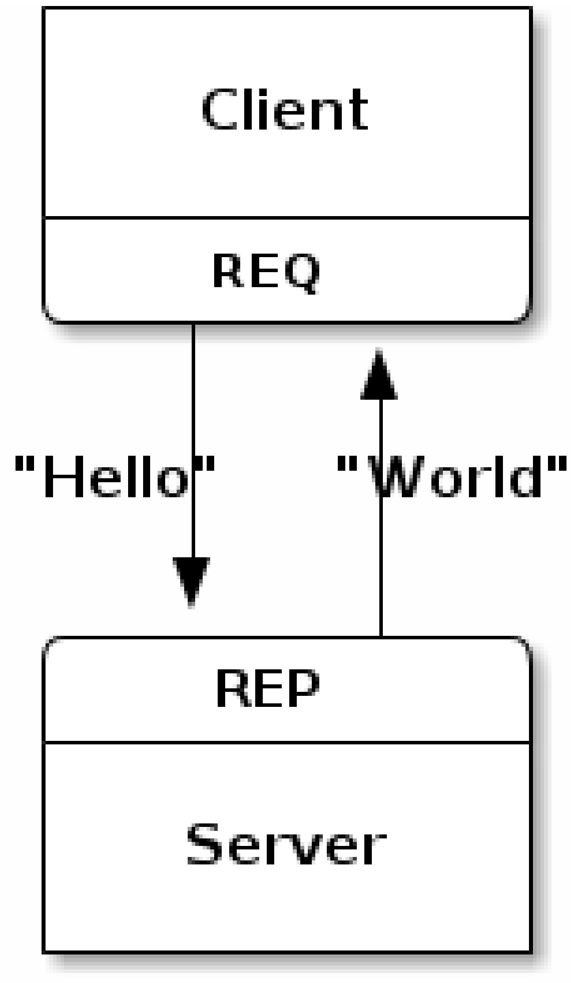


Figure 1 — Request-Reply

Publish-Subscribe

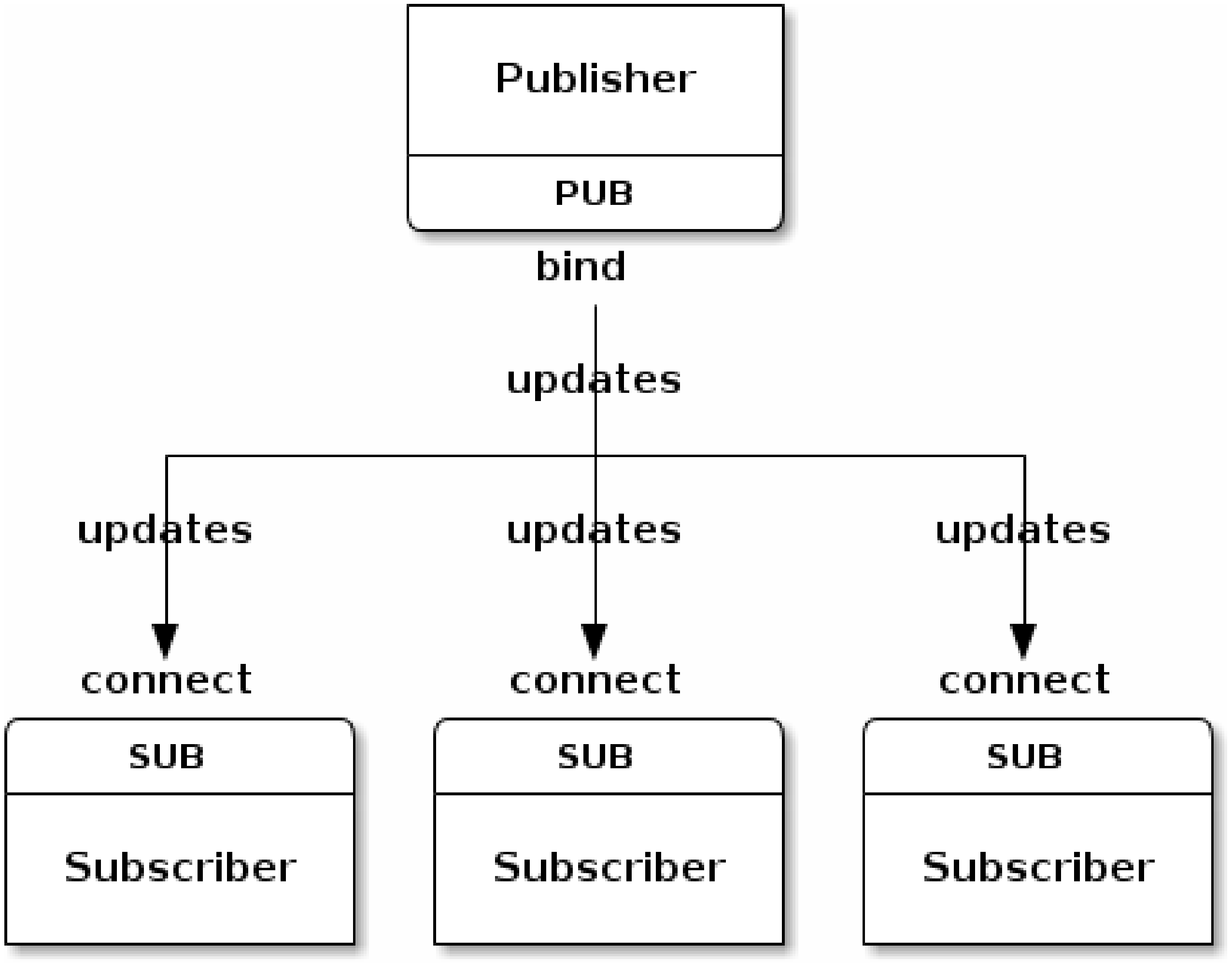


Figure 4 — Publish-Subscribe

Pipeline or Push-Pull

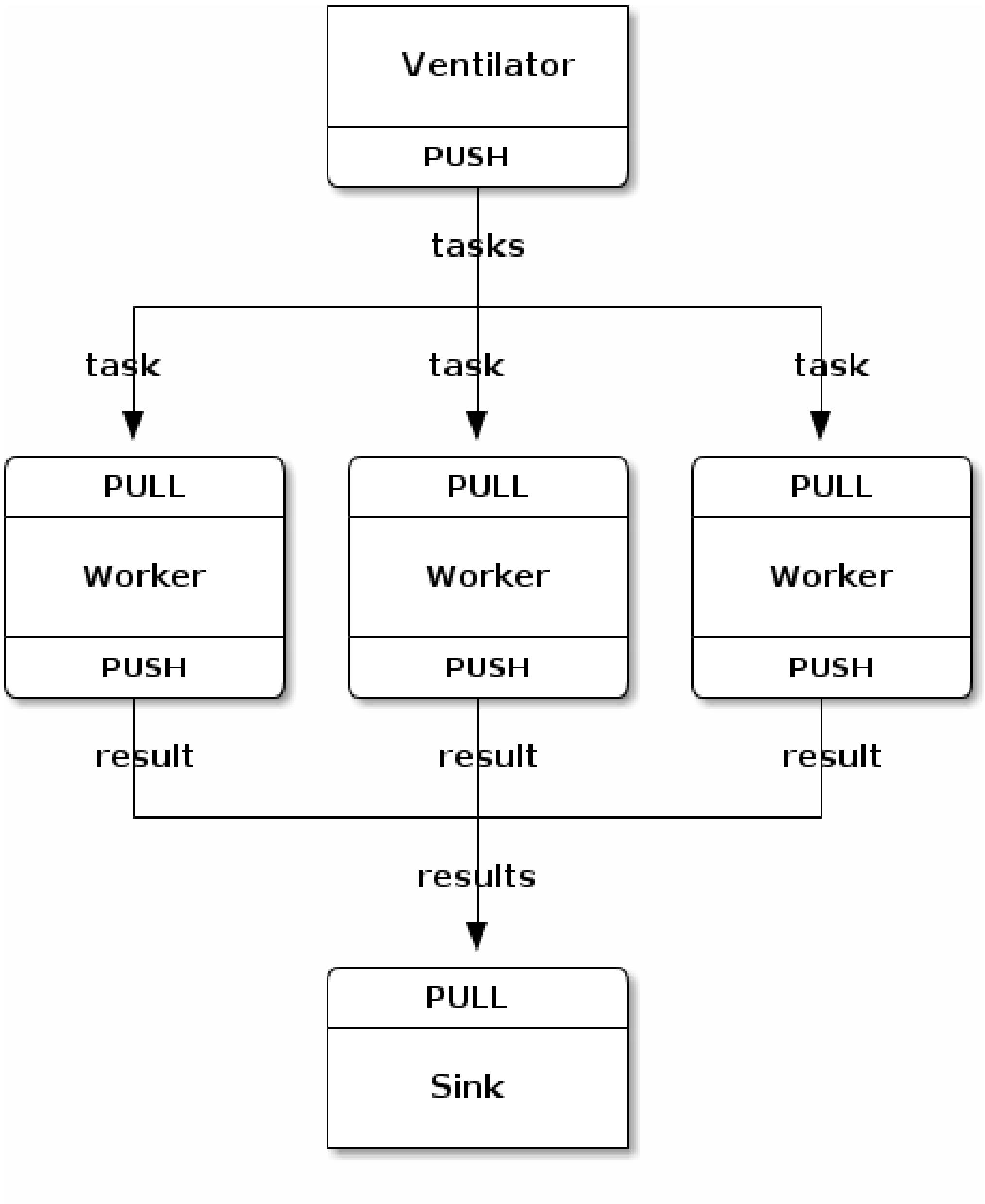


Figure 5 – Parallel Pipeline

Multicore, Multithreading?

ZeroMQ

we don't need mutexes, locks, or any other form of inter-thread communication except messages sent across ØMQ sockets

Network Programming?

ZeroMQ

It gives you sockets that carry whole messages across various transports like in-process, inter-process, TCP, and multicast.

You can connect sockets N-to-N with patterns like fanout, pub-sub, task distribution, and request-reply.

Use all cores and machines?

ZeroMQ

It presents a familiar BSD socket API but that hides a bunch of message-processing machines that will slowly fix your world-view about how to design and write distributed software.

ZeroMQ Keeps on Giving

- Great open source community example
- Excellent documentation
- Superbly engineered C++ core
- Very active mailing list

Publisher in C (1)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "zmq.h"

int main (void)
{
    void *context = zmq_init(1);
    void *publisher = zmq_socket(context, ZMQ_PUB);
    zmq_bind(publisher, "tcp://*:5556");

    srand((unsigned) time(NULL));
    while(1) {

        int zipcode, temperature, relhumidity;
        zipcode     = rand() % 100000;
        temperature = (rand() % 215) - 80;
        relhumidity = (rand() % 50) + 10;

        char update[20];
```

Publisher in C (2)

```
        sprintf(update, "%05d %d %d", zipcode, temperature, relhumidity);
        zmq_msg_t message;
        zmq_msg_init_size(&message, strlen(update));

        memcpy(zmq_msg_data(&message), update, strlen(update));

        zmq_send(publisher, &message, 0);
        zmq_msg_close(&message);
    }

    zmq_close(publisher);
    zmq_term(context);
    return 0;
}
```

Subscriber in Python (1)

