2600hz

CLOUD TELECOM

Crossing Language Barriers for Fun and Profit

Presented by Karl Anderson

Karl Anderson

Senior Bit Herder

Why am I

No. 700,986. Patented May 27, 1902.

G. D. SMITH.

MUSICAL TOY.

(Application filed May 15, 1901.)

(No Model.)

Fig. 1.

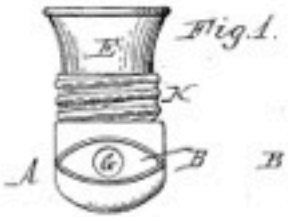


Fig. 2.

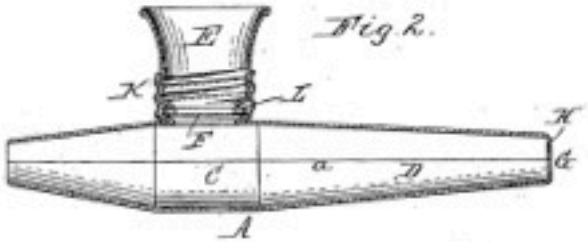


Fig. 3.

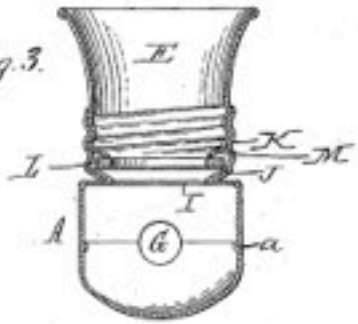


Fig. 4.

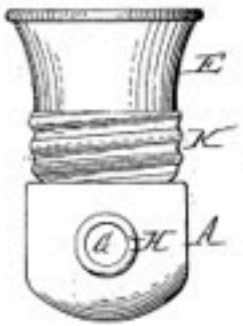


Fig. 5.

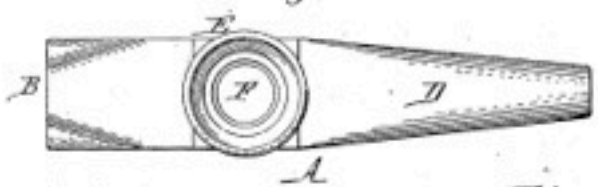


Fig. 6.

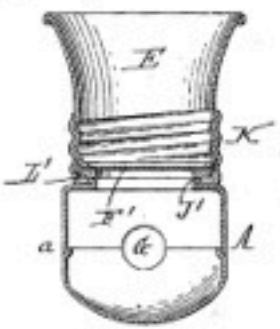
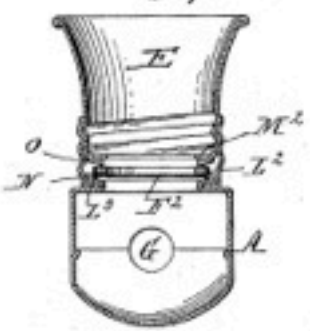


Fig. 7.

