Francesco Cesarini

presents

ERLANG/OTP

Francesco Cesarini
Erlang Solutions

@FrancescoC
francesco@erlang-solutions.com
www.erlang-solutions.com


Erlang
SOLUTIONS

What Is Scalability?

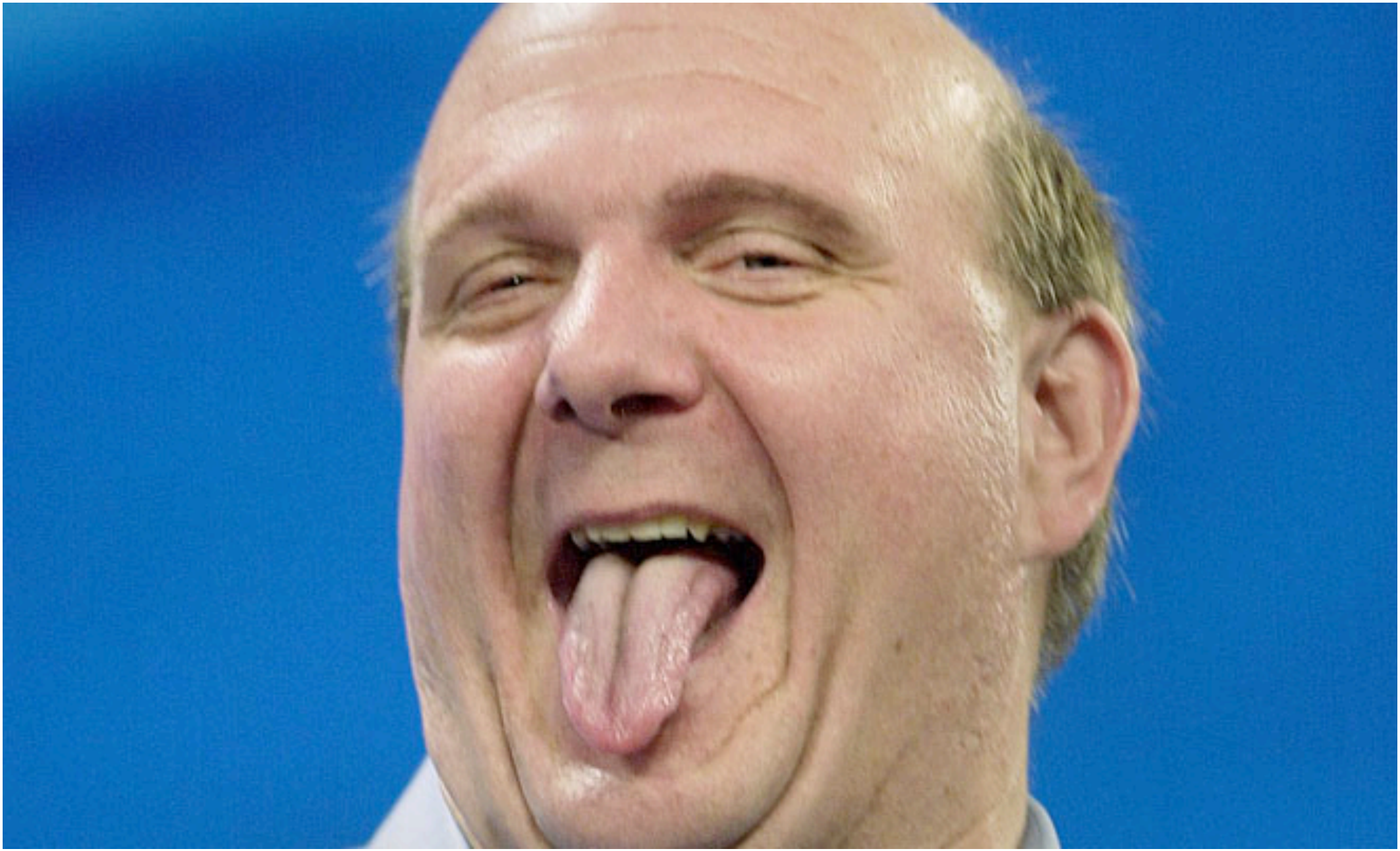
What Is (massive) Concurrency?

What Is High Availability?

What Is Fault Tolerance?


Thereifixedit.com

What Is Distribution Transparency?



Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system? Do you need a massively concurrent system? Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system? Do

YES,

PLEASE!

TO THE RESCUE

WHAT IS ERLANG

- Open source
- Concurrency-oriented
- Lightweight processes
- Asynchronous message passing
- Share-nothing model
- Process linking / monitoring
- Supervision trees and recovery strategies
- Transparent distribution model
- Soft-real time
- Let-it-fail philosophy
- Hot-code upgrades

WELL, IN FACT YOU NEED more.

ERLANG IS JUST
A Programming
Language.

YOU NEED Architecture patterns.

YOU NEED Middleware.

YOU NEED Libraries.

YOU NEED Tools.

You need OTP.

What is Middleware?

Design
Patterns
Fault
Tolerance
Distribution
Upgrades
Packaging

MIDDLEWARE

What are Libraries?

Storage
O&M
Interfaces
Communicati
on

LIBRARIES

What Tools?

Development
Test
Frameworks
Release &
Deployment
Debugging &
Monitoring

OTP TOOLS

OPEN SOURCE

OTP IS

PART OF THE ERLANG DISTRIBUTION

Less Code
Less Bugs
More Solid
Code
More
Tested
Code More
Free Time

Servers
Finite State
Machines
Event
Handlers
Supervisors
Applications

OTP

Let It Fail

```
convert(Day) ->
  case Day of
      monday    -> 1;
      tuesday   -> 2;
      wednesday -> 3;
      thursday  -> 4;
      friday    -> 5;
      saturday  -> 6;
      sunday    -> 7;
      Other ->
          {error, unknown_day}
  end.
```

