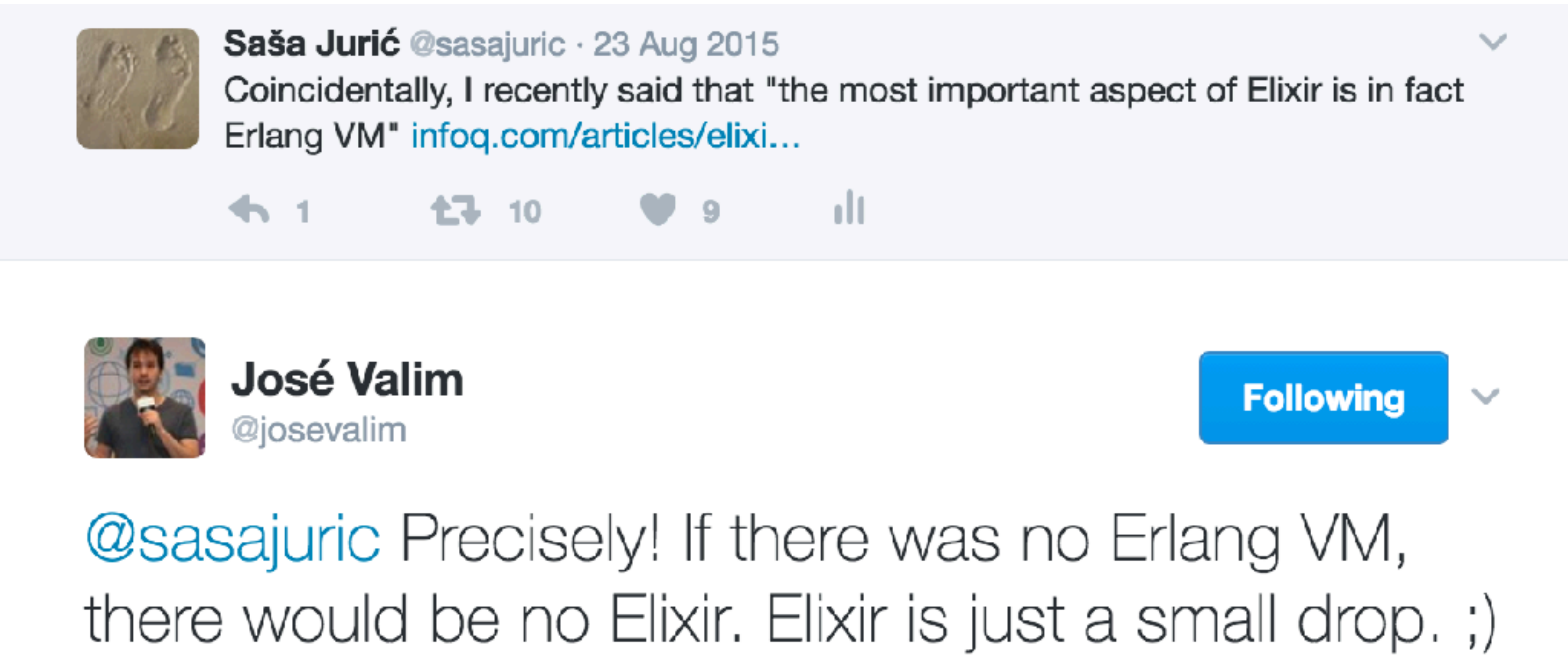
Elixir

@sasajuric

aircloak.com



Saša Jurić @sasajuric · 23 Aug 2015

Coincidentally, I recently said that "the most important aspect of Elixir is in fact Erlang VM" infoq.com/articles/elixi...

1 10 9

José Valim
@josevalim

Following

@sasajuric Precisely! If there was no Erlang VM, there would be no Elixir. Elixir is just a small drop. ;)

