



Sneaking Erlang in Through the Back Door

and convincing people you're not mad

Bernard Duggan - Sr. Telephony Engineer
ShoreTel
Erlang Factory SF - 2014-03-07



whatsapp.com



2/37



erlang.org



3/37

I Come from the Land Down Under



maps.google.com

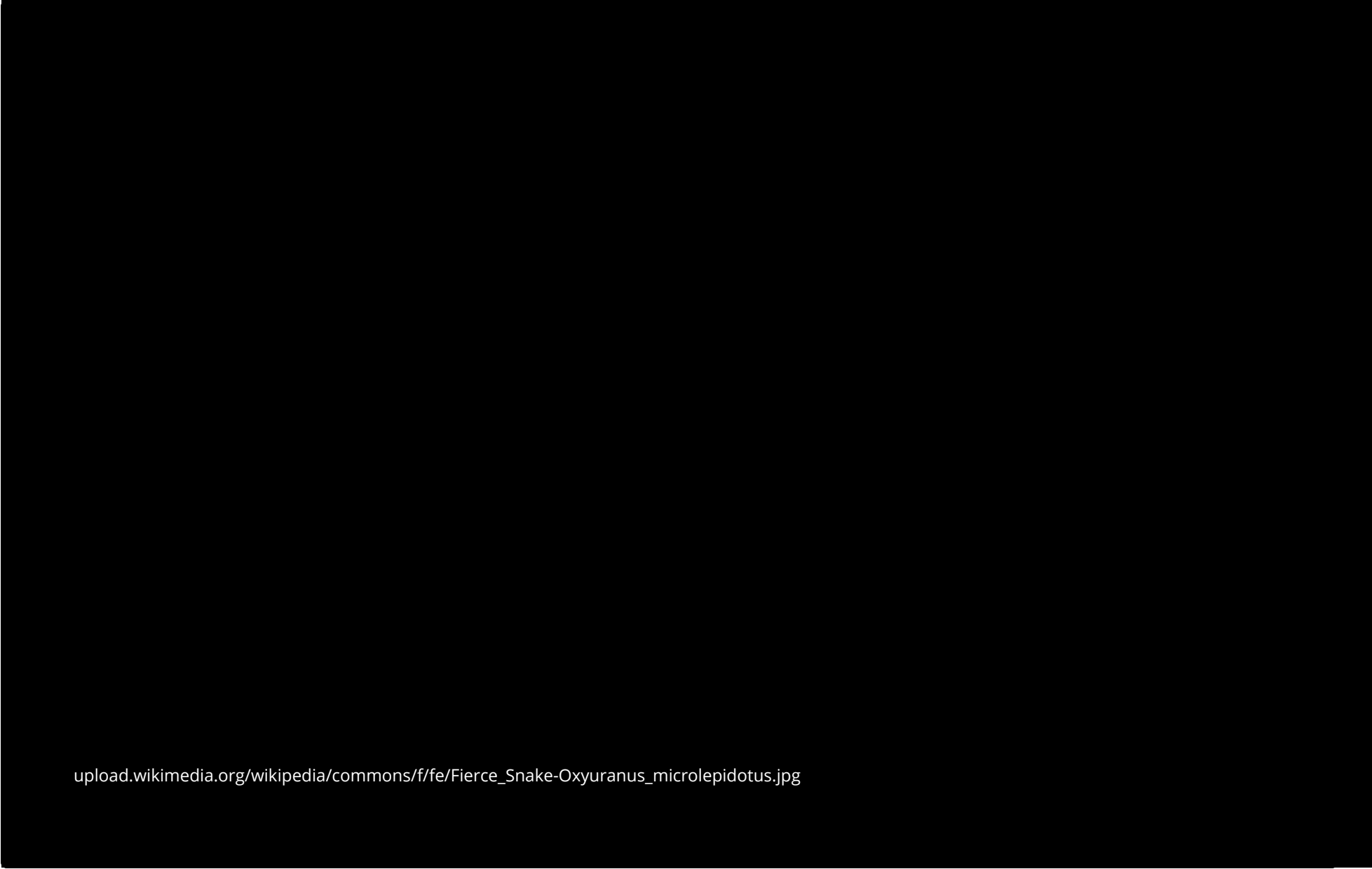


4/37



www.canberrayourfuture.com.au/workspace/images/banner-three.jpg





upload.wikimedia.org/wikipedia/commons/f/fe/Fierce_Snake-Oxyuranus_microlepidotus.jpg



