# Introductory Guide to Message Passing
## In Distributed Systems

Florian Willich

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences Berlin
Course: Distributed Systems
Lecturer: Prof. Dr. Christin Schmidt

June 15, 2015

# Table of Contents

# Introduction

What is message passing?

# Introduction

What is message passing?

What is message passing in distributed systems?

(Tanenbaum & Steen, 2007)

# Requirements in Theory

- Connectivity

# Requirements in Theory

- Connectivity
- Ability

# Requirements in Theory

- Connectivity
- Ability
- Integrity

# Requirements in Theory

- Connectivity
- Ability
- Integrity
- Intelligibility

# Requirements in Theory

- Connectivity
- Ability
- Integrity
- Intelligibility

Executability is not part of the message passing model!

# Requirements Provider in Practice

TCP/IP provides several *socket primitives*:

- Connectivity: *Socket* and *Bind*

# Requirements Provider in Practice

TCP/IP provides several *socket primitives*:

- Connectivity: *Socket* and *Bind*
- Ability: *Send* and *Receive*

# Requirements Provider in Practice

TCP/IP provides several *socket primitives*:

- Connectivity: *Socket* and *Bind*
- Ability: *Send* and *Receive*
- Integrity: Several mechanisms provided

# Requirements Provider in Practice

TCP/IP provides several *socket primitives*:

- Connectivity: *Socket* and *Bind*
- Ability: *Send* and *Receive*
- Integrity: Several mechanisms provided
- Intelligibility: Do not fall within the responsibility of TCP/IP

# Requirements Provider in Practice

TCP/IP provides several *socket primitives*:

- Connectivity: *Socket* and *Bind*
- Ability: *Send* and *Receive*
- Integrity: Several mechanisms provided
- Intelligibility: Do not fall within the responsibility of TCP/IP

There is still plenty to discuss! (Tanenbaum & Steen, 2007)

# Asynchronous vs. Synchronous Message Passing

- Synchronous:
  Communication primitives called directly

# Asynchronous vs. Synchronous Message Passing

- Synchronous:
  Communication primitives called directly
- Asynchronous:
  Middleware takes place (Message Queues)

(Tanenbaum & Steen, 2007)

# Message-Passing Interface Standard

- designed by the Message-Passing Interface Forum

# Message-Passing Interface Standard

- designed by the Message-Passing Interface Forum
- defines a set of functions and datatypes for message passing

# Message-Passing Interface Standard

- designed by the Message-Passing Interface Forum
- defines a set of functions and datatypes for message passing
- aims High-Performance Computing Distributed Systems

(Message Passing Interface Forum, 2012)
(Tanenbaum & Steen, 2007)

# Definition

*Message passing in distributed systems is a model to exchange messages within a process pair by making use of several standards and implementation details. The specifically used message passing model can diverge extremely in its provided facilities and defines how to establish the connection, send and receive messages.*

# Definition

*Message passing in distributed systems is a model to exchange messages within a process pair by making use of several standards and implementation details. The specifically used message passing model can diverge extremely in its provided facilities and defines how to establish the connection, send and receive messages.*

*The physical conditions and the implementation of executing the desired instructions is not defined by any message passing model.*

# Erlang Programming Language

- functional

# Erlang Programming Language

- functional
- declarative

# Erlang Programming Language

- functional
- declarative
- for real-time, non-stop, concurrent, very large and distributed system applications

(Armstrong, 1996)
(Däcker, 2000)

# Concurrency in Erlang

- Function spawn(*Module*, *Exported Function*, *List of Arguments*)

# Concurrency in Erlang

- Function spawn(*Module*, *Exported Function*, *List of Arguments*)
- Construct receive

# Concurrency in Erlang

- Function spawn(*Module*, *Exported Function*, *List of Arguments*)
- Construct receive
- operator !

# Concurrency in Erlang

- Function spawn(*Module*, *Exported Function*, *List of Arguments*)
- Construct receive
- operator !
- Function self()

(Ericsson AB, 2015)
(Hebert, 2013)

# Live Demo

Diffie-Hellman Key Exchange Algorithm implemented in Erlang

# Q&A

Thank you for your kind attention!
Any Questions?

This document was written with LaTeX
Typeface: Open Sans by Steve Matteson.

# References


Armstrong, Joe. 1996.
*Erlang - A survey of the language and its industrial applications. In: Proceedings of the symposium on industrial applications of Prolog (INAP96)*.
http://www.erlang.se/publications/inap96.ps.
The 9'th Exhibitions and Symposium on Industrial Applications of Prolog. 16-18, October 1996. Hino, Tokyo Japan. [Online. Accessed 5th May 2015].

Däcker, Bjarne. 2000.
*Concurrent functional programming for telecommunications: A case study of technology introduction*.
http://www.erlang.se/publications/bjarnelic.ps.
Licenciate Thesis, Department of Teleinformatics. TRITA-IT AVH 00:08, ISSN 1403-5286. Royal Institute of Technology, Stockholm, Sweden. [Online. Accessed 20th May 2015].

Ericsson AB. 2015.
*Erlang/OTP System Documentation 6.4*.
http://www.erlang.org/doc/pdf/otp-system-documentation.pdf.
[Online. Accessed 31th May 2015].

Hebert, Fred. 2013.
*Learn You Some Erlang for Great Good! : A Beginner's Guide*.
http://learnyousomeerlang.com/content.
No Starch Press. [Online. Accessed 20th May 2015].

Message Passing Interface Forum. 2012.
*MPI: A Message-Passing Interface Standard Version 3.0*.
http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf.
[Online. Accessed 20th May 2015].

Tanenbaum, Andrew S., & Steen, Maarten Van. 2007.
*Distributed Systems: Principles and Paradigms (Second Edition)*.
Pearson Prentice Hall.
ISBN 0-13-239227-5.