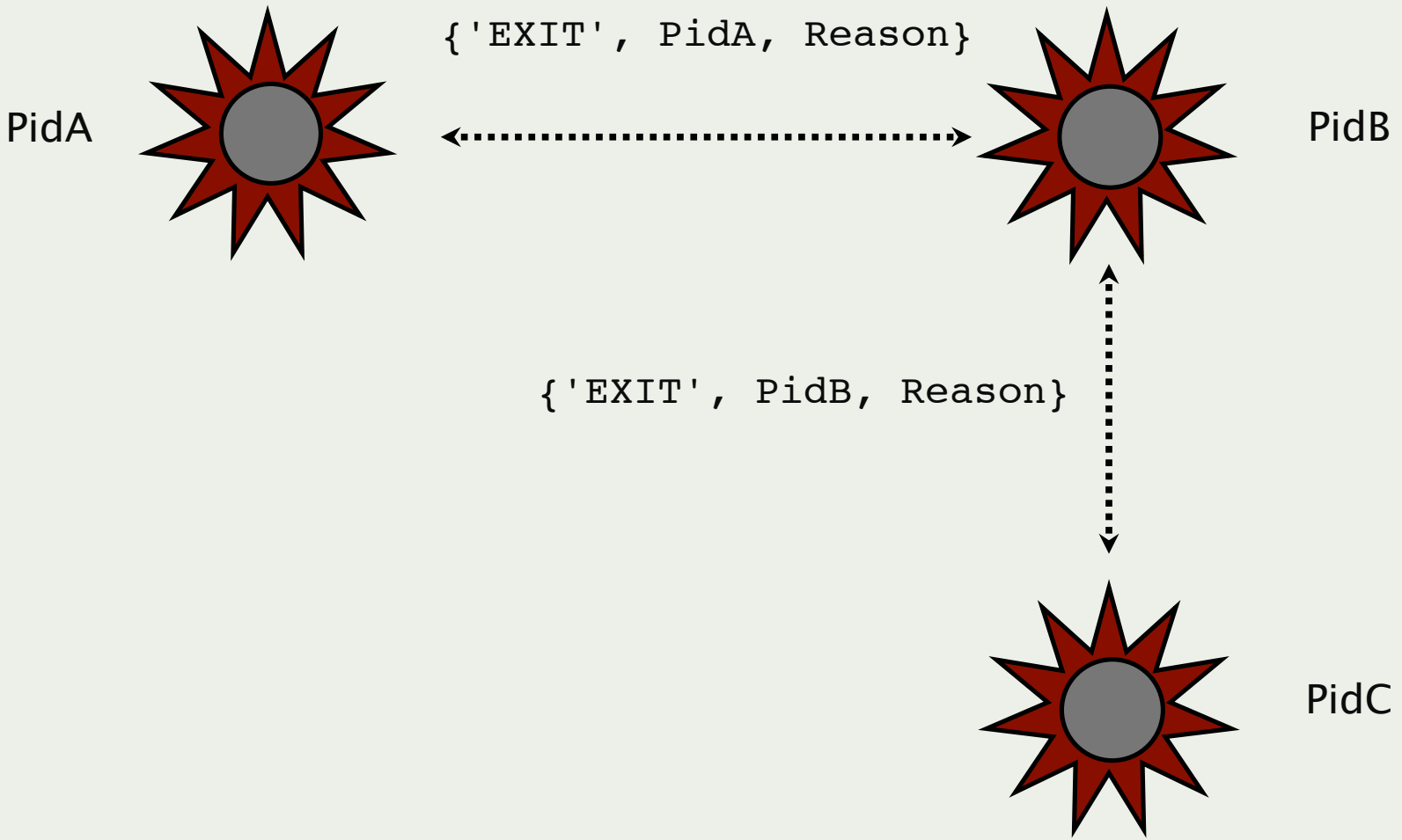
```
convert(Day) ->
  case Day of
      monday    -> 1;
      tuesday   -> 2;
      wednesday -> 3;
      thursday  -> 4;
      friday    -> 5;
      saturday  -> 6;
      sunday    -> 7;


  end.
```

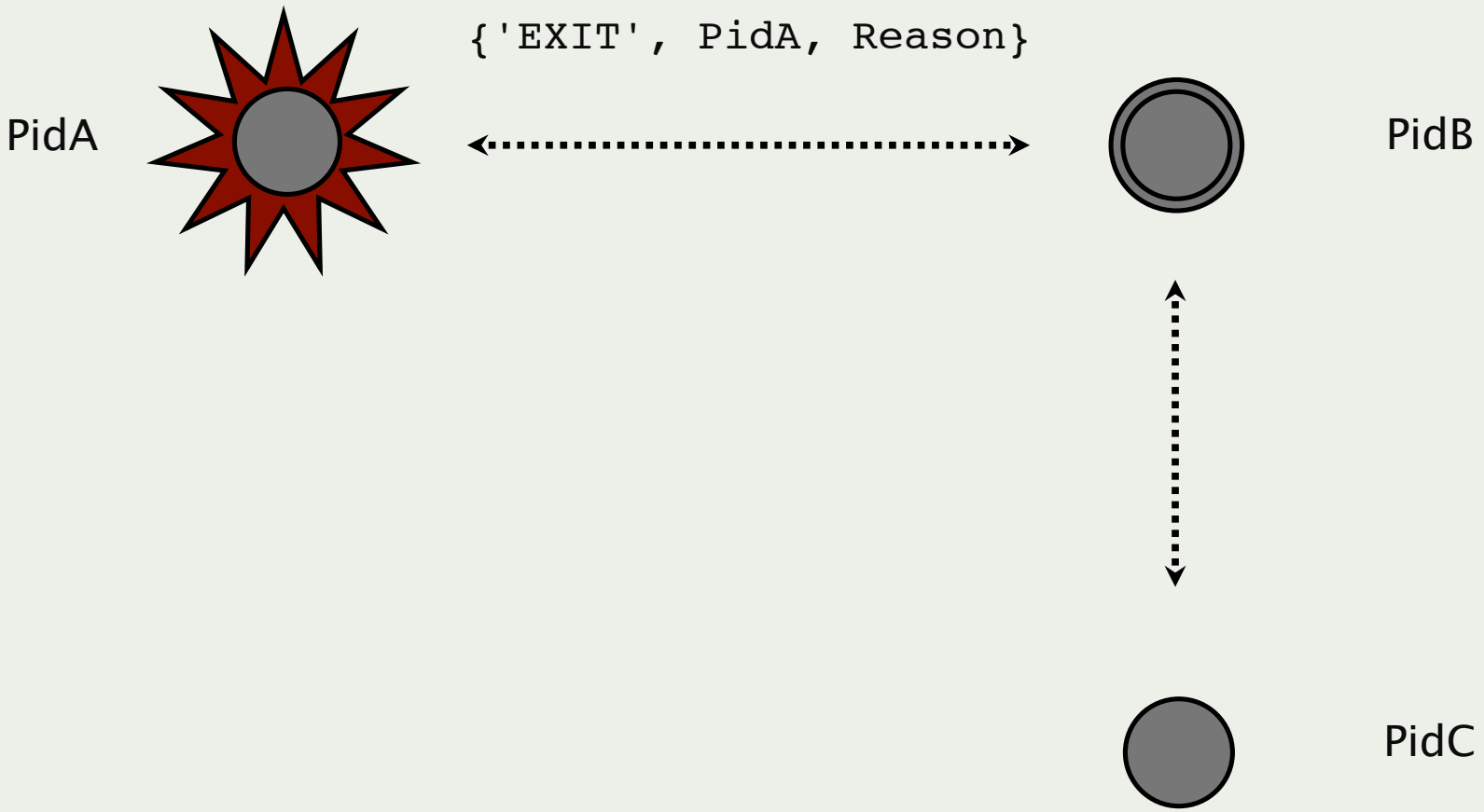
Let It Fail

Fail Safe!