WhatsApp

ejabberd

riak

heroku

RabbitMQ
Messaging that just works

IRCCloud

bet365

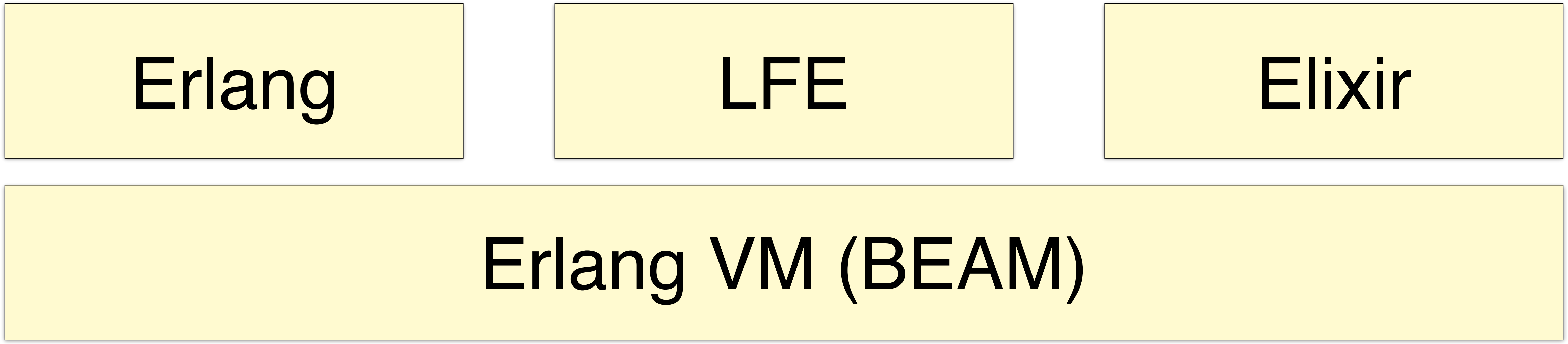
Klarna

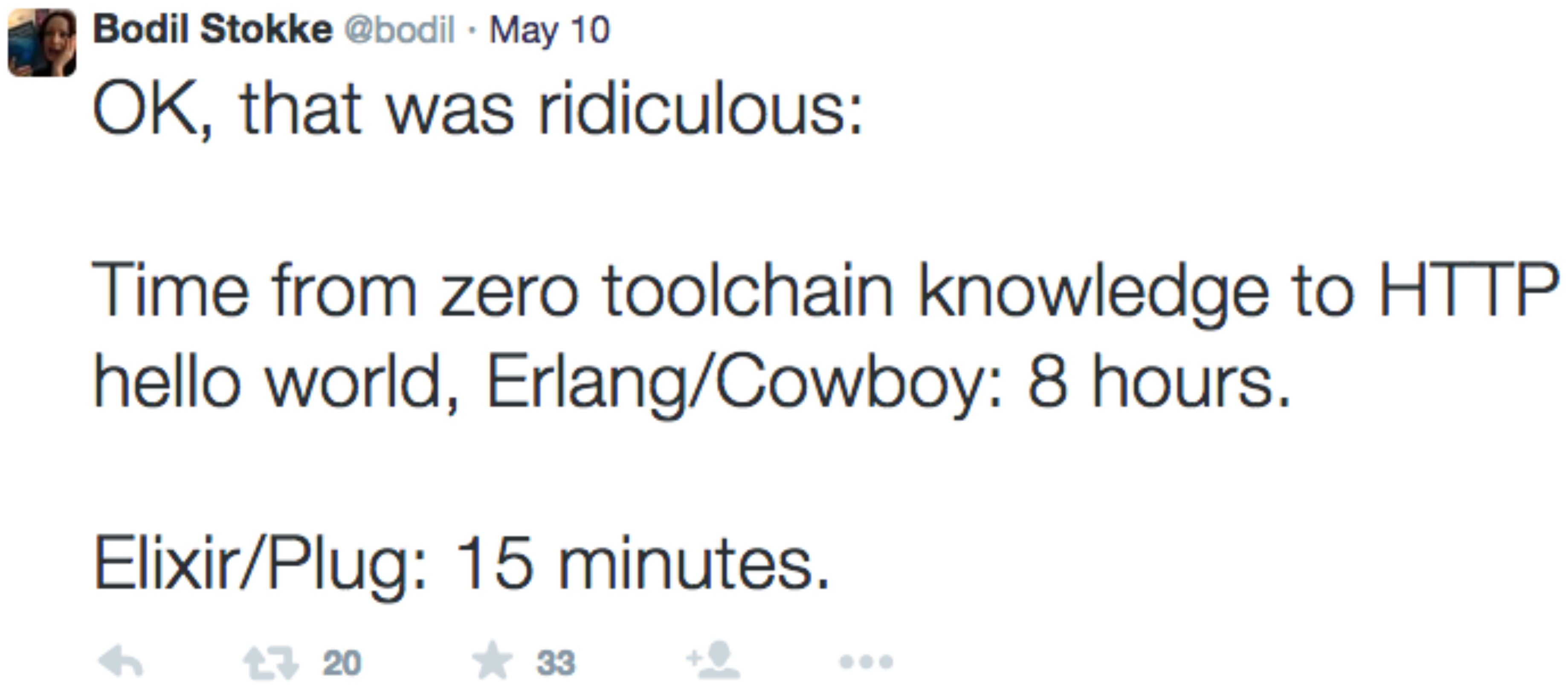
AdRoll

ERICSSON

wooga
world of gaming

otpornost na pogreške
skalabilnost
responzivnost


Erlang
LFE
Elixir
Erlang VM (BEAM)



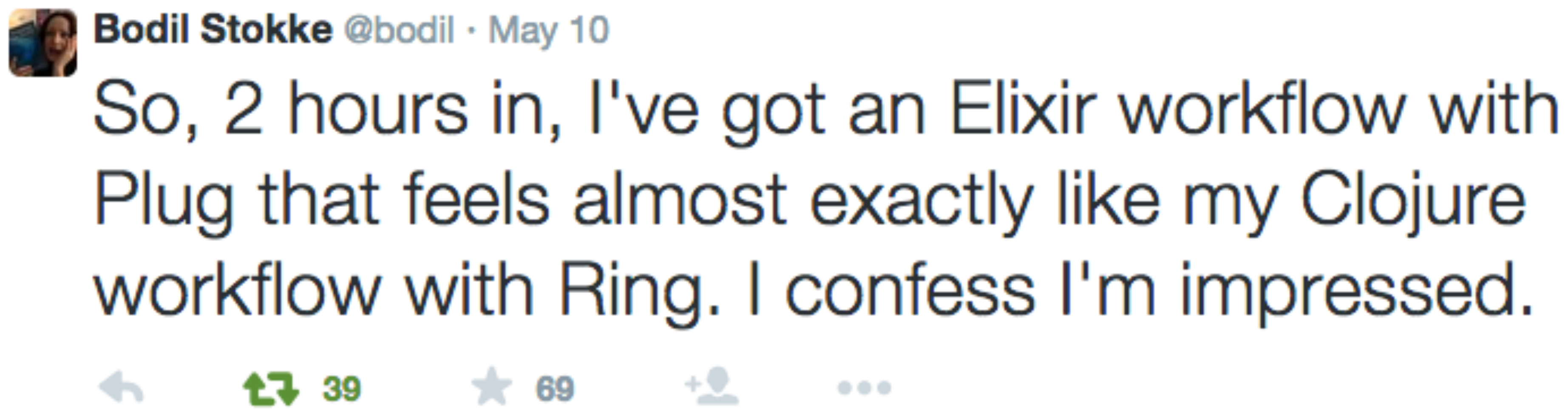
Bodil Stokke @bodil · May 10

OK, that was ridiculous:

Time from zero toolchain knowledge to HTTP hello world, Erlang/Cowboy: 8 hours.

Elixir/Plug: 15 minutes.

