Computer Network

Lecture 9

Transport Layer Part.2

2019. 03. 01

Sungwon Lee

Department of Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Contents

- **Socket Programming**
- ZeroMQ
- RabbitMQ
- Book Recommendation

Network Socket

- An internal endpoint for sending or receiving data within a node on a computer network.
- Concretely, it is a representation of this endpoint in networking software (protocol stack), such as an entry in a table (listing communication protocol, destination, status, etc.), and is a form of system resource.

Socket Programming

- The fundamental technology behind communications on TCP/IP networks.
- The socket provides a bidirectional communication endpoint for sending and receiving data with another socket.
- Socket connections normally run between two different computers on a local area network (LAN) or across the internet, but they can also be used for interprocess communication on a single computer.

Reference: https://www.lifewire.com/definition-of-socket-817934

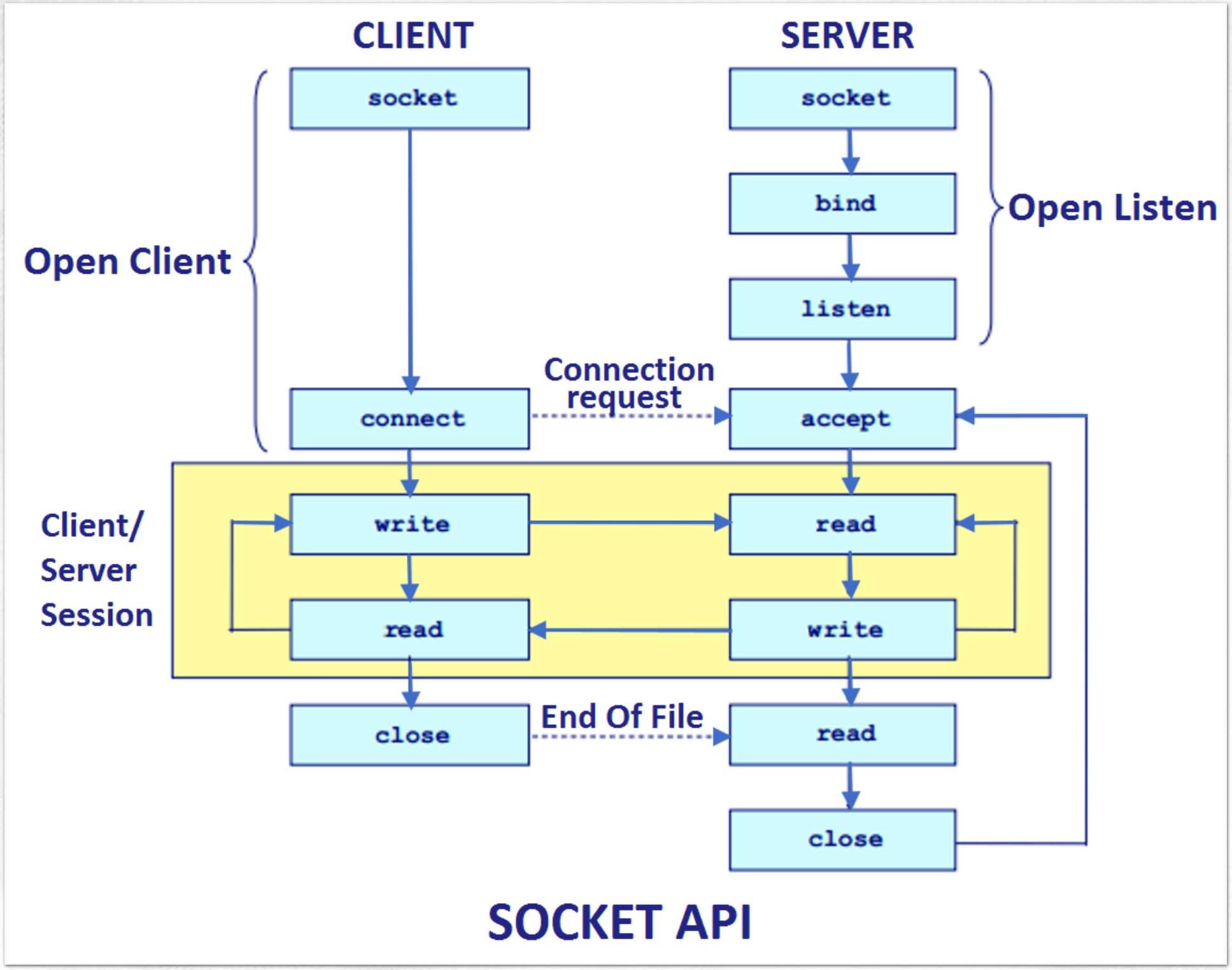
Client and Server Architecture

- Socket Server: Any program that listens for other programs to connect to it. For example the Web Server, or your Email Server, etc.
- Socket Client: Any program that is designed to connect to another program. For example, the Web Browser you are using to read this answer. Your email app on your phone. Most all Internet games on mobile devices. (Some run standalone, meaning no multi-user chat, published leader board, etc).

Reference:https://www.quora.com/What-is-socket-programming-a-socket-server-and-a-socket-client

Socket Programming

Procedures



Reference: https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/02dd04be-0dbb-4732-8bc5-1961644e8875.xhtml

Echo Server in Python

```python
#!/usr/bin/env python3

import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 65432        # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```


Reference: https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/02dd04be-0dbb-4732-8bc5-1961644e8875.xhtml

Echo Client in Python

```
#!/usr/bin/env python3

import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 65432        # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b'Hello, world')
    data = s.recv(1024)

