UNORTHORDOX DESIGN PATTERNS IN RABBITMQ

AYANDA DUBE

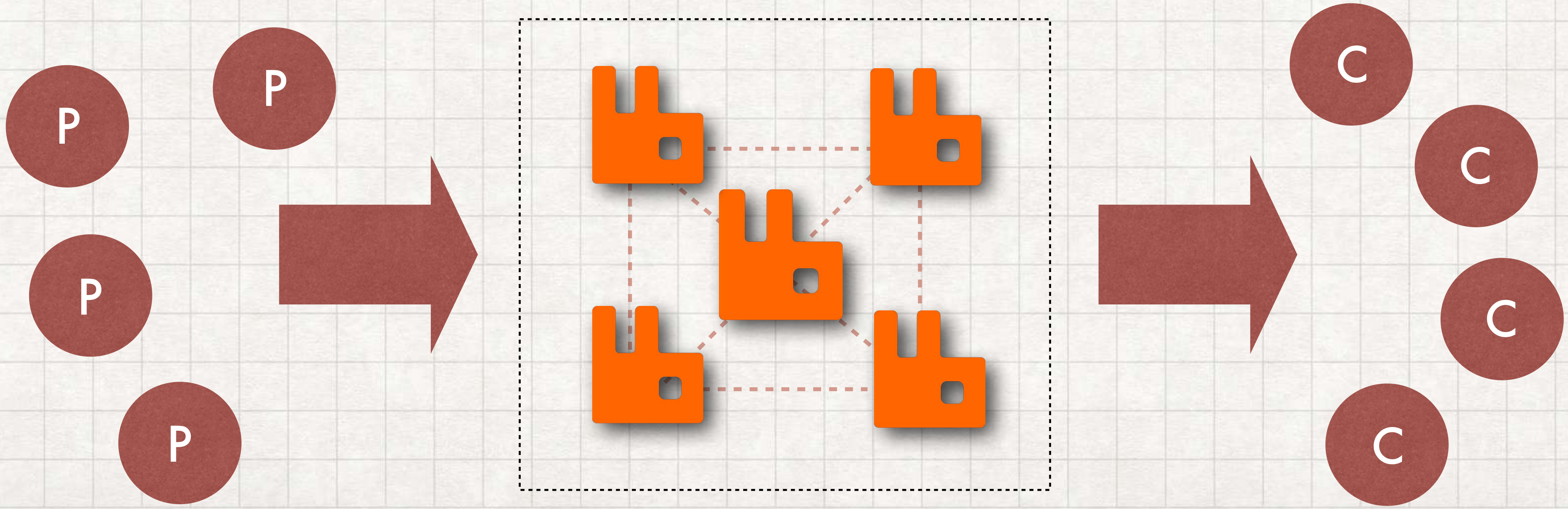
ERLANG SOLUTIONS

ACKNOWLEDGEMENTS

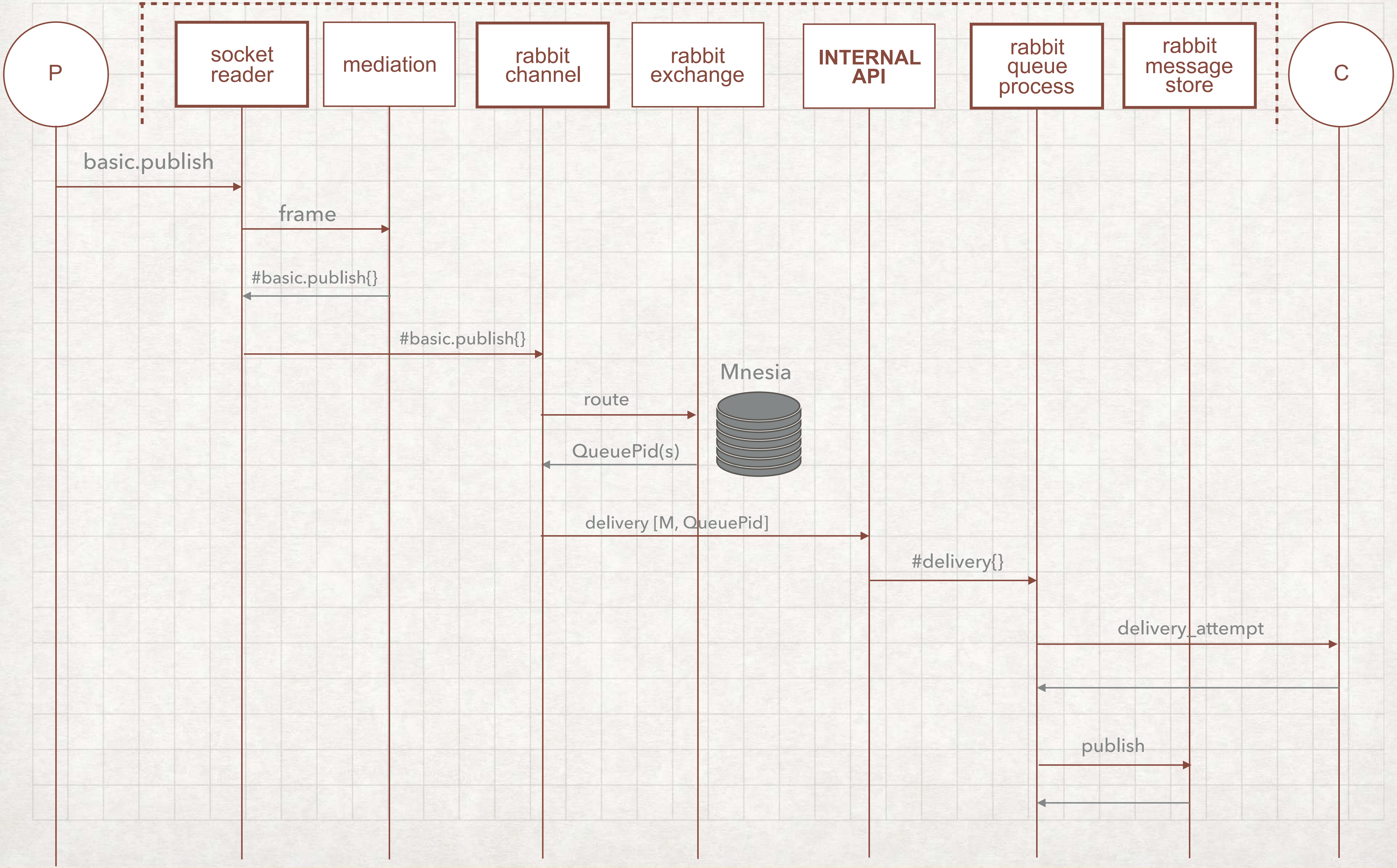
- **Joe Armstrong** - the legacy
- Erlang Solutions conferences (Gurpreet & crew)

OVERVIEW: RABBITMQ

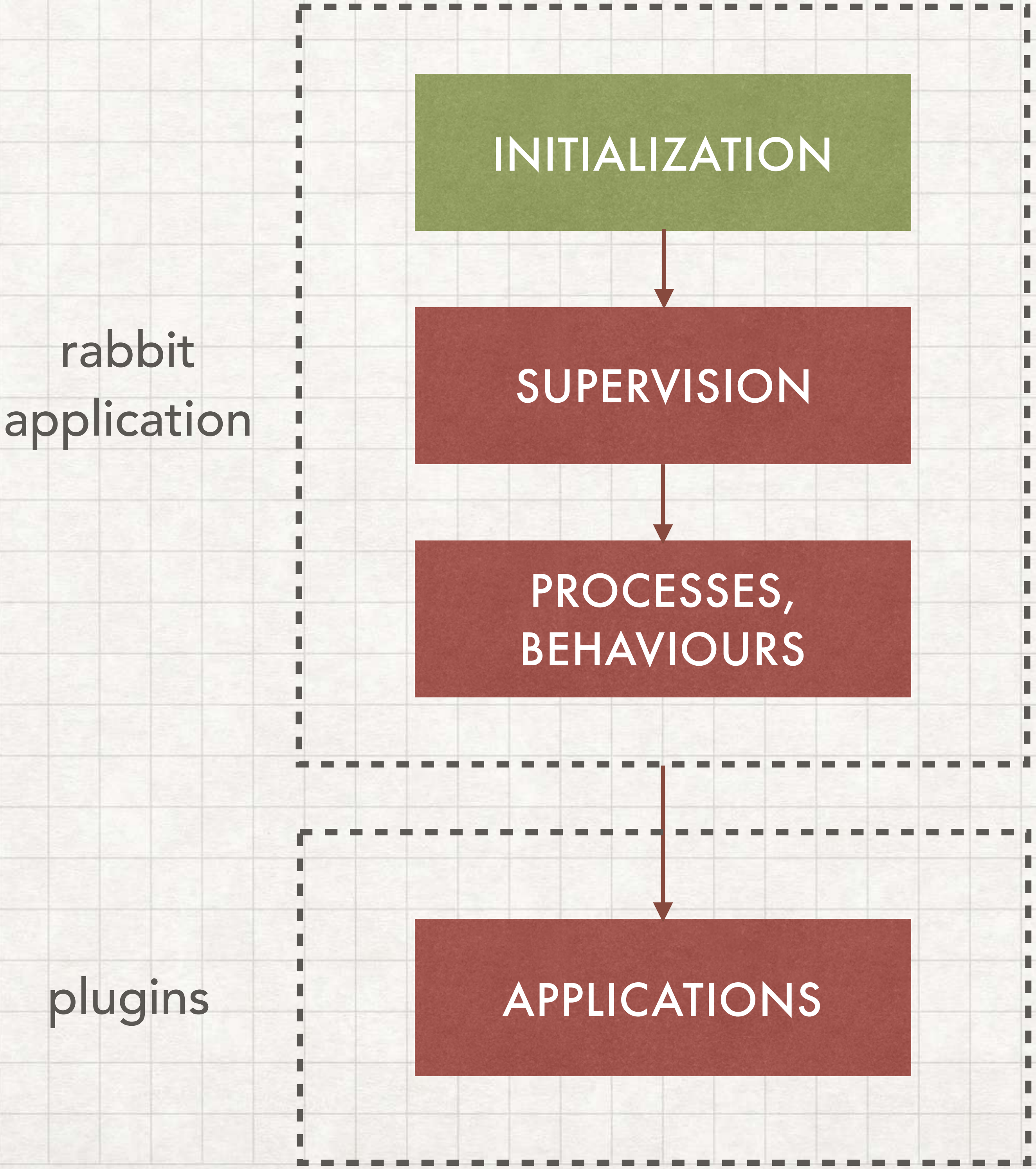
- Erlang & Elixir AMQP implementation
- Client libraries (Java, .Net, Objective-C, JMS, PHP,)
- Been around for a while - over 10years
- More than 35,000 known deployments