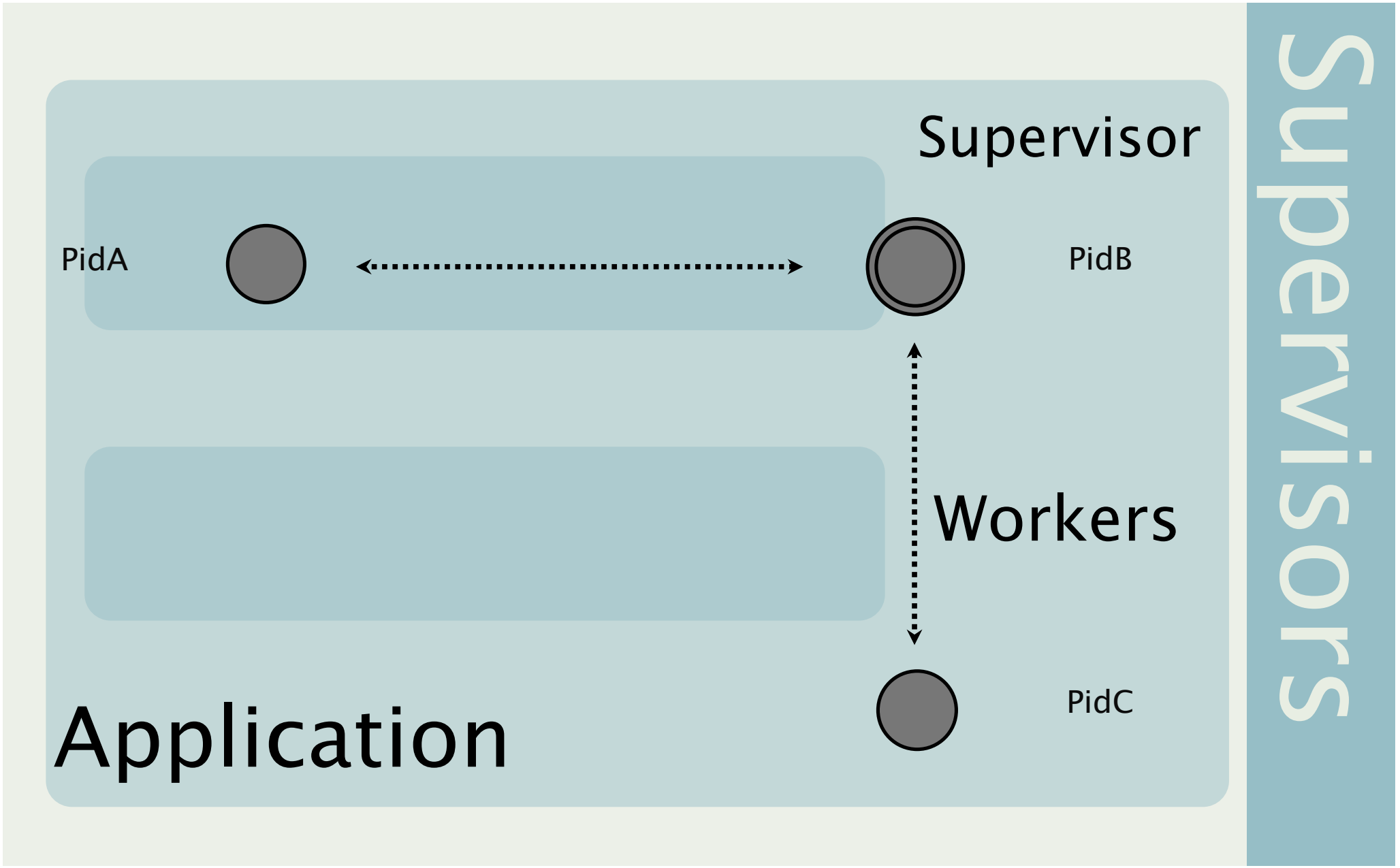
Propagating Exit Signals



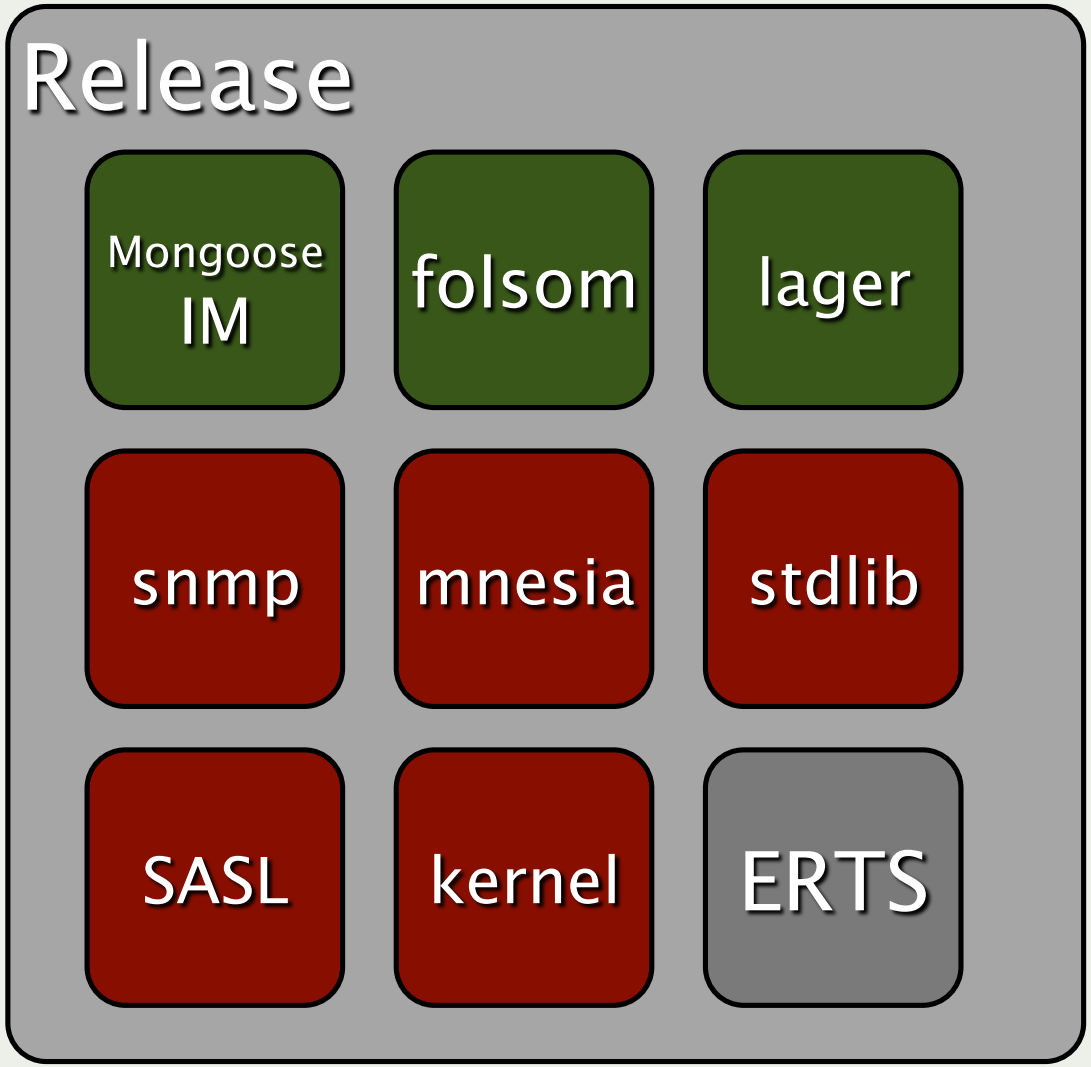
Trapping an Exit Signal



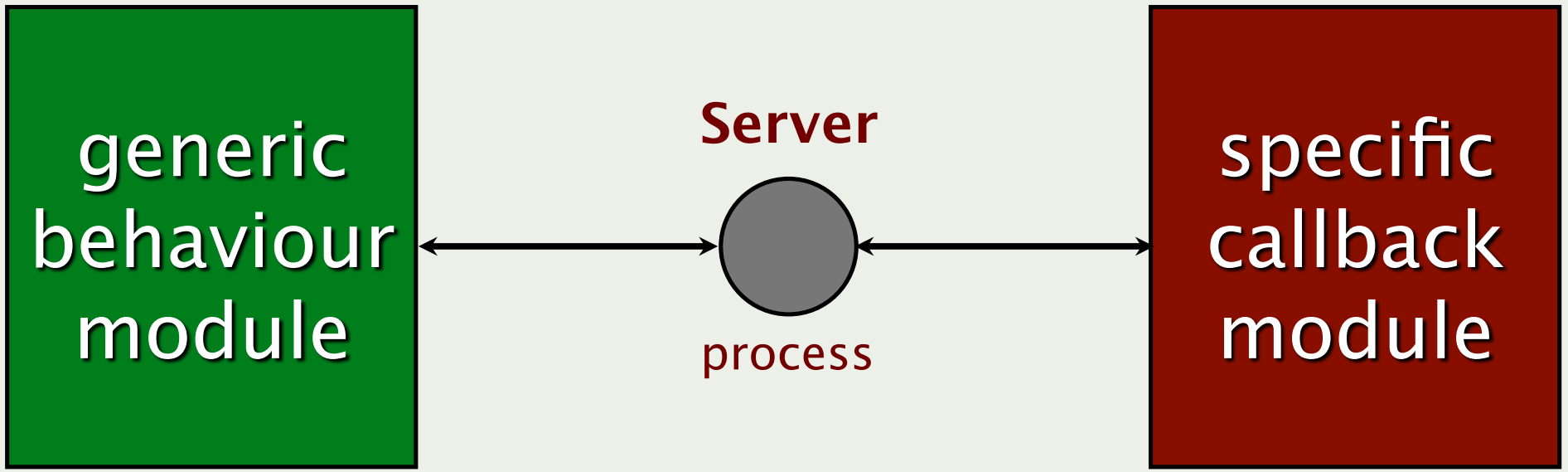
Trap Exit

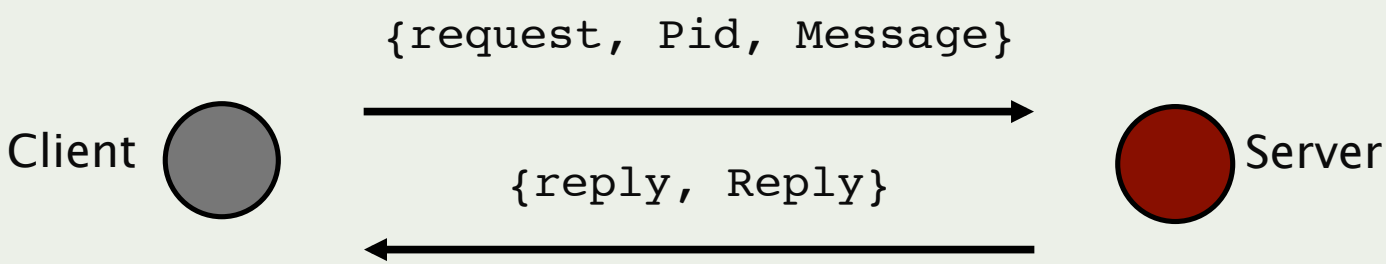

Supervisors
Supervisor
PidA
PidB
Workers
PidC
Application

Releases


Release
Mongoose IM
folsom
lager
snmp
mnesia
stdlib
SASL
kernel
ERTS

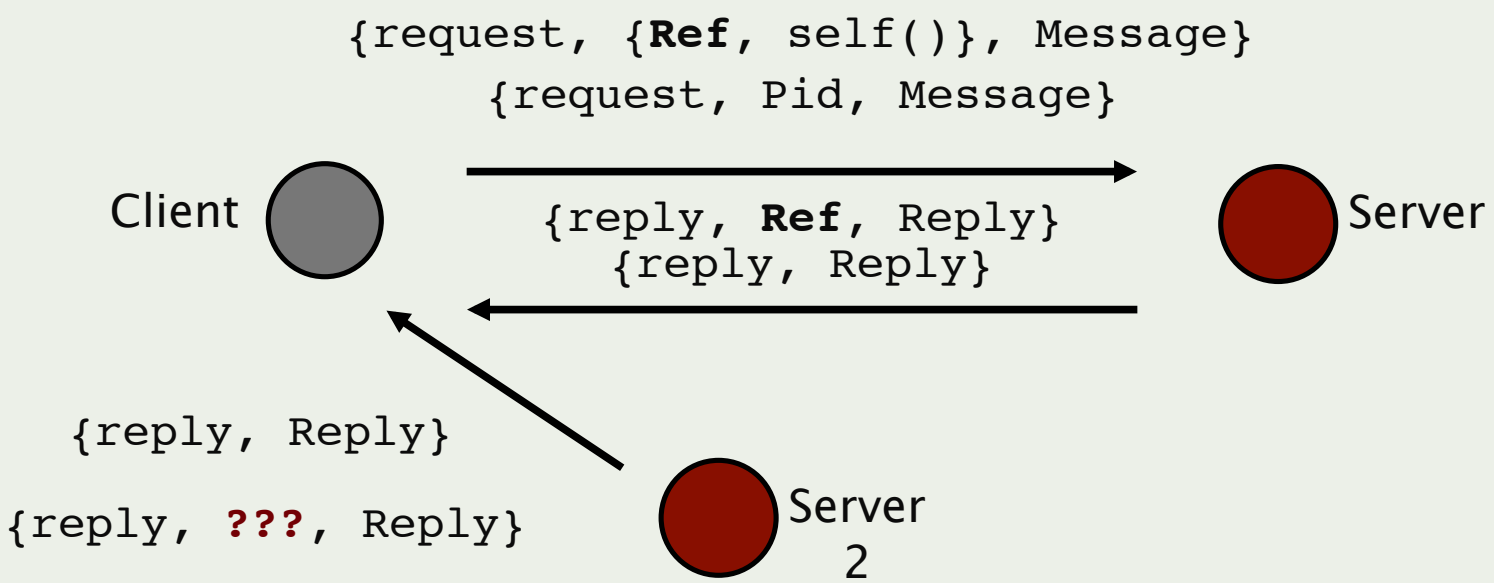