print('Received', repr(data))
```


Reference: https://www.oreilly.com/library/view/distributed-computing-in/9781787126992/02dd04be-0dbb-4732-8bc5-1961644e8875.xhtml

Echo Client & Server in C

https://gist.github.com/suyash/2488ff6996c98a8ee3a84fe3198a6f85

Reference: https://gist.github.com/suyash/2488ff6996c98a8ee3a84fe3198a6f85

Contents

- Socket Programming
- **ZeroMQ**
- RabbitMQ
- Book Recommendation

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Concept

- ZeroMQ (also spelled ØMQ, 0MQ or ZMQ) is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ZeroMQ system can run without a dedicated message broker. The library's API is designed to resemble that of Berkeley sockets.
- ZeroMQ is developed by a large community of contributors, founded by iMatix, which holds the domain name and trademarks. There are third-party bindings for many popular programming languages.

Reference: https://en.wikipedia.org/wiki/ZeroMQ

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Concept

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Features

- The ZeroMQ API provides sockets (a kind of generalization over the traditional IP and Unix domain sockets), each of which can represent a many-to-many connection between endpoints. Operating with a message-wise granularity, they require that a messaging pattern be used, and are particularly optimized for that kind of pattern.
- Each pattern defines a particular network topology. Request-reply defines a so-called "service bus", publish-subscribe defines a "data distribution tree", and push-pull defines "parallelised pipeline". All the patterns are deliberately designed in such a way as to be infinitely scalable and thus usable on Internet scale.

Reference: https://en.wikipedia.org/wiki/ZeroMQ

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Patterns

- **Request–reply**: Connects a set of clients to a set of services. This is a remote procedure call and task distribution pattern.
- **Publish–subscribe**: Connects a set of publishers to a set of subscribers. This is a data distribution pattern.
- **Push–pull (pipeline)**: Connects nodes in a fan-out / fan-in pattern that can have multiple steps, and loops. This is a parallel task distribution and collection pattern.
