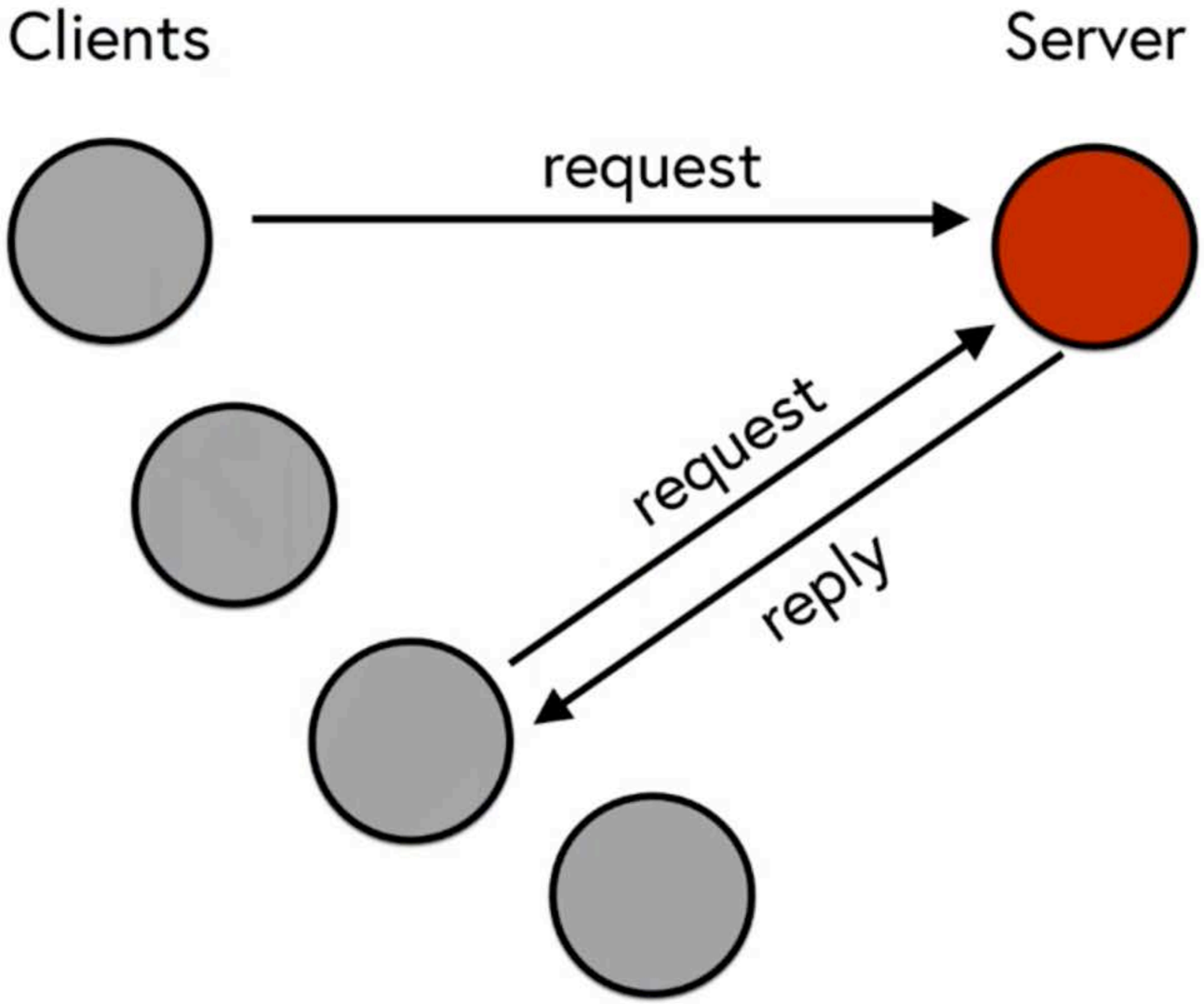

University of
Kent

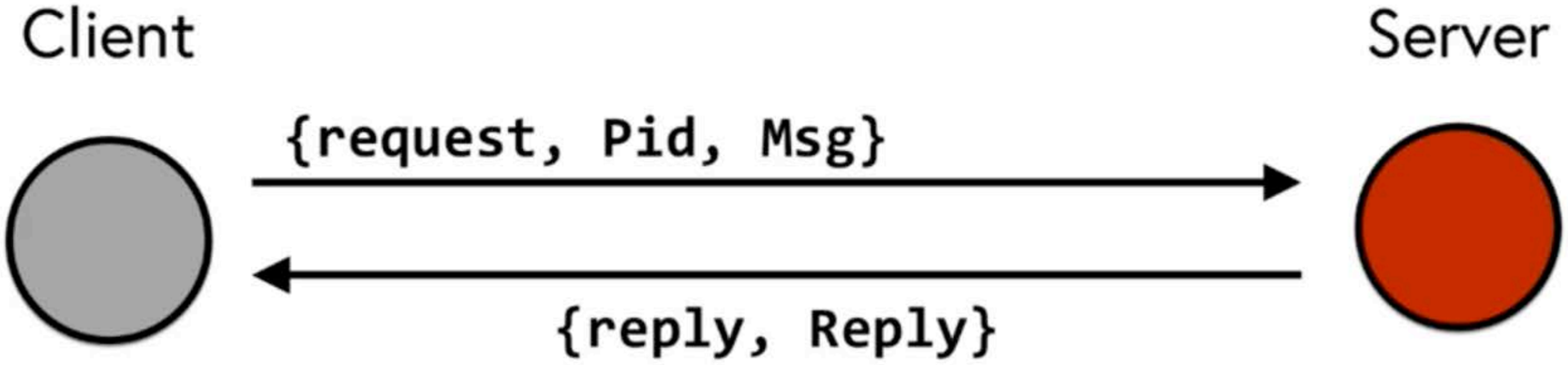
Francesco Cesarini

FOUNDER AND TECHNICAL DIRECTOR, ERLANG SOLUTIONS

University of Kent