Behaviours


generic
behaviour
module
Server
process
specific
callback
module

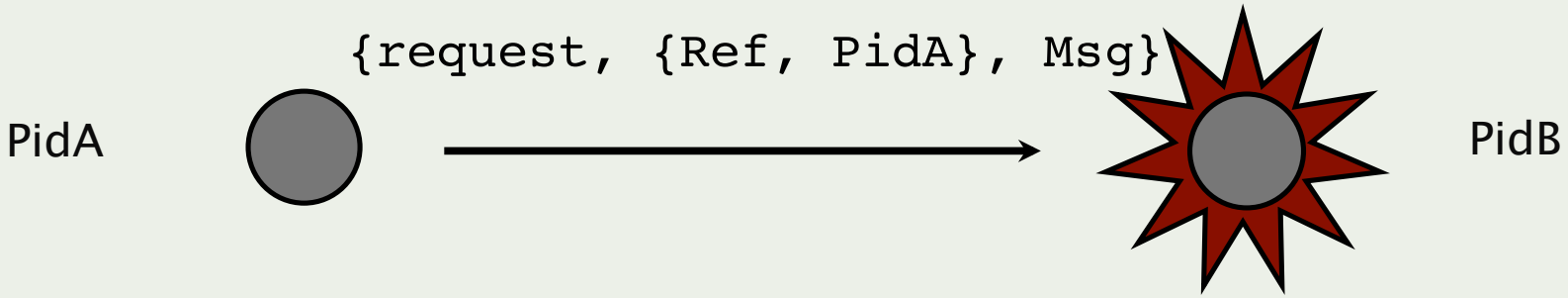



```
call(Name, Message) ->    Name !
{request, self(), Message},    receive
{reply, Reply} -> Reply    end.
```

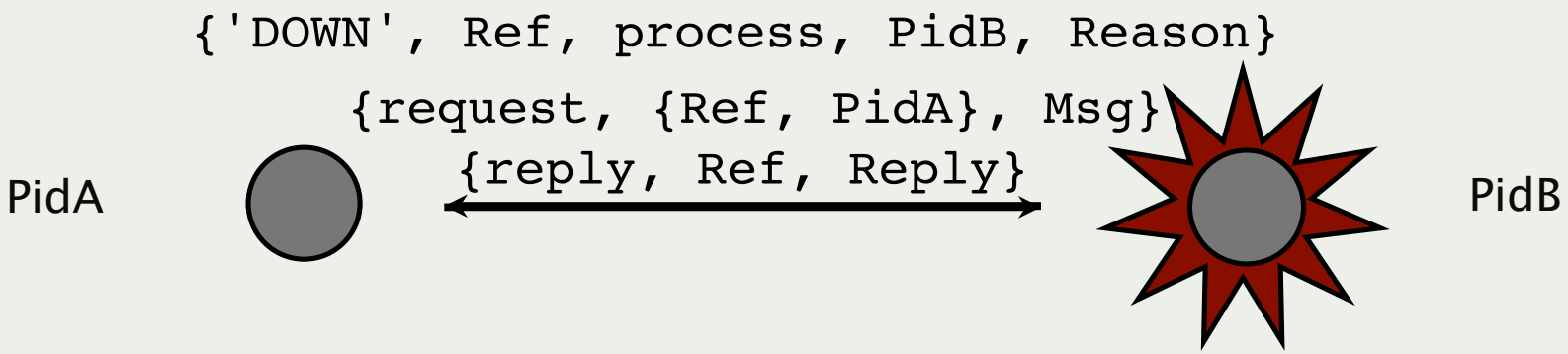
```
reply(Pid, Reply) ->
Pid ! {reply, Reply}.
```




```
call(Name, Msg) ->    Ref = make_ref(),
Name ! {request, {Ref, self()}, Msg},
receive {reply, Ref, Reply} -> Reply end.reply
({Ref, Pid}, Reply) ->    Pid ! {reply, Ref,
Reply}.
```




```
call(Name, Msg) ->    Ref = erlang:monitor
(process, Name),    Name ! {request, {Ref, self
()}, Msg},    receive      {reply, Ref, Reply} -
>          erlang:demonitor(Ref),          Reply;
{'DOWN', Ref, process, _Name, _Reason} ->
{error, no_proc}    end.
```