- **Exclusive pair**: Connects two sockets in an exclusive pair. (This is an advanced low-level pattern for specific use cases.)

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Example: client & worker

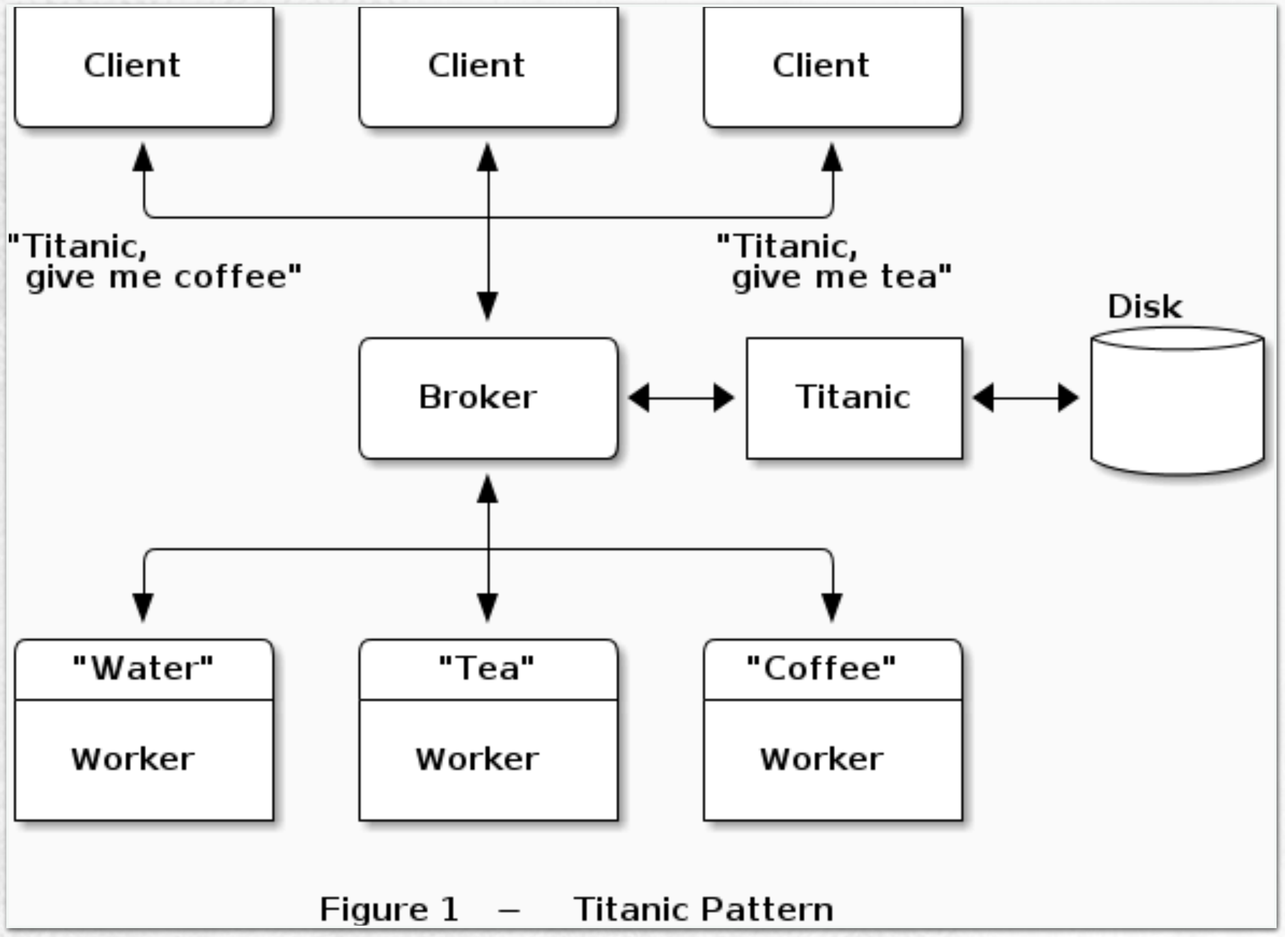
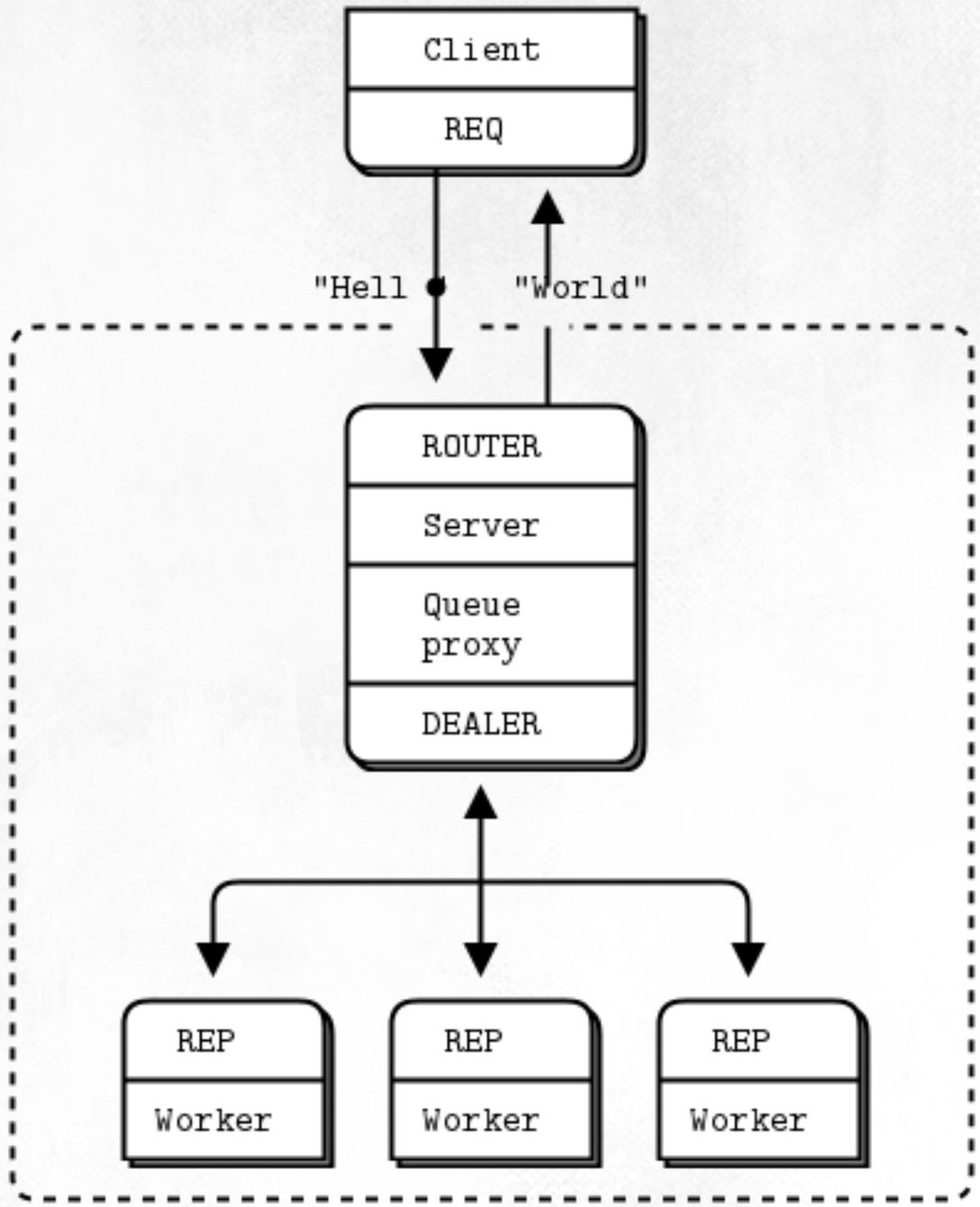


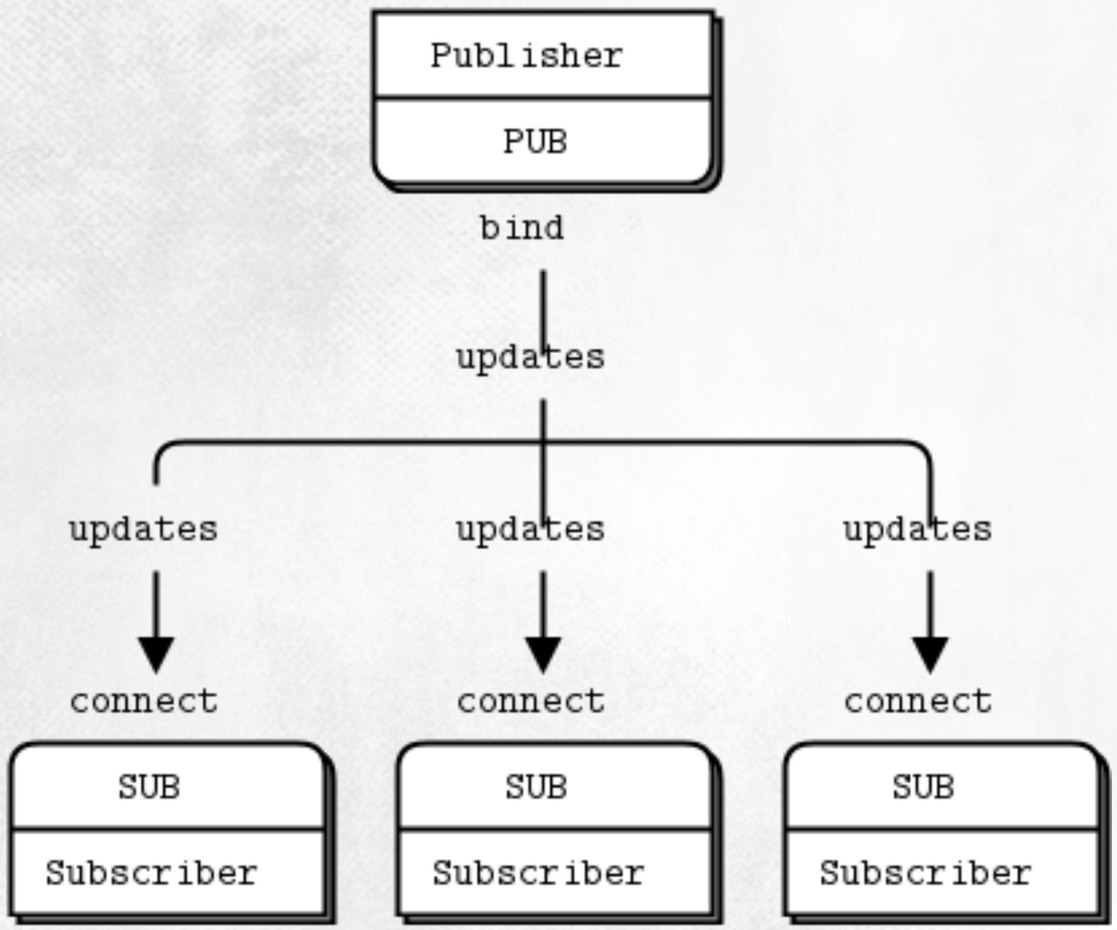
Figure 1 – Titanic Pattern

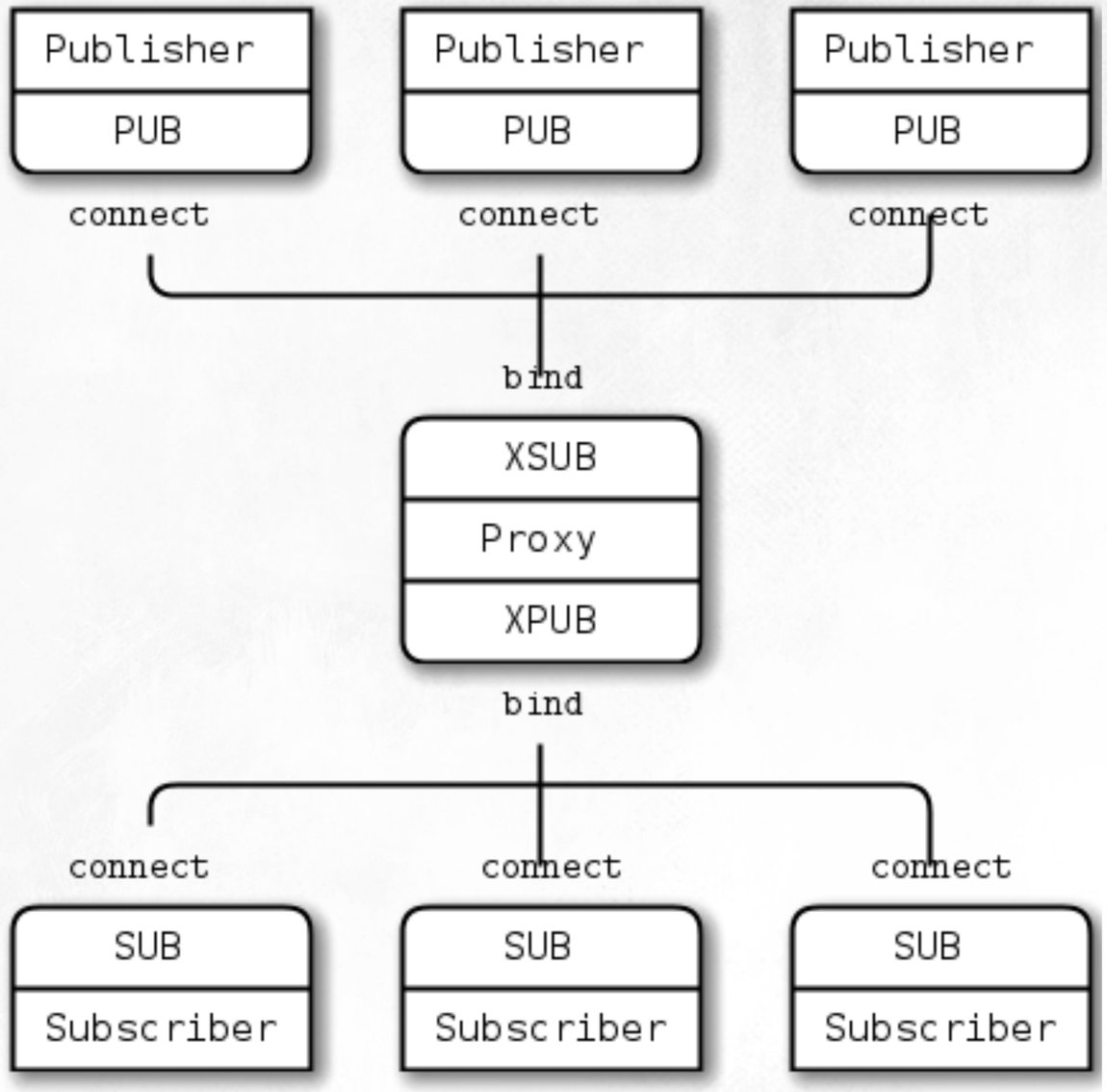


Reference: http://u.delta9.pl/k/ReadThis/%C3%98MQ%20-%20The%20Guide%20-%20%C3%98MQ%20-%20The%20Guide.html
https://rfc.zeromq.org/spec:9/TSP/

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Example: publish & scribe





Reference: https://stackoverflow.com/questions/14516102/how-to-implement-pub-sub-network-with-a-proxy-by-using-xpub-and-xsub-in-zeromqj
http://landcareweb.com/questions/8702/zeromqzhong-pub-subhe-push-pullmo-shi-zhi-jian-de-qu-bie

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Example: publish & scribe code

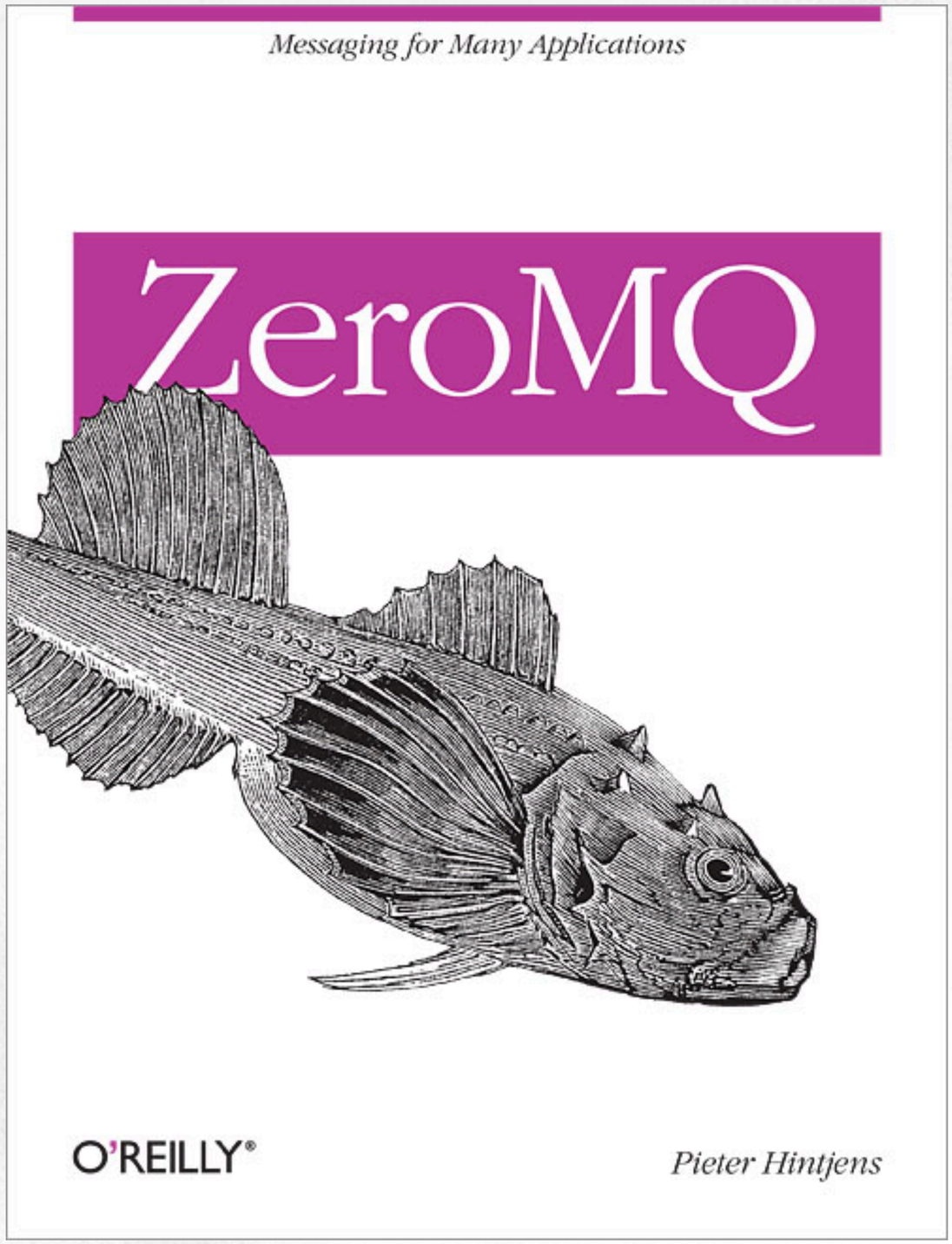
This shows a filter that collects data from two different remote publishers and sends it to local subscribers:

