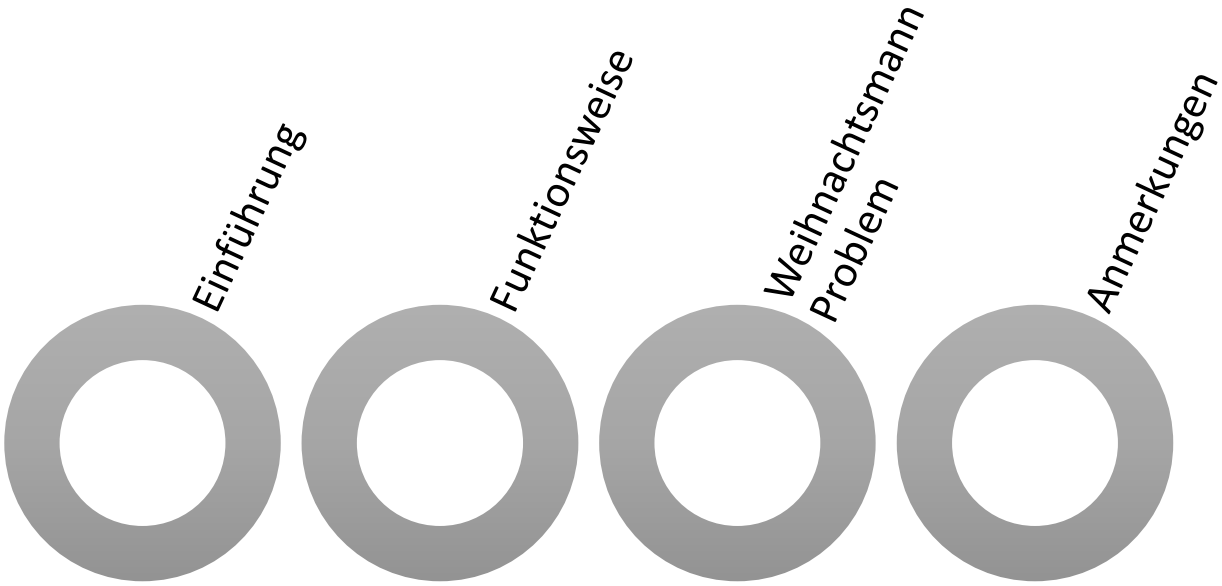
ZeroMQ

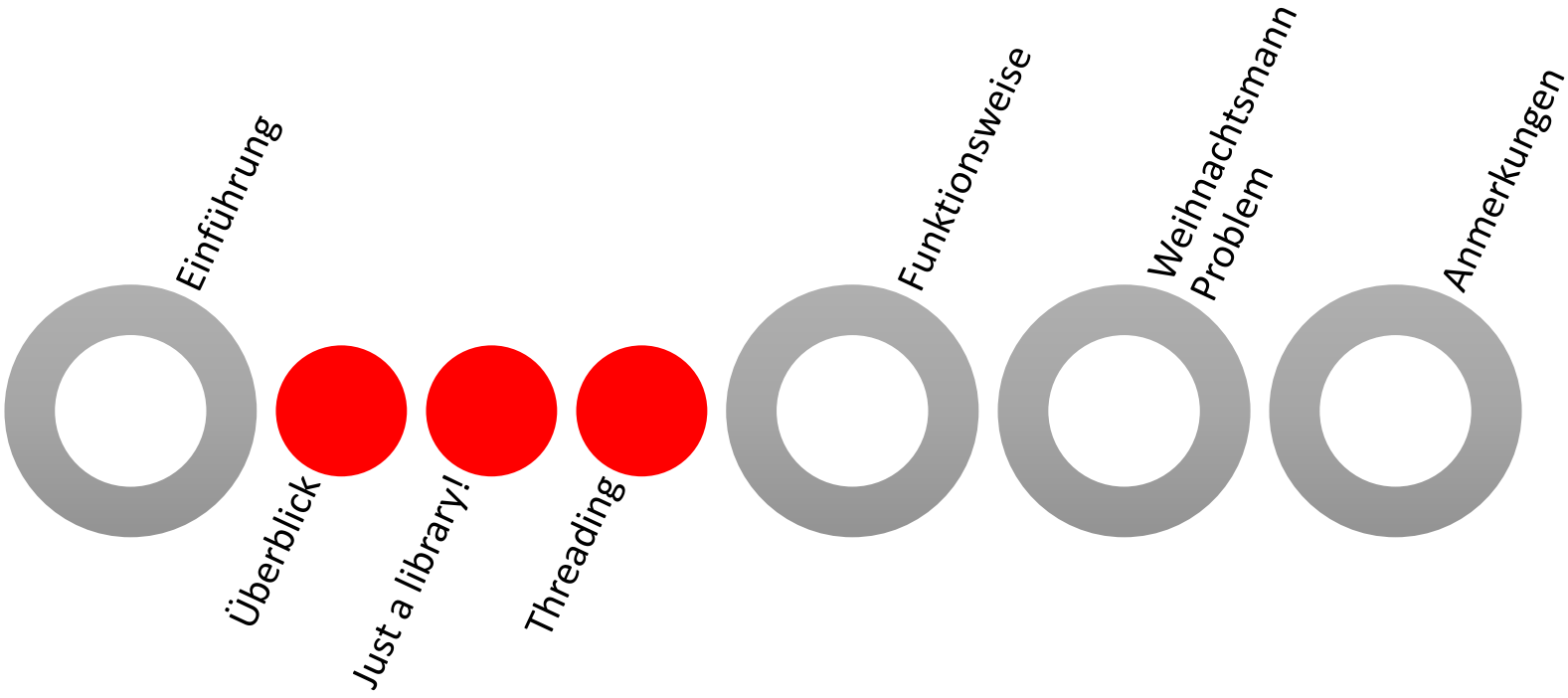
Vorlesung Betriebssysteme

Wintersemester 2016 / 2017

Inhalt



Präsentation & Code auf GitHub: lekoll.de/santa


Einführung
Überblick
Just a library!
Threading
Funktionsweise
Weihnachtsmann Problem
Anmerkungen

