Building Distributed systems with LabVIEW and ØMQ.

How to build a distributed system without a large team.

Maciej Kolosko, Partner Composed Systems, CLA , CLED

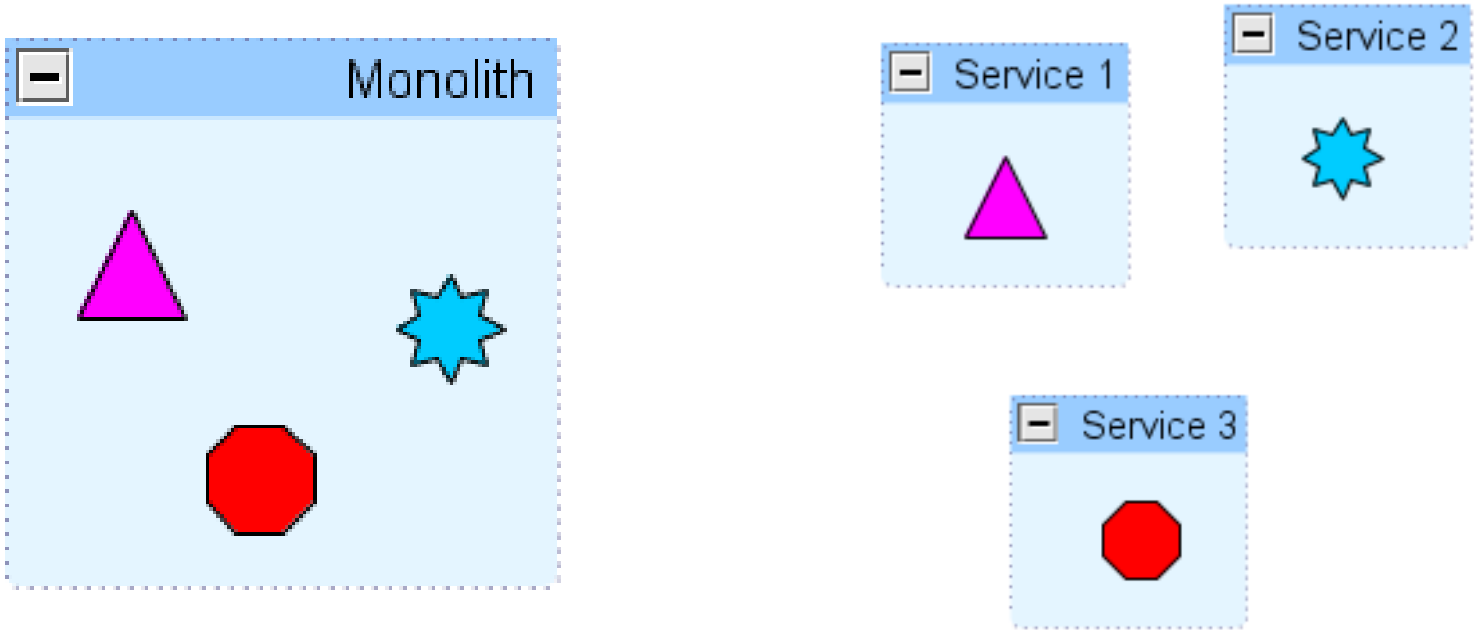
Presentation overview

- Microservices – how distributed cloud systems are built
- What is ØMQ and where is it available to use
- Demo of connecting Current Gen and NXG with ØMQ
- How ØMQ can be used in the kind of systems we build with LabVIEW
- Stable vs Unstable networks
- Wrap up followed by Q&A

Aspects of distributed systems

- Componentization via Services
- Segregation oriented to Business Capabilities rather than Technlology
- Decentralized governance
- Decentralized data management
- Design for failure
- Smart endpoints and dumb pipes