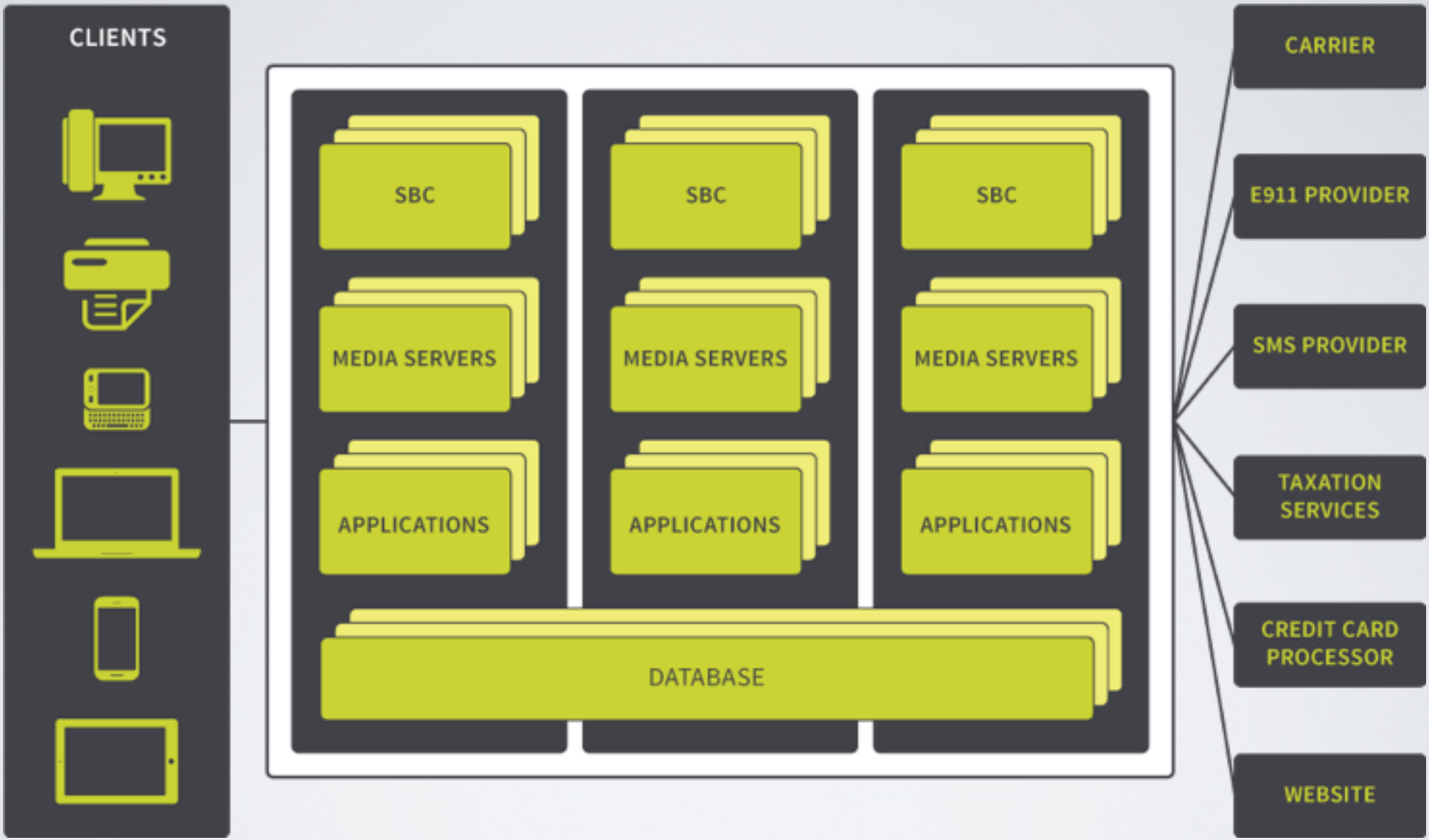


Witnesses:
Henry L. Deck.
F. F. Schreiner.