Überblick

„Distributed Messaging“

Netzwerk

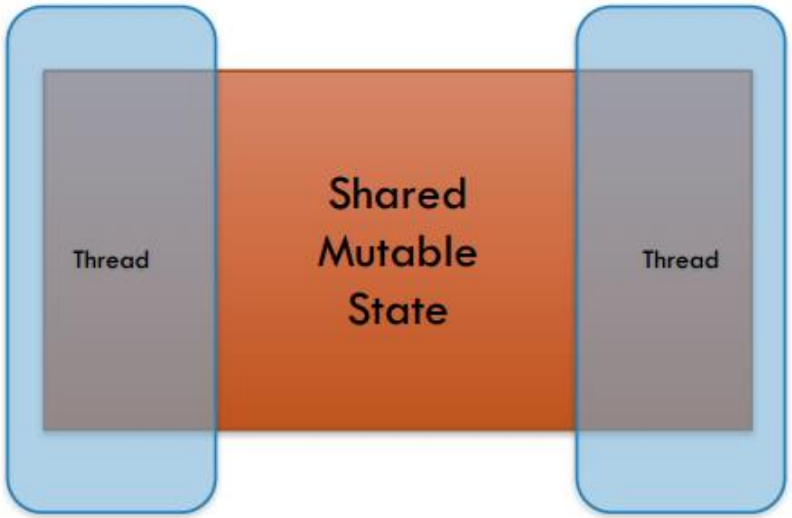
IPC

Threading

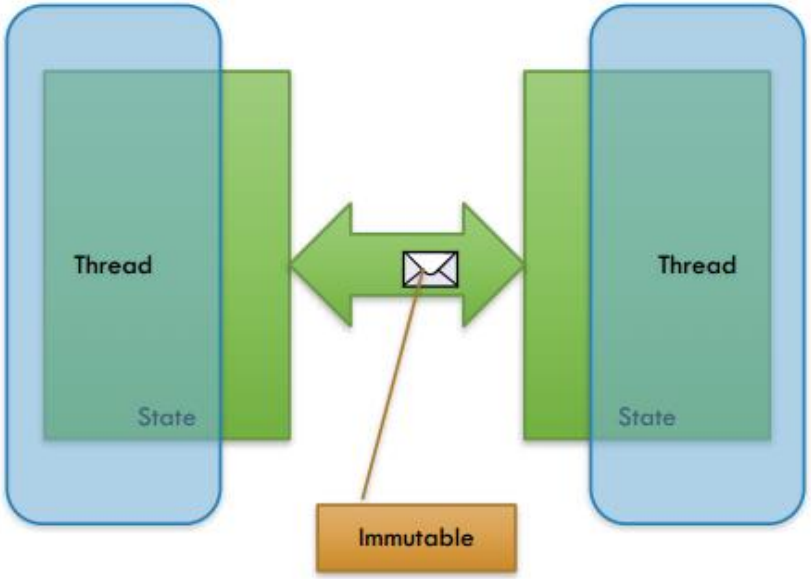
Just a library!

Threading

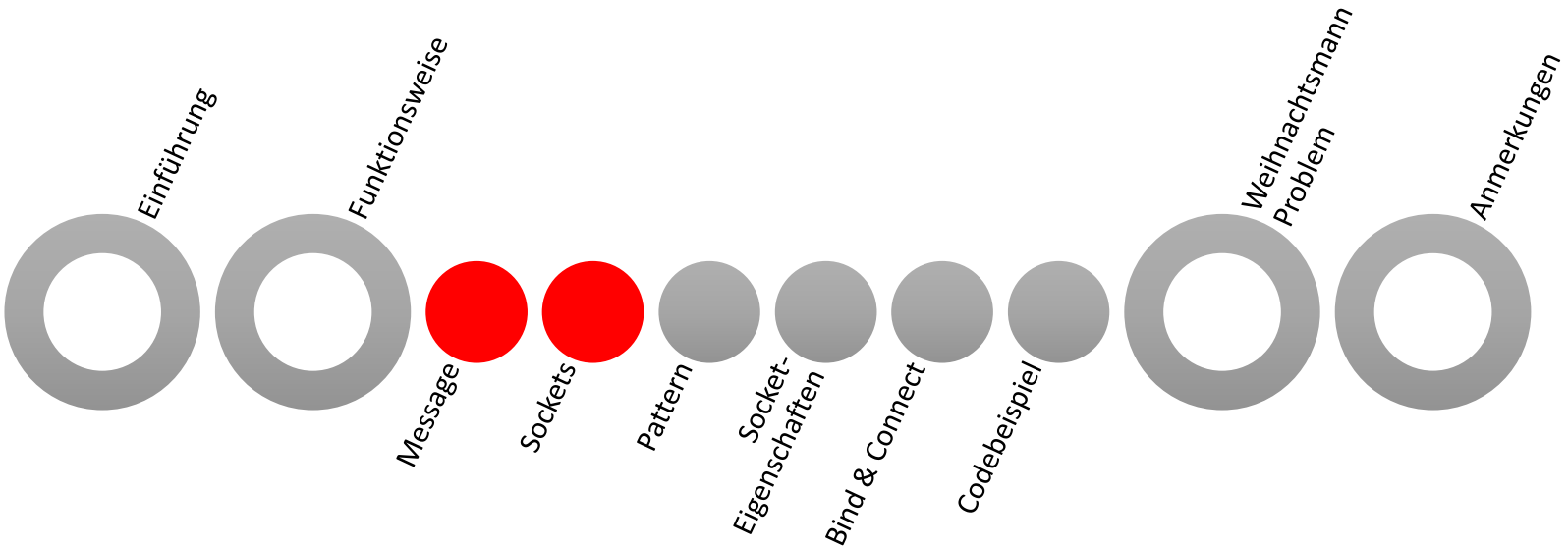
Grundproblematik



ZMQ Lösungsansatz



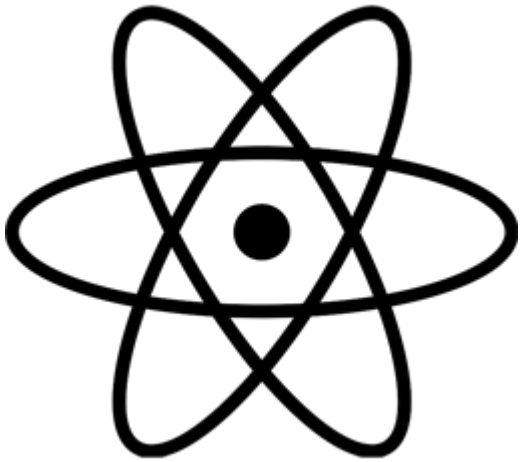
© Peter Sturm, Universität Trier


Einführung
Funktionsweise
Message
Sockets
Pattern
Socket-Eigenschaften
Bind & Connect
Codebeispiel
Weihnachtsmann Problem
Anmerkungen

Message

```
2inR2aoDR4WXMQ12e5BX
2Qw1W1ptcPUbDjcQ5Mge
DYpyMph8SpzbpW5K7hS8
gCvCdBqYeWLTAKSvJWeS
FiA1PorymmKhaKns7zXr
h5vBmhX0EVtxErwL6HkB
fRwLksMvvWWBwDreKM03
j8Wl9QBd6P2yX45zPOH8
8migVgPTHLfwWyfoJ7qY
lXm1LYDypvvV6vFnp3tS
Afbtwdg3j9yX6H99RlzX
```