OVERVIEW: INTERNALS



OVERVIEW: ROADMAP



rabbit application

INITIALIZATION

SUPERVISION

PROCESSES, BEHAVIOURS

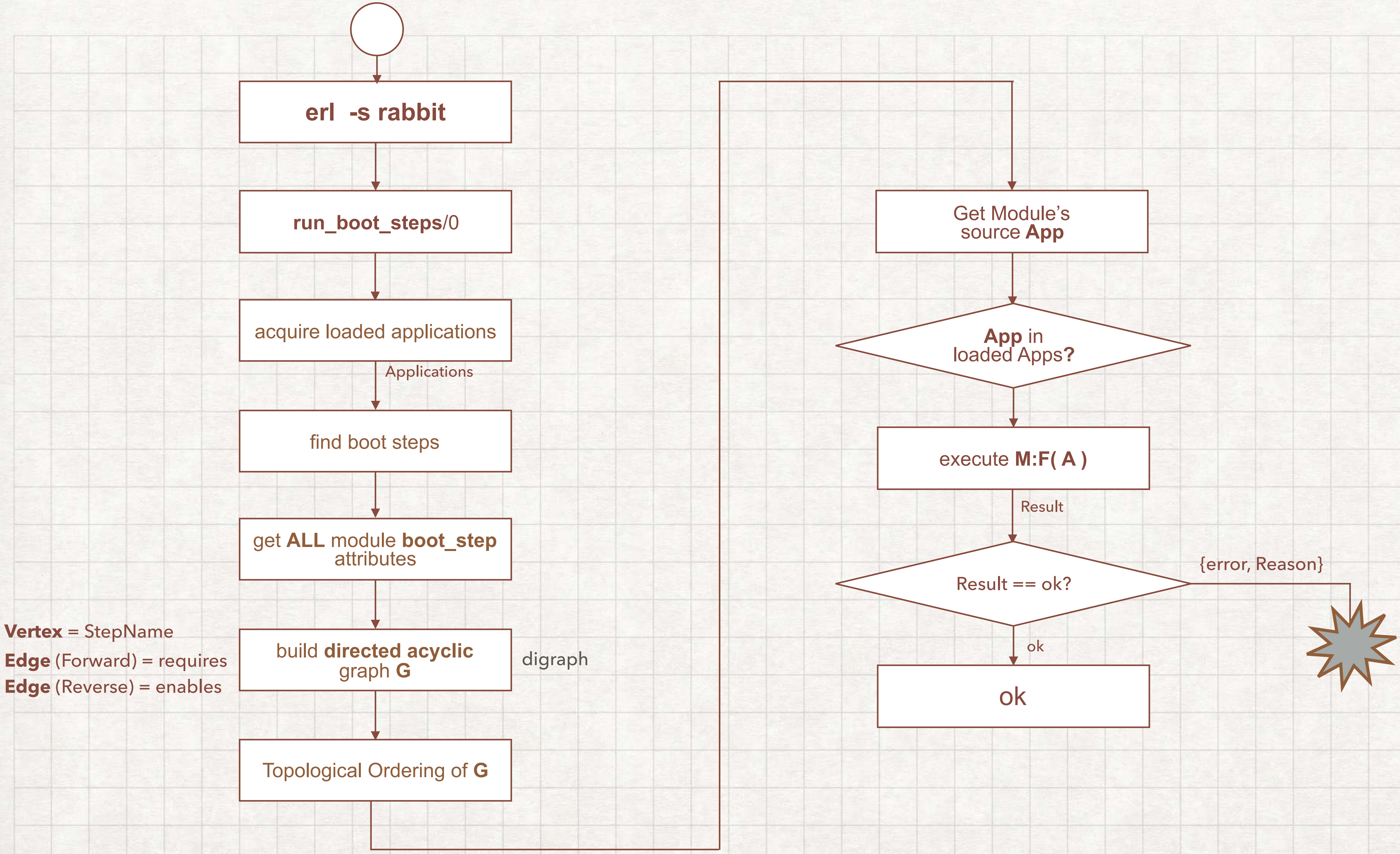
plugins

APPLICATIONS

INITIALIZATION: BOOTSTEPS

- Initialisation: **boot steps**
- Loosely coupled application procedures/steps
- Ordered execution of initialisation steps
- Defined and set as **module attributes**
- Pre-conditions and post-conditions
- Cleanup capabilities on shutdown
- Alternative to **OTP** application start-phases

INITIALIZATION: BOOTSTEPS

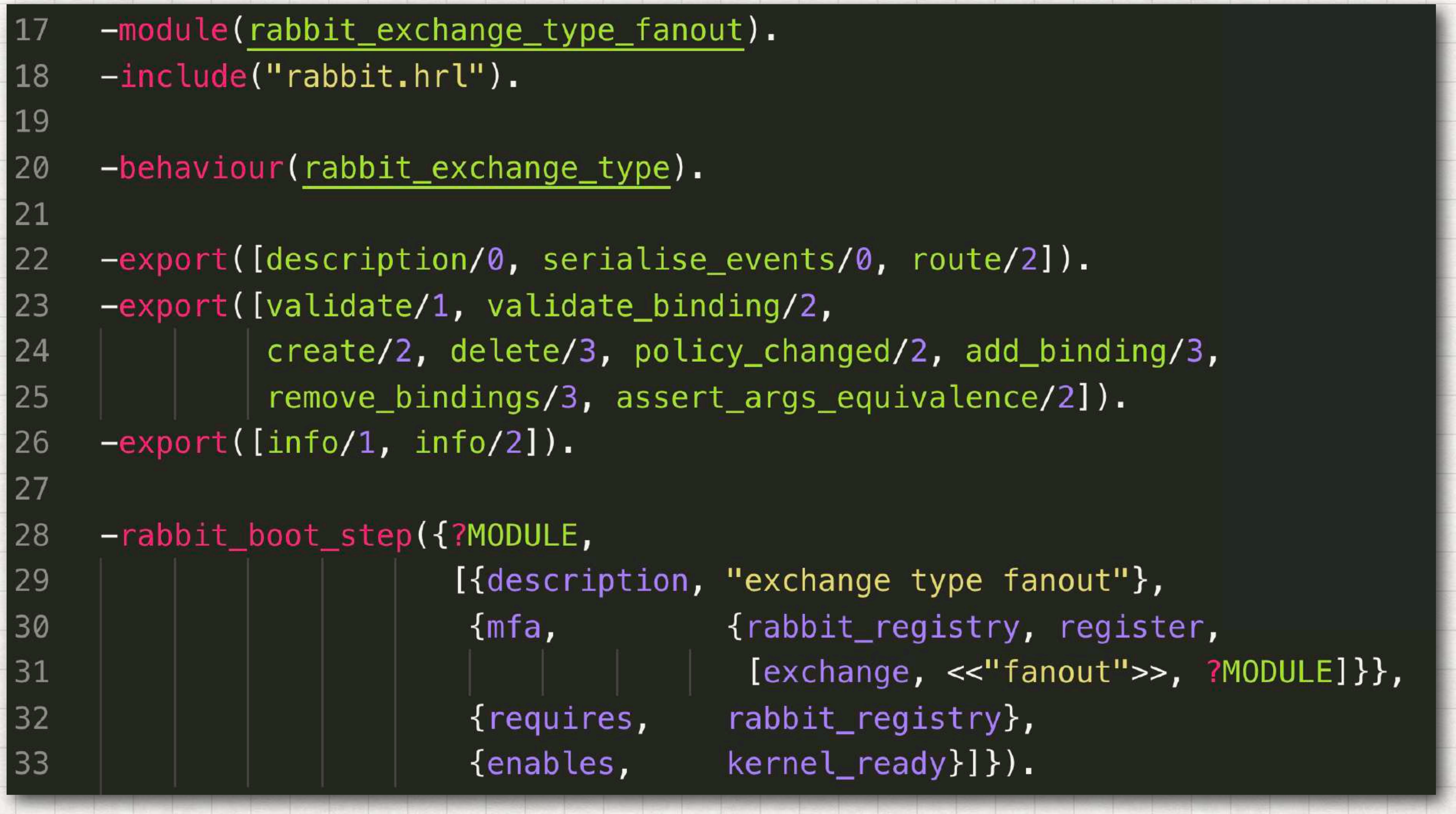


Vertex = StepName

Edge (Forward) = requires

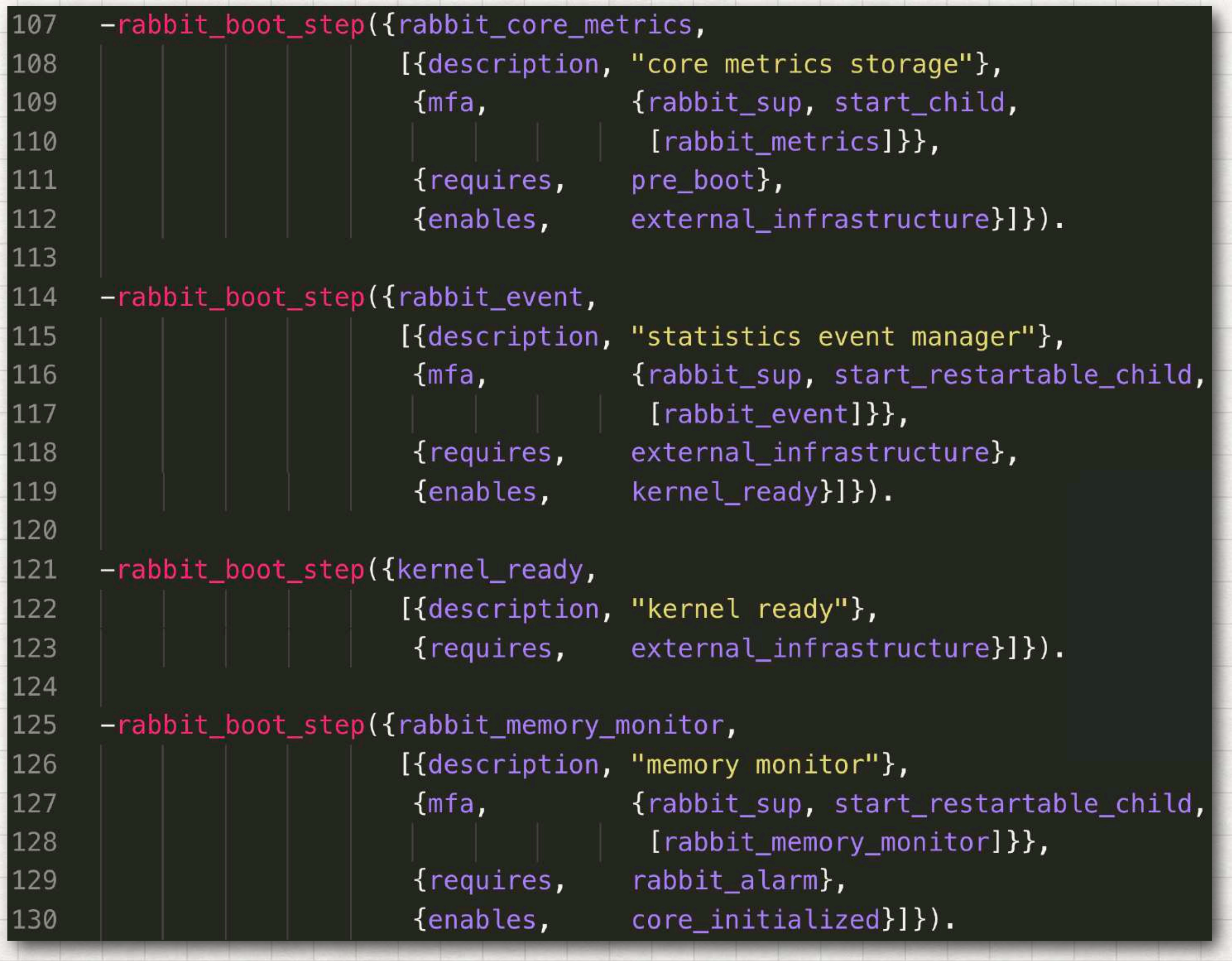
Edge (Reverse) = enables

INITIALIZATION: BOOTSTEPS



```erlang
-module(rabbit_exchange_type_fanout).
-include("rabbit.hrl").

-behaviour(rabbit_exchange_type).

-export([description/0, serialise_events/0, route/2]).
-export([validate/1, validate_binding/2,
         create/2, delete/3, policy_changed/2, add_binding/3,
         remove_bindings/3, assert_args_equivalence/2]).
-export([info/1, info/2]).

-rabbit_boot_step({?MODULE,
                   [{description, "exchange type fanout"},
                    {mfa,         {rabbit_registry, register,
                                   [exchange, <<"fanout">>, ?MODULE]}},
                    {requires,    rabbit_registry},
                    {enables,     kernel_ready}]}).
```