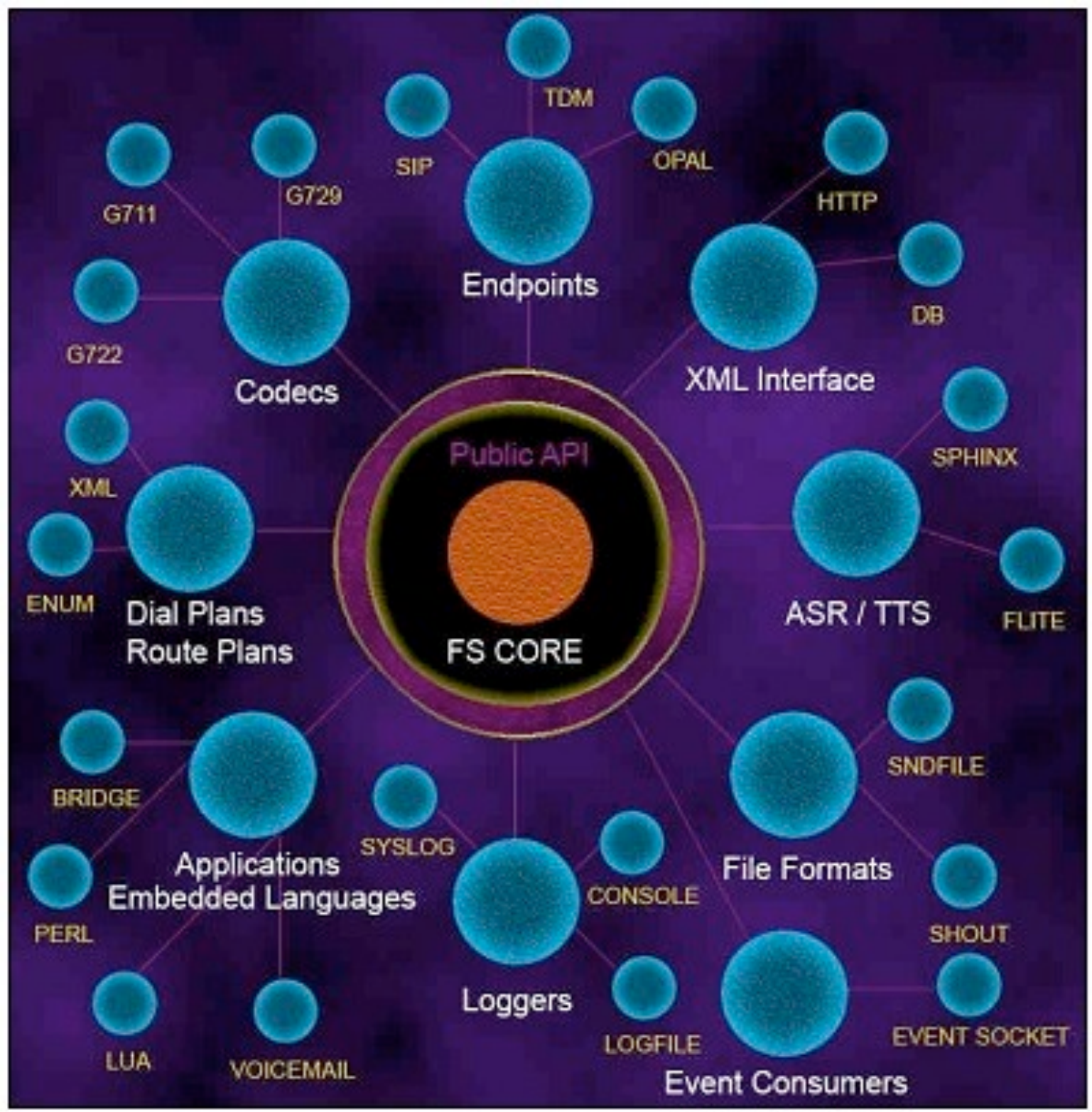
George D. Smith Inventor
By Wilkinson & Bonner
Attorneys.