```
#!/usr/bin/env python
import sys
import zmq

context = zmq.Context()
socket = context.socket(zmq.SUB)

socket.connect("tcp://localhost:5556")

filter = "10001"
socket.setsockopt(zmq.SUBSCRIBE, filter)
```

Subscriber in Python (2)

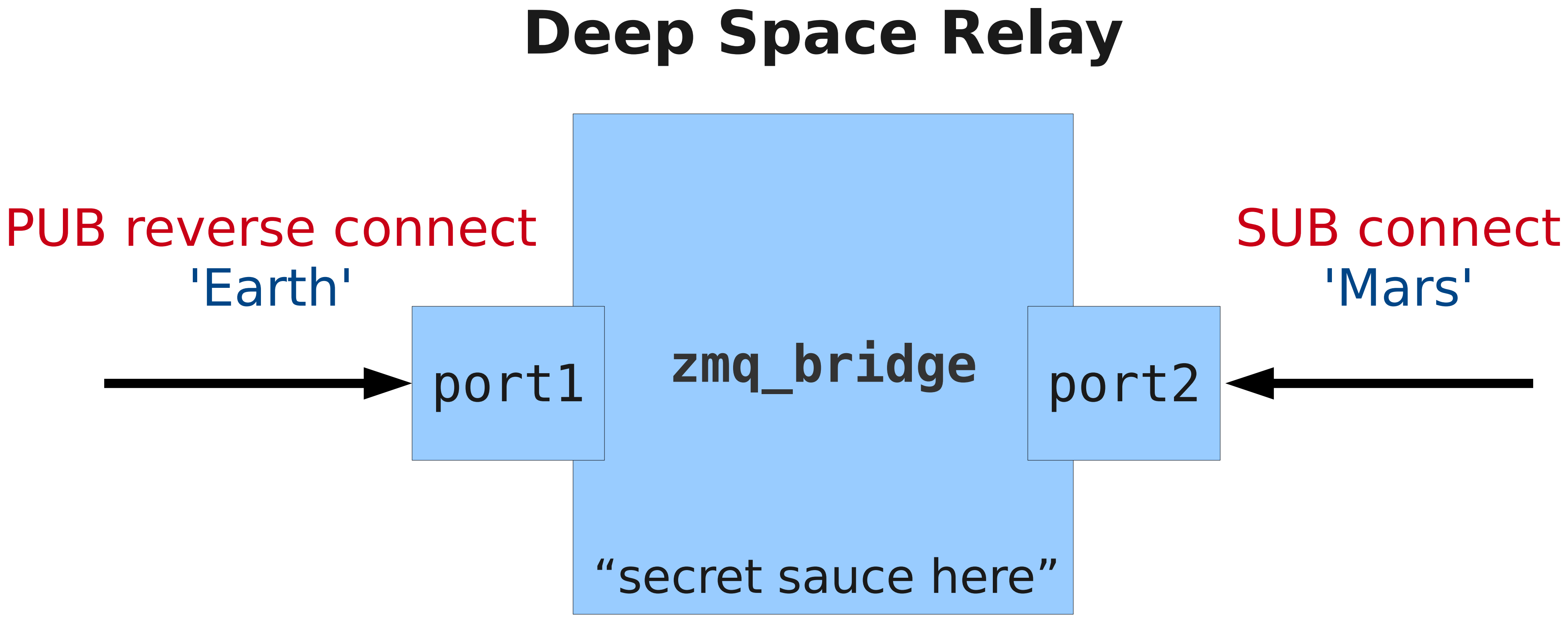
```
total_temp = 0
for update_nbr in range(5):
    string = socket.recv()
    print string
    zipcode, temperature, relhumidity =\
    string.split()
    total_temp += int(temperature)

print "Average temperature was %dF" % (
      total_temp / update_nbr)
```

PJ3 Extra Credit [10 Points]

- Create a ZeroMQ bridge w/ your protocol
- **Email Wolf telling you did this...**
- Use the reliable data transport protocol
- ZMQ message size cap at 256MB
- Produce a '`zmq_bridge`' executable on '`make ec`'
- Take two parameters:
  - '`zmq_bridge <port1> <port2>`'
- `port1` – SUB socket
- `port2` – PUB socket

PJ3 EC Picture



Don't waste time in the future reinventing the wheel.

This is an engineering snafu.

Fork me on GitHub

GitHub:

Git it, got it, good.

```
git clone git://github.com/theonewolf/15-441-Recitation-Sessions.git
```