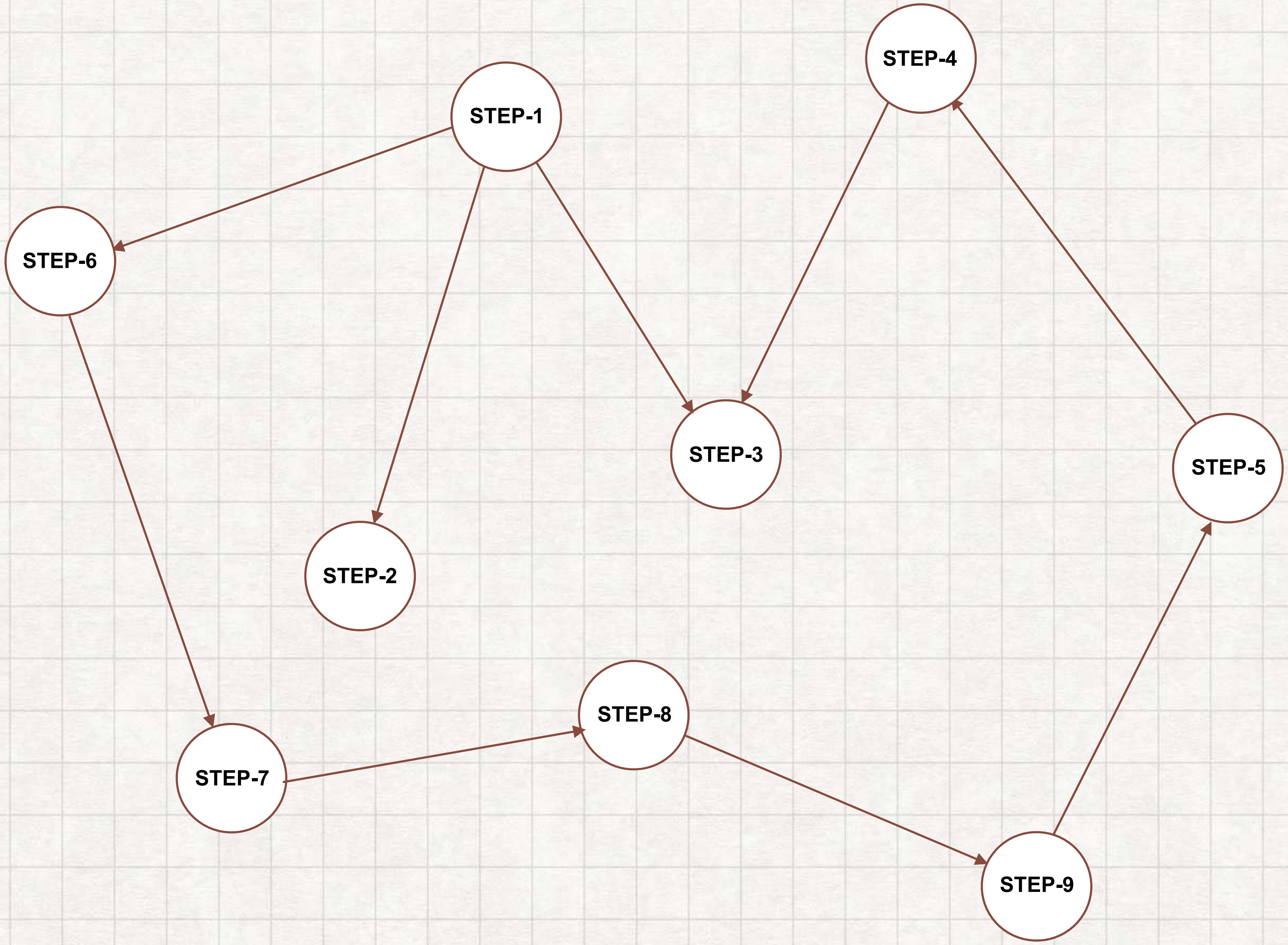
INITIALIZATION: BOOTSTEPS



```
-rabbit_boot_step({rabbit_core_metrics,
                   [{description, "core metrics storage"},
                    {mfa,         {rabbit_sup, start_child,
                                   [rabbit_metrics]}},
                    {requires,    pre_boot},
                    {enables,     external_infrastructure}]}).

-rabbit_boot_step({rabbit_event,
                   [{description, "statistics event manager"},
                    {mfa,         {rabbit_sup, start_restartable_child,
                                   [rabbit_event]}},
                    {requires,    external_infrastructure},
                    {enables,     kernel_ready}]}).

-rabbit_boot_step({kernel_ready,
                   [{description, "kernel ready"},
                    {requires,    external_infrastructure}]}).

-rabbit_boot_step({rabbit_memory_monitor,
                   [{description, "memory monitor"},
                    {mfa,         {rabbit_sup, start_restartable_child,
                                   [rabbit_memory_monitor]}},
                    {requires,    rabbit_alarm},
                    {enables,     core_initialized}]}).
```

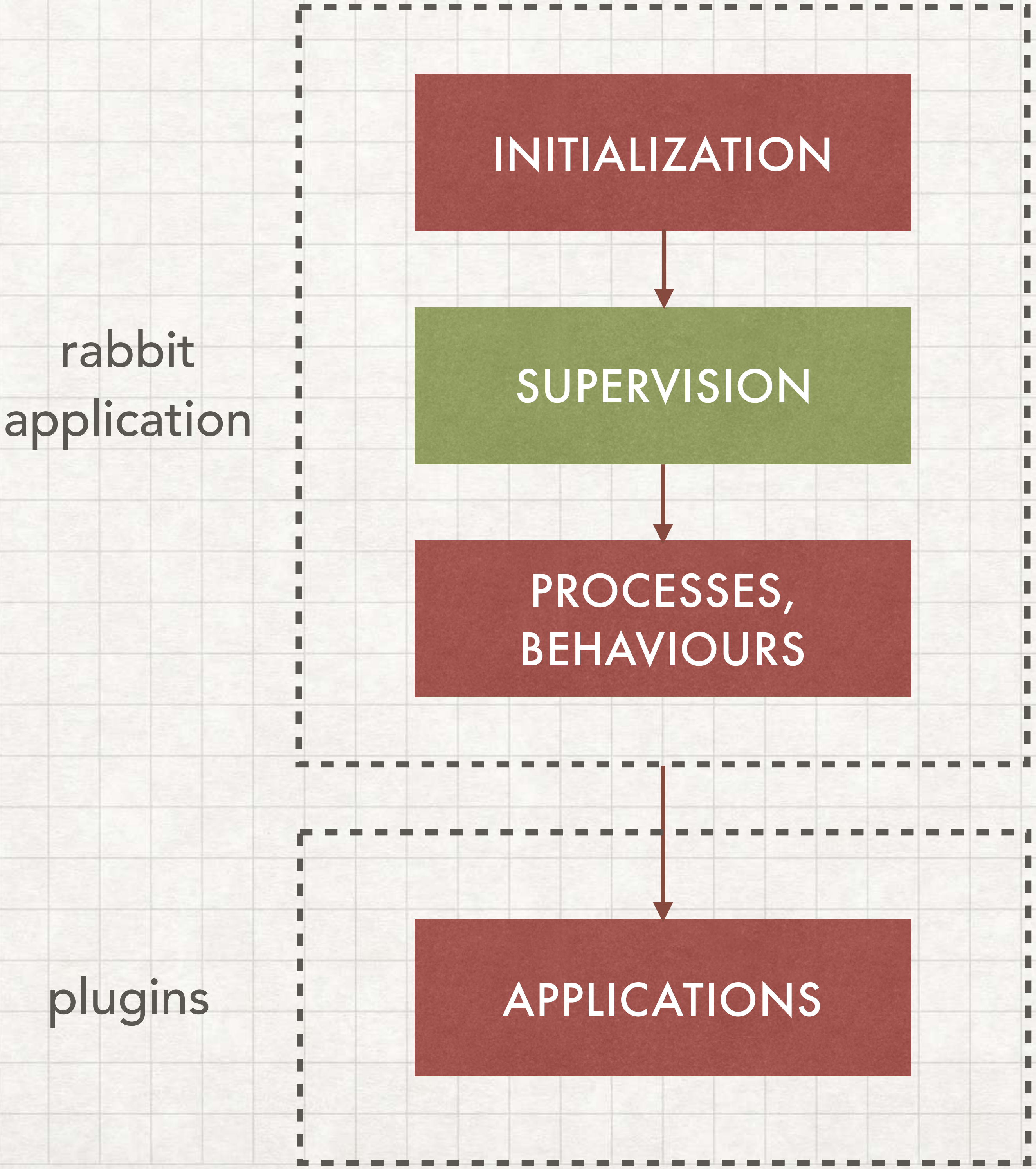
INITIALIZATION: BOOTSTEPS

- Topologically sorted: digraph_utils:topsort(G)



[STEP-1, STEP-6, STEP-7, STEP-8, STEP-9, STEP-5, STEP-4, STEP-2, STEP-3]

SUPERVISION



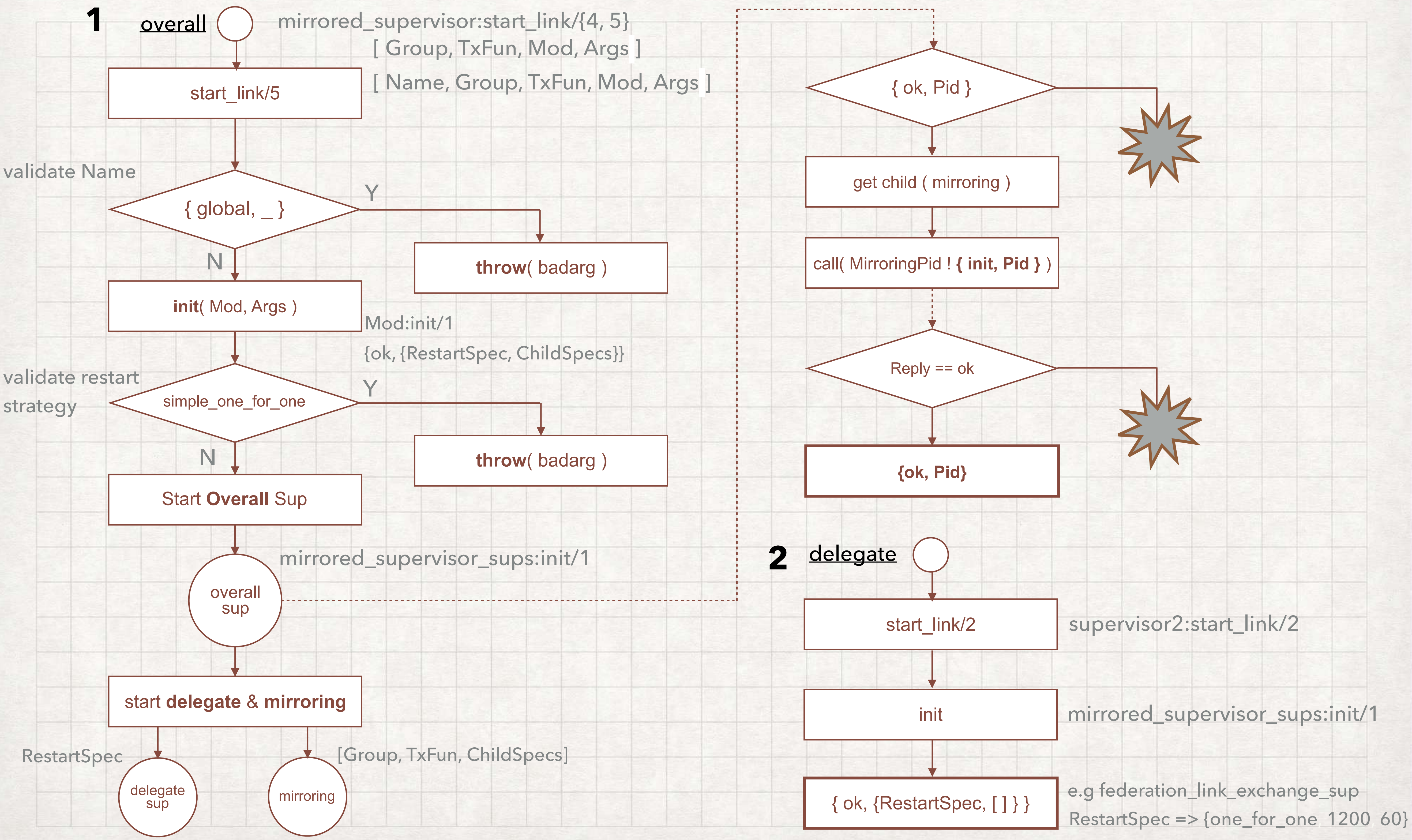
SUPERVISORS: SUPERVISOR2

- Extension of OTP supervisor
- **Intrinsic** restart type (restarts on abnormal exists)
  - If child exists normally, sup also exits normally
- Delayed restart types, e.g. {intrinsic, Delay}
  - Sup continues after Delay to restart child if MaxRestarts and MaxTime were exceeded
- Find child utilities, …

SUPERVISORS: MIRRORED SUPERVISOR

- Multiple supervisors within a single process group
- Child specifications retained in Mnesia
- Processes than need to exist once in a cluster
- Low state footprint
- Process recovery on separate node in case of node failure
- {global, Name} registration **not** supported