Componentization via Services

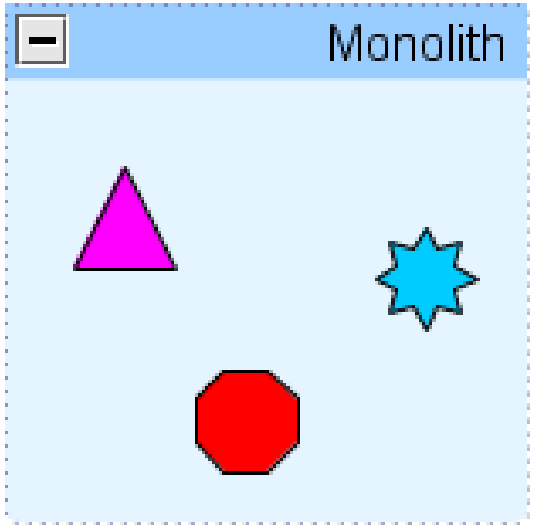


Organizing vs capabilities

User Interface

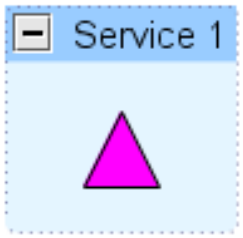
Business
Logic

Database



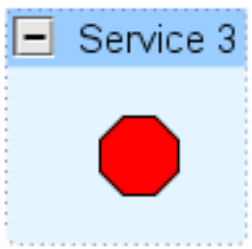
Organizing vs **Business** Capabilities

Orders



Catalog

Shipping



Building for failure

- Randomly shuts down your services and VM's in your deployment structure
- https://github.com/Netflix/chaosmonkey

The ESB : The smart pipe.

SMART ENDPOINTS DUMB PIPES: What is zero ØMQ?

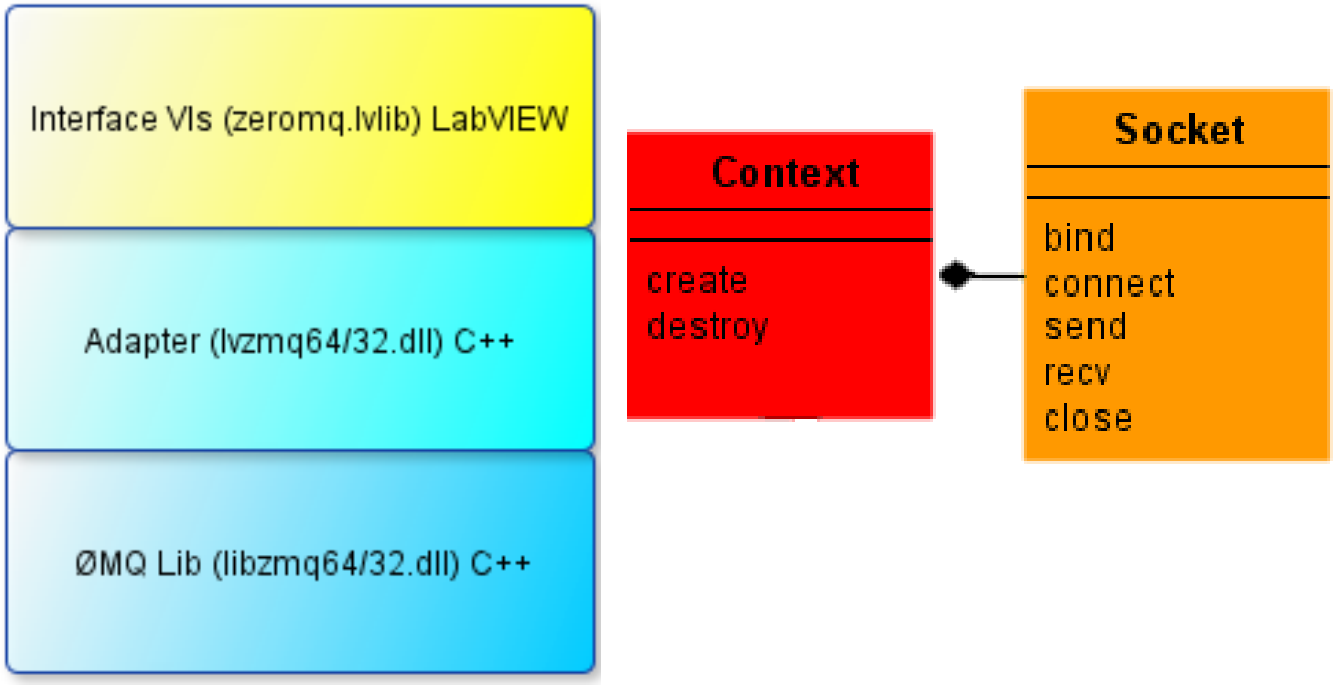
- It's not another message queue.
- ØMQ – not a daemon not a server. It is a library.
- It does not bring any additional dependencies with it.
- It is a very lightweight socket system.
- It was built for financial markets, to handle a high volume of messages.