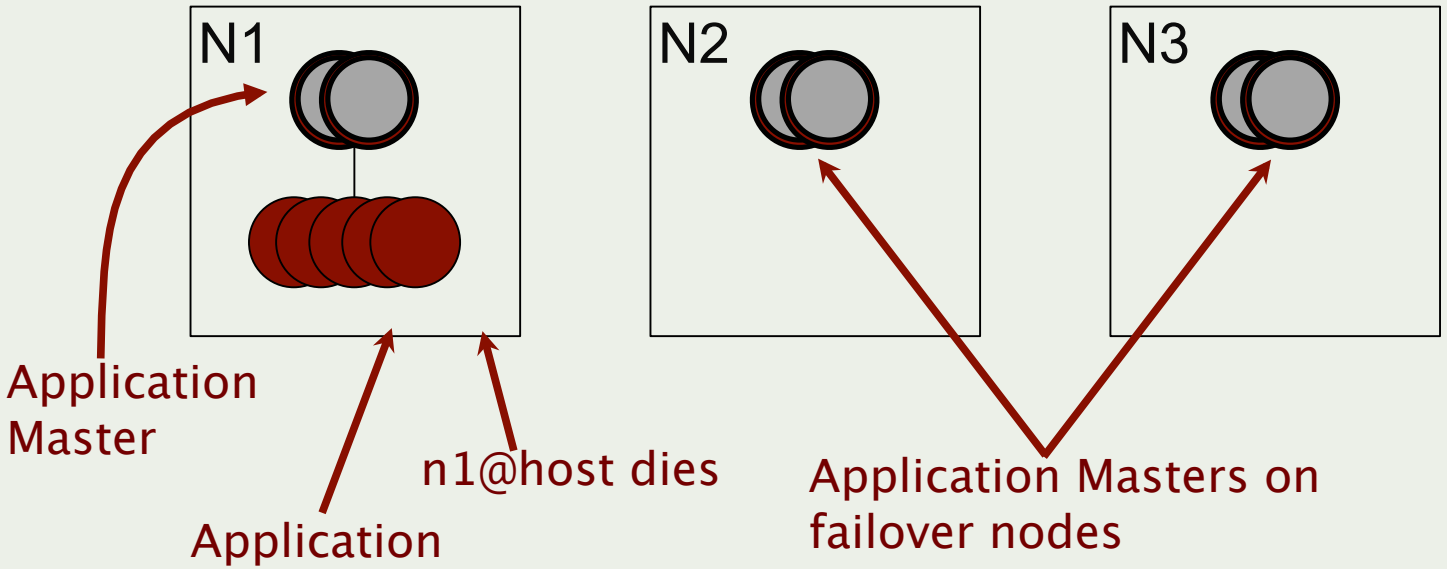

```
call(Name, Msg) ->    Ref = erlang:monitor
(process, Name),    Name ! {request, {Ref, self
()}, Msg},    receive      {reply, Ref, Reply} -
>          erlang:demonitor(Ref, [flush]),
Reply;          {'DOWN', Ref, process, _Name,
_Reason} ->      {error, no_proc}    end.
```

BEHAVIOURS

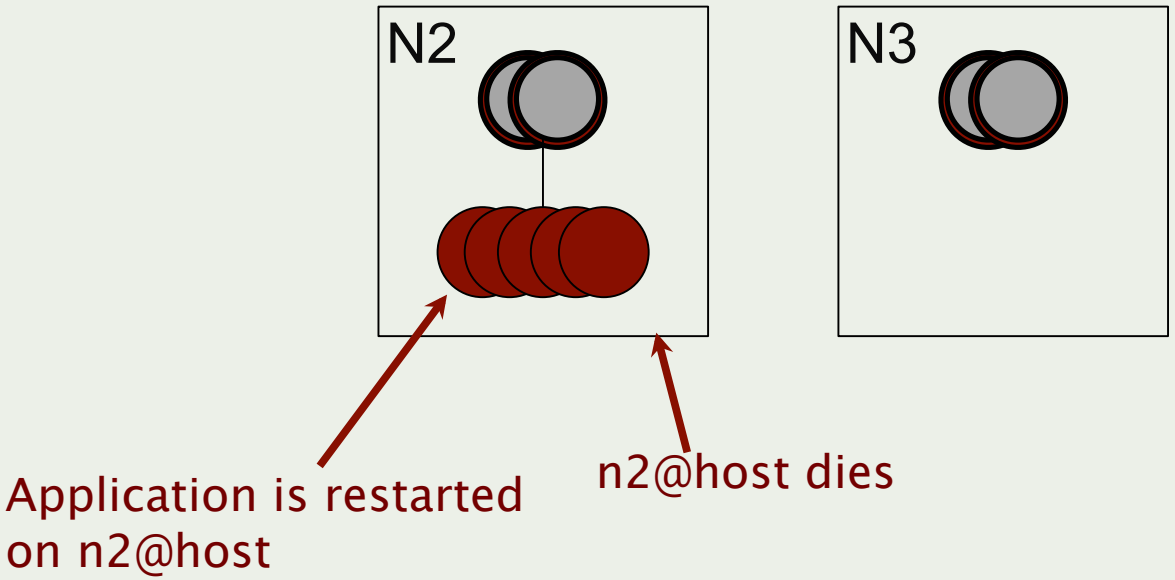
Timeouts
Deadlocks
tracing
Monitoring
Distribution

Automatic Takeover and Failover

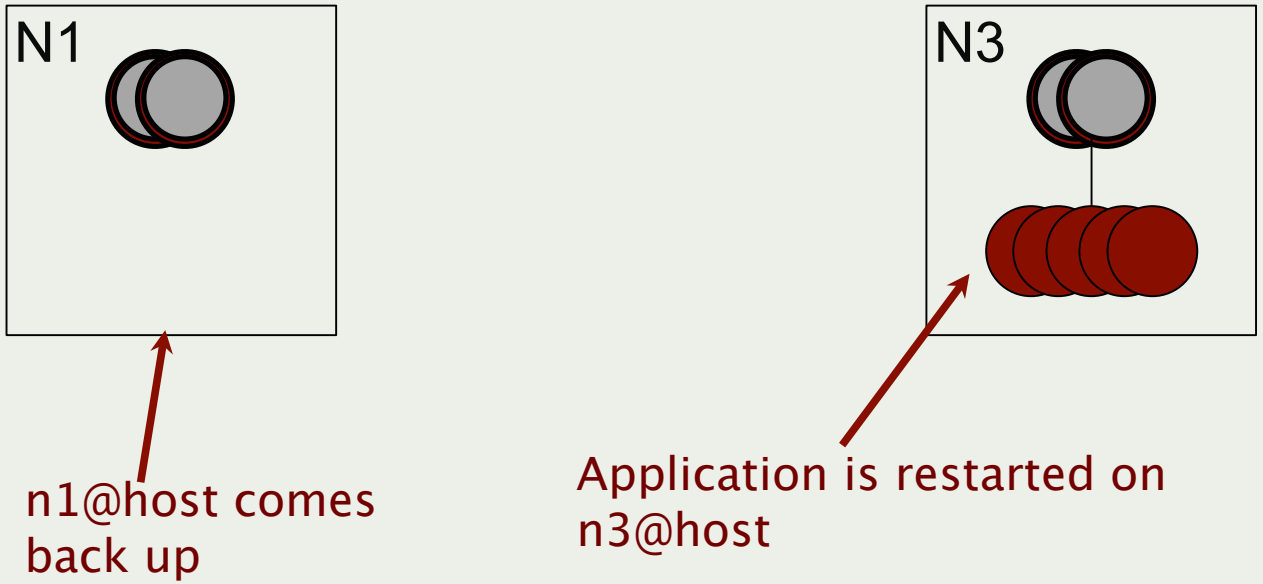
```
{myApp, 2000, {n1@host, {n2@host, n3@host}]}
```