String

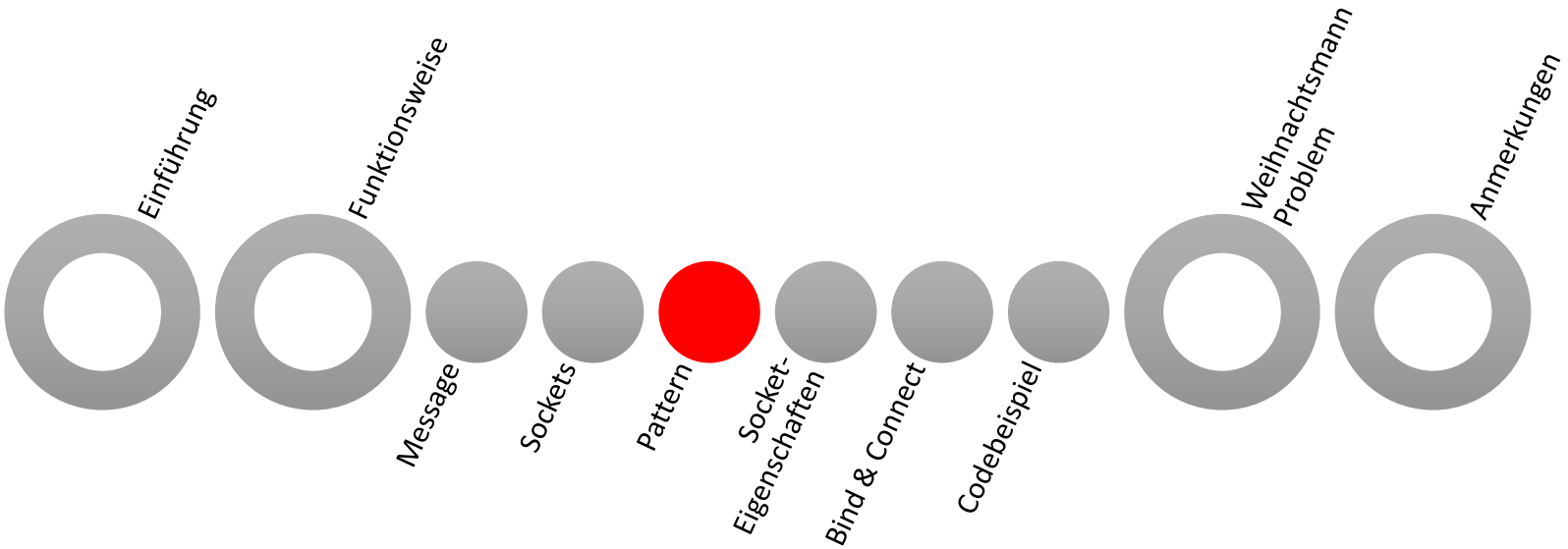


Atomic

Sockets

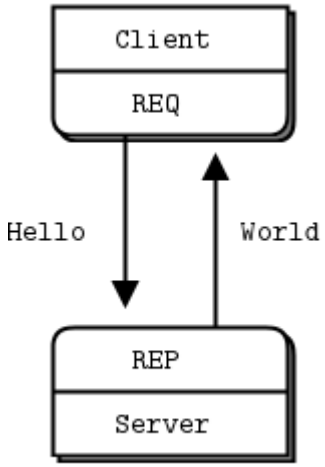
- ZMQ-Sockets != Netzwerksockets
- Bind & Connect

```
meinSocket = context.socket(zmq.PAIR)
meinSocket.bind("tcp://*:5555")
meinSocket.send_string("Hello World")
```

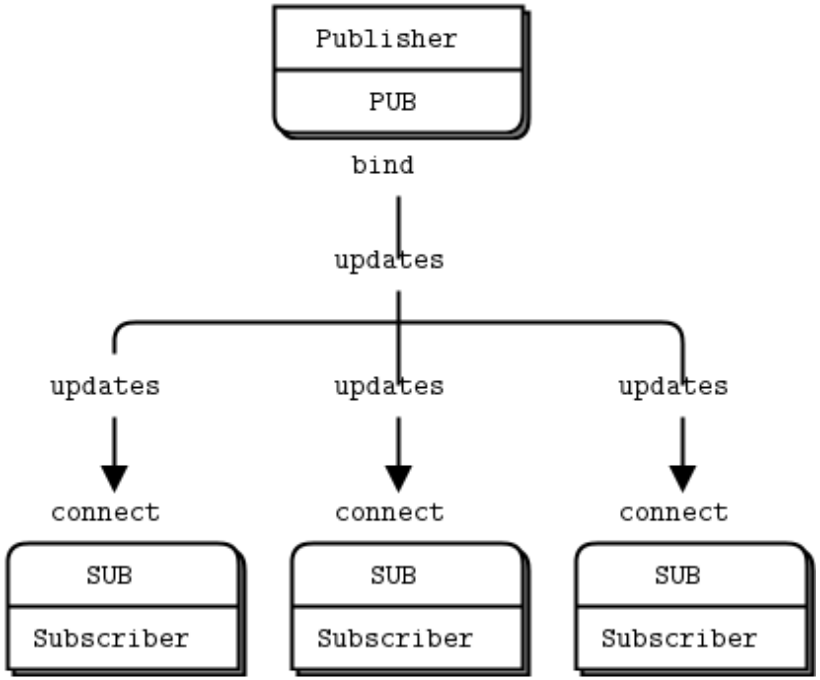

Einführung
Funktionsweise
Message
Sockets
Pattern
Socket-Eigenschaften
Bind & Connect
Codebeispiel
Weihnachtsmann Problem
Anmerkungen