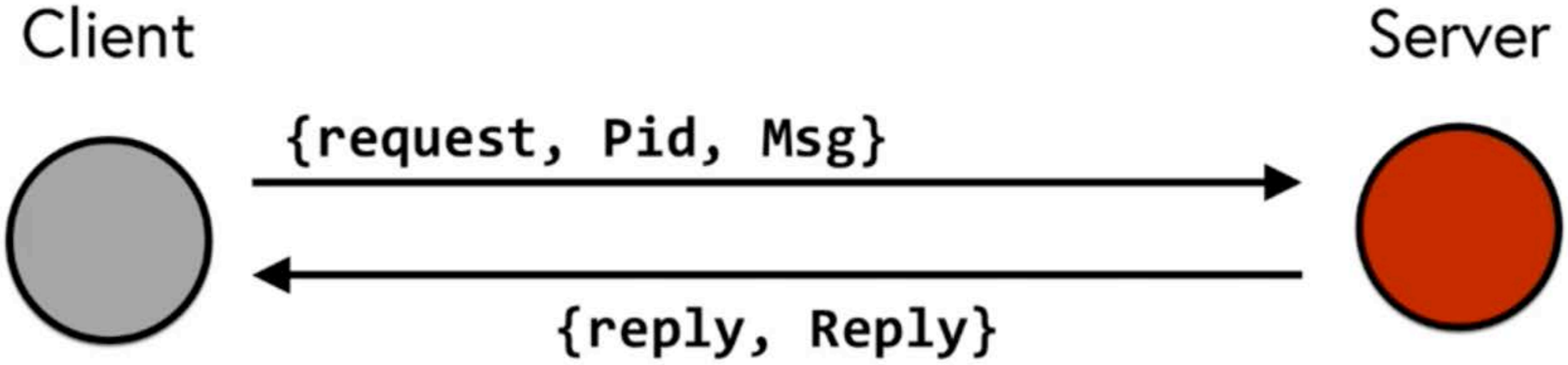


- If the client using the service needs a reply to the request, the call to the server has to be synchronous
- If the client does not need a reply, the call to the server can be asynchronous