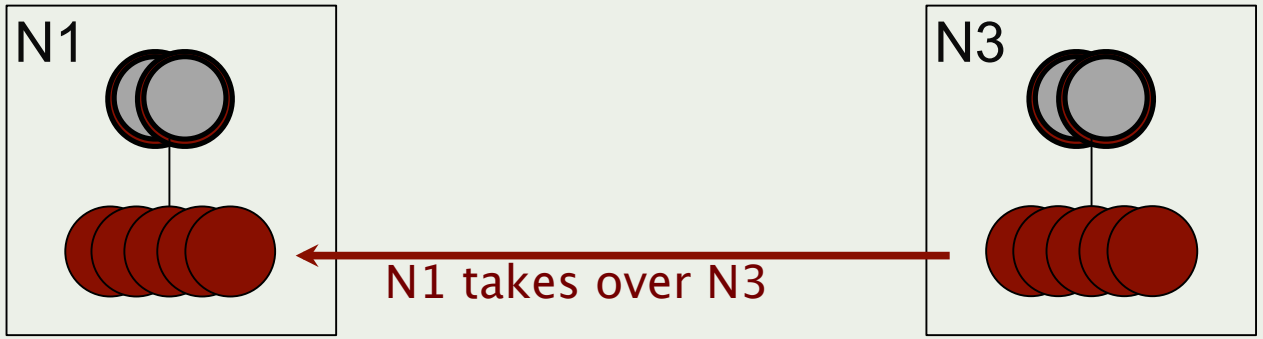
```
{myApp, 2000, {n1@host, {n2@host, n3@host}]}
```



```
{myApp, 2000, {n1@host, {n2@host, n3@host}]}
```



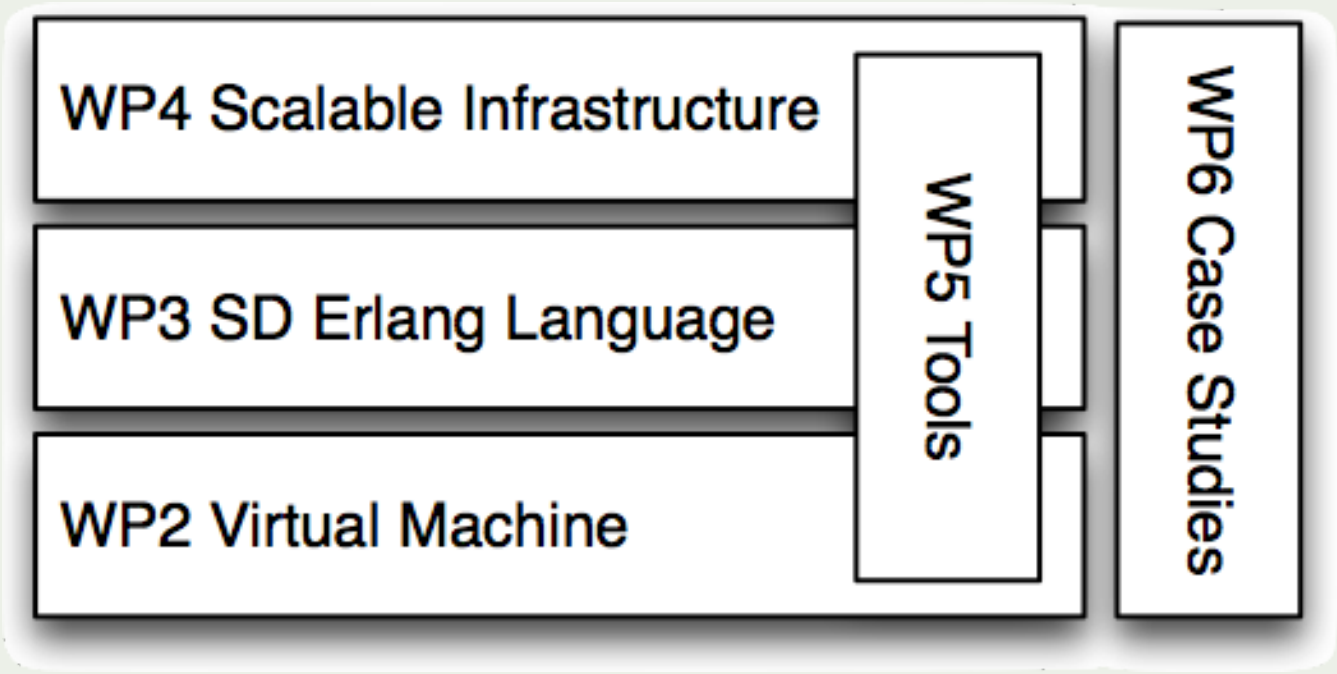
```
{myApp, 2000, {n1@host, {n2@host, n3@host}]}
```



RELEASE STATEMENT OF AIMS

"To scale the radical concurrency-oriented programming paradigm to build reliable general-purpose software, such as server-based systems, on massively parallel machines (10^5 cores)."