ØMQ AVAILIBLITY

- Language bindings which use the C++ implementation are available for C#, JAVA, C , LabVIEW , Python , F#, Lua, PHP and many … many more http://zeromq.org/bindings:_start
- Java and C# stack supported as direct implementations.
- Supported operating systems include :
  - Windows
  - Unix / Linux
  - iOS, Android
  - WxWorks

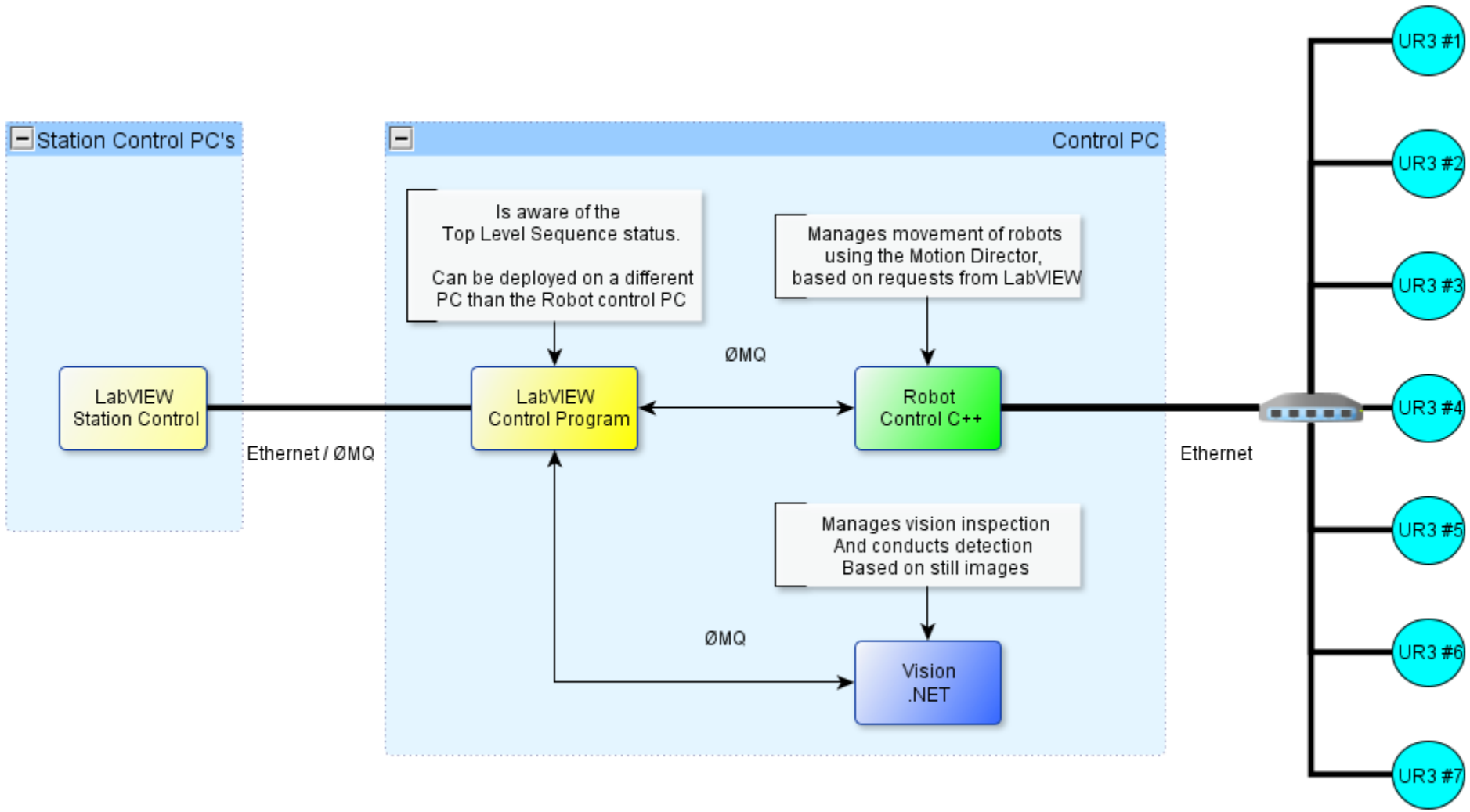
ØMQ binding for current gen

- Three layers of libraries
- Prominent Classes
  - Context
    - Thread safe
    - Manages I/O threads
  - Socket
    - Manages a single endpoints
    - Not thread safe

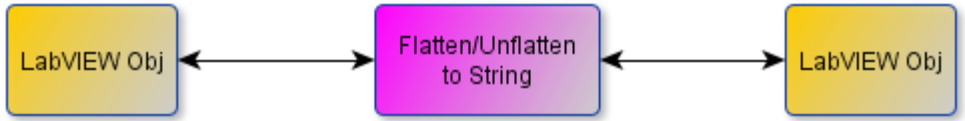


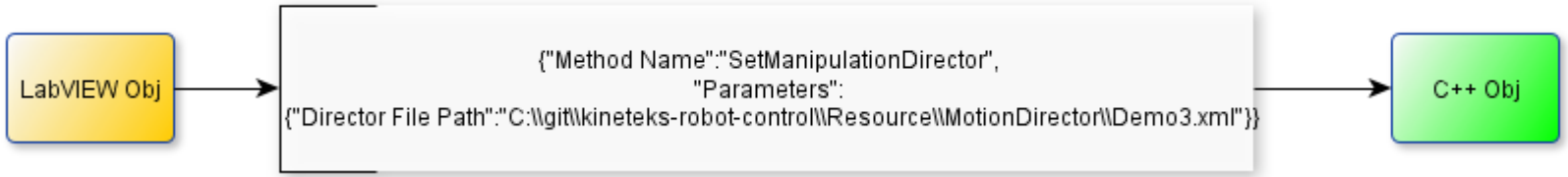
Examples : connecting current gen and NXG using ØMQ

- Example of connecting LabVIEW current GEN and NXG though ØMQ and the use of REQ / REP pattern
- Overview of the example code, and brief discussion about the process of porting a portion of the Zero MQ binding from current gen to next gen.


Station Control PC's
Control PC
Is aware of the
Top Level Sequence status.
Can be deployed on a different
PC than the Robot control PC
Manages movement of robots
using the Motion Director,
based on requests from LabVIEW
ØMQ
LabVIEW
Station Control
LabVIEW
Control Program
Robot
Control C++
Ethernet / ØMQ
Ethernet
Manages vision inspection
And conducts detection
Based on still images
ØMQ
Vision
.NET
UR3 #1
UR3 #2
UR3 #3
UR3 #4
UR3 #5
UR3 #6
UR3 #7