SRY, KANT DOO THAT
KTHXBYE

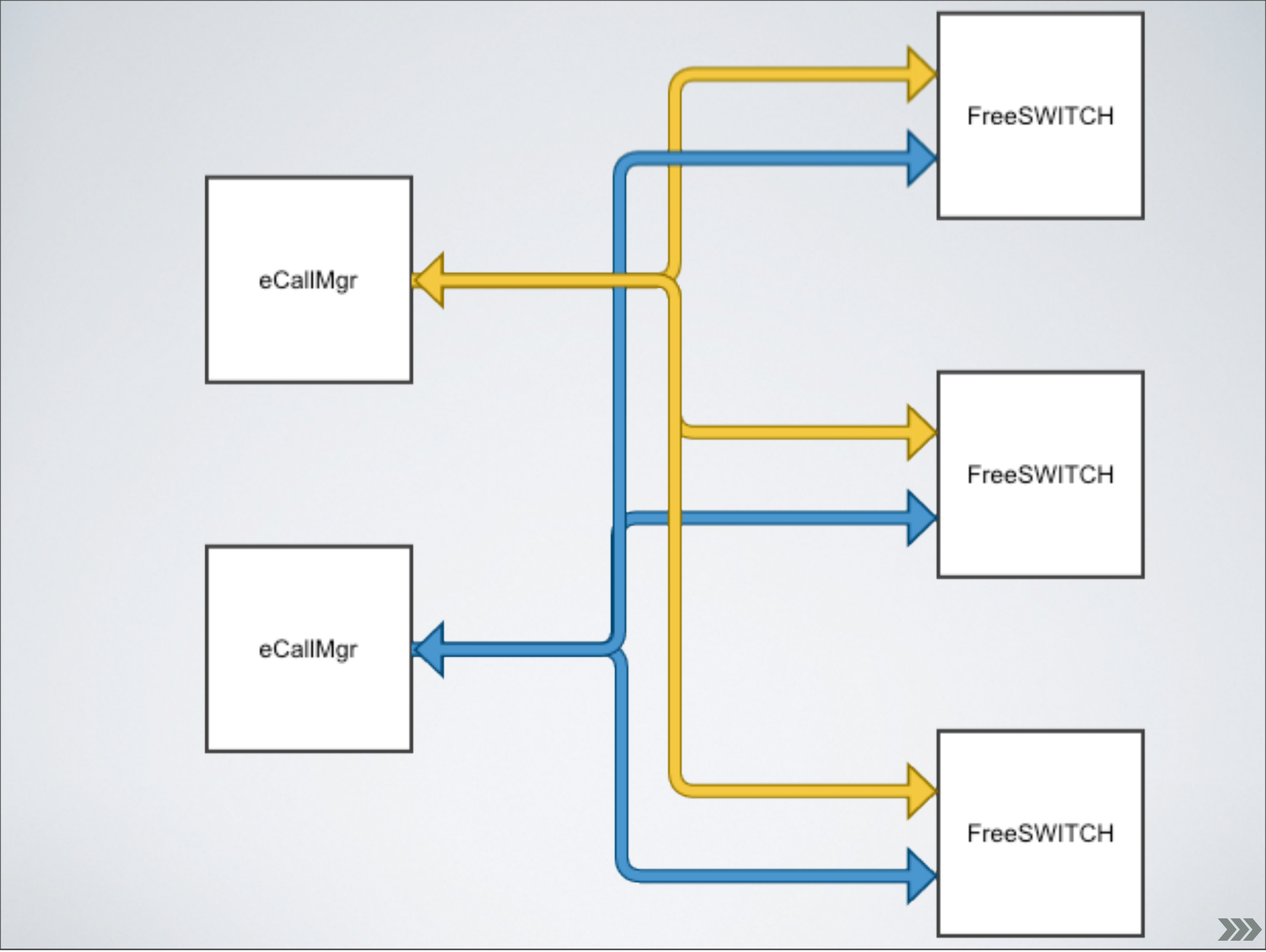
What is Kazoo


CLIENTS
SBC
SBC
SBC
MEDIA SERVERS
MEDIA SERVERS
MEDIA SERVERS
APPLICATIONS
APPLICATIONS
APPLICATIONS
DATABASE
CARRIER
E911 PROVIDER
SMS PROVIDER
TAXATION SERVICES
CREDIT CARD PROCESSOR
WEBSITE

