University of Kent

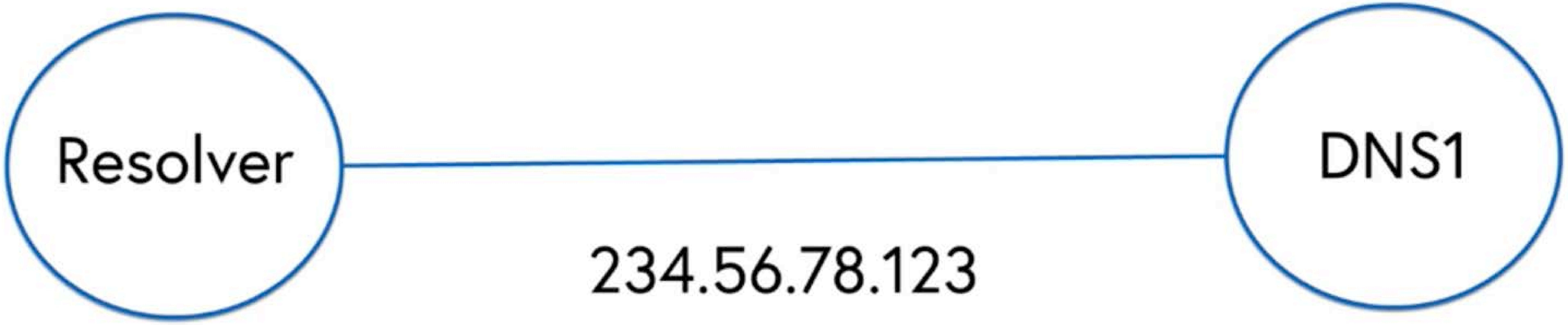
Joe Armstrong

EXPERT SYSTEM DEVELOPER, AND ONE OF THE CREATORS OF ERLANG AT ERICSSON

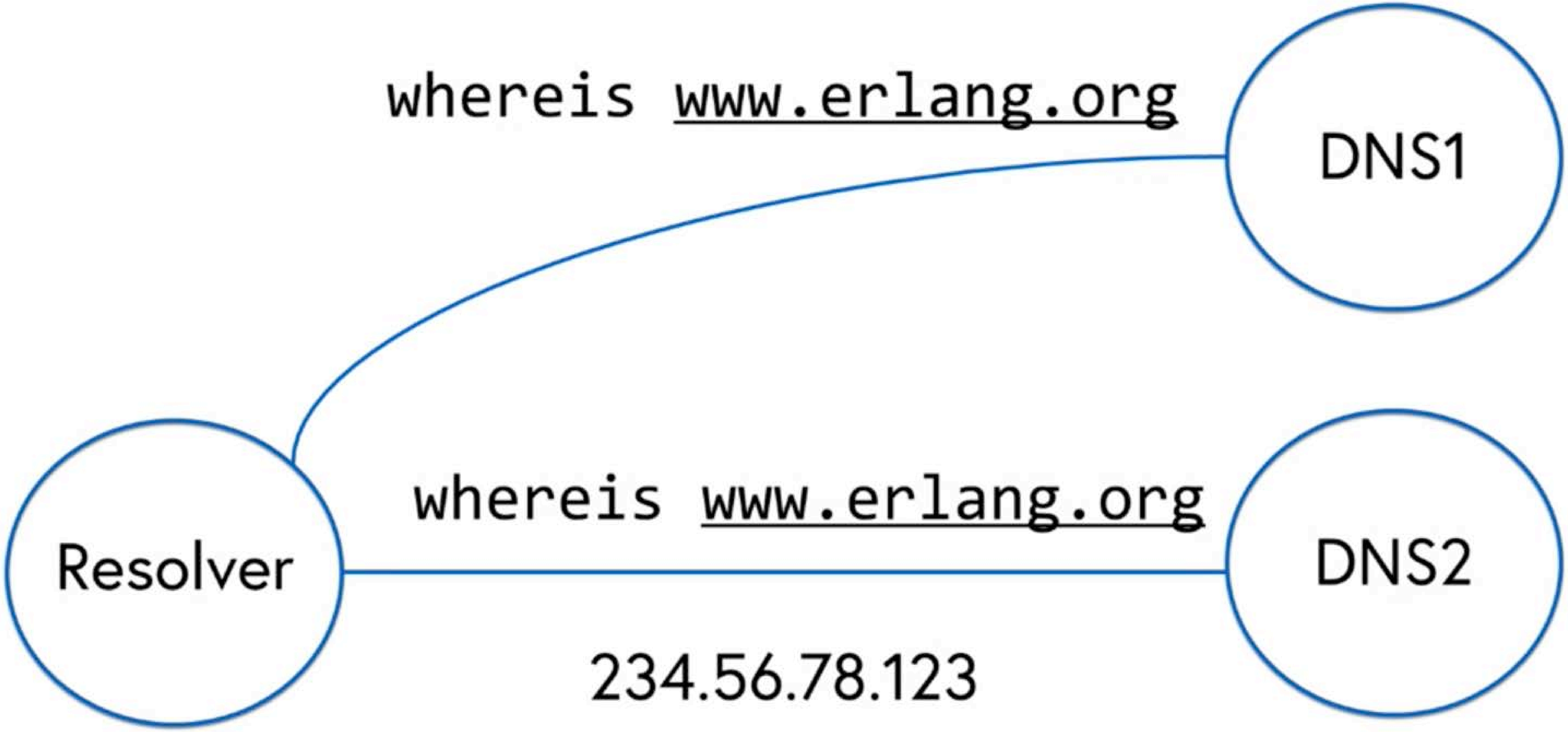
University of Kent

Domain Name System

`whereis www.erlang.org`



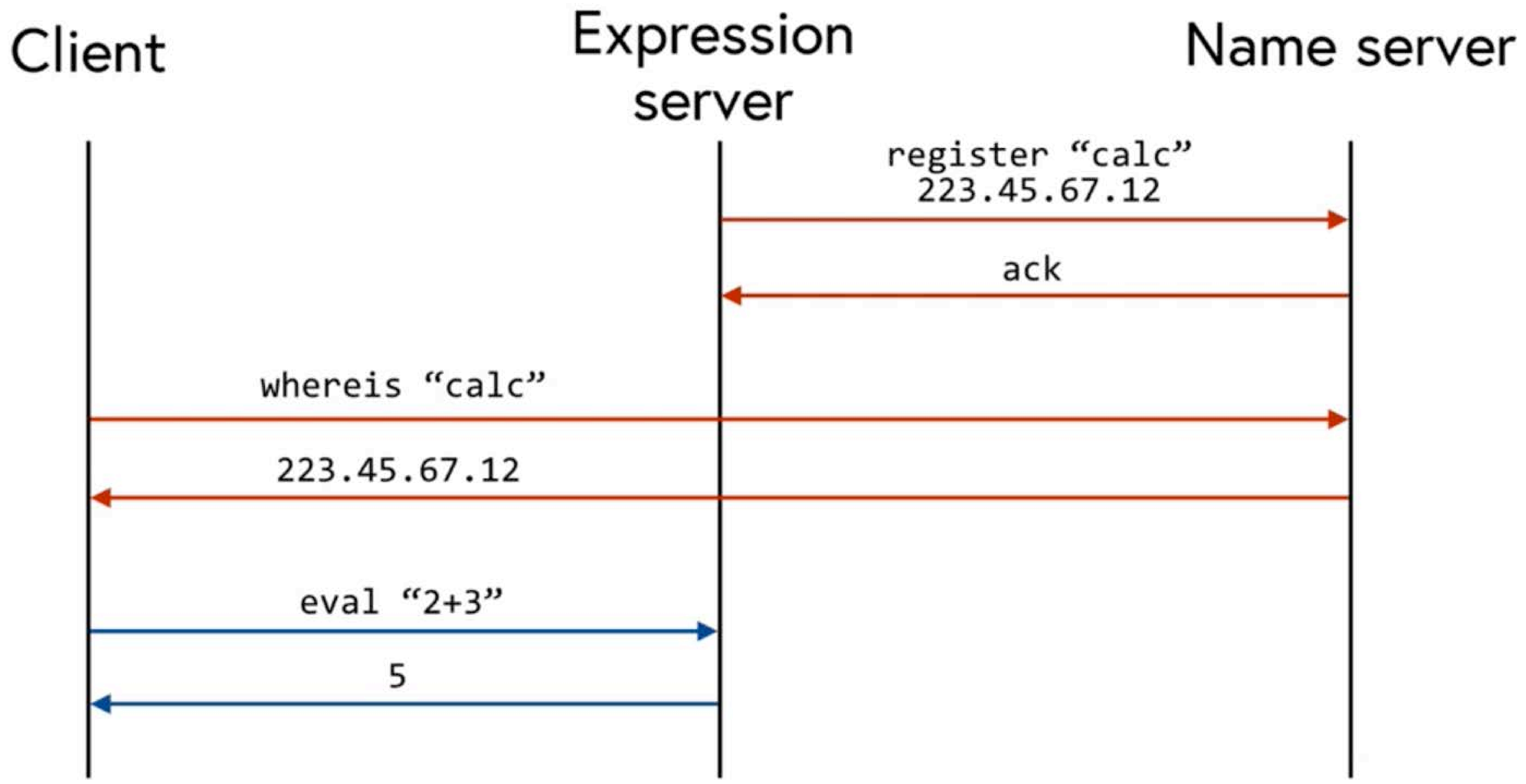
Fault tolerance



Fault tolerance

- Name server
- Converts names to addresses
- Replicated
- Hides a lot of complexity
- Resolver
- Client software that accesses the name server

Client + Server + NameServer



Reality

- Consistent data replication is a hard problem
- Security is tricky
- Caching is used (a lot) – cache coherency is a hard problem

```erlang
-module(calc).
-export([start/0, stop/0, execute/1]).
-export([init/0]).

start() -> spawn(calc, init, []).

init() ->
    io:format("Starting...~n"),
    register(calc, self()),
    loop().

loop() ->
    receive
       {request, From, Expr} ->
            From ! {reply, expr:eval(Expr)},
            loop();
        stop ->
            io:format("Terminating...~n")
    end.
```

```erlang
stop() ->
    calc ! stop.

execute(X) ->
    calc ! {request, self(), X},
    receive
        {reply, Reply} ->
            Reply
    end.
```

Notes

- Mirrors the (abstract) DNS design
- Can be used to test the logic of the server
- Most of the work is in writing the semantics of the server; the rest is boilerplate
- Starting / stopping / logging is more complicated than suggested here

Summary

- We turned sequential code into concurrent code
- Looked at primitives for writing concurrent code
- Started on a name-server – resolver – client – server framework

University of Kent