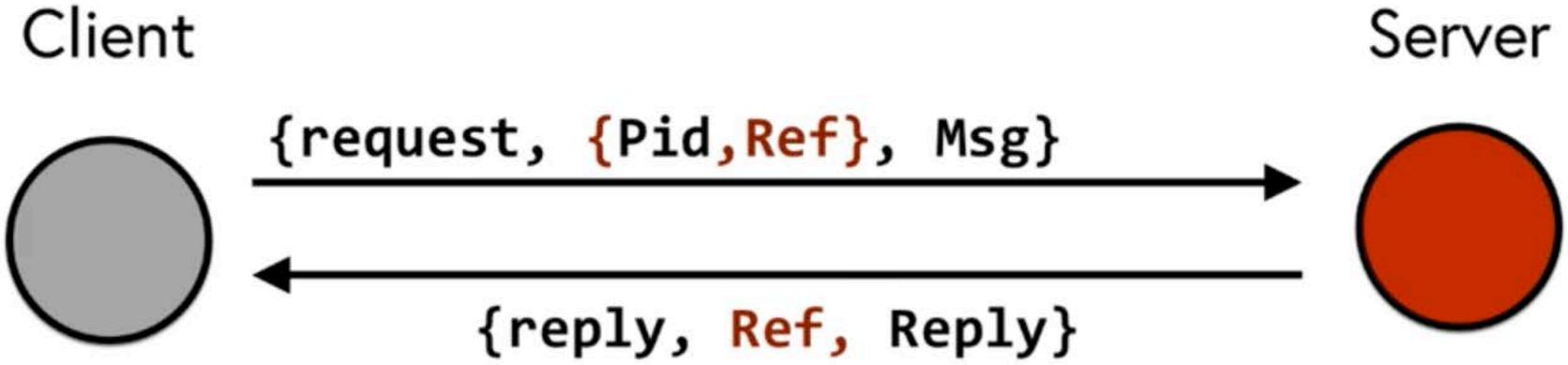


becomes

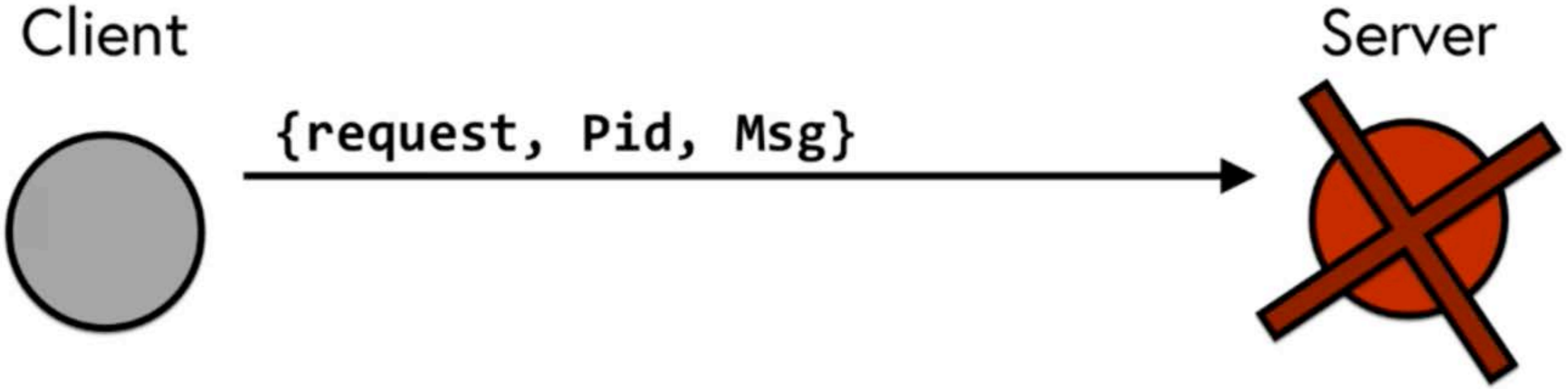
```
server:request(Msg) -> Reply
```

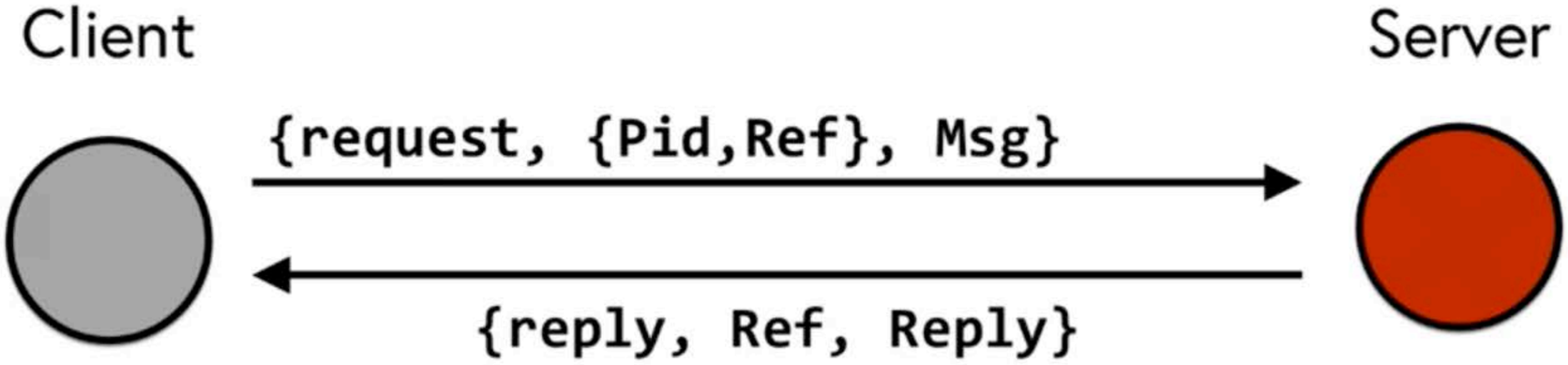



```
request(Name, Msg) ->
  Name ! {request, self(), Msg},
  receive
    {reply, Reply} ->
       Reply
  end.
```