SUPERVISORS: MIRRORED SUPERVISOR



1 overall

mirrored_supervisor:start_link/{4, 5}

[Group, TxFun, Mod, Args]

[Name, Group, TxFun, Mod, Args]

start_link/5

validate Name

{ global, _ }

Y

throw(badarg)

N

init(Mod, Args)

Mod:init/1

{ok, {RestartSpec, ChildSpecs}}

validate restart strategy

simple_one_for_one

Y

throw(badarg)

N

Start **Overall** Sup

mirrored_supervisor_sups:init/1

overall sup

start **delegate** & **mirroring**

RestartSpec

delegate sup

[Group, TxFun, ChildSpecs]

mirroring

{ ok, Pid }

get child (mirroring)

call(MirroringPid ! **{ init, Pid }**)

Reply == ok

{ok, Pid}

2 delegate

start_link/2

supervisor2:start_link/2

init

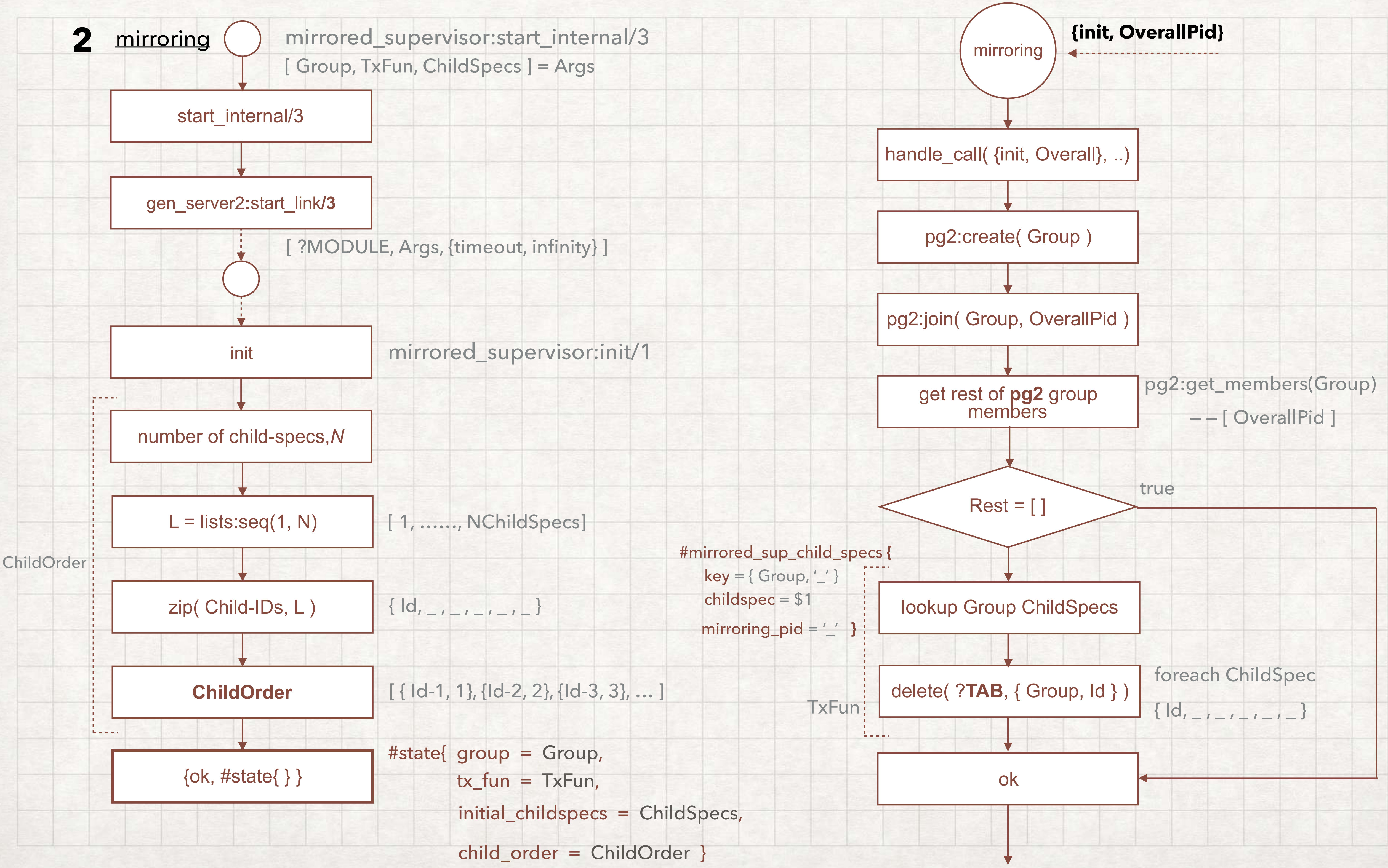
mirrored_supervisor_sups:init/1

{ ok, {RestartSpec, [ ] } }

e.g federation_link_exchange_sup

RestartSpec => {one_for_one, 1200, 60}

SUPERVISORS: MIRRORED SUPERVISOR



2 mirroring

mirrored_supervisor:start_internal/3

[Group, TxFun, ChildSpecs] = Args

start_internal/3

gen_server2:start_link**/3**

[?MODULE, Args, {timeout, infinity}]

init

mirrored_supervisor:init/1

number of child-specs,*N*

L = lists:seq(1, N)

[1, ……, NChildSpecs]

ChildOrder

zip(Child-IDs, L)

{ Id, _ , _ , _ , _ , _ }

ChildOrder

[{ Id-1, 1}, {Id-2, 2}, {Id-3, 3}, …]

{ok, #state{ } }

#state{ group = Group,
tx_fun = TxFun,
initial_childspecs = ChildSpecs,
child_order = ChildOrder }

mirroring

{init, OverallPid}

handle_call({init, Overall}, ..)

pg2:create(Group)

pg2:join(Group, OverallPid)

get rest of **pg2** group members

pg2:get_members(Group)

– – [OverallPid]

Rest = [ ]

true

#mirrored_sup_child_specs {
key = { Group, '_' }
childspec = $1
mirroring_pid = '_' }

lookup Group ChildSpecs

TxFun

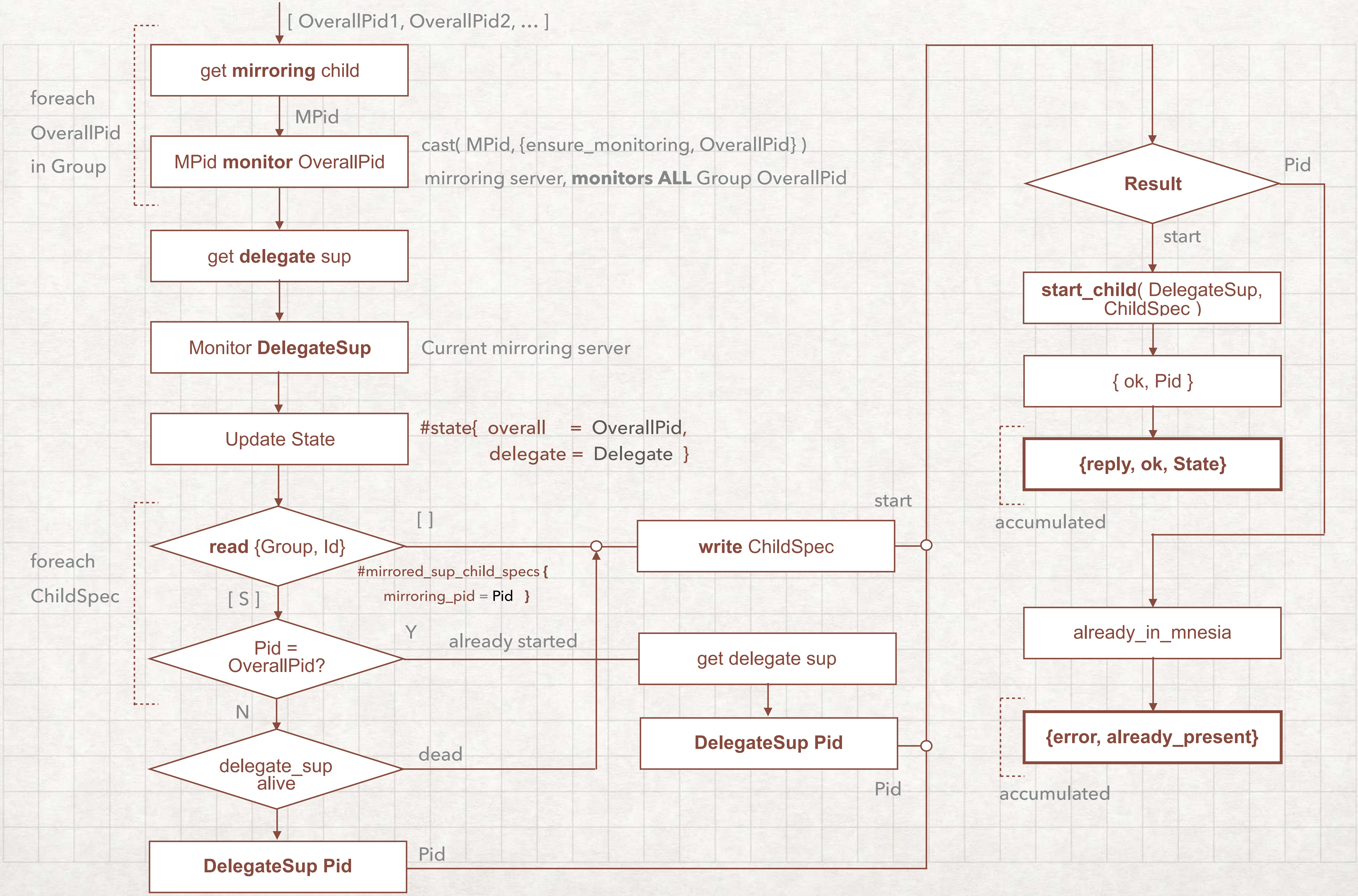
delete(?**TAB**, { Group, Id })