20 33

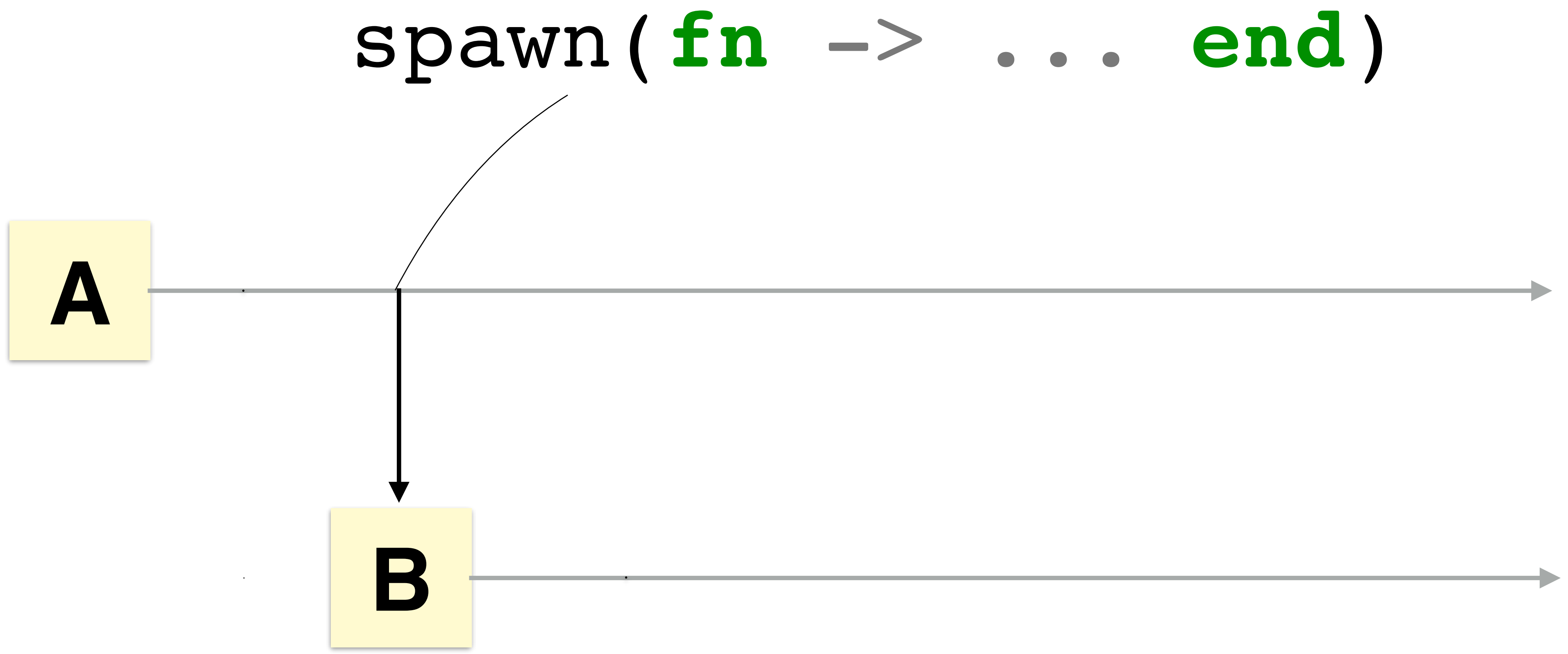


Bodil Stokke @bodil · May 10

So, 2 hours in, I've got an Elixir workflow with Plug that feels almost exactly like my Clojure workflow with Ring. I confess I'm impressed.

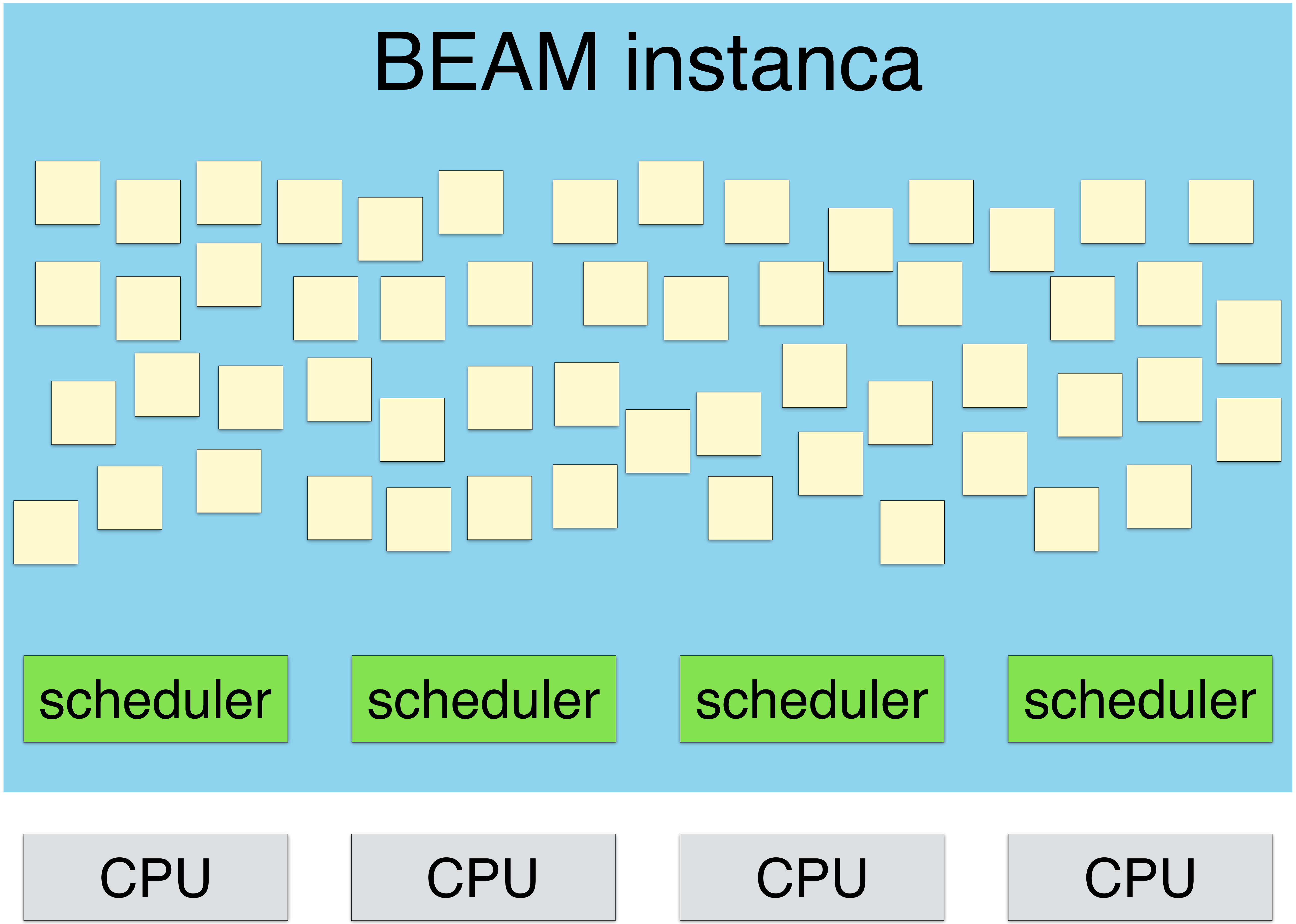
39 69

```
v1 = f1(p1, p2, ...)
v2 = f2(...)
...
```


```
spawn(fn -> ... end)
```

OS proces

BEAM proces


BEAM instanca
scheduler
scheduler
scheduler
scheduler
CPU
CPU
CPU
CPU

```
pid = spawn(fn -> ... end)
pid = spawn(fn -> ... end)
send(pid, :ping)
```

```
spawn(fn ->
  receive do
    :ping ->
      # ...
  end
end)
```