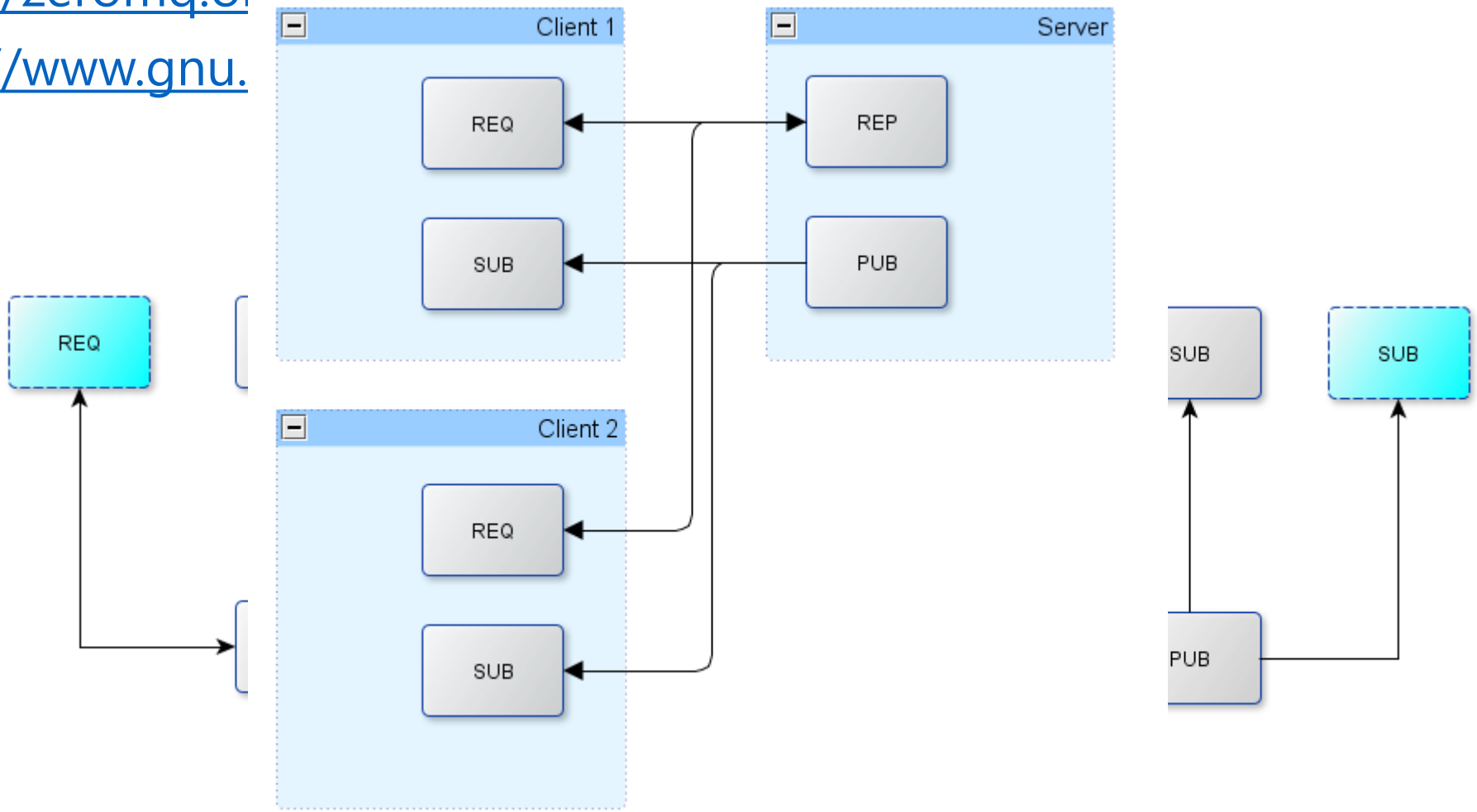
Messages ØMQ



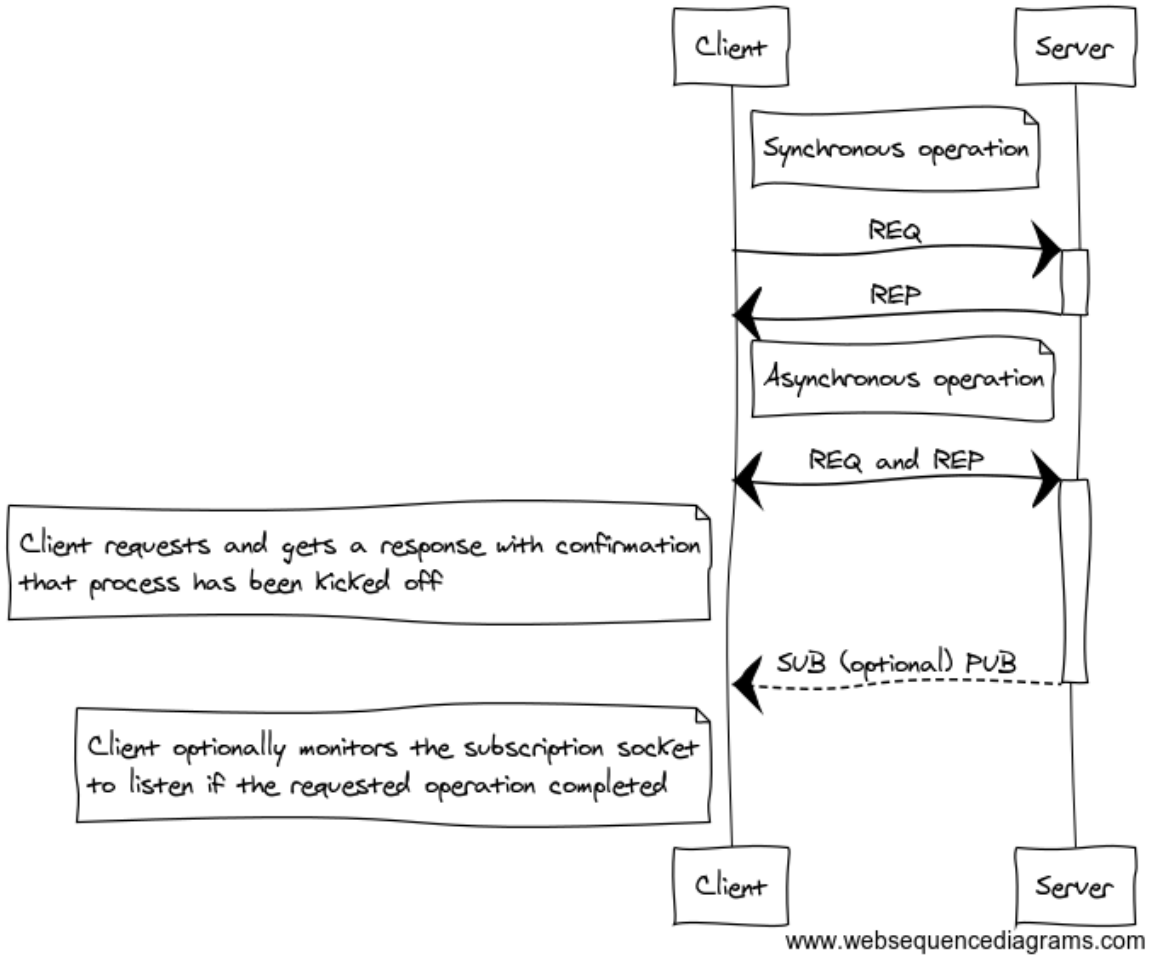


Zero MQ Communication scheme

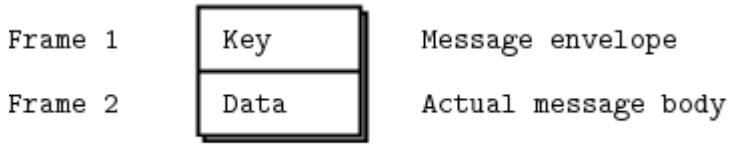
- http://zeromq.org
- http://www.gnu.



