Powerful, Open-Source VoIP with Erlang

Presented by James Aimonetti

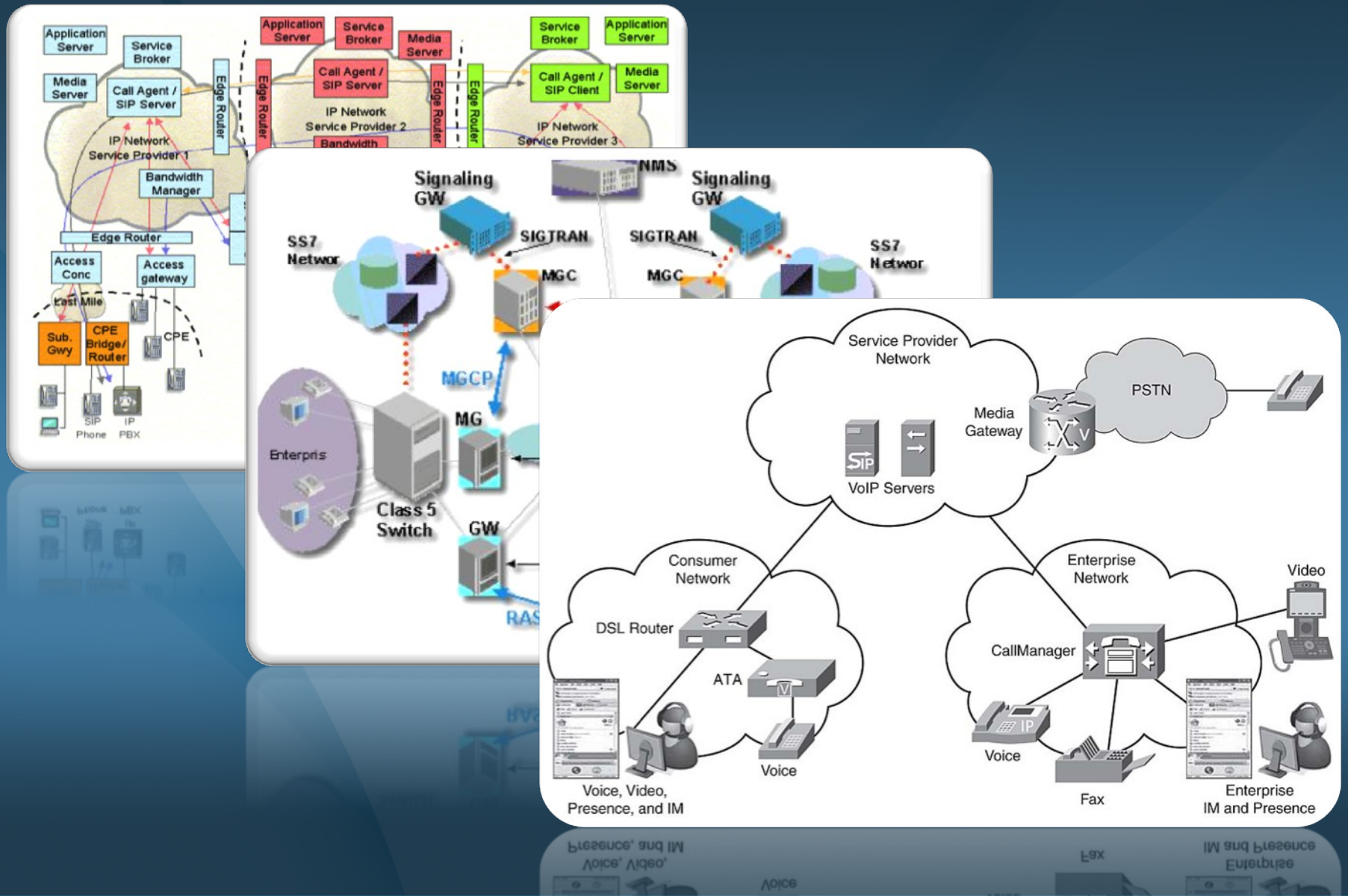
2600hz
THE FUTURE OF CLOUD TELEPHONY

About Me

James Aimonetti

- Senior Distributed Software Engineer
- I <3 Erlang
- Background in Comp Sci & Mathematics
- Sports Enthusiast
- mc_ on #2600hz, #erlang

Scaling VoIP is Complicated



VoIP Tools Are Fragmented

Lots of Mature Tools

- OpenSIPs
- FreeSWITCH
- CDR Tool
- DTH / BillSoft
- Yate
- Soft Clients
- Etc.