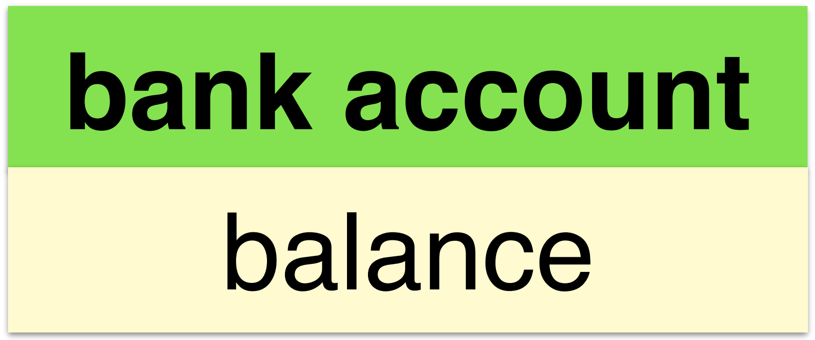
```
send(pid, {:ping, self()})
```

```
spawn(fn ->
  receive do
    {:ping, caller_pid} ->
      send(caller_pid, :pong)
  end
end)
```

```
send(pid, {:ping, self()})

receive do
  :pong ->
    # ...
  after :timer.seconds(5) ->
    # ...
end
```

```elixir
pid = spawn(fn ->
  server_loop(state)
end)
```

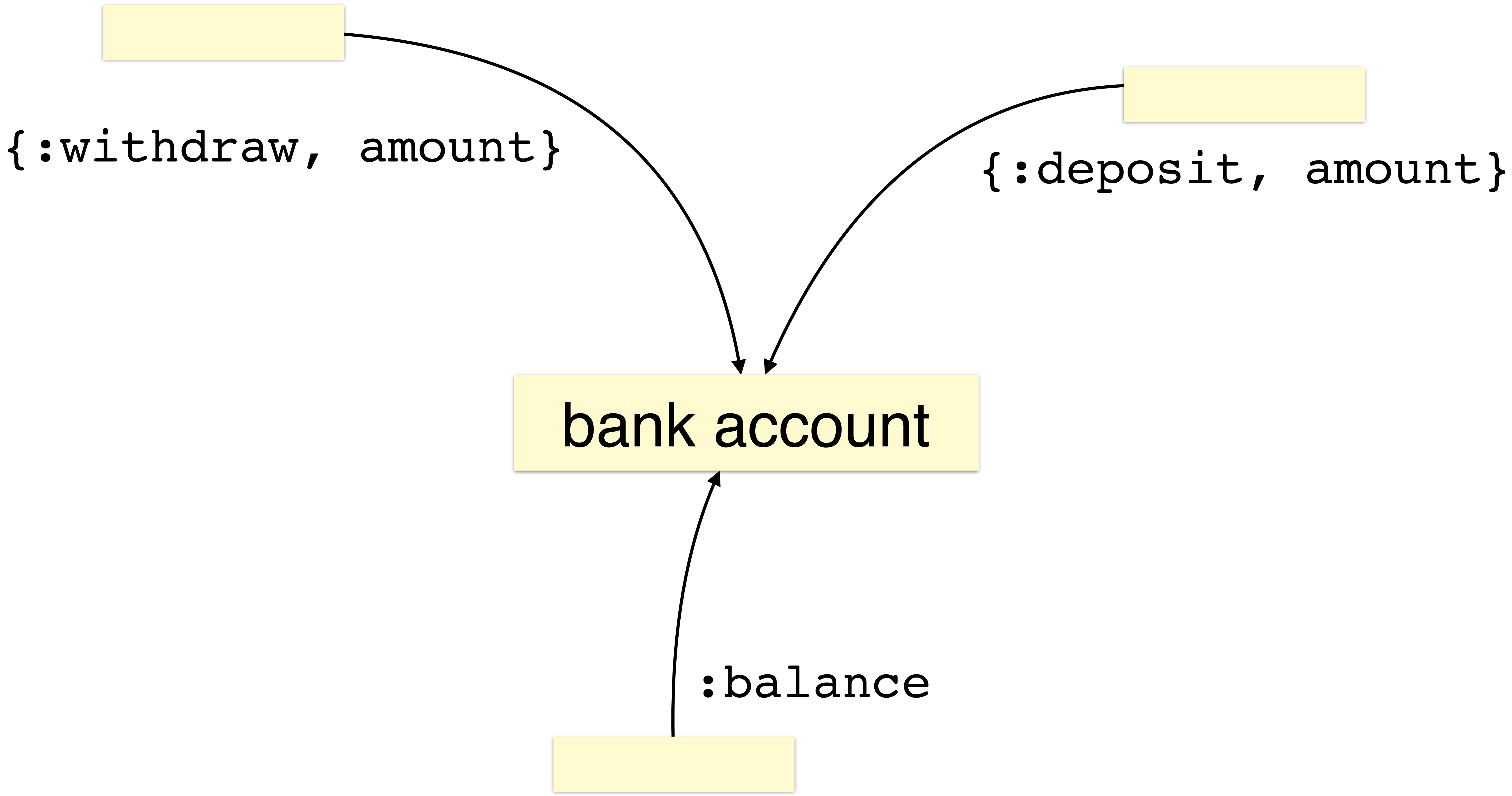
```elixir
defp server_loop(state) do
  receive do
    message ->
      new_state = handle(state, message)
      server_loop(new_state)
  end
end
```