Pattern

Synchrones Request/Response

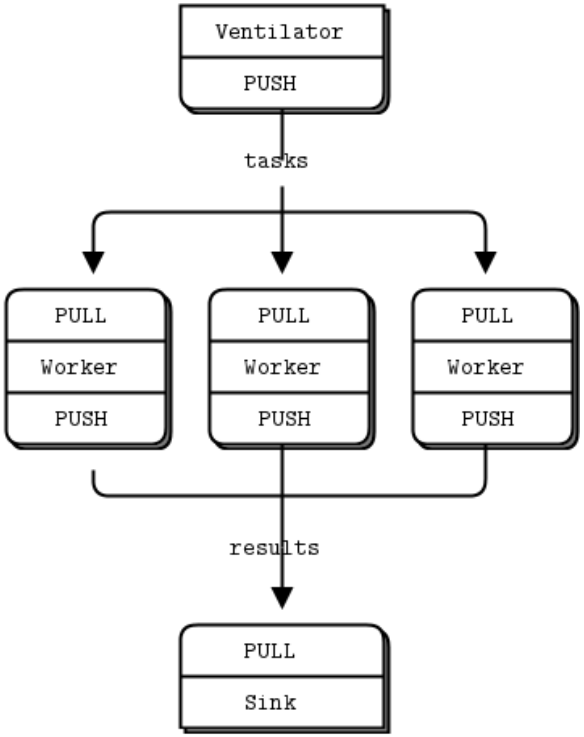


Publish/Subscribe

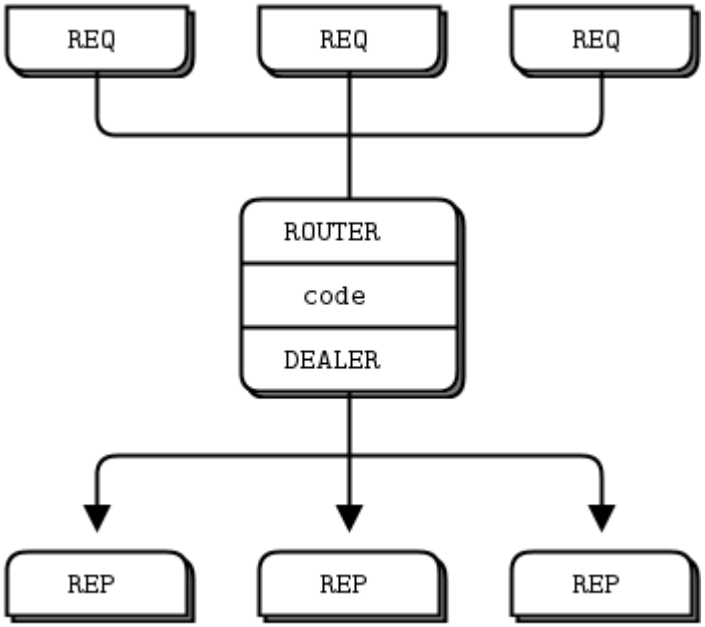


Pattern

Push/Pull

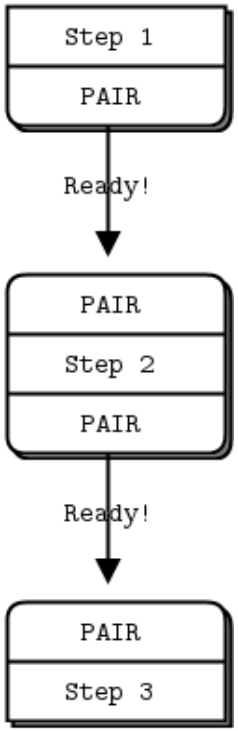


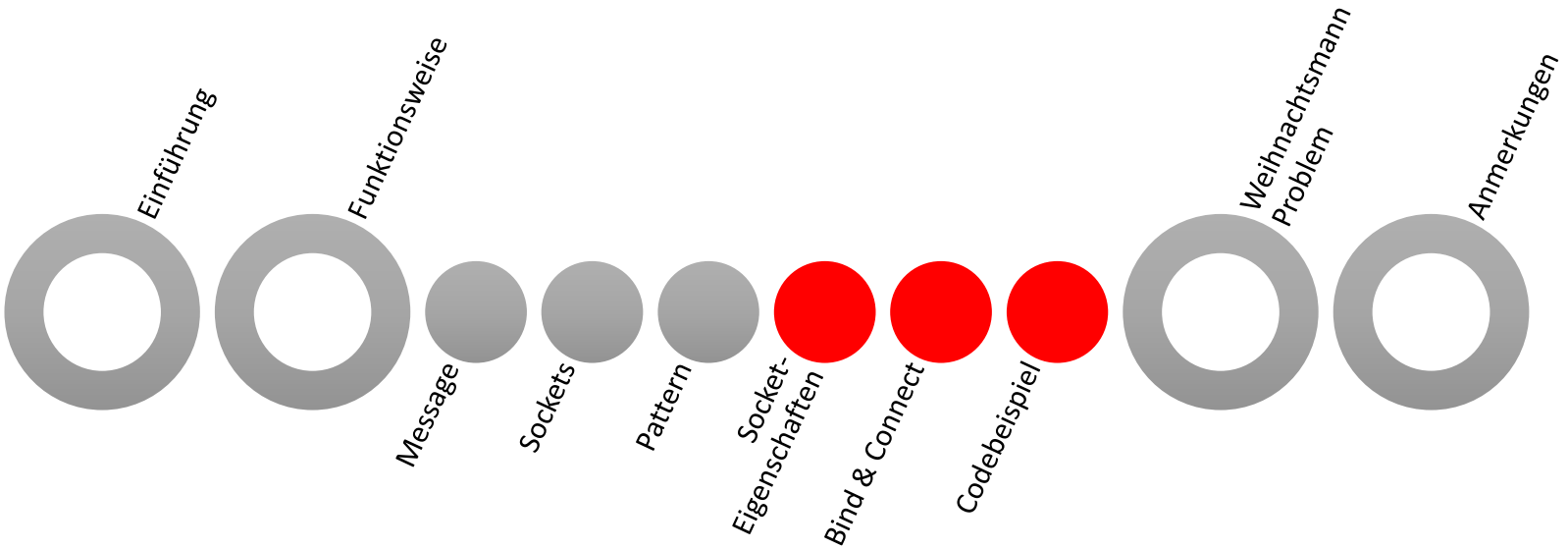
Asynchrones Request/Response



Pattern

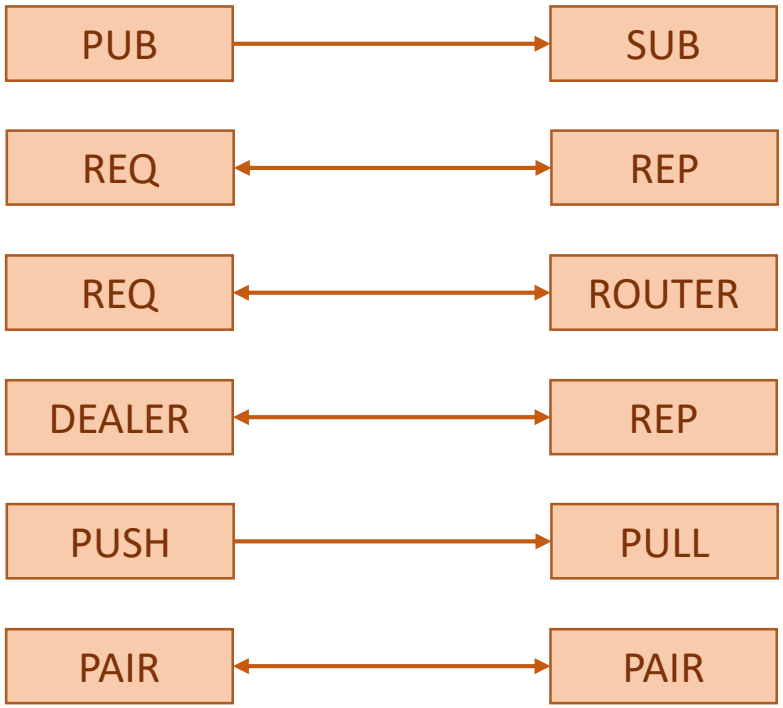
Pair



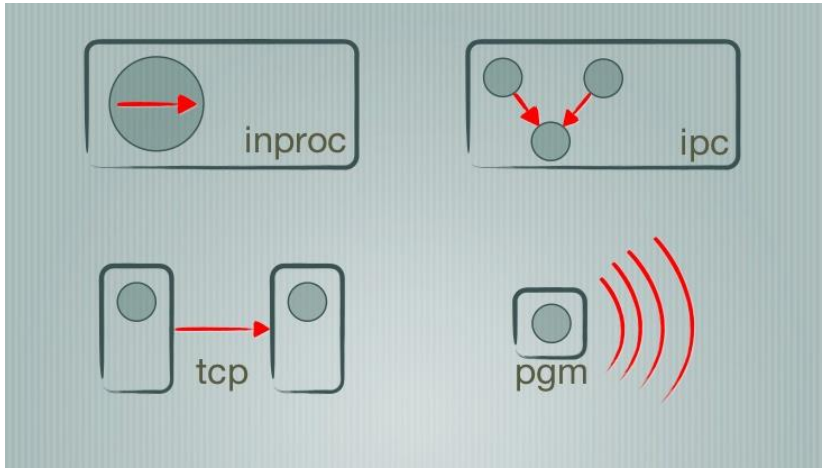

Einführung
Funktionsweise
Message
Sockets
Pattern
Socket-Eigenschaften
Bind & Connect
Codebeispiel
Weihnachtsmann Problem
Anmerkungen