All of them live on their own

- ... yet depend on each other

We Need Glue.

Choosing Our Glue

2600hz

Need to Understand Common Needs

- Scale easily
  - # of VoIP channels
- Reliability
  - Redundancy of Call Processing
  - Consistent Call Handling
- Flexibility
  - Access to ANY APIs in a SoftSwitch
  - No lock-in to specific development languages
- Ownership
  - Own data / VoIP circuits / Software / etc.

Whistle VoIP Platform

Whistle VoIP Platform = Glue

- AMQP - RabbitMQ
- CouchDB
- FreeSWITCH
- OpenSIPs
- HTTP / REST - Webmachine
- Business Logic (WhApps)

Why These Components?

Our Research: the Core

Erlang = Super Glue

- Built for Telecom
- Strong Supervision
- Inherently Distributable
- Highly Concurrent
- Asynchronous Design is Easy
- Code is Short, Concise, Powerful
- Cross-Platform (even Windows & MIPS!)
- Fast
- EASY

Our Research : the Core

Event Processing in PHP

- 191 Lines to Parse Events (text)

```php
<?php defined('SYSPATH') or die('No direct access allowed.');
/**
 * ESLevent.php - This is used when the native ESL extension is not avaliable
 *
 * @author K Anderson
 * @license LGPL
 * @package Esl
 */
class ESLevent {
    private $headers = array('Event-Name' => 'COMMAND');

    private $body = NULL;

    private $hdrPointer = NULL;

    public function __construct($event) {
        if (!is_array($event)) {
            $this->addHeader('Event-Name', $event);
            return $this;
        }

        foreach ($event as $line) {
            if ($line == "\n") {
                continue;
            } else if (strstr($line, ':')) {
                list($key, $value) = explode(':', $line);
                $this->addHeader($key, $value);
            } else {
                $this->addBody($line);
            }
        }

        $this-> convertPlainEvent();
```

Event Processing in Erlang

- 23 Lines to Parse Events (native)

```
spec(start_handler/3 :: (Node :: atom(), Options :: prop
start_handler(Node, _Options, Host) ->
    HState = #handler_state{fs_node=Node, app_vsn=list_t
    case freeswitch:start_fetch_handler(Node, dialplan,
        timeout -> {error, timeout};
        {error, _Err}=E -> E;
        {ok, RPid} when is_pid(RPid) -> RPid
    end.

fetch_route(Node, #handler_state{lookups=LUs, stats=Stat
    receive
        {fetch, dialplan, _Tag, _Key, _Value, ID, [UUID
                    Self = self(),
                    LookupPid = spawn_link(?MODULE, look
                    LookupsReq = Stats#handler_stats.loo
                    format_log(info, "FETCH_ROUTE(~p): t
                                ,[self(), ID, UUID, Looku
                    ?MODULE:fetch_route(Node, State#hand

            end;
    end.
```

Our Research : Messaging

Need : Real-time Messaging

- Call Control
- Resource Monitoring

Why AMQP

- Built-in Messaging is Fast
  - Designed for Financial Systems
- Easy to Scale & Cluster
- Most Important: Directed Messaging
  - Messages only go where they need to go
  - On a busy switching environment, this is critical
  - Multiple Strategies for Directing Messages

Our Research : Messaging

How it relates to telecom

- One VoIP channel is going to produce:
  - 1 Request / Multiple Initial Responses
  - 100 or so call events published
  - 0->Many messages for call manipulation
  - Expecting 300 calls/second per box
  - Expecting 3,000-4,000 events per second max