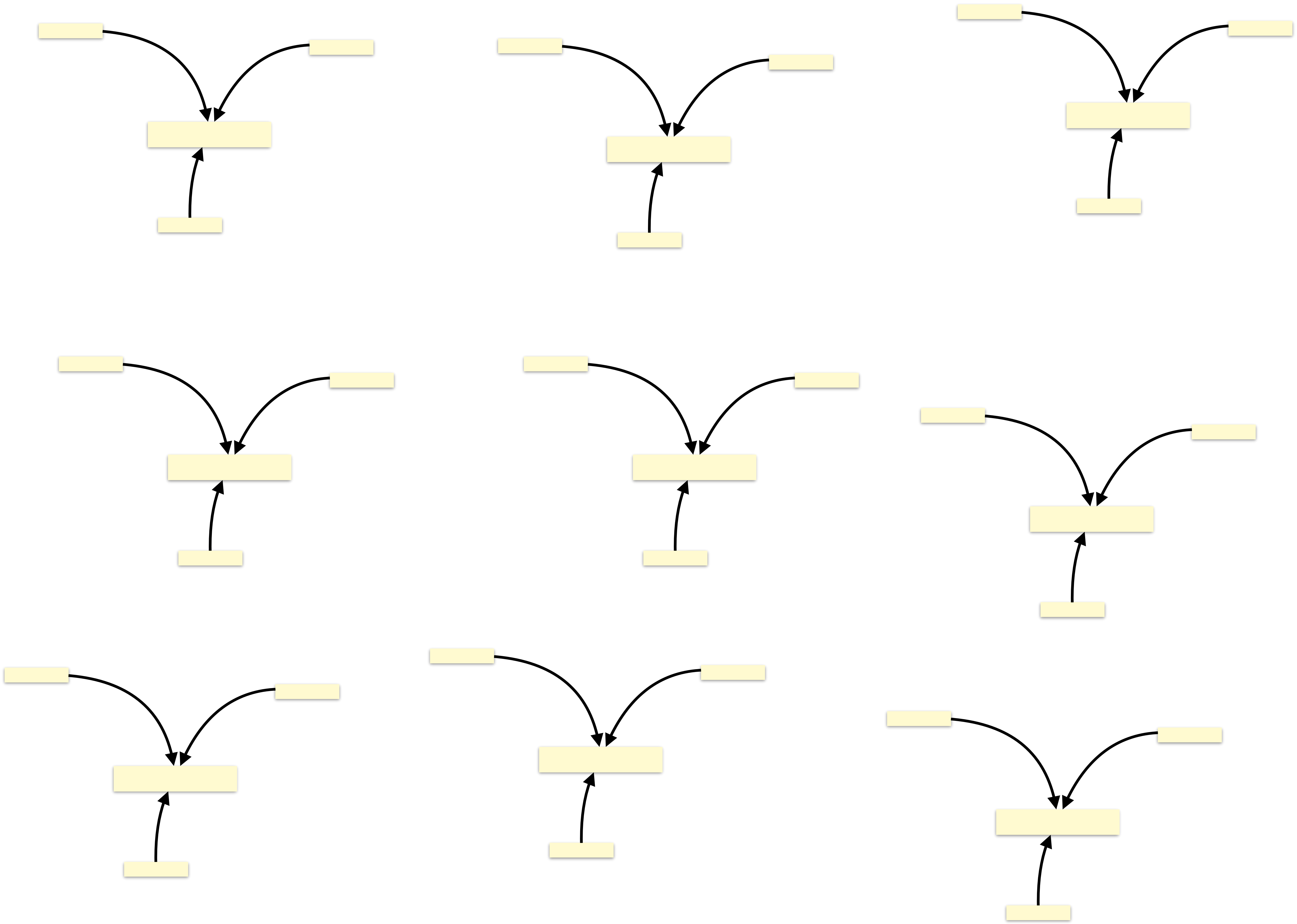

bank account
balance

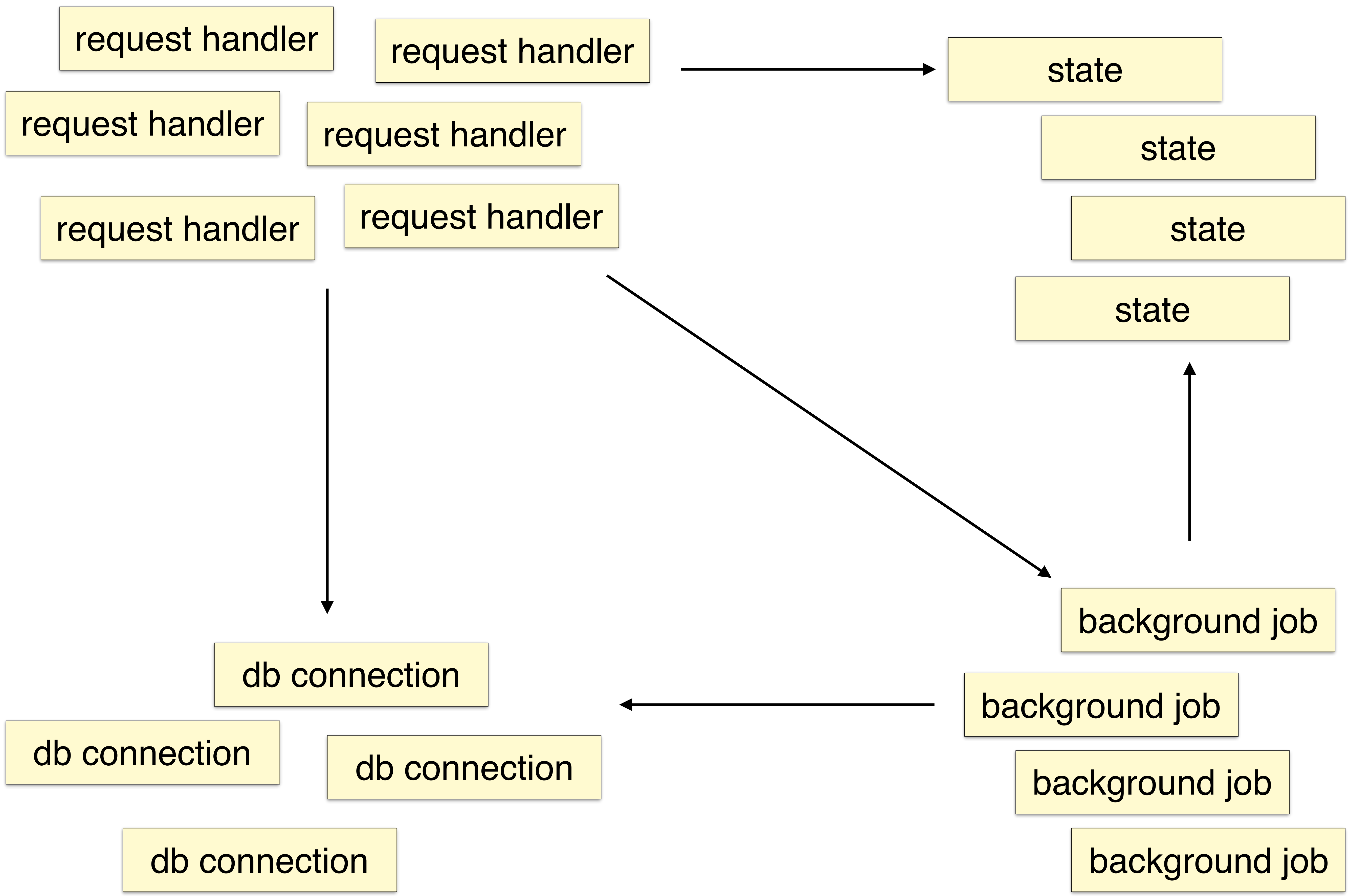
```elixir
send(pid, {:deposit, 100})

send(pid, {:withdraw, 100})

send(pid, {:balance, self()})
receive do
  balance ->
    # ...
end
```