foreach ChildSpec

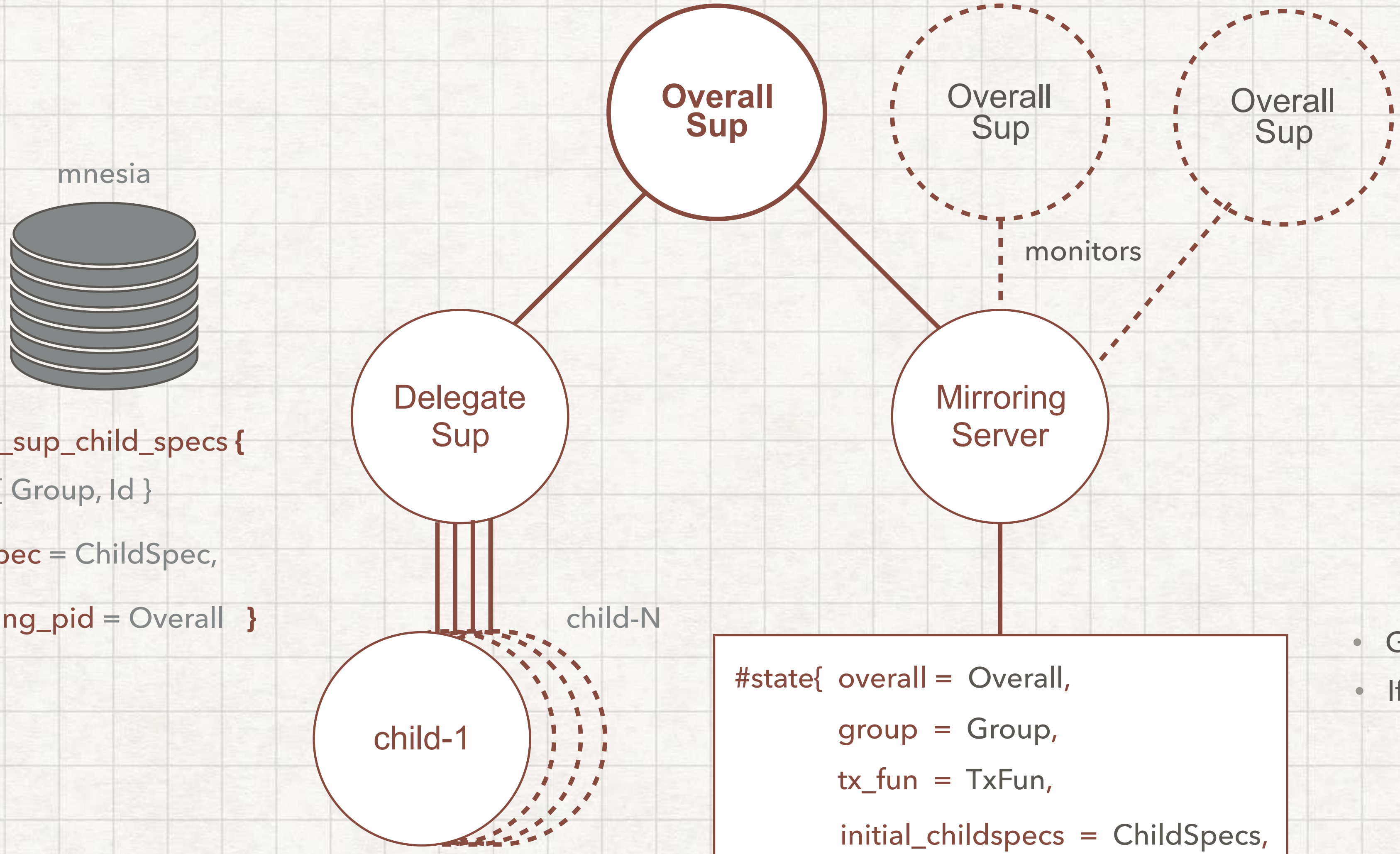
{ Id, _ , _ , _ , _ , _ }

ok

SUPERVISORS: MIRRORED SUPERVISOR



SUPERVISORS: MIRRORED SUPERVISOR



```
#mirrored_sup_child_specs {
    key = { Group, Id }
    childspec = ChildSpec,
    mirroring_pid = Overall   }
```

```
#state{  overall =  Overall,
         group  =  Group,
         tx_fun  =  TxFun,
         initial_childspecs  =  ChildSpecs,
         child_order  =  ChildOrder  }
```

start_child/2

delete_child/2

restart_child/2

terminate_child/2

which_children/1

count_children/1

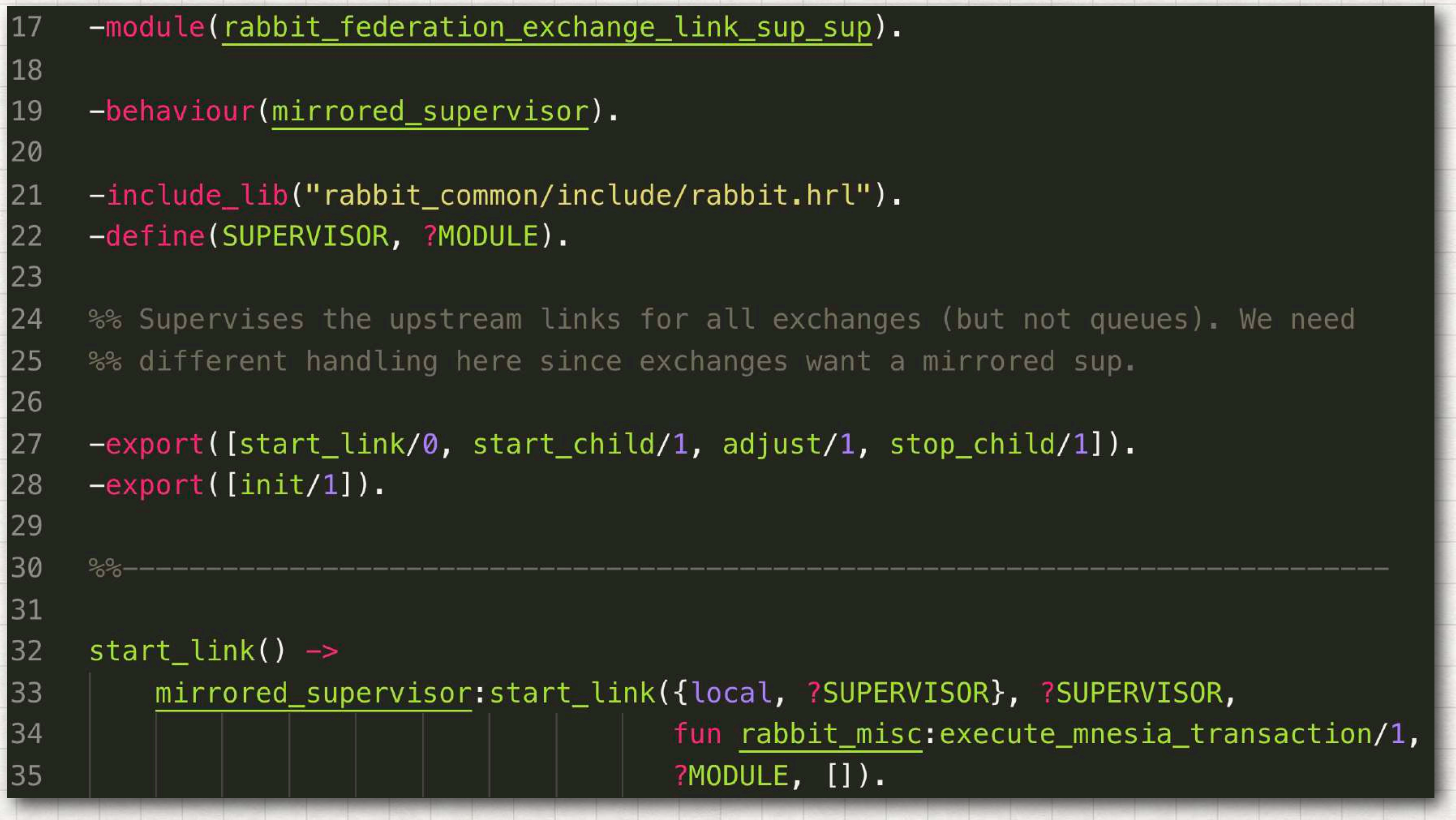
SUPERVISOR API CALLS ←

{'DOWN', Ref, process, Pid, Info} ←

- Get **NewOverallPID** from PG2
- If current **Overall** of mirror is head of list
  - Update all **OldOverall** ChildSpecs in MNESIA, with **NewOverallPID**
  - Restore child-ordering
  - Start child specs

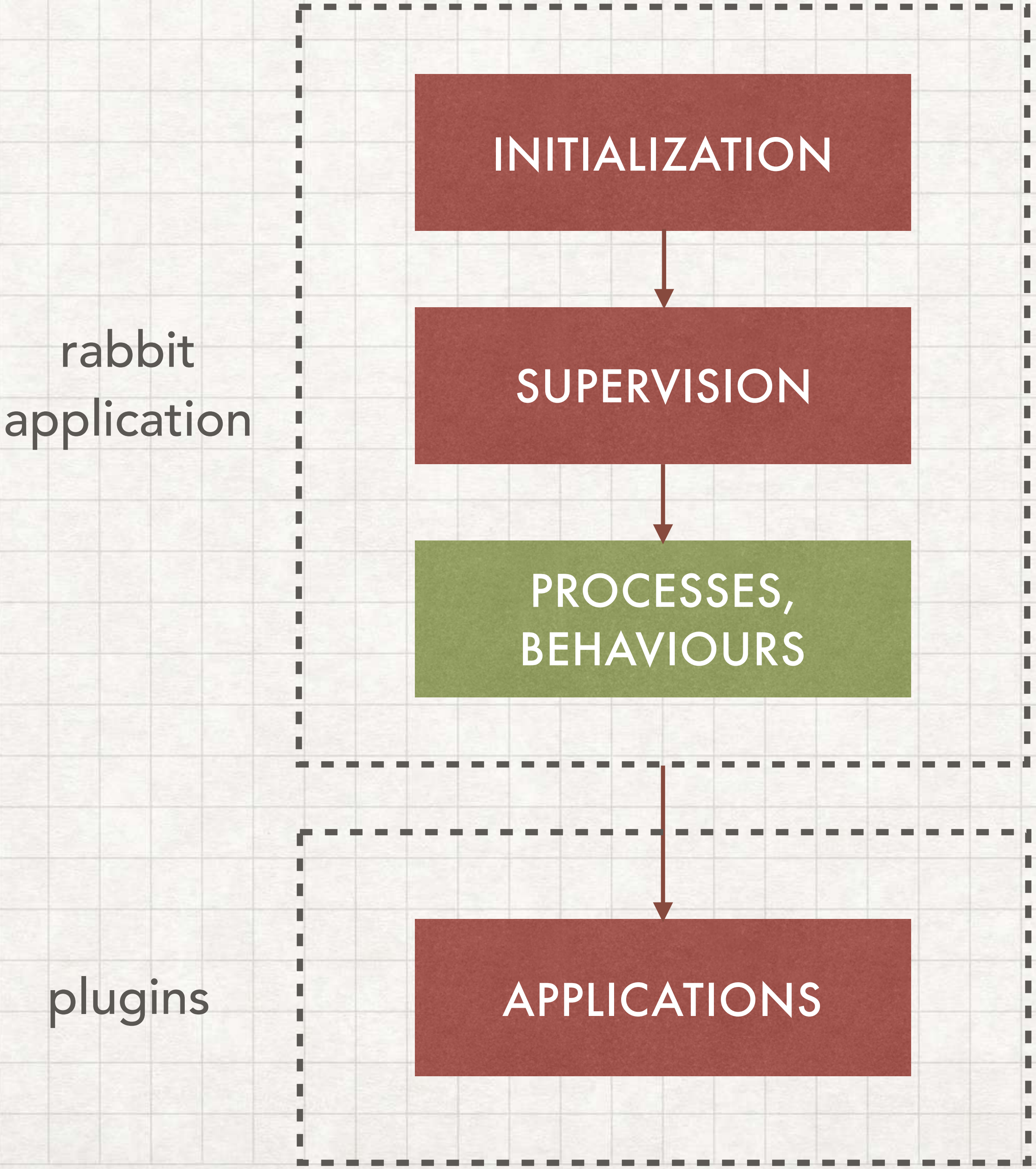
SUPERVISORS: MIRRORED SUPERVISOR

- Federation link top-level supervisors



```erlang
-module(rabbit_federation_exchange_link_sup_sup).

-behaviour(mirrored_supervisor).

-include_lib("rabbit_common/include/rabbit.hrl").
-define(SUPERVISOR, ?MODULE).

%% Supervises the upstream links for all exchanges (but not queues). We need
%% different handling here since exchanges want a mirrored sup.

-export([start_link/0, start_child/1, adjust/1, stop_child/1]).
-export([init/1]).

%%----------------------------------------------------------------------------

start_link() ->
    mirrored_supervisor:start_link({local, ?SUPERVISOR}, ?SUPERVISOR,
                                   fun rabbit_misc:execute_mnesia_transaction/1,
                                   ?MODULE, []).
```