Our Research : Storage

2600hz

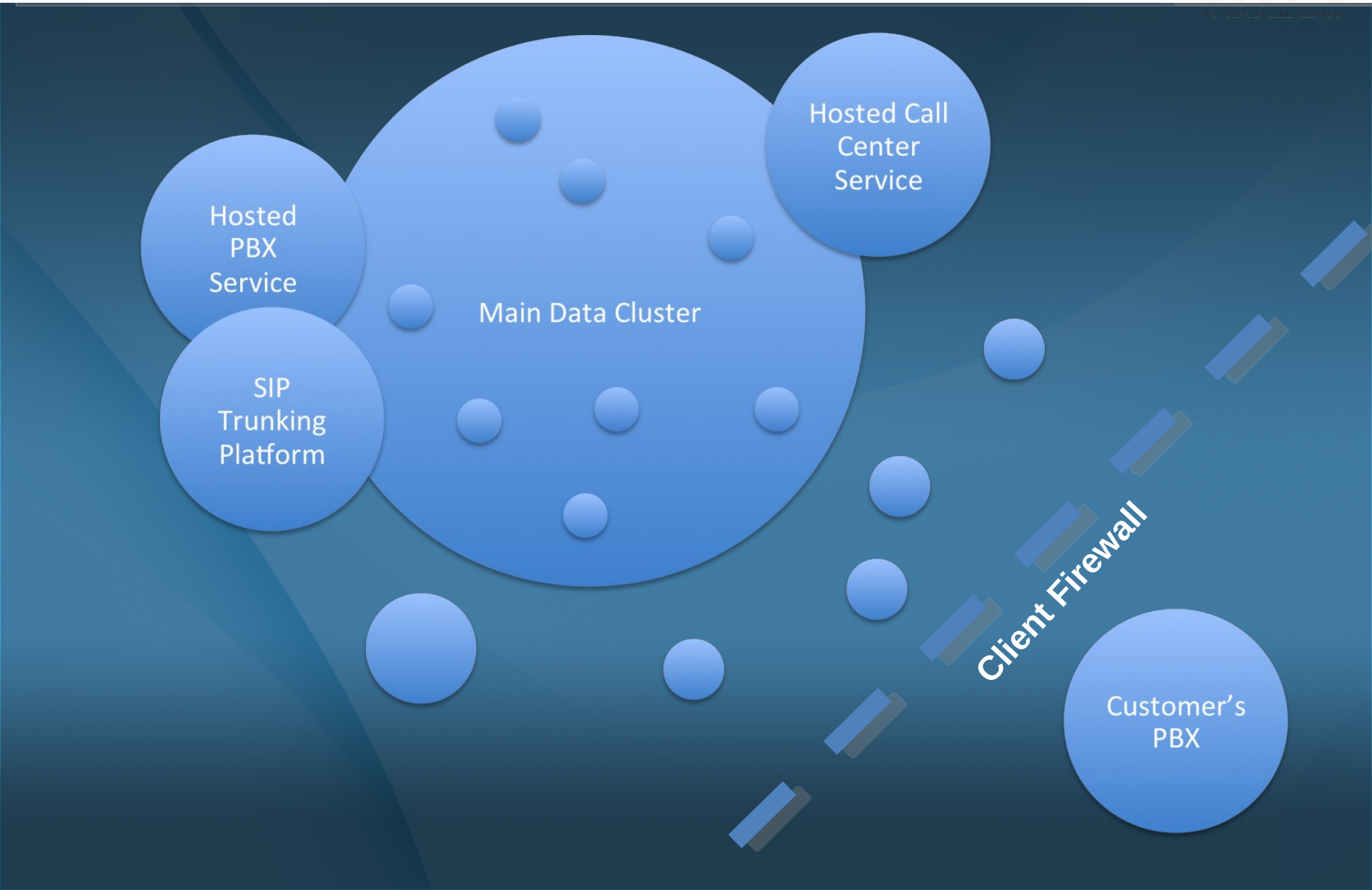
Need : Scalable Storage, Flexible Schema

- Heavy Read, Less Write (reconfigure infrequently)
- Features Change Constantly
  - 0 downtime for maintenance is goal

Why CouchDB?

- NoSQL – based
  - Schema changes regularly, but usually based on core object (translates well to a document)
- Databases are Lightweight Concepts
- Replication is Stupidly Simple
  - A database, list of documents, or a view