media.brainz.org/uploads/2011/03/12-great-white.jpg



www.nedmartin.org/v3/amused/kangaroo-kicks-boy-into-dam

Then: M5 Networks

- Hosted business phone systems
- VoIP as a Service
- Just a phone and an Internet connection on the client side
- Thousands of companies hosted on our servers
- >10,000 phones per server
- <100 employees



Now: ShoreTel

- Sells both hosted and premise systems
- Approx 975 employees (according to Wikipedia)
- 3rd largest business VoIP provider in the US (according to some guy I met in the corridor)



Legacy Platform

- C/C++ code (~250k lines)
- Massively parallel*
- Robust
- Fast in-memory database (home-grown)
 - Speed
 - Reliability
- Surprisingly scalable

* Sort of



12/37

Far from Perfect

- C/C++ code (~250k lines)
- One invalid pointer can take down thousands of phones
- No hot upgrades
- Ever tried to debug a crash in the STL?
- State machines are not an easy concurrency model (no blocking allowed)
- Threads/Processes are not a practical concurrency model at this scale (in C++)





Enter Erlang

"Hey, we're not the first people to have these problems!"

Erlwhat?

- Colleague returned from LCA 2007 raving about Erlang
- Naturally we dismissed him as being a lunatic
- Yet...on the face of it it did seem a good fit:
 - Highly concurrent
 - Robust
 - High level language
 - Even had its own distributed in-memory DB



First Experiment

Elvis TFTP

- Happened to have a small, stand-alone task - a multiplexing TFTP server for Cisco phones
- World didn't end when we started using it in production



PHLEM

The PHone Logic EMulator

- Somehow we'd gotten where we were with no test clients other than real phones
- Initial bit of code was written in "spare time" to emulate a single phone
- I picked it up and with minimal effort made it work for thousands of phones
- Okay, maybe our lunatic was on to something here...



Spread the Word!



w.minecraft-smp.de/e107_files/public/1366197563_42_FT9337_qantas-747.jpg

© AP6 PHOTOGRAPHY

Convincing the rest of the Team

- Gave talks explaining Erlang from the basics
 - What it looks like
 - Why it isn't scary
 - Why it was a perfect fit for us
- Demonstrated what we'd done so far and **how little code it took**





Objections

"We shouldn't do this because..."

“There's no inheritance”



“No global data at all which is like killing a dog because it has flies”



“The Law of Leaky Abstraction”



“Obscure language - NetBSD has more websites than Erlang”



“If someone likes the syntax, it can be easily [reimplemented] in C++ using operators”





Hooking It In

You mean we can't just rewrite everything from scratch?

So There's All This Pesky Legacy Code...

And for some reason people will be upset if it stops working

- Erlang use initially restricted to new functionality
- Pick features that weren't "core":
 - Call Recording
 - CNAM
 - Monitoring
- Has to interface with the old code
- Which means it has to talk...



Binary Struct-based Protocols

It seemed like such a good idea at the time

```
typedef struct {  
    int id;  
    long long key;  
    char value[10];  
} message_t;
```

C++

Sender

```
message_t msg = {5, 10, "Hello"};  
write(sock, &msg, sizeof(msg));
```

C++

Receiver

```
message_t msg;  
read(sock, &msg, sizeof(msg));
```

C++



28/37

Erlang Side

Receiver

```
parse(<<Id:32/signed-little, Key:64/signed-little, Value:(10*8)/bitstring>>) - ERLANG
    #message_t{
        id = Id,
        key = Key,
        value = lists:takewhile(fun(A) -> A /= 0 end, bitstring_to_list(Value)).
```

Sender

```
encode(#message_t{id = Id, key = Key, value = Value}) -> ERLANG
    <<
        Id:32/signed-little,
        Key:64/signed-little,
        (list_to_bitstring(string:substr(Value, 1, 10-1)))/bitstring,
        0:((10 - lists:min([10-1, length(Value)]))*8)
    >>.
```



Oops

Forgot the padding...