PROCESSES & BEHAVIOURS



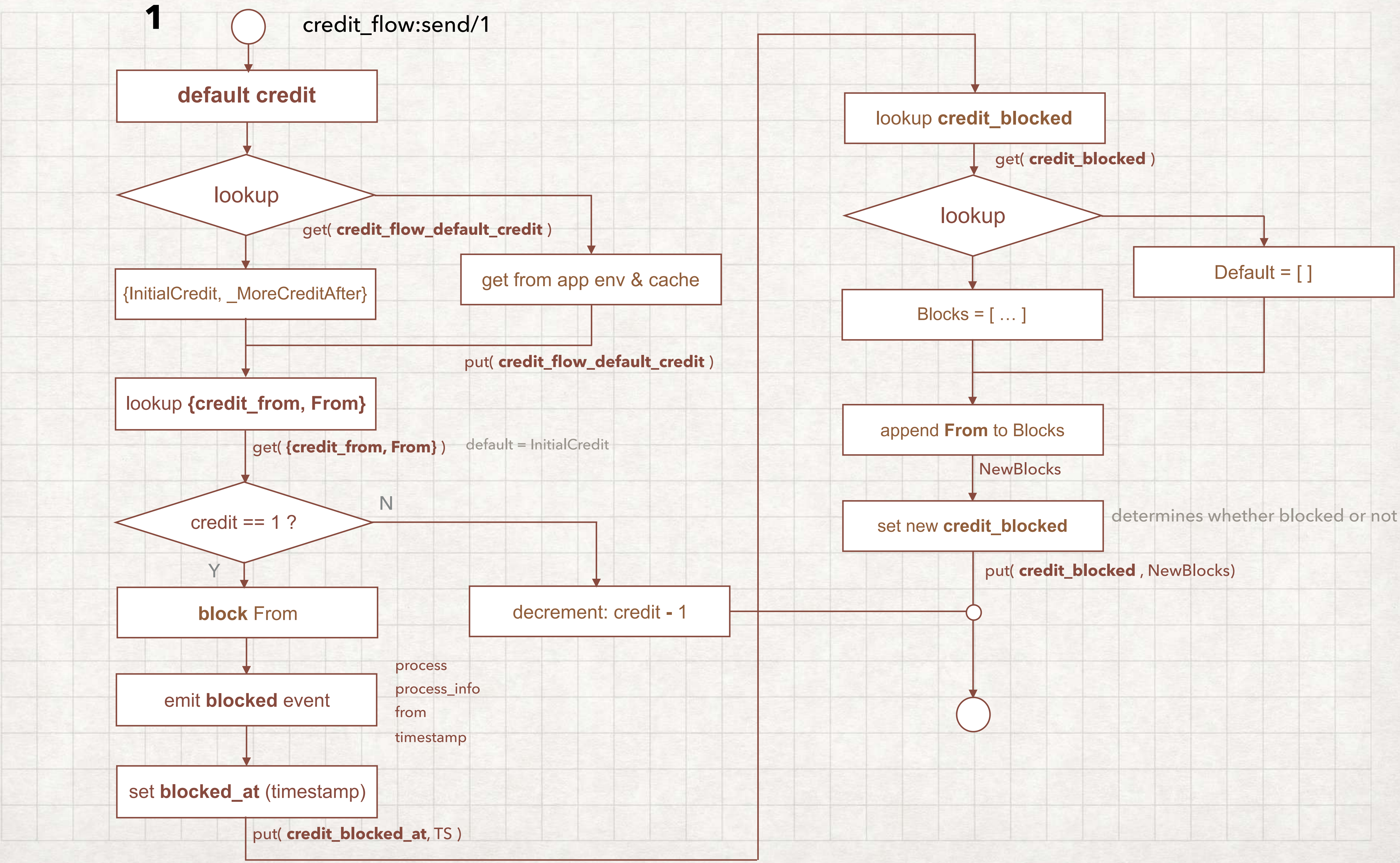
PROCESSES & BEHAVIOURS: GEN-SERVER-2

- Optimised selective receives - internal buffer extending (& draining) the message queue
- Additional callbacks - prioritised **call**, **casts** & **info**
- Pre- and Post- hibernation callbacks
- Backoff capabilities for delayed hibernation and variable timeouts
- Dynamic switching of callbacks (**become**)
- Debugging and formatting capabilities

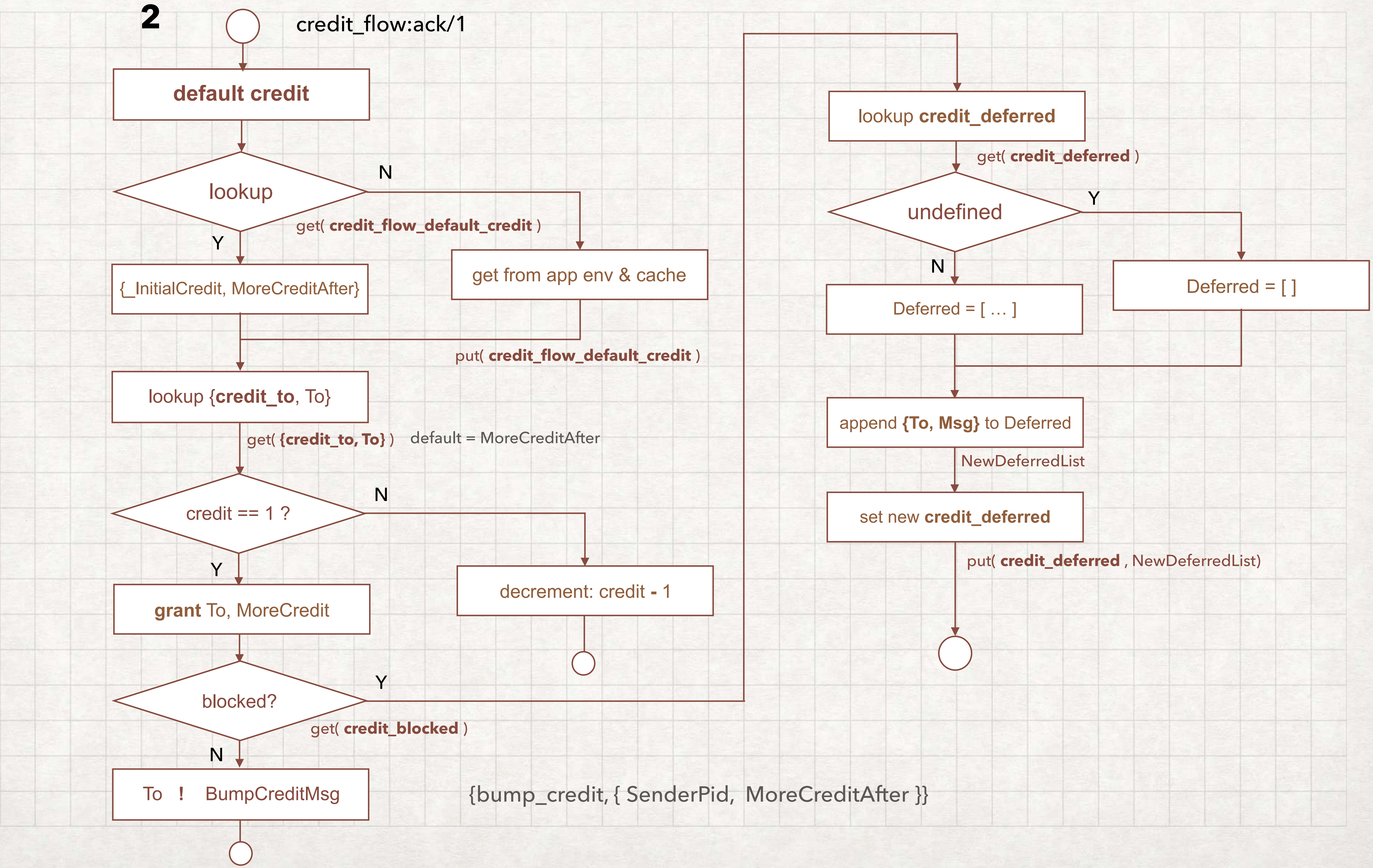
PROCESSES & BEHAVIOURS: CREDIT FLOW

- Flow control on peer Erlang processes
- Lightweight - based on process dictionary
- Single control Erlang message (on demand)
- Simple, effective principle of operation
  - Sender **granted credit** by receiver, to send more
  - Sender **blocks** if it runs out of credit
  - Transceivers cannot grant more credit if blocked
- {InitialCredit, MoreCreditAfter}

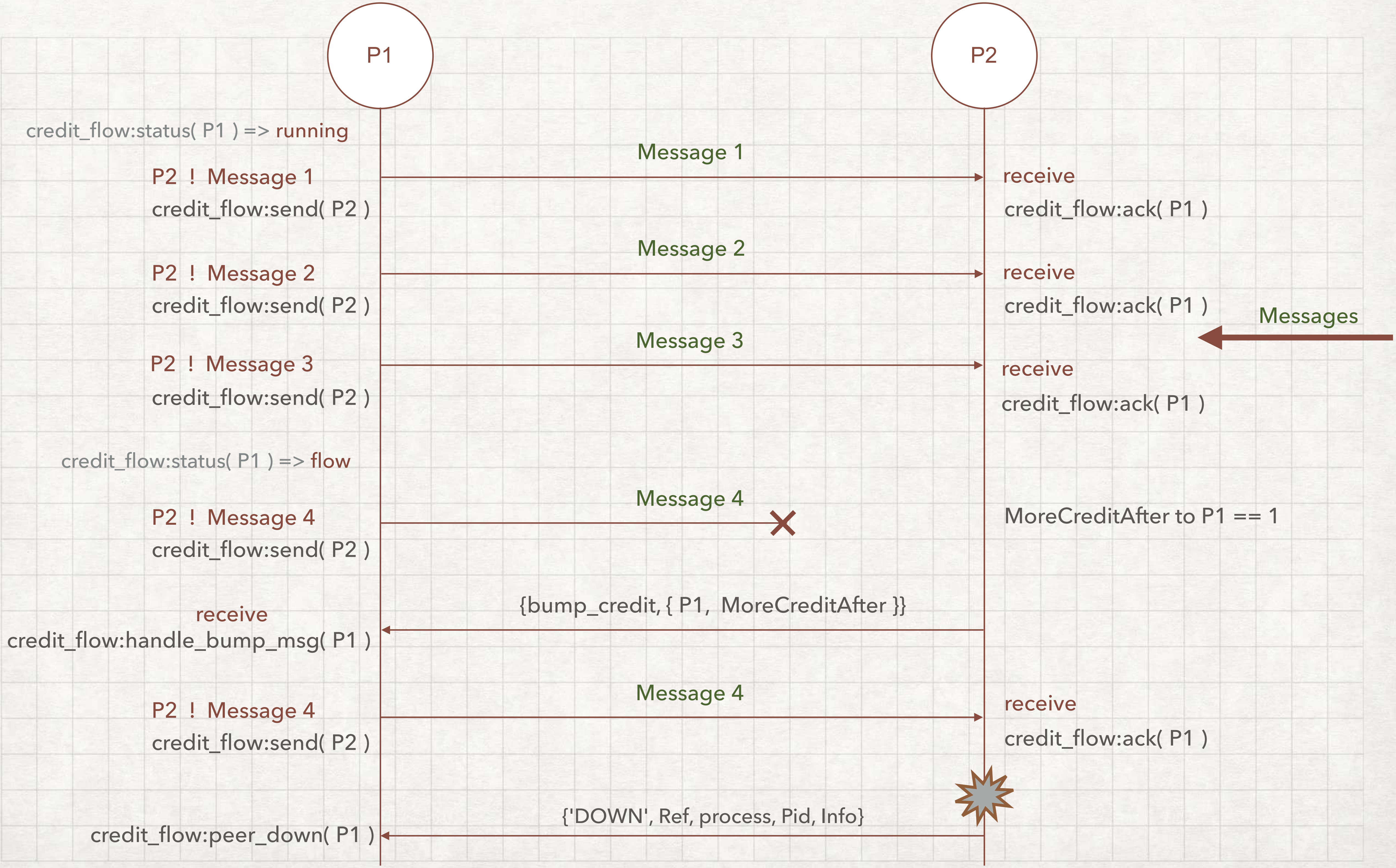
PROCESSES & BEHAVIOURS: CREDIT FLOW



PROCESSES & BEHAVIOURS: CREDIT FLOW



PROCESSES & BEHAVIOURS: CREDIT FLOW



P1

P2

credit_flow:status(P1) => running

Message 1

P2 ! Message 1

receive

credit_flow:send(P2)

credit_flow:ack(P1)

Message 2

P2 ! Message 2

receive

credit_flow:send(P2)

credit_flow:ack(P1)

Messages

Message 3

P2 ! Message 3

receive

credit_flow:send(P2)

credit_flow:ack(P1)

credit_flow:status(P1) => flow

Message 4

P2 ! Message 4

MoreCreditAfter to P1 == 1

credit_flow:send(P2)

{bump_credit, { P1, MoreCreditAfter }}

receive

credit_flow:handle_bump_msg(P1)