Socket-Eigenschaften

Typen



Protokolle



Bind & Connect

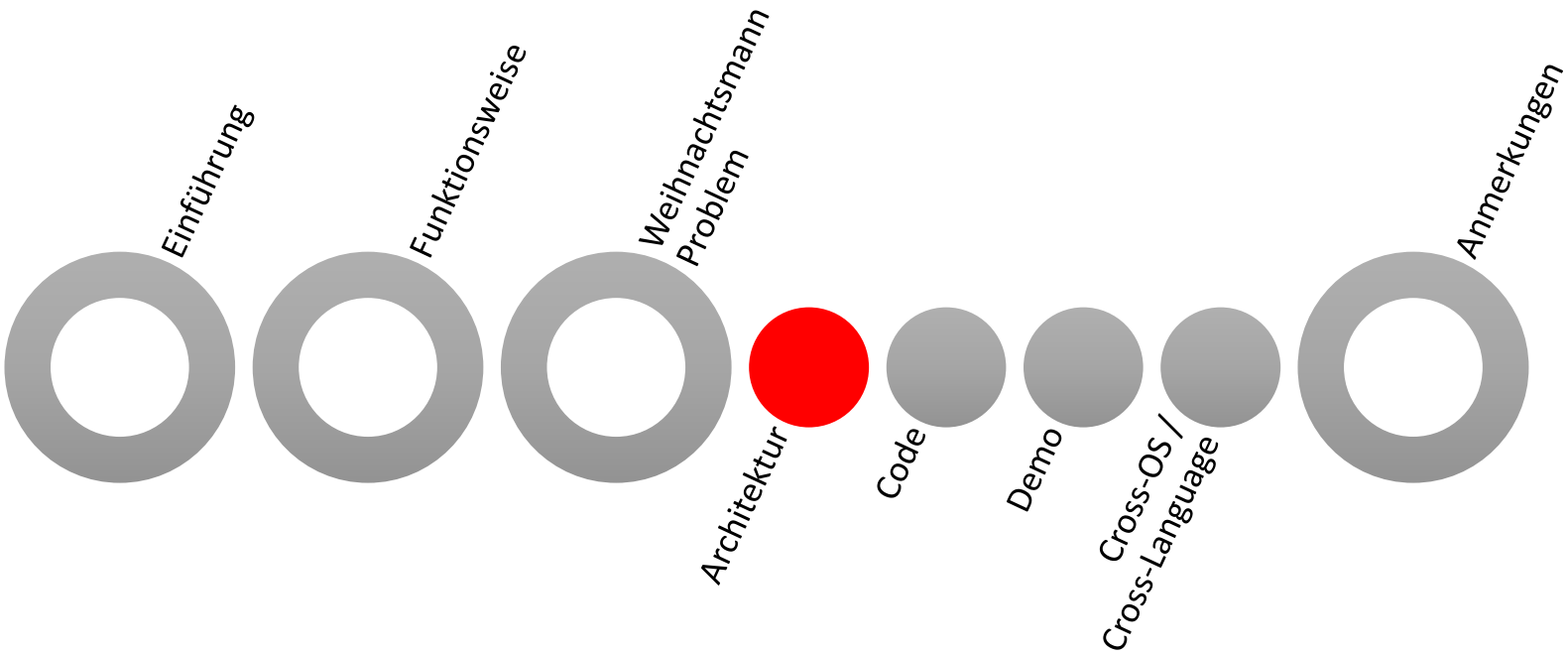
“with ZeroMQ sockets,
it does not matter which end connects and which end binds.”

“think in terms of ‘servers’ as static parts
[…]
and ‘clients’ as dynamic parts”

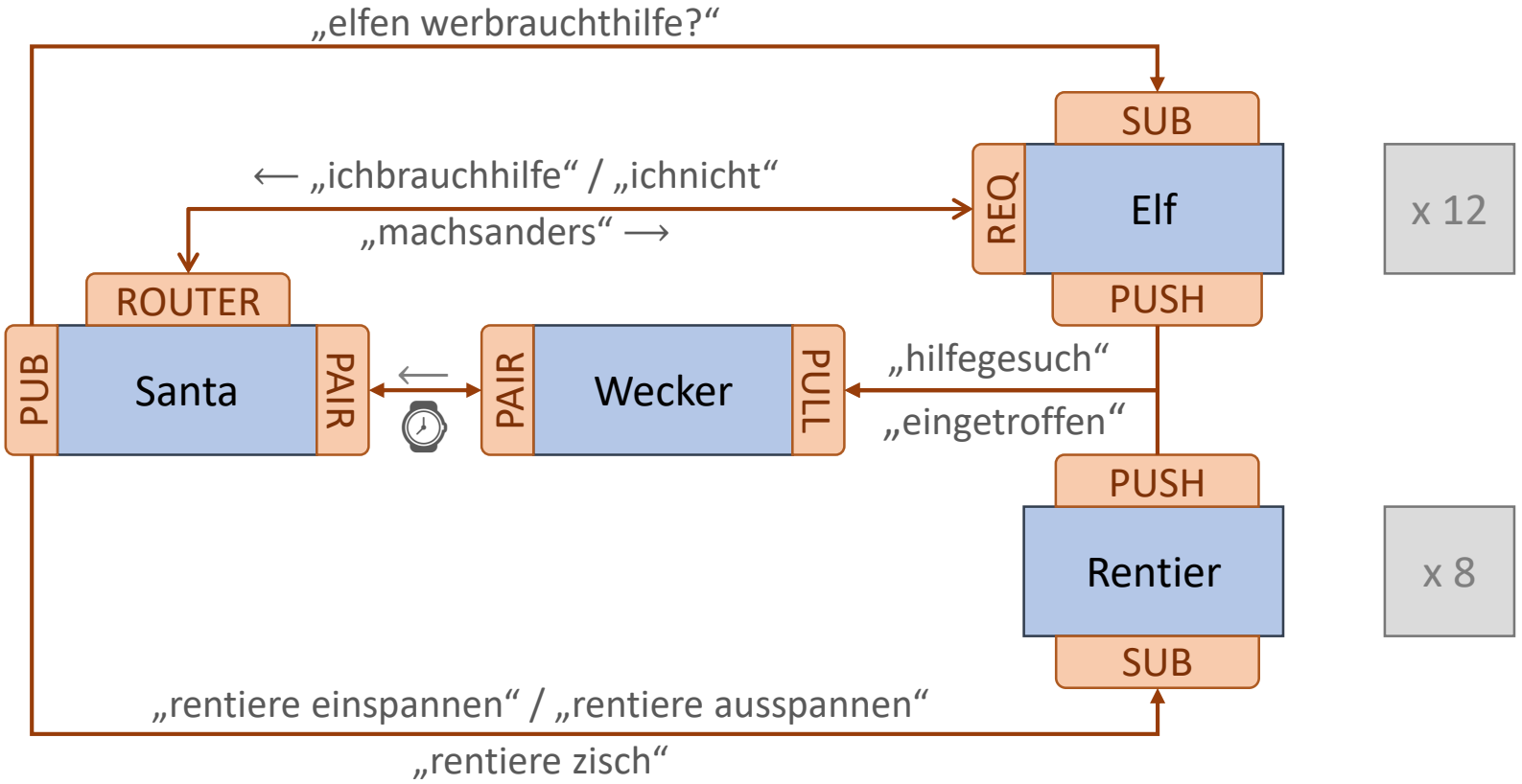
(zguide.zeromq.org)