Our Research : Storage



Our Research : Scalability

2600hz

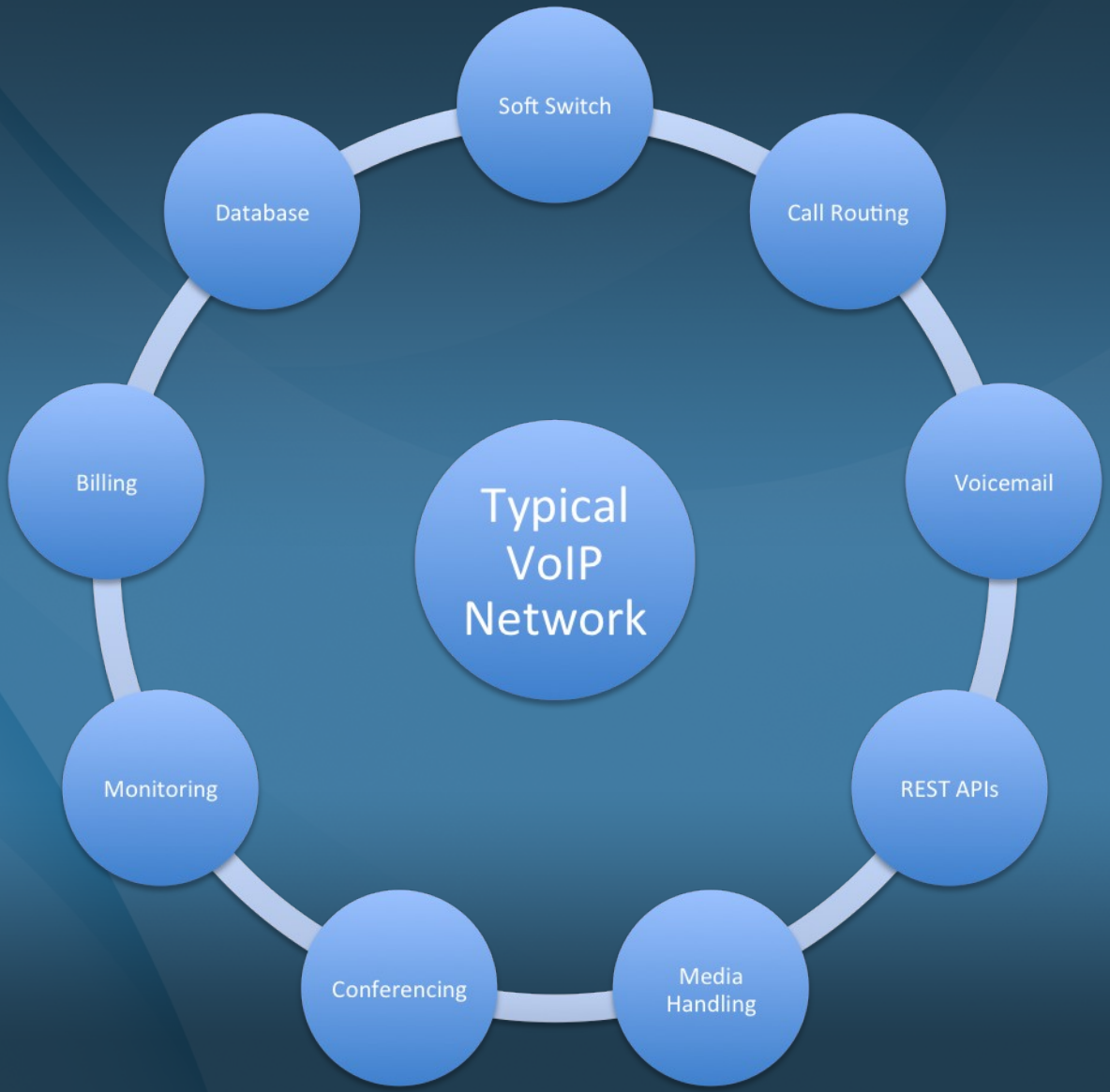
Need : Fast Deployment, Easy Management

- Growth in Customers Usually Inconsistent
- Resource Demand Varies by Situation
- Resource Demand Varies by Component