```
import zmq
import time
context = zmq.Context()

subscriber = context.socket (zmq.SUB)
subscriber.connect ("tcp://192.168.55.112:5556")
subscriber.connect ("tcp://192.168.55.201:7721")
subscriber.setsockopt (zmq.SUBSCRIBE, "NASDAQ")

publisher = context.socket (zmq.PUB)
publisher.bind ("ipc://nasdaq-feed")

while True:
    message = subscriber.recv()
    publisher.send (message)
```


Reference: http://u.delta9.pl/k/ReadThis/%C3%98MQ%20-%20The%20Guide%20-%20%C3%98MQ%20-%20The%20Guide.html
https://rfc.zeromq.org/spec:9/TSP/

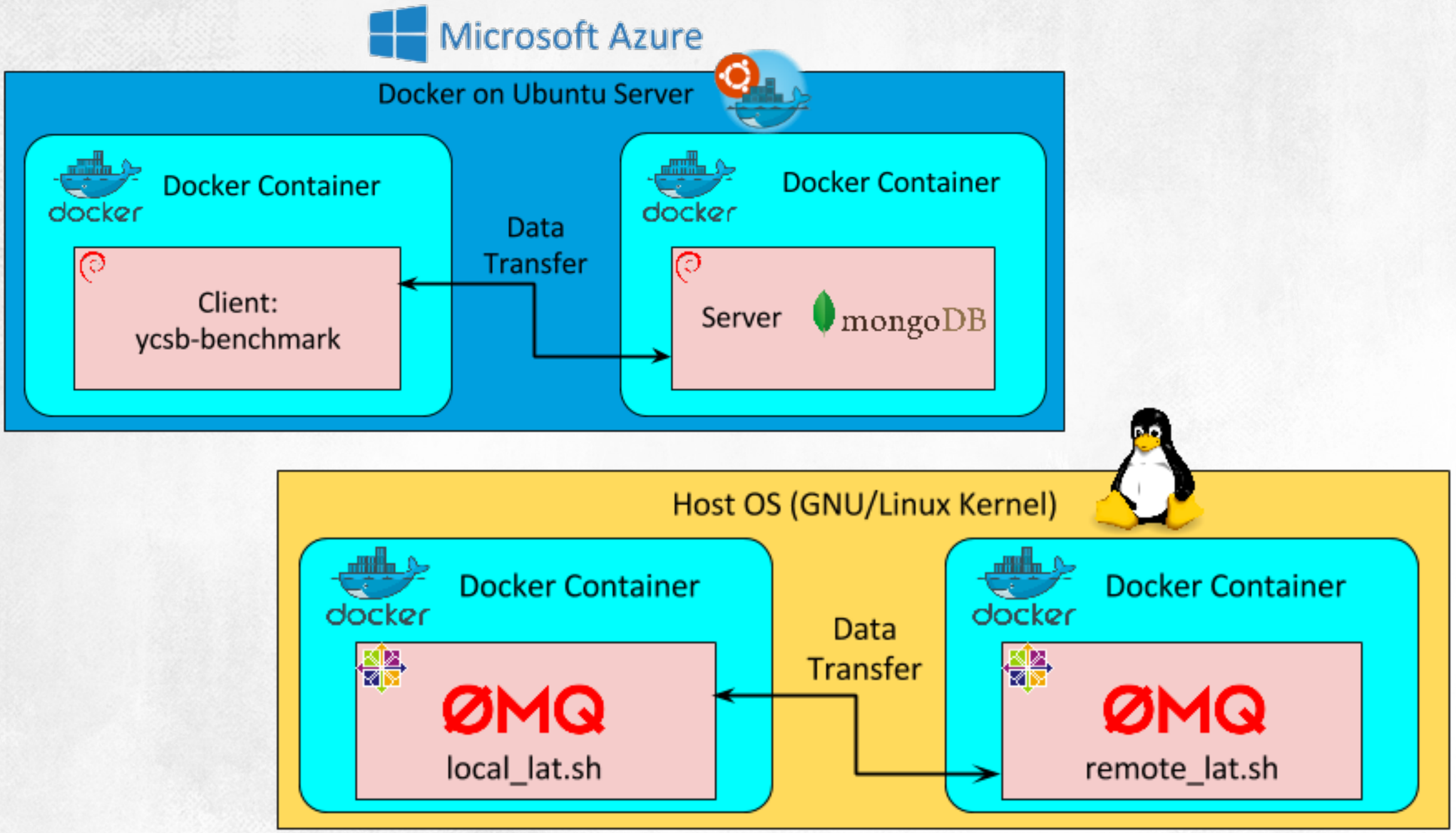
ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Reference Recommendation