Receiver

```
parse(<<Id:32/signed-little, Key:64/signed-little, Value:(10*8)/bitstring, _Pad:(2*8)>>) ->  
  #message_t{  
    id = Id,  
    key = Key,  
    value = lists:takewhile(fun(A) -> A /= 0 end, bitstring_to_list(Value)).
```

ERLANG

Sender

```
encode(#message_t{id = Id, key = Key, value = Value}) ->  
  <<  
    Id:32/signed-little,  
    Key:64/signed-little,  
    (list_to_bitstring(string:substr(Value, 1, 10-1)))/bitstring,  
    0:((10 - lists:min([10-1, length(Value)]))*8),  
    0:(2*8)  
  >>.
```

ERLANG



30/37

Oops again

x86_64 pads differently...

Receiver

```
parse(<<Id:32/signed-little, _Pad/(4*8), Key:64/signed-little,  
      Value:(10*8)/bitstring, _Pad2:(6*8)>>) ->  
    #message_t{  
      id = Id,  
      key = Key,  
      value = lists:takewhile(fun(A) -> A /= 0 end, bitstring_to_list(Value)).
```

ERLANG

Sender

```
encode(#message_t{id = Id, key = Key, value = Value}) ->  
    <<  
      Id:32/signed-little,  
      0:(4*8), % Pad to long long boundary  
      Key:64/signed-little,  
      (list_to_bitstring(string:substr(Value, 1, 10-1)))/bitstring,  
      0:((10 - lists:min([10-1, length(Value)]))*8),  
      0:(6*8) % Pad to boundary of largest struct member  
    >>.
```

ERLANG



31/37

Yarn Code Generator

- Erlang-like canonical files
- Erlang parser + yecc + erlydtl
- Many output formats:
 - Erlang (records, encoders/decoders)
 - C++ (structs)
 - Java (classes, encoders/decoders)
 - Python (DB modification)
 - SQL (DB schema and updates)



Rec files

```
constant(MESSAGE_SIZE = 10) % length of the message

record(
  message_t {
    id :: integer,
    key :: long_long,
    value :: string : [{max_len, 'MESSAGE_SIZE'}],
  })
```

REC

```
typedef struct {
  int id;
  long long key;
  char value[10];
} message_t;
```

C++

```
-record( message_t, {
  id :: integer(),
  key :: integer(),
  value :: string()
}).
```

ERLANG



33/37

Other Things

- Keep your sysadmins happy - use `erld`
- Monitor your system - Erlang's fault tolerance is a blessing and an occasional curse
- Use Dialyzer to catch beginner (and advanced) mistakes



Where are we now?

- All new major features written in Erlang
- Rewritten two legacy features in Erlang
- 385k lines of C++
- 171k lines of Erlang
- Which makes Erlang the majority (in terms of functionality*)
- Erlang used for
 - Hosted call recording
 - CNAM
 - Web admin UI
 - Support for latest phone models
 - DB sync
 - Call tracking
 - TFTP
 - Parked call management
 - Load testing
 - System monitoring (SNMP etc)
 - SIP director

* According to a metric I just made up



35/37

Building Acceptance

Start Small:

- Start with non-core elements
- Don't build anything critical first - you will make mistakes
- One piece at a time - you're not going to get a year off to rewrite your whole system
- Erlang is easy^{[[citation-needed1](#)]} but building concurrent, high load, reliable production systems still isn't

[1] Mike Williams, Erlang Factory SF 2014 keynote



36/37

Thank You! Questions?



Send whiskey to:
email bduggan@shoretel.com

www shoretel.com
github github.com/bernardd