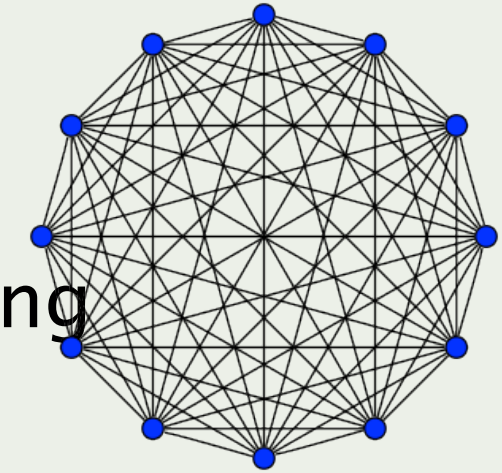
LIMITATIONS ARE PRESENT AT THREE LEVELS

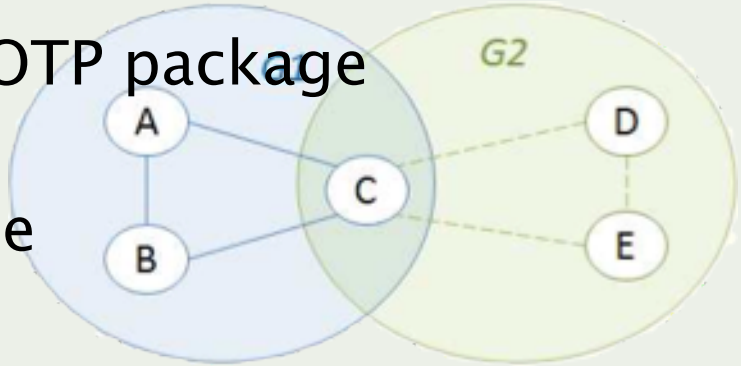


VM | LANGUAGE | INFRASTRUCTURE

- Push the responsibility for scalability from the programmer to the vm
- Analyze performance and scalability
- Identify bottlenecks and prioritize changes and extensions
- Tackle well-known scalability issues
  - Ets tables (shared global data structure)
  - Message passing, copying and frequently communicating processes

- Two major issues
  - Fully connected clusters
  - Explicit process placement
- Scalable Distributed (SD) Erlang
  - Nodes grouping
  - Non-transitive connections
  - Implicit process placement
  - Part of the standard Erlang/OTP package
- New concepts introduced
  - Locality, Affinity and Distance





VM | LANGUAGE | INFRASTRUCTURE

CCL/sɪˈsɪlɪ/

- Middleware layer
- Set of Erlang Applications
- Create and manage clusters of (heterogeneous) erlang nodes
- API to monitor and control erlang distributed systems
- Existing tracing/logging/debugging tools pluggable
- Broker layer between users and cloud providers
- Auto-scaling

... And Much More

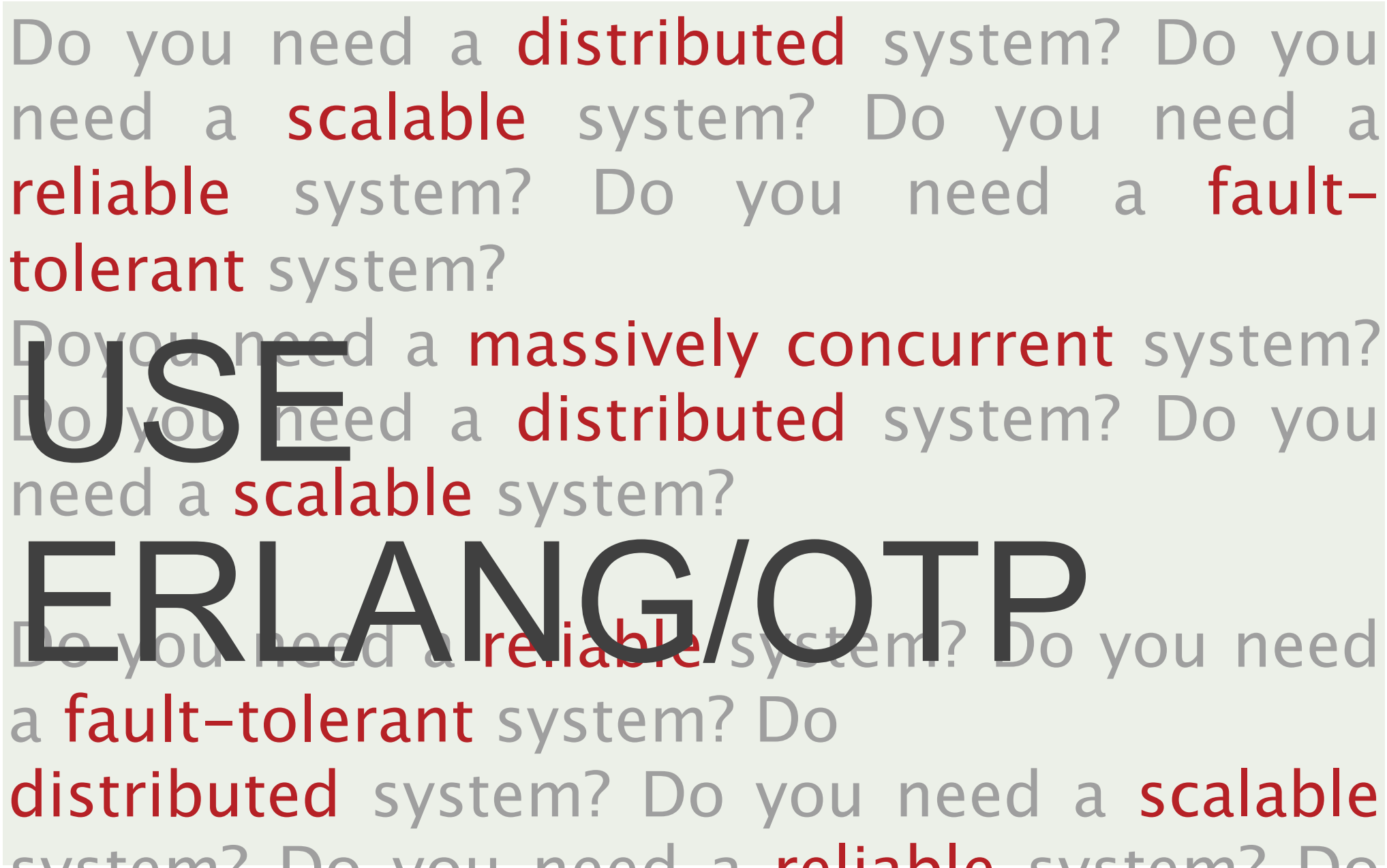
Conclusions

Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system? Do you need a massively concurrent system? Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system?

USE

Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system? Do you need a massively

ERLANG


Do you need a distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system?
Doyou need a massively concurrent system? Do you need a distributed system? Do you need a scalable system?
USE
ERLANG/OTP
Do you need a reliable system? Do you need a fault-tolerant system? Do
distributed system? Do you need a scalable system? Do you need a reliable system? Do you need a fault-tolerant system? Do you

Questions?

@francescoC