Codebeispiel

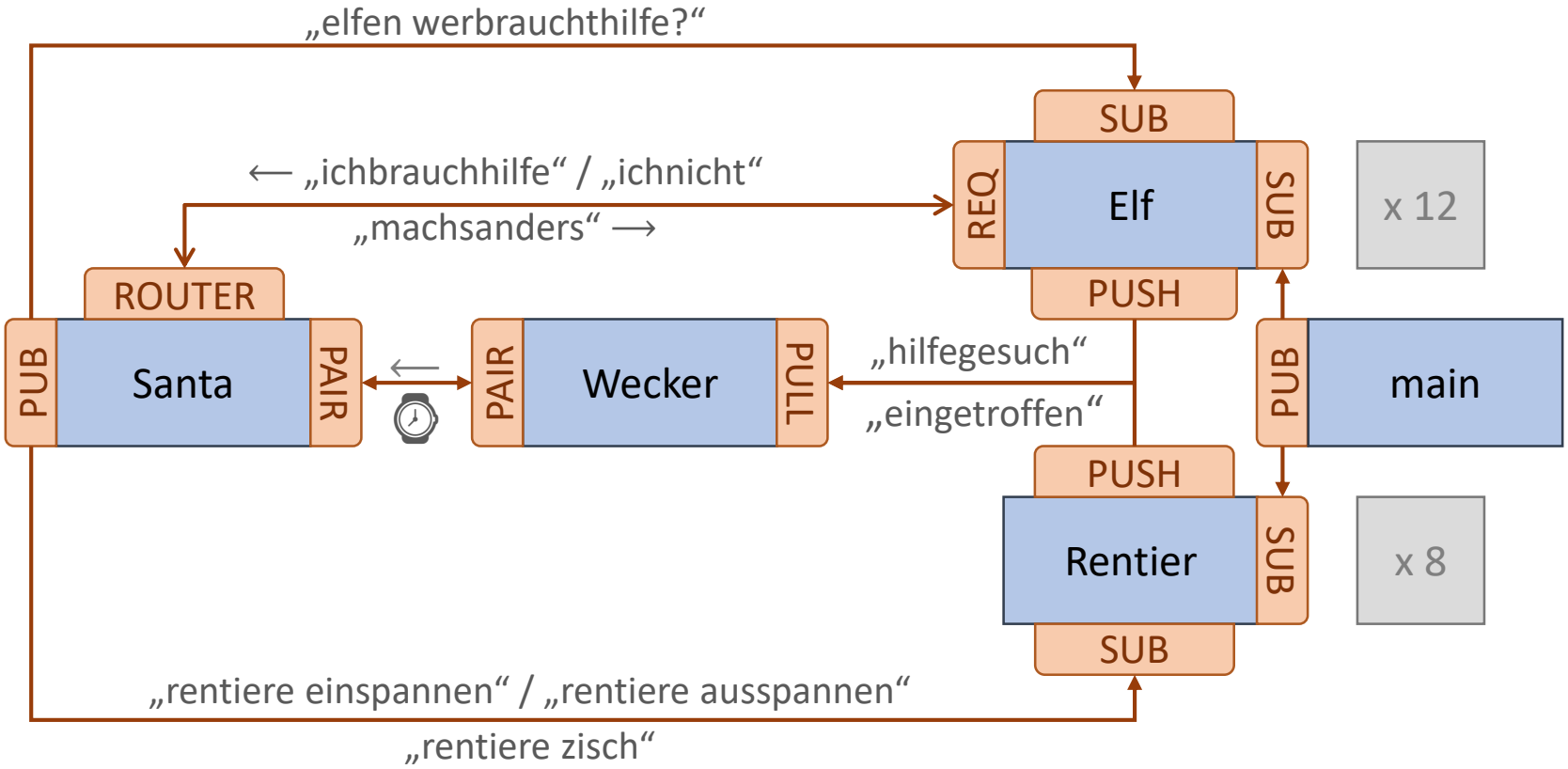
```
5   def sender (context):
6       senderSocket = context.socket(zmq.PAIR)
7       #senderSocket.bind("tcp://*:5555")
8       senderSocket.bind("inproc://meinKanal")
9       senderSocket.send_string("Hello World")
10
11  def receiver (context):
12      recvSocket = context.socket(zmq.PAIR)
13      #recvSocket.connect("tcp://localhost:5555")
14      recvSocket.connect("inproc://meinKanal")
15      print(recvSocket.recv())
16
17  zmqContext = zmq.Context.instance()
18
19  sThread = threading.Thread(target=sender, args=(zmqContext,))
20  sThread.start()
21  rThread = threading.Thread(target=receiver, args=(zmqContext,))
22  rThread.start()
```