Message 4

P2 ! Message 4

receive

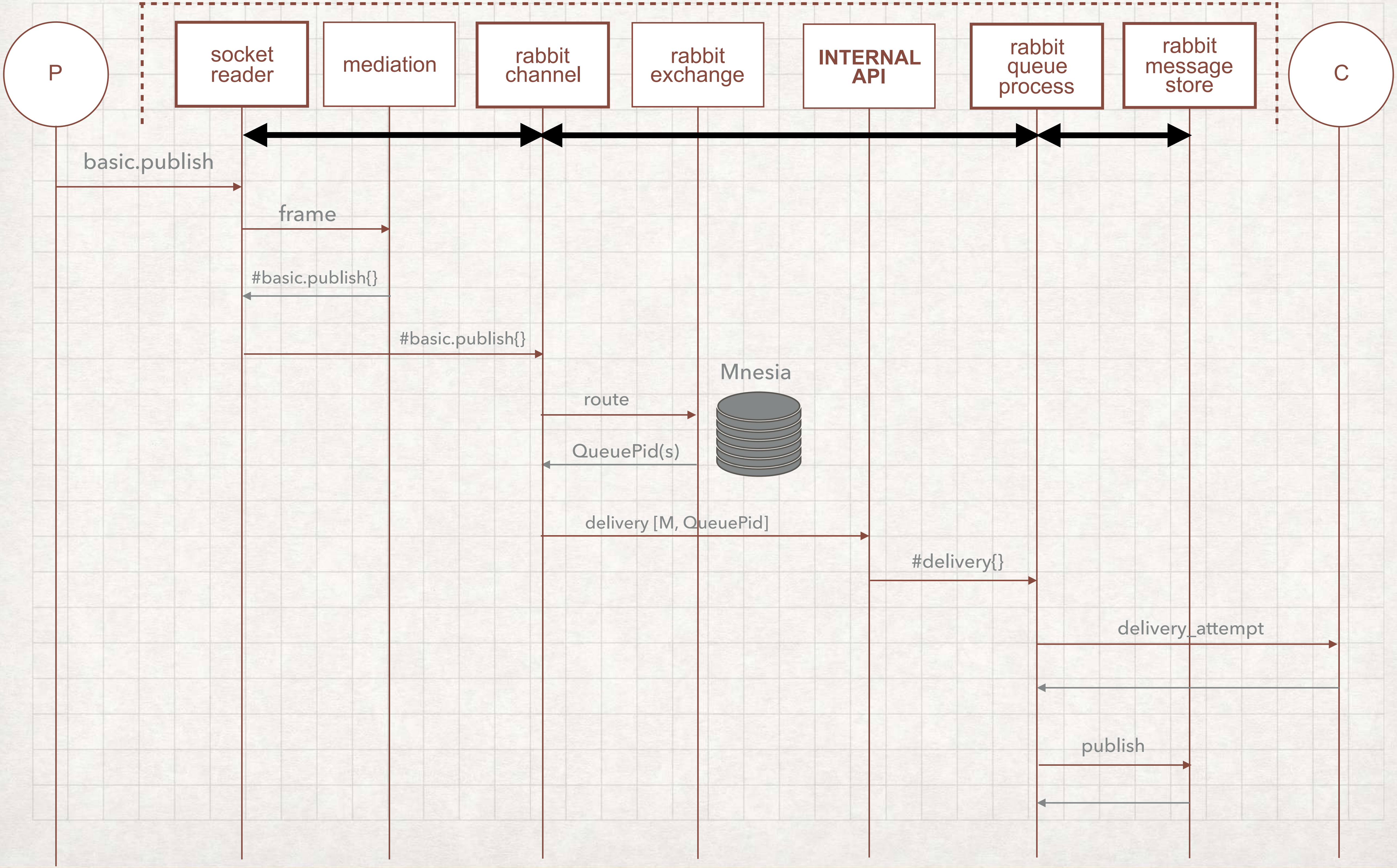
credit_flow:send(P2)

credit_flow:ack(P1)

{'DOWN', Ref, process, Pid, Info}

credit_flow:peer_down(P1)

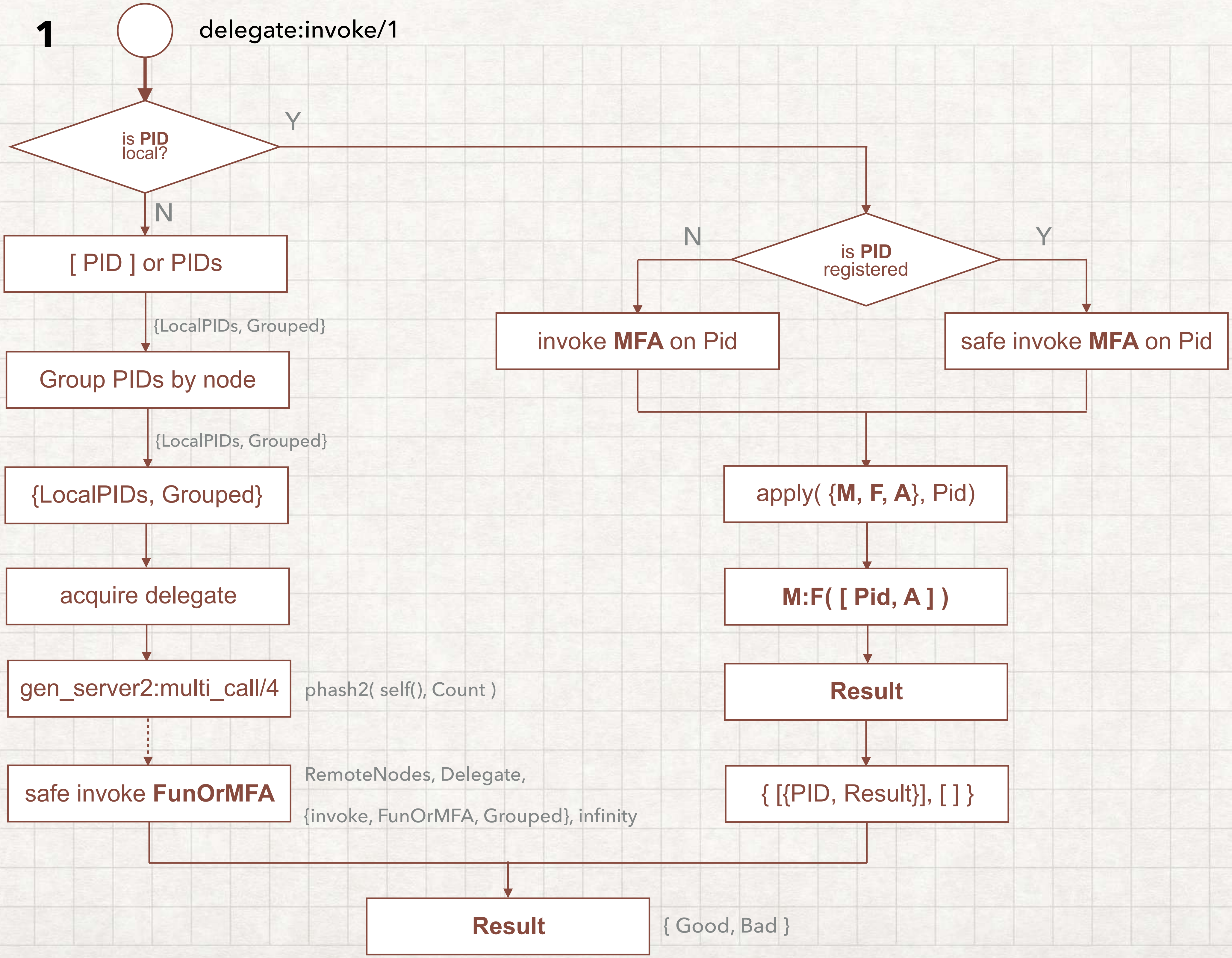
PROCESSES & BEHAVIOURS: CREDIT FLOW



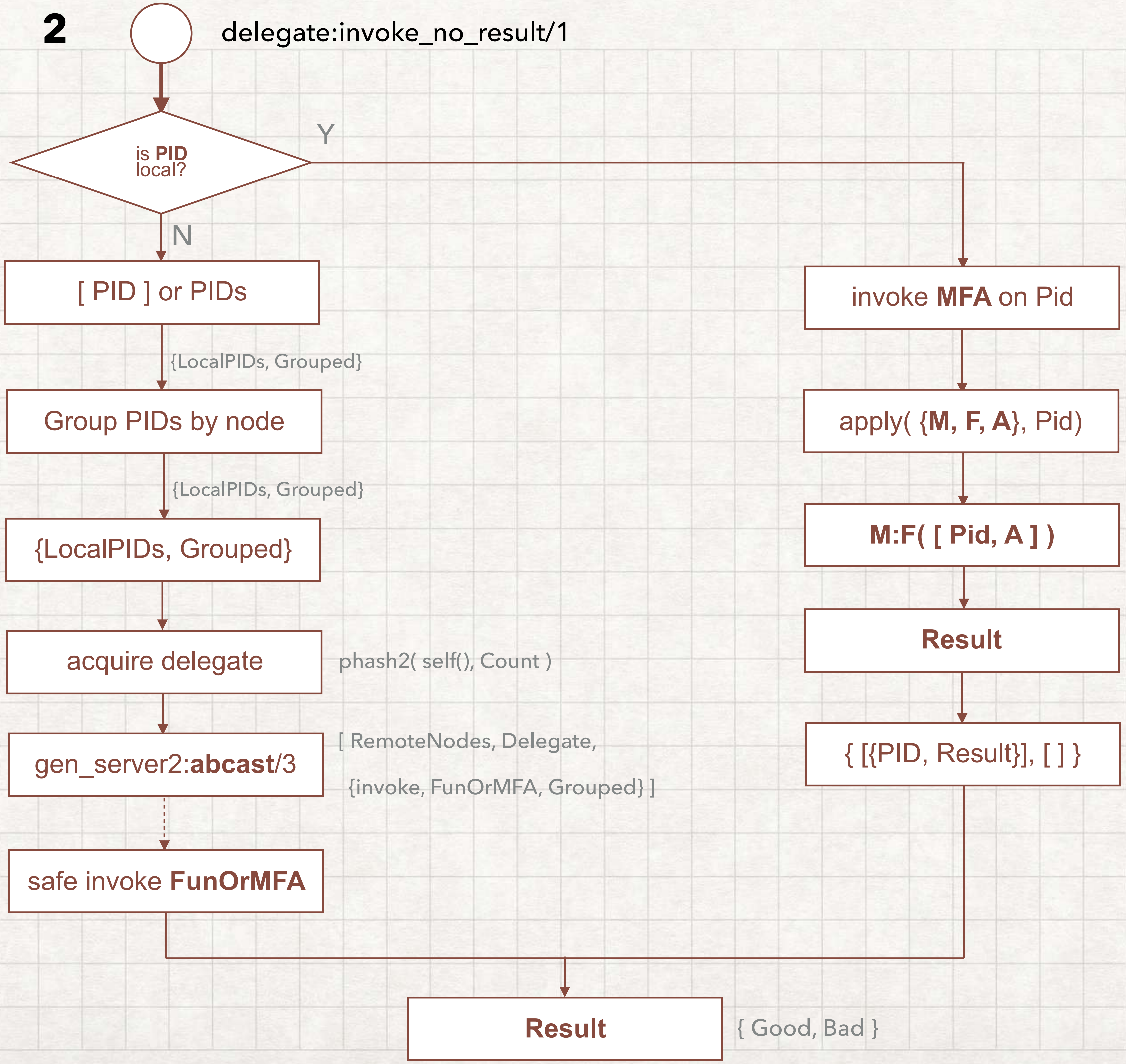
PROCESSES & BEHAVIOURS: DELEGATES

- Optmized internode communication
- Synchronous and Asynchronous operations
- Minimum blocking - configurable pool size
- Optmized process monitoring (on local node only)
- Low bandwidth usage on distribution links