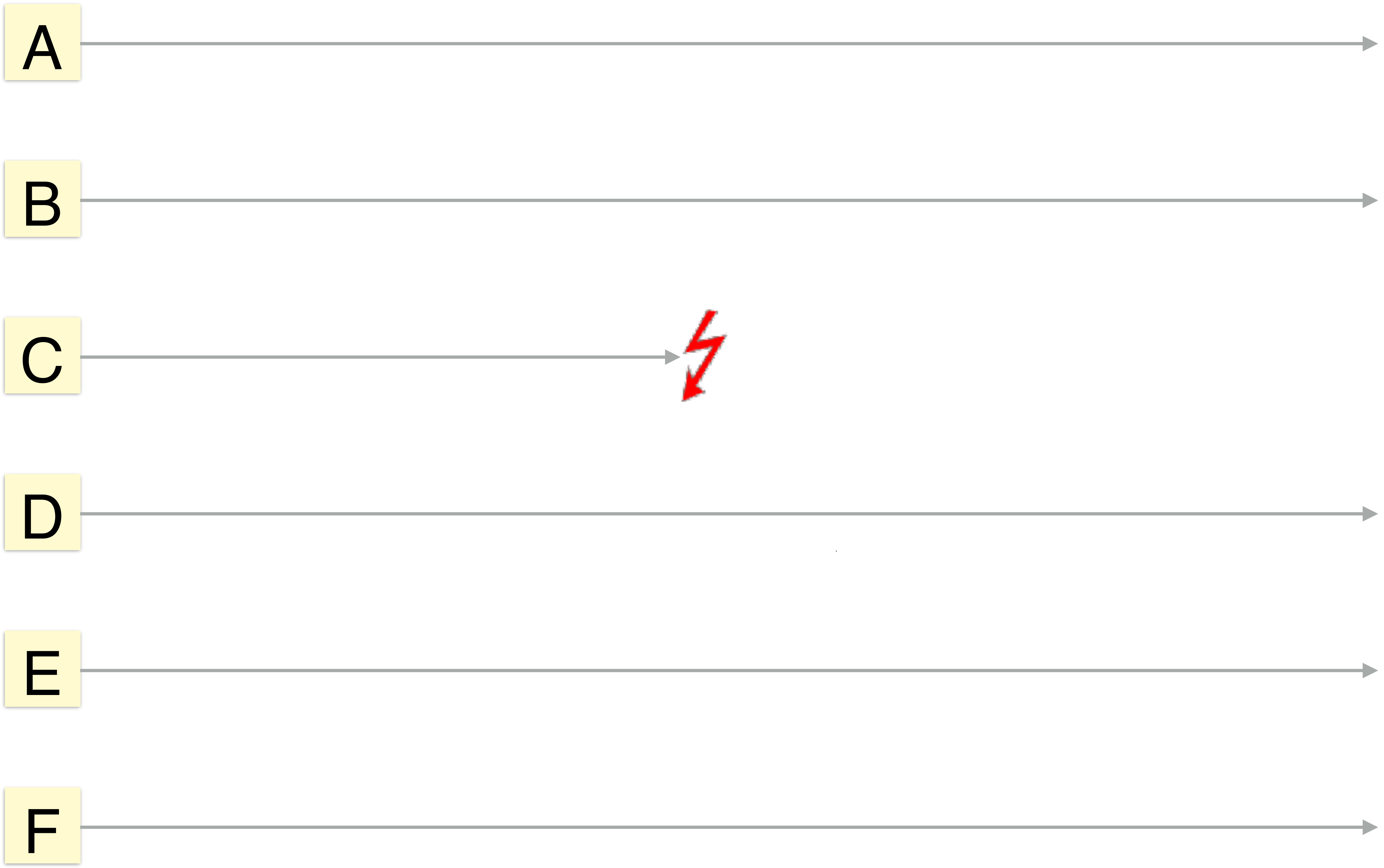
```
spawn(fn ->
  bank_account_loop(0)
end)
```

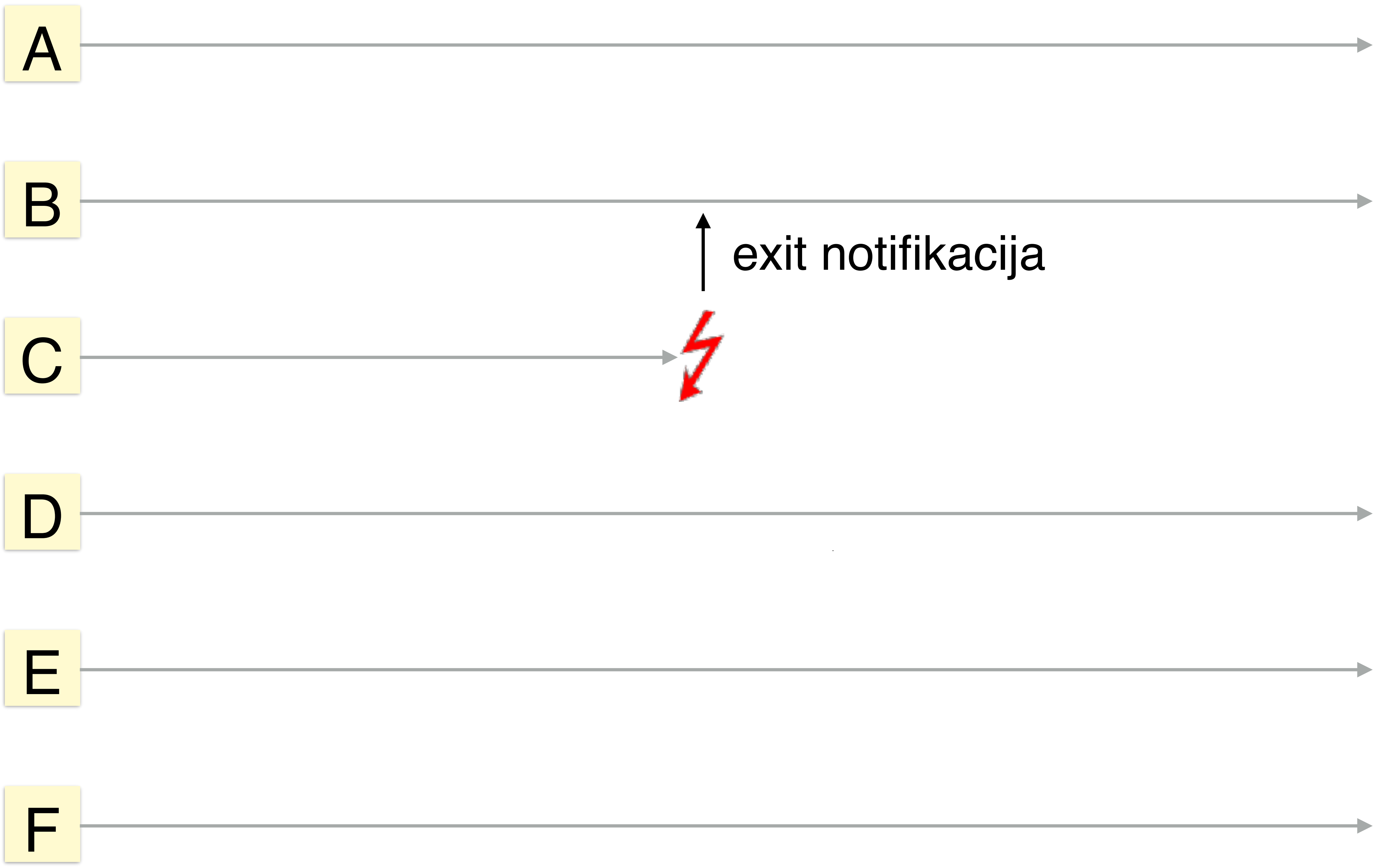
```elixir
def bank_account_loop(balance) do
  receive do
    {:deposit, amount} ->
      bank_account_loop(balance + amount)

    {:withdraw, amount} ->
      bank_account_loop(balance - amount)

    {:balance, caller_pid} ->
      send(caller_pid, balance)
      bank_account_loop(balance)
  end
end
```