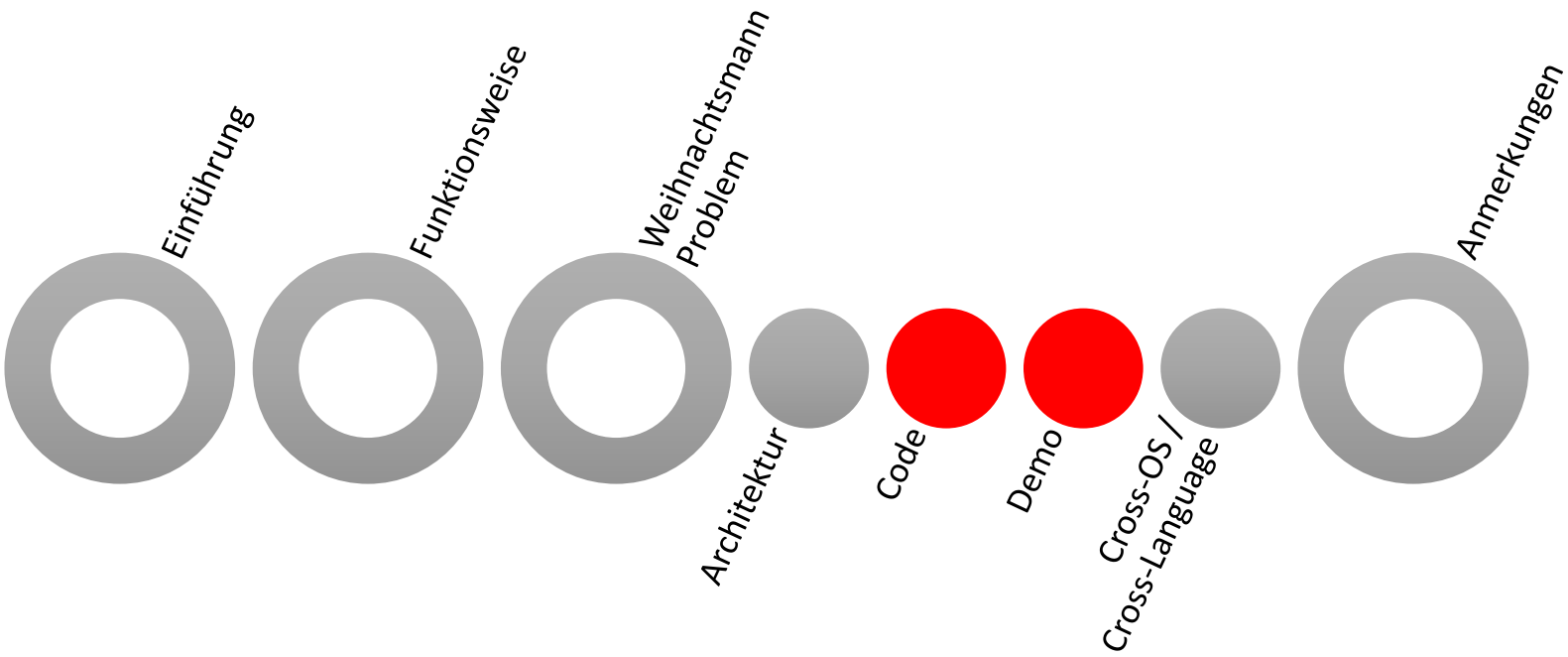

Einführung
Funktionsweise
Weihnachtsmann Problem
Architektur
Code
Demo
Cross-OS / Cross-Language
Anmerkungen