Synchronous vs Asynchronous operation

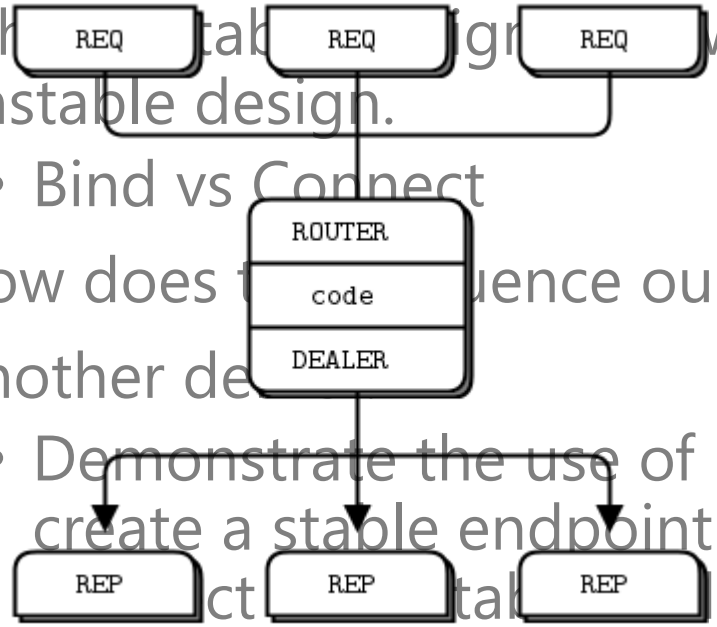


PUB : socket enveloping



STABLE vs UNSTABLE Design

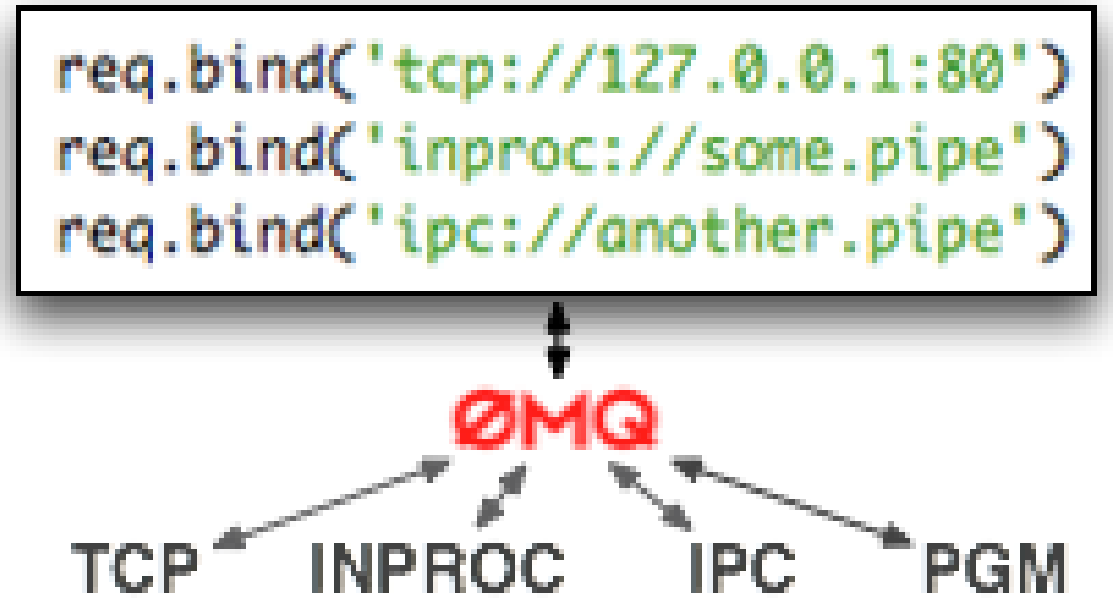
- Wh tab ign what is unstable design.
  - Bind vs Connect
- How does t uence our design.
- Another de
  - Demonstrate the use of Proxy to create a stable endpoint and ct tab points.



Node Presence / Discovery

- Video demo of an unstable network. Go to http://labviewcraft.com to watch.

ØMQ Types of transport



Zero MQ Termination

- It’s not straightforward at the beginning.
- http://zeromq.org/whitepapers:0mq-termination

Questions

- Thank you!
- Email maciej.kolosko@composedsystems.com
- Web https://composed.io
- See Examples / Demo Videos http://labviewcraft.com upload coming soon!