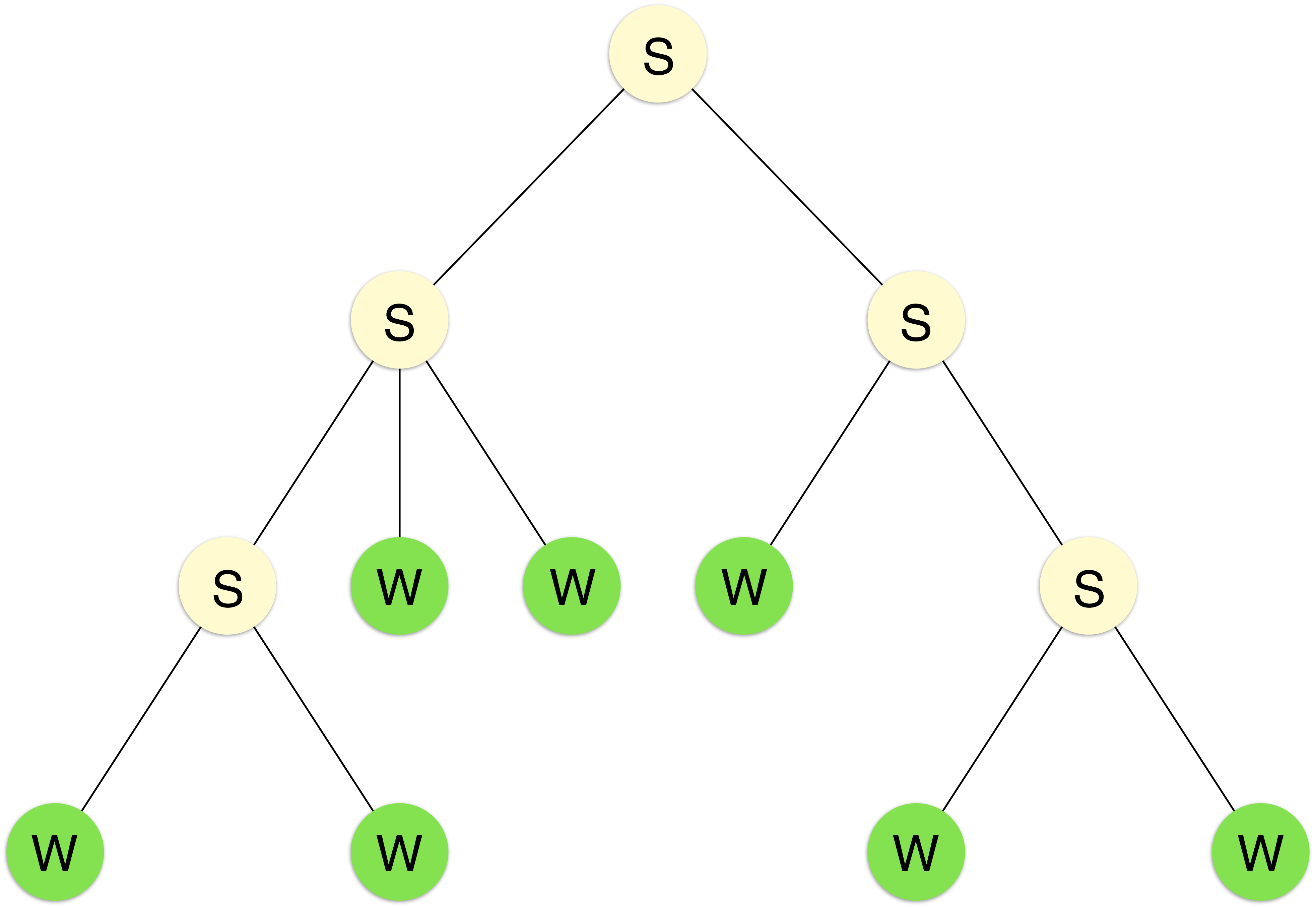

{:withdraw, amount}
{:deposit, amount}
bank account
:balance

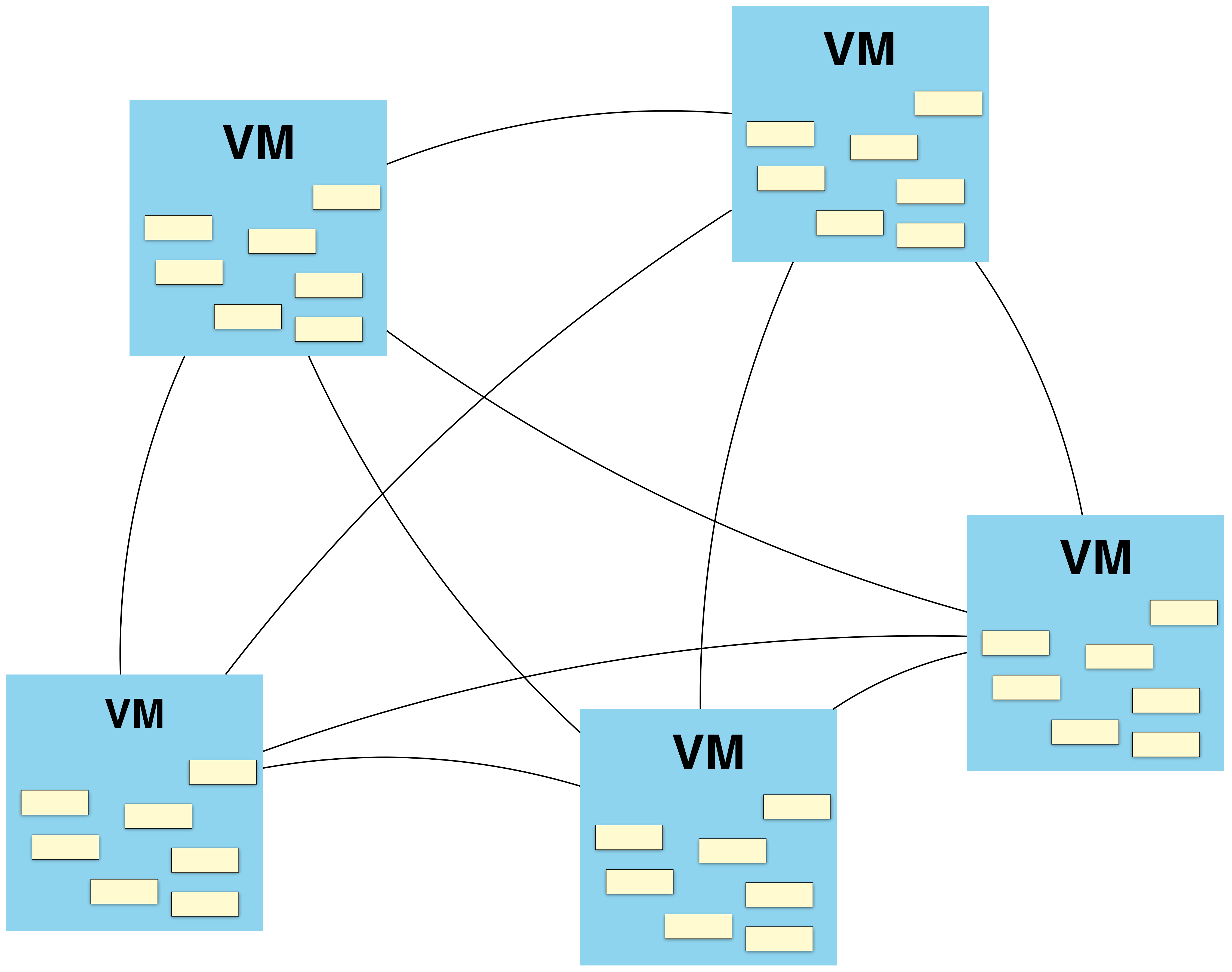


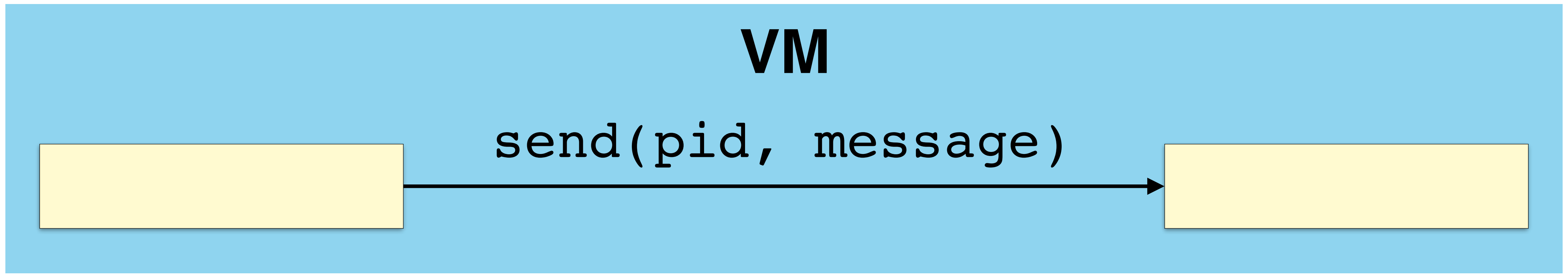

request handler
request handler
request handler
request handler
request handler
request handler
state
state
state
state
db connection
db connection
db connection
db connection