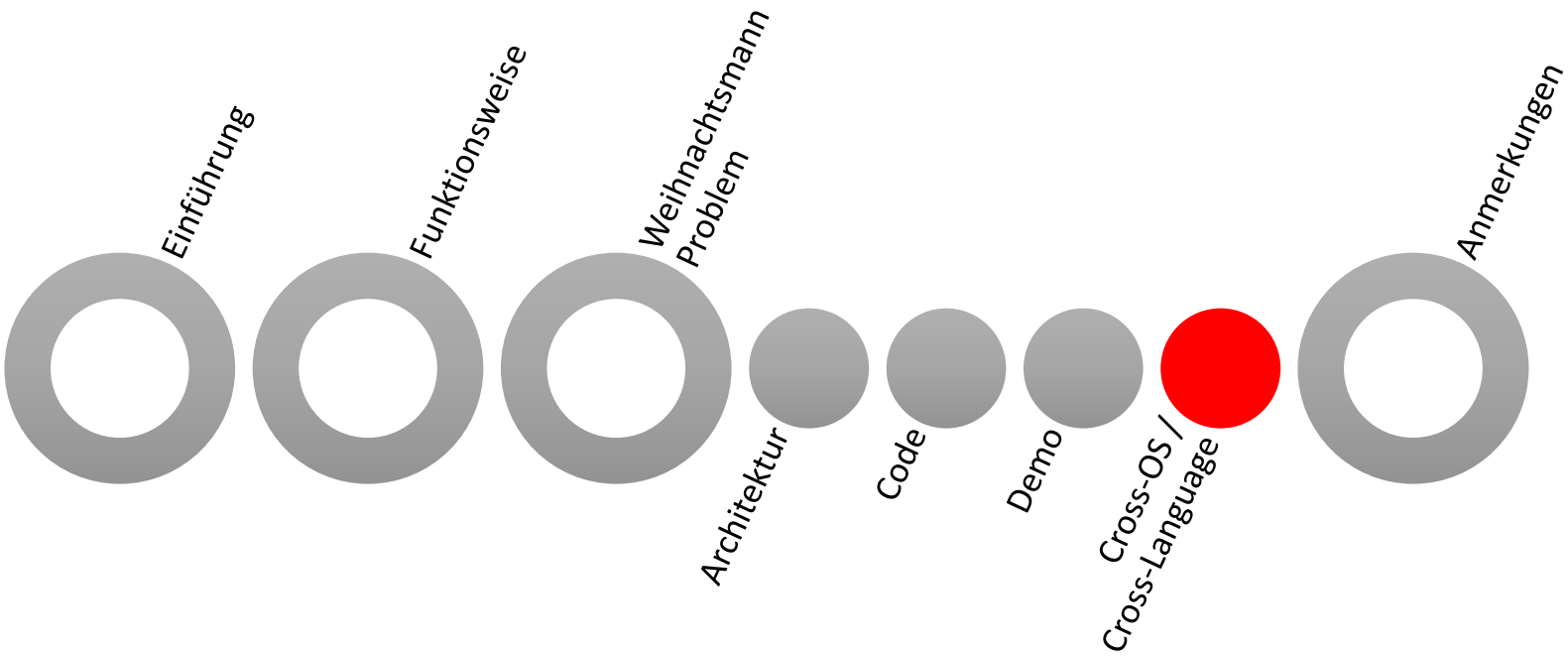
Architektur



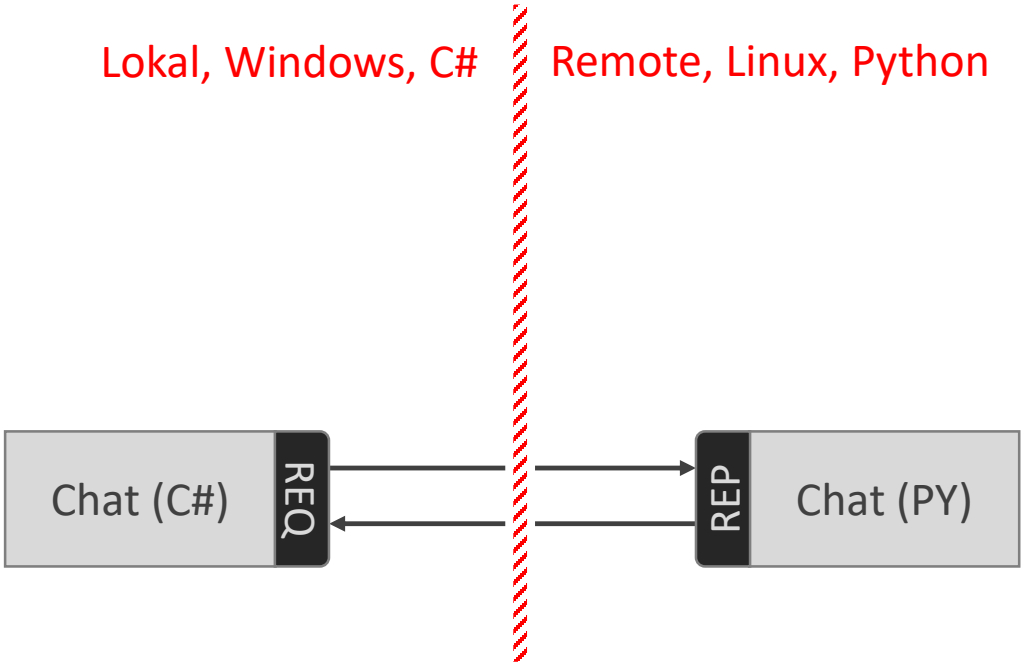
Architektur

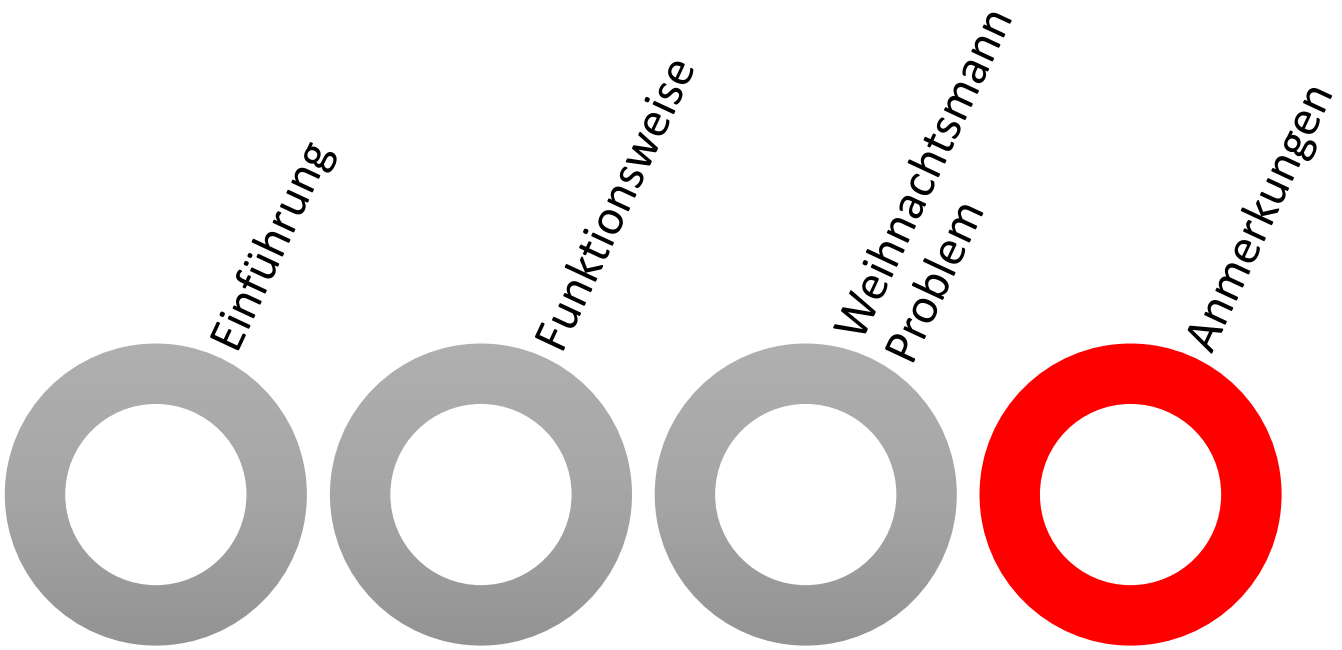



Einführung
Funktionsweise
Weihnachtsmann Problem
Architektur
Code
Demo
Cross-OS / Cross-Language
Anmerkungen


Einführung
Funktionsweise
Weihnachtsmann Problem
Architektur
Code
Demo
Cross-OS / Cross-Language
Anmerkungen

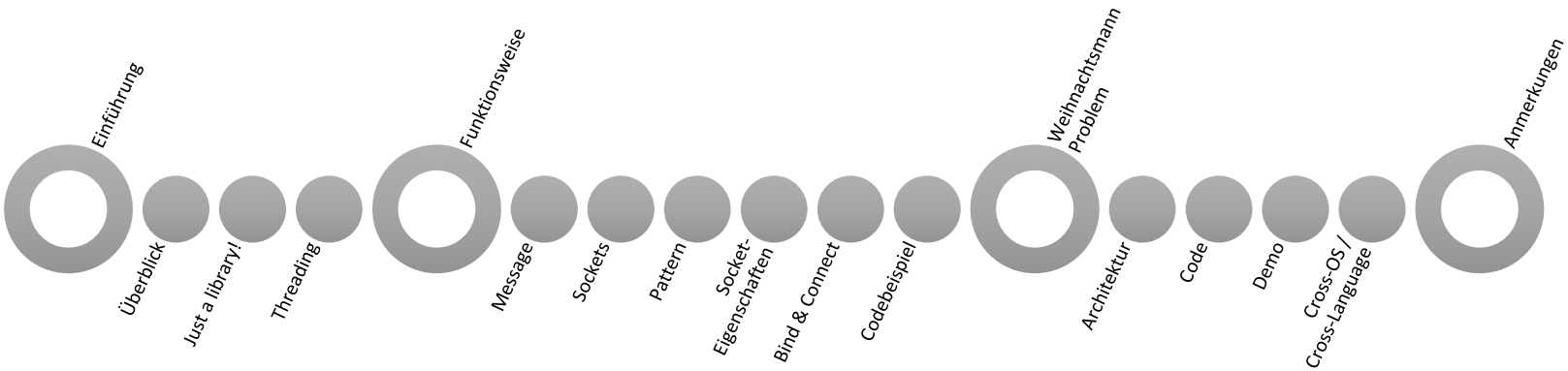
Cross-OS / Cross-Language




Einführung
Funktionsweise
Weihnachtsmann Problem
Anmerkungen

Anmerkungen

- Für Inproc nur bedingt geeignet
  - Serialisierung
- Schnelles Setup von leistungsstarkem Messaging
  - Cross-OS / Cross-Language
  - Performant
  - Abstraktion


Einführung
Überblick
Just a library!
Threading
Funktionsweise
Message
Sockets
Pattern
Socket-Eigenschaften
Bind & Connect
Codebeispiel
Weihnachtsmann Problem
Architektur
Code
Demo
Cross-OS / Cross-Language
Anmerkungen

Quellen

- Vortrag "ZeroMQ is the Answer“ by Ian Barber at the PHP UK Conference 2011 (https://vimeo.com/20605470)
- Offizielle Website (http://zeromq.org/)
- “A quick and dirty Einführungduction to ZeroMQ”, Scott Logic Ltd. (http://blog.scottlogic.com/2015/03/20/ZeroMQ-Quick-Einführung.html)