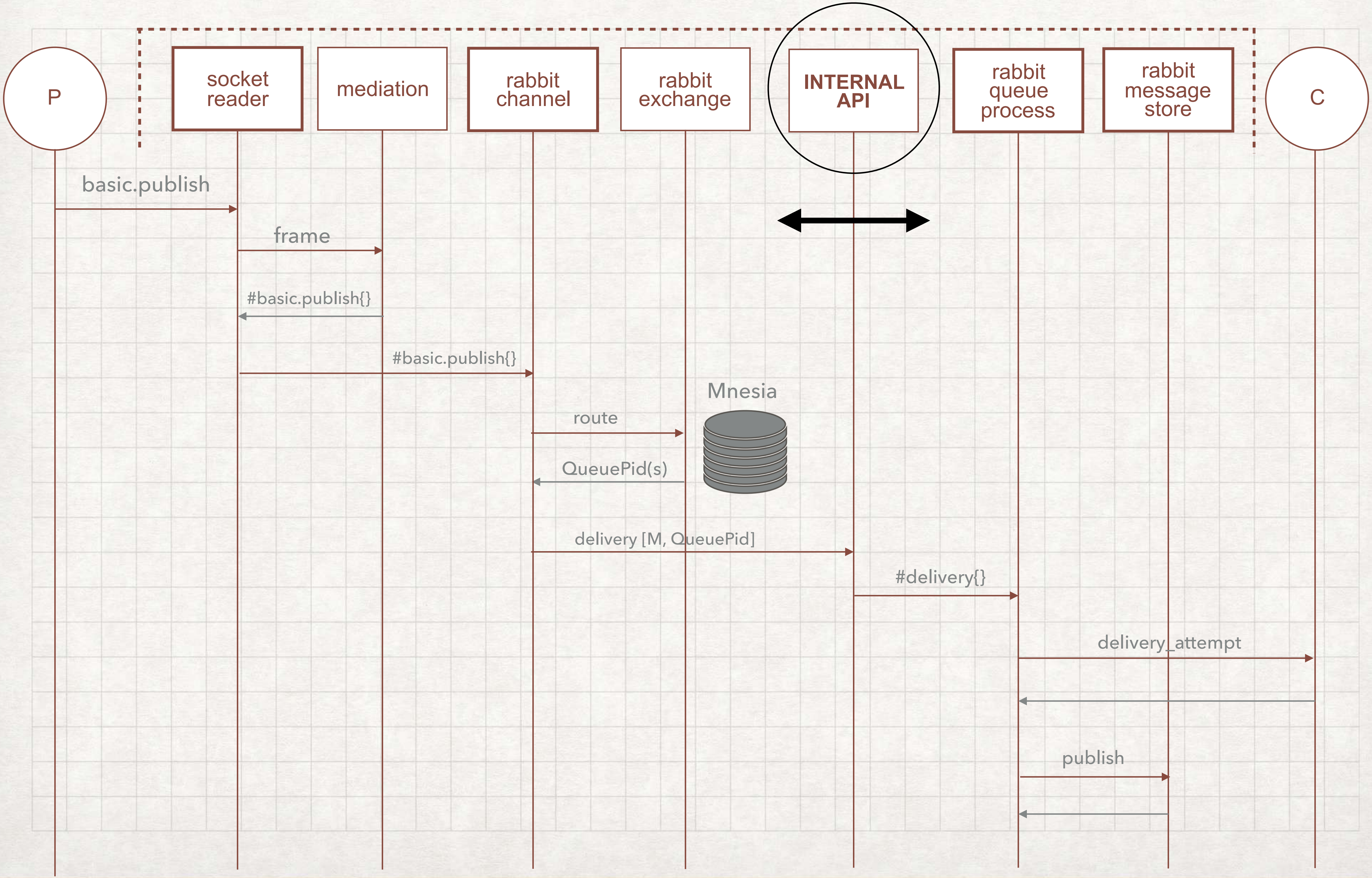
PROCESSES & BEHAVIOURS: DELEGATES



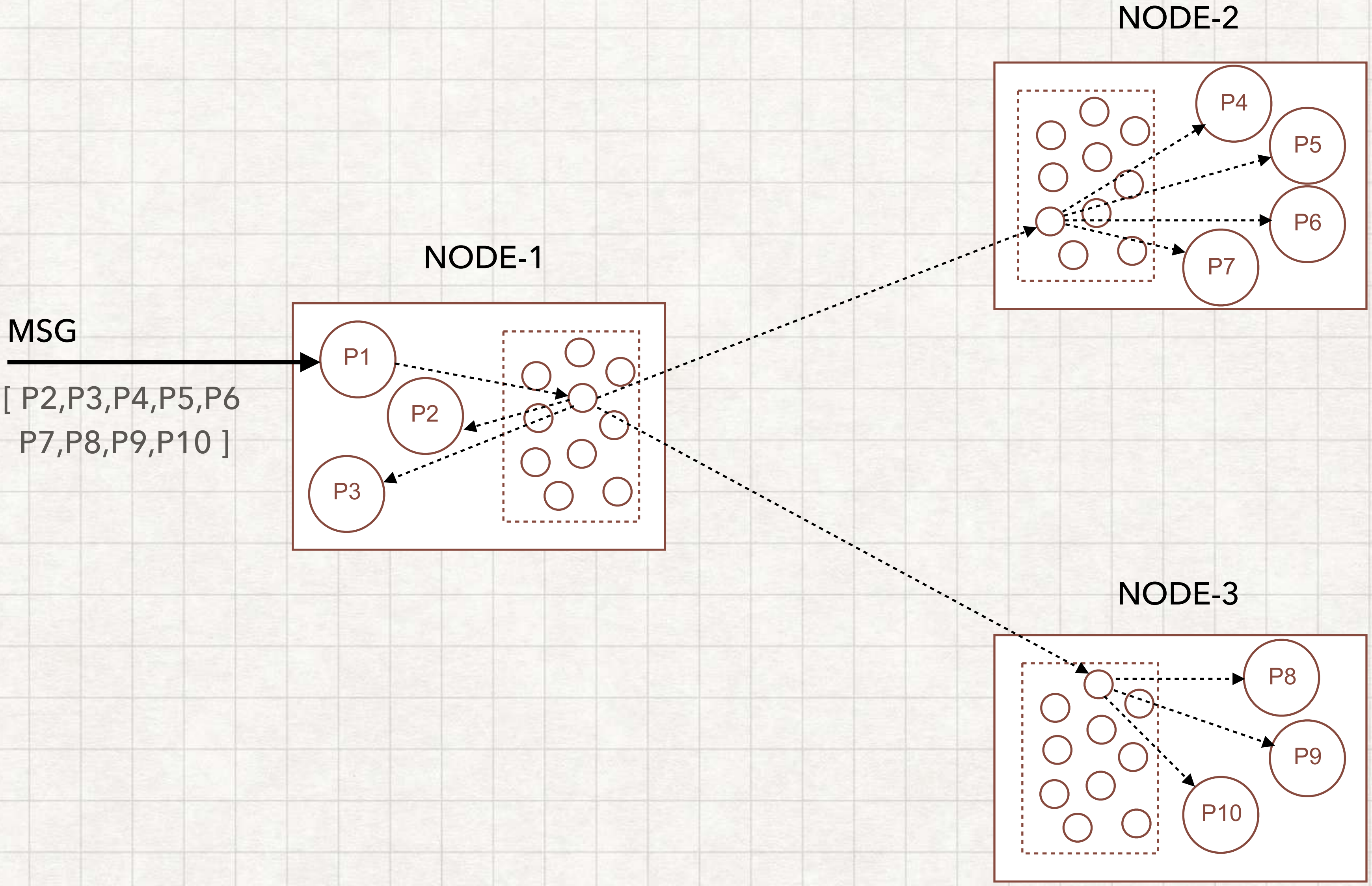
PROCESSES & BEHAVIOURS: DELEGATES



PROCESSES & BEHAVIOURS: DELEGATES



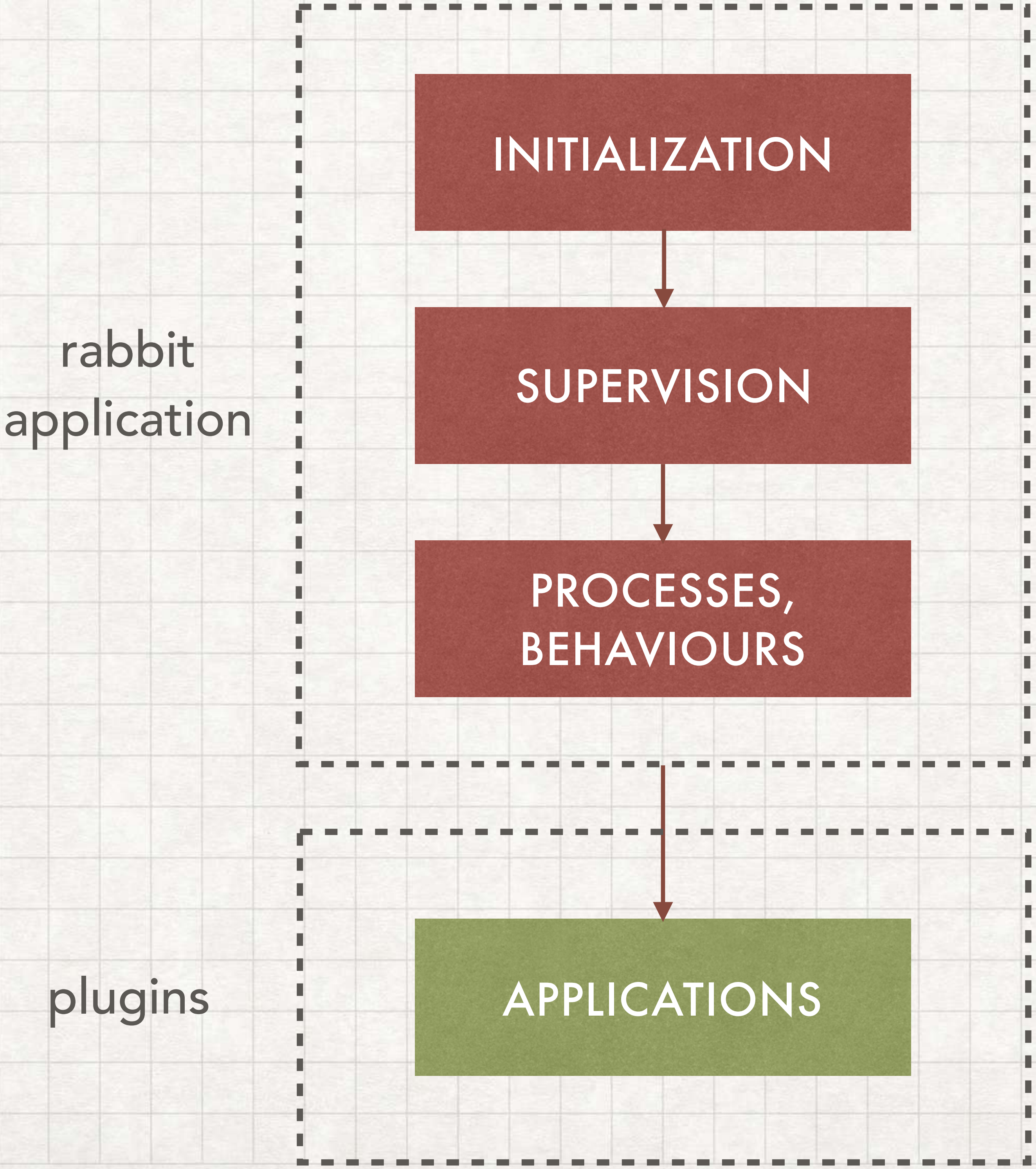
PROCESSES & BEHAVIOURS: DELEGATES



PROCESSES & BEHAVIOURS: MORE ...

- **Decorators**
  - Dynamic state updates of implementing processes
  - Facilitate OAM, CLI tools, e.g. live policy updates
- **GM (Guaranteed Multicast)**
  - Behaviour for attaining consensus on a group of processes
- **PMon**
  - Optimized monitors, at most 1 monitor per process
  - Querying capabilities, e.g. **is_monitored/1**

PLUGINS


rabbit application
INITIALIZATION
SUPERVISION
PROCESSES, BEHAVIOURS
plugins
APPLICATIONS

APPLICATIONS: PLUGINS

- Plugin architecture “pattern”: highly extensible
- Plugins are simply OTP applications (zipped)
- Dynamically started/stopped (& expanded) via CLI
  - **rabbitmq-plugins enable** <PLUGIN/APP-NAME>
  - **rabbitmq-plugins disable** <PLUGIN/APP-NAME>
- Active plugins maintained in an **enabled_plugins** file
- Queried and updated on runtime
- Enabler for multi-protocol handling (MQTT, STOMP, ..)
- **Awesome** for abstracting Erlang/Elixir expertise!

CODEBASE: RABBITMQ

https://github.com/rabbitmq/rabbitmq-server

https://github.com/rabbitmq/rabbitmq-common

END: THANK YOU

QUESTIONS

Twitter: dube_aya

Github: Ayanda-D