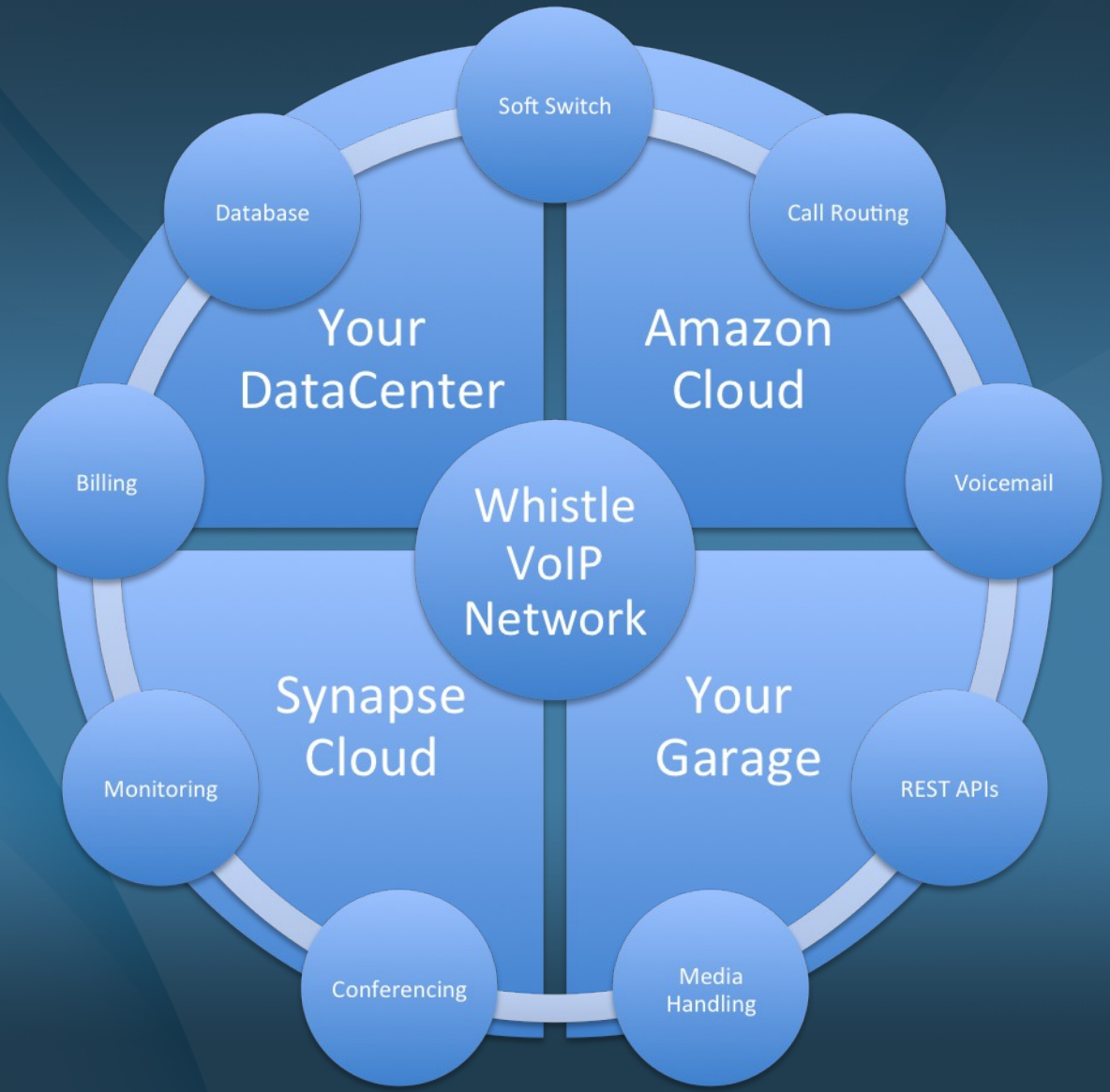
Why Erlang?

- One VM and Library to Deploy
- Networking is Built-In
- Many, many lightweight threads possible
- Everything can live anywhere

Our Research : Scalability



Our Research : Scalability



Our Research : Maintenance

Need : Monitoring, Up-Time

- This needs to be built-in because it’s expected
- Nothing standard really out there

Why FreeSWITCH + Erlang?

- Round-trip media monitoring with audio
- Test true audio latency on circuits
- Test true up-time across all call paths

Our Research : Maintenance

Audio Sent

Audio Received

Latency of X miliseconds

Our Research : Simplicity

Need : APIs, Easy Mashups, Simple

- Whistle = The Ultimate Mashup Tool

Why REST / Crossbar?

- Layer 1: Abstraction of real-time events
- Layer 2: Abstraction of common features
- Layer 3: Provide Common Interface
  - REST keeps it easy and language agnostic

Our Research : Simplicity

Our Research : Simplicity

```
review_recording(#mailbox{prompts=Prompts, file_id=FileId, keys=Keys}=Box, Call) ->
    play(Prompts#prompts.press, ?ANY_DIGIT, Call),
    say(Keys#keys.listen, <<"name_spelled">>, Call),
    play(Prompts#prompts.to_listen, ?ANY_DIGIT, Call),
    play(Prompts#prompts.press, ?ANY_DIGIT, Call),
    say(Keys#keys.record, <<"name_spelled">>, Call),
    play(Prompts#prompts.to_save, ?ANY_DIGIT, Call),
    play(Prompts#prompts.press, ?ANY_DIGIT, Call),
    say(Keys#keys.save, <<"name_spelled">>, Call),
    play(Prompts#prompts.to_rerecord, ?ANY_DIGIT, Call),
    case cf_call_command:wait_for_dtmf(10000) of
        {error, _} ->
            store(Box, Call),
            {stop};
        {ok, Digit} ->
            flush(Call),
            if
                Digit == Keys#keys.listen ->
                    cf_call_command:b_play(FileId, ?ANY_DIGIT, Call),
                    review_recording(Box, Call);
                Digit == Keys#keys.save ->
                    record_voicemail(Box, Call);
                true ->
                    store(Box, Call),
                    {stop}
            end
    end.
```

The Full Picture