ZeroMQ (also spelled ØMQ, 0MQ or ZMQ)

Coming Cloud Computing with ZeroMQ

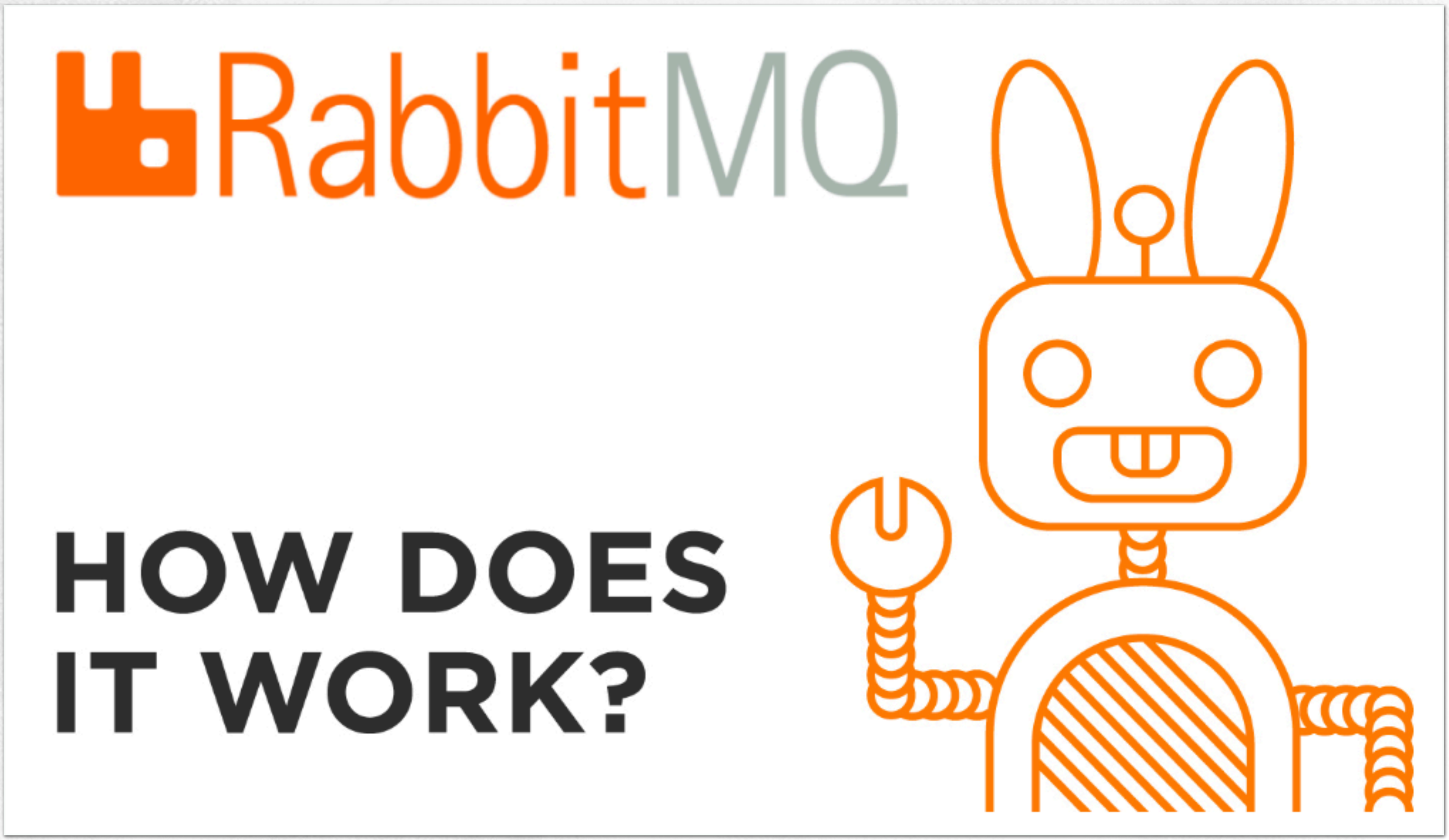


Reference: http://speedus.torusware.com/docs/config/app-setup-examples/

Contents

- Socket Programming
- ZeroMQ
- **RabbitMQ**
- Book Recommendation

Centralized Message Queue



Reference: https://www.rabbitmq.com/#features

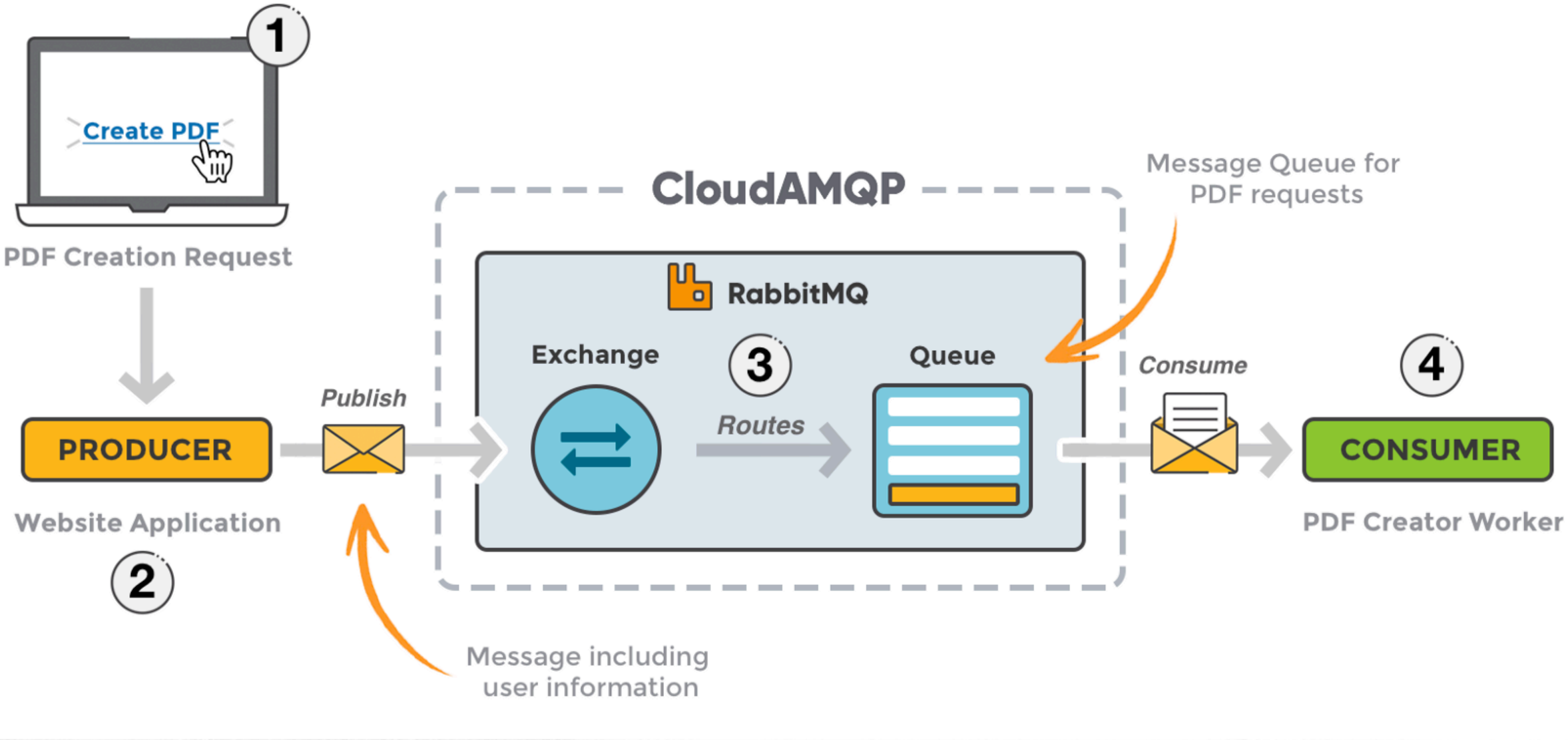
Centralized Message Queue

- RabbitMQ is an open source message broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols.
- The RabbitMQ server program is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

Reference: https://en.wikipedia.org/wiki/RabbitMQ