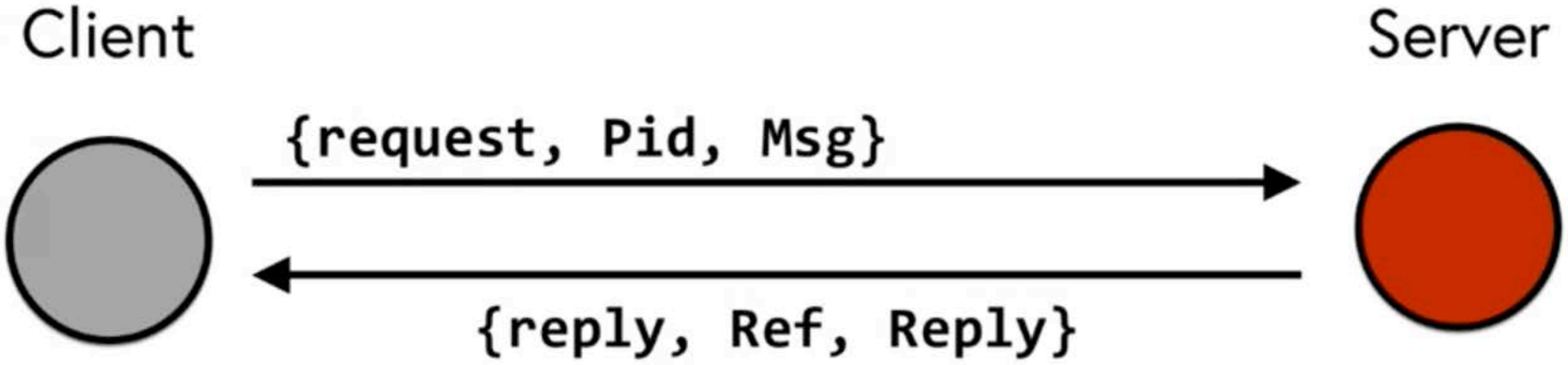


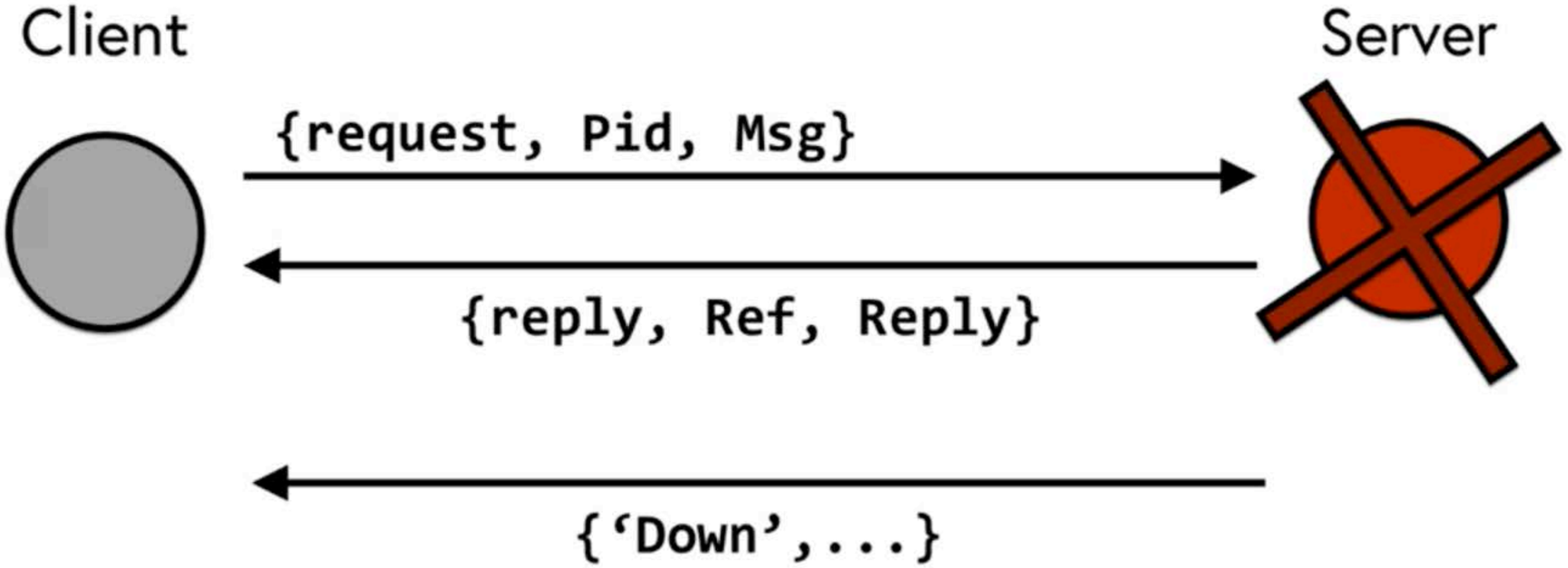

```
request(Name, Msg) ->
Ref = make_ref(),
Name ! {request, {self(), Ref}, Msg},
  receive
    {reply, Ref, Reply} ->
       Reply
  end.
```

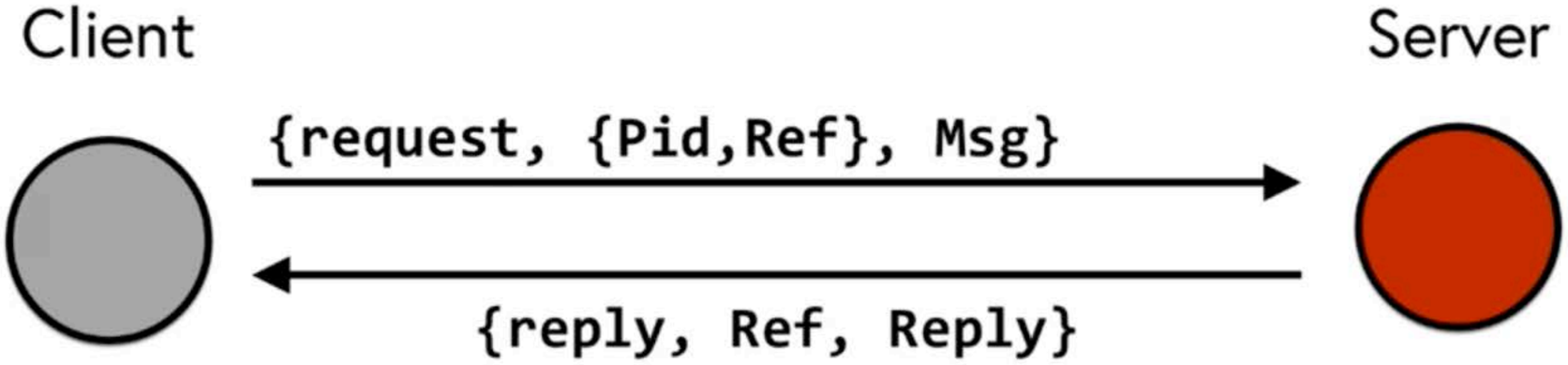





```
request(Name, Msg) ->
Ref = erlang:monitor(process, Pid),
Name ! {request, {self(), Ref}, Msg},
  receive
    {reply, Ref, Reply} ->
       erlang:demonitor(Ref, []),
          Reply
    {‘DOWN’, Ref, process, Pid, _} ->
       error(noproc)
  end.
```








```
request(Name, Msg) ->
Ref = erlang:monitor(process, Pid),
Name ! {request, {self(), Ref}, Msg},
  receive
    {reply, Ref, Reply} ->
       erlang:demonitor(Ref, [flush]),
          Reply
    {‘DOWN’, Ref, process, Pid, _} ->
       error(noproc)
  end.
```

University of Kent

Summary

- Understood the importance of abstracting message passing
- Looked at race conditions in concurrent systems
- A safe approach to sending and receiving requests


University of
Kent