background job
background job
background job
background job

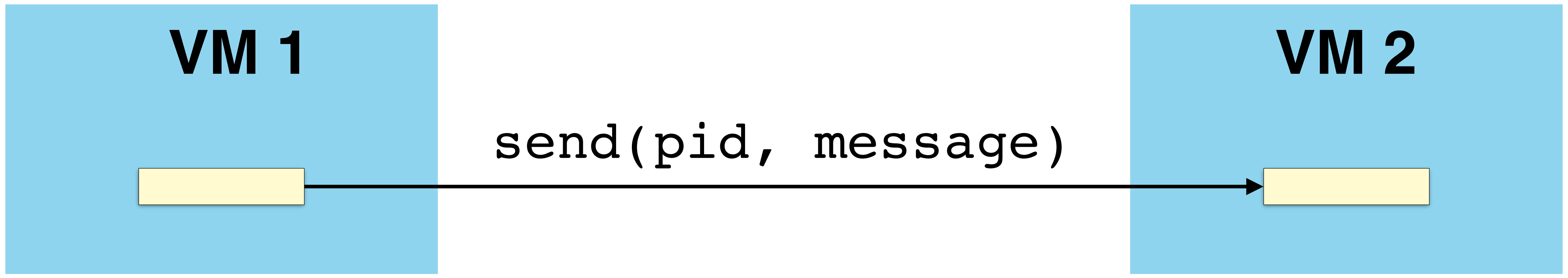

A
B
C
D
E
F


A
B
exit notifikacija
C
D
E
F


S
S
S
S
W
W
W
S
W
W
W
W


VM
VM
VM
VM
VM


VM
send(pid, message)


VM 1
VM 2
send(pid, message)

elixir-lang.org
phoenixframework.org

adventofcode.com
exercism.io

elixirforum.com
meetup.com/lambdazagreb/