RabbitMQ

Centralized Message Queue

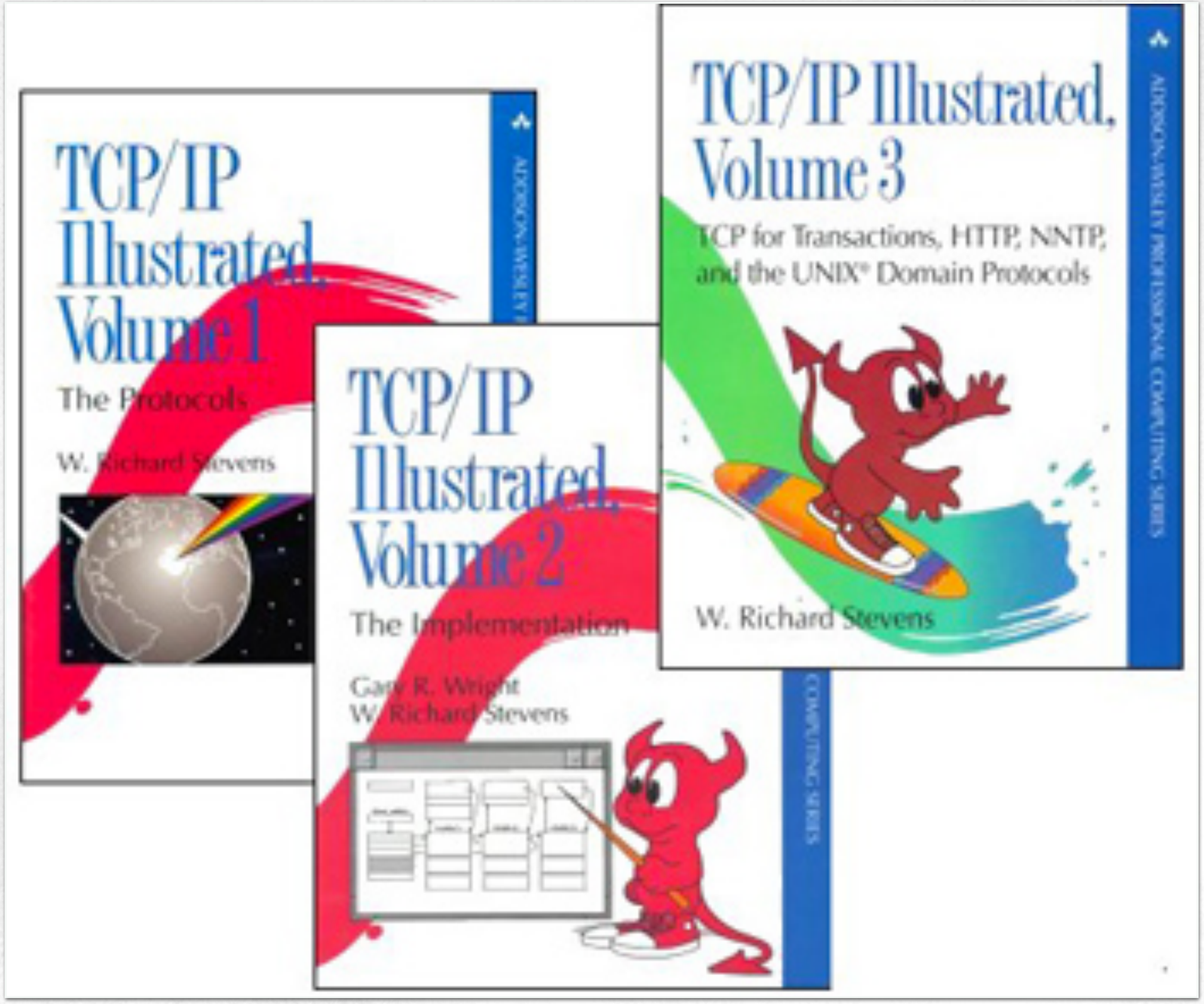


Reference: https://medium.com/naukri-engineering/naukriengineering-rabbitmq-messaging-that-just-works-dafcd0cfd194

Contents

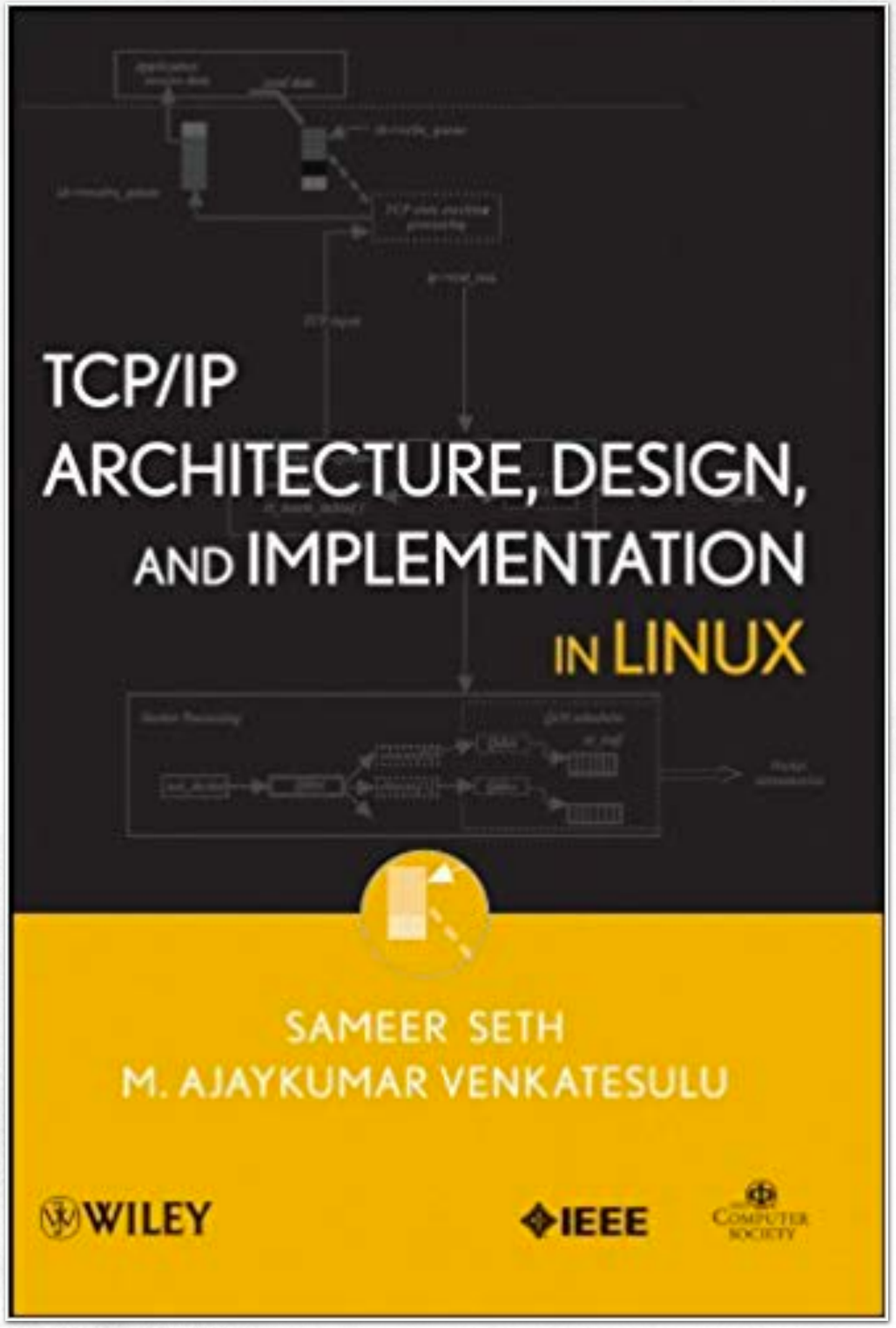
- Socket Programming
- ZeroMQ
- RabbitMQ
- **Book Recommendation**

UNIX Kernel


TCP/IP Illustrated, Volume 1
The Protocols
W. Richard Stevens
TCP/IP Illustrated, Volume 2
The Implementation
Gary R. Wright
W. Richard Stevens
TCP/IP Illustrated, Volume 3
TCP for Transactions, HTTP, NNTP, and the UNIX® Domain Protocols
W. Richard Stevens

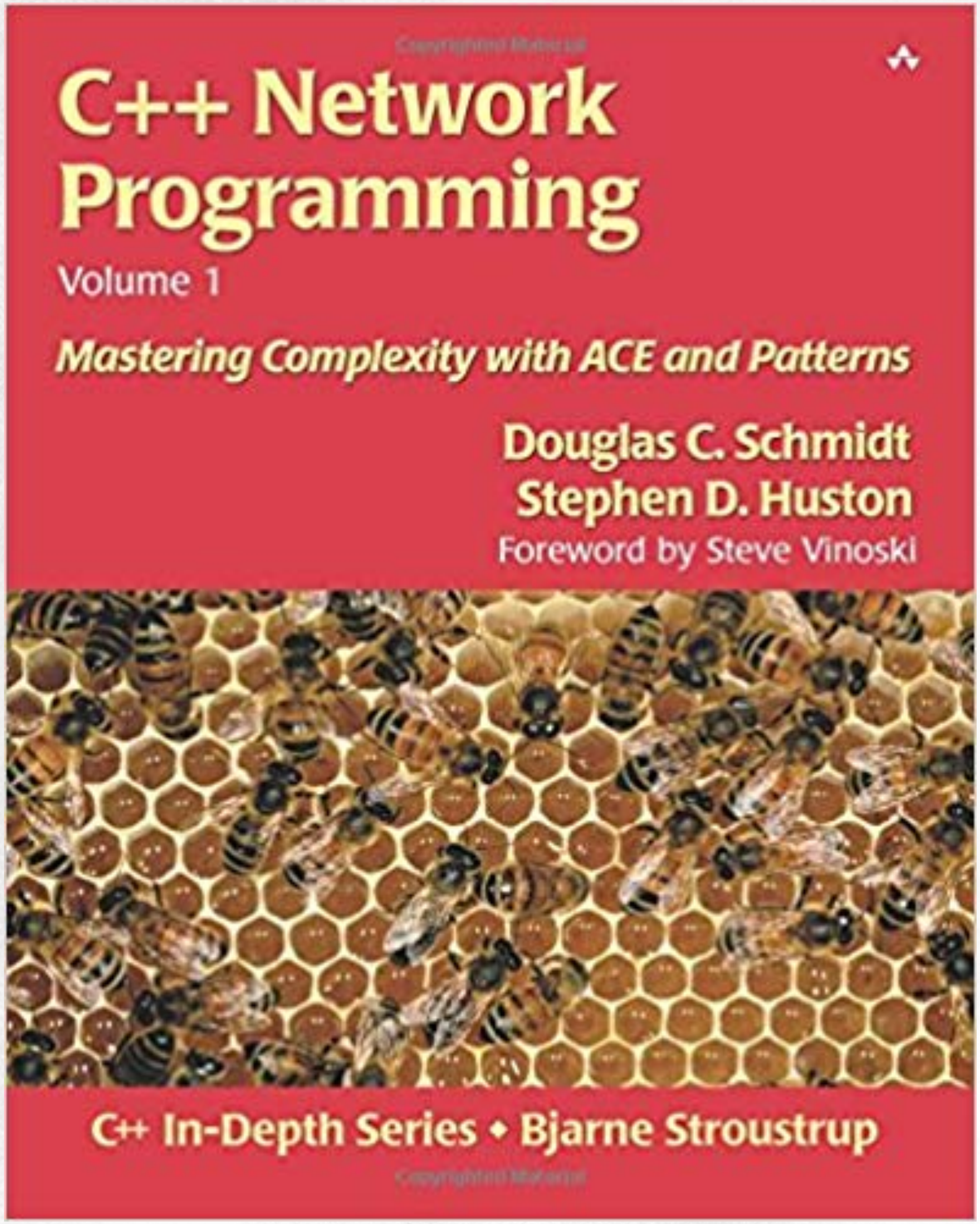
Network Protocol Anatomy

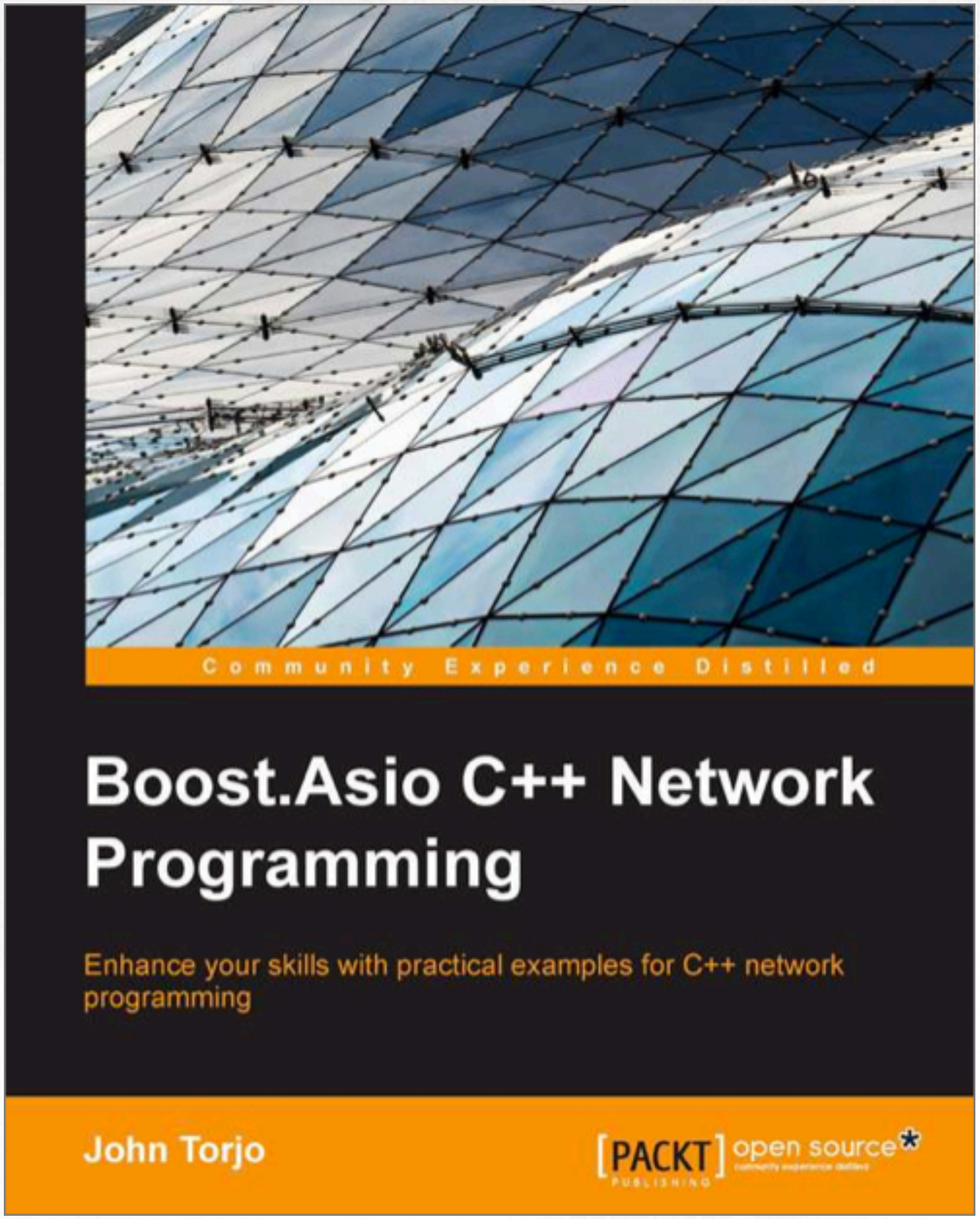
Linux Kernel



General Network Programming

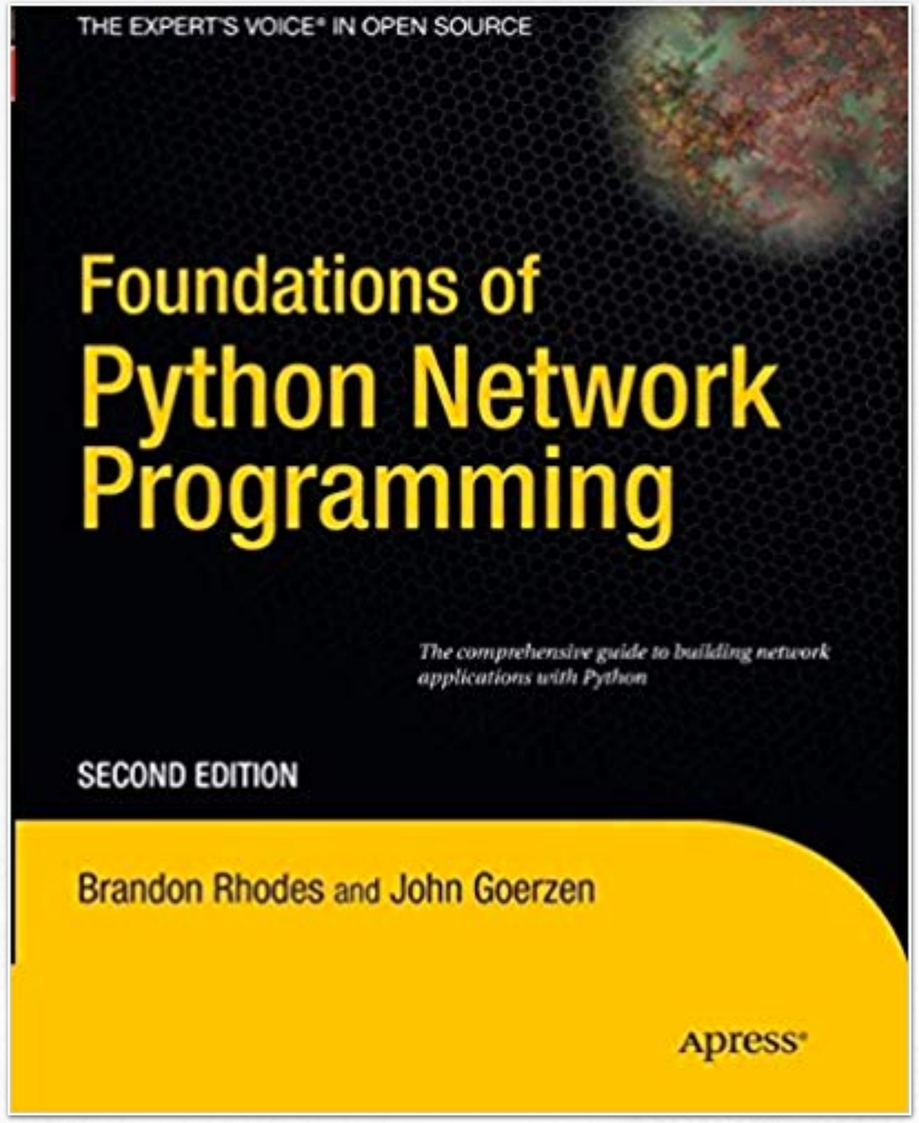
C++





Python

https://www.tutorialspoint.com/python/python_network_programming.htm



Contents

- One more …

C++ embraces Boost.asio (private C++ library for networking)

C++17

- C++ Extensions for Networking
  - https://cplusplus.github.io/networking-ts/draft.pdf
- C++Now 2017
  - Michael Caisse “Networking TS Workshop (part 1 of 2)
    - https://youtu.be/dZdTOH9bFvs
  - Michael Caisse “Networking TS Workshop (part 1 of 2)
    - https://youtu.be/5H-hI98TYIE

Thank you

경희대학교
KYUNG HEE UNIVERSITY