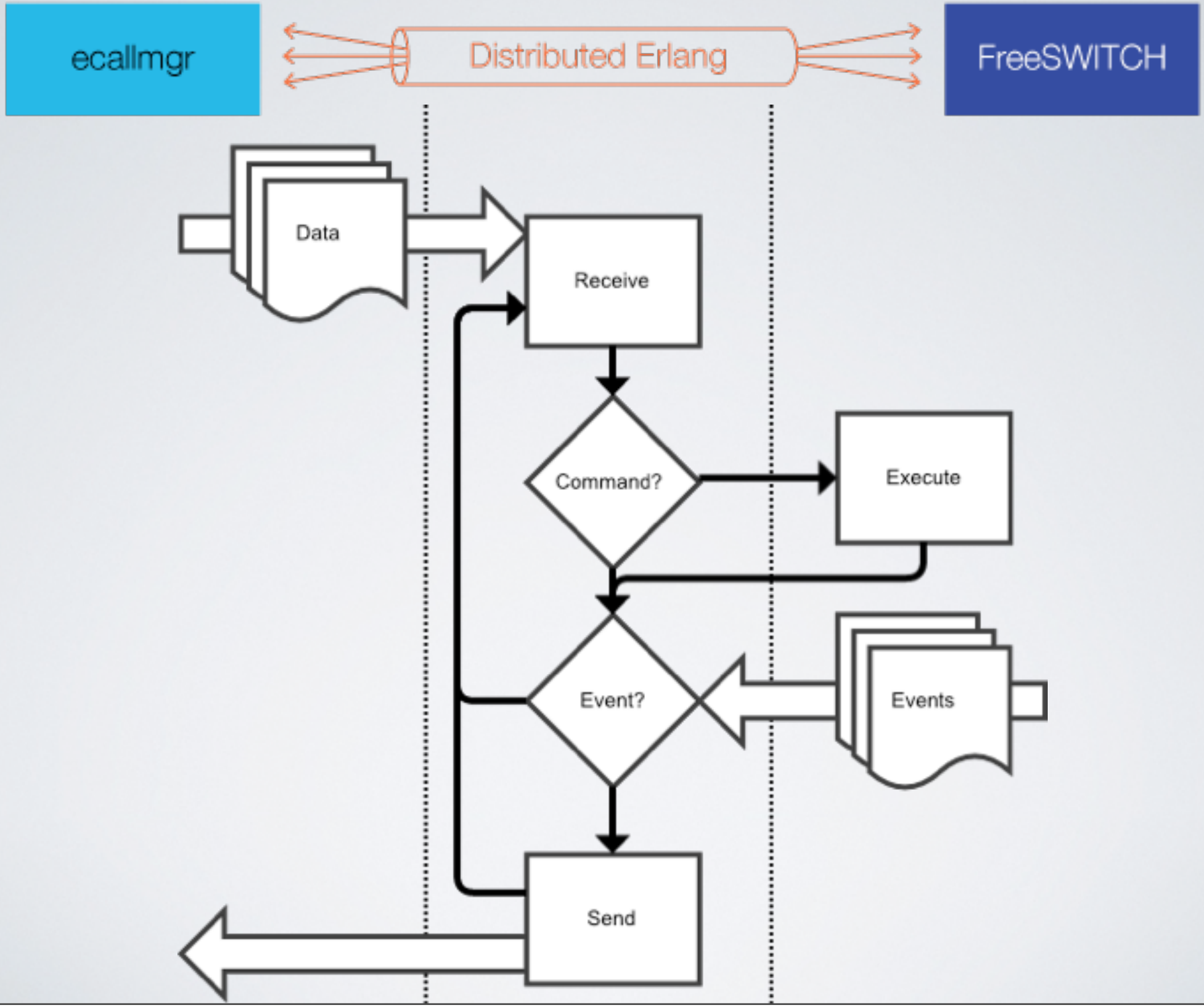
What is FreeSWITCH


TDM
SIP
OPAL
G711
G729
HTTP
Endpoints
G722
Codecs
DB
XML Interface
Public API
XML
SPHINX
ENUM
Dial Plans
Route Plans
FS CORE
ASR / TTS
FLITE
BRIDGE
SNDFILE
SYSLOG
Applications
Embedded Languages
File Formats
CONSOLE
PERL
SHOUT
Loggers
LUA
VOICEMAIL
LOGFILE
EVENT SOCKET
Event Consumers

What is mod_kazoo


FreeSWITCH
eCallMgr
FreeSWITCH
eCallMgr
FreeSWITCH

How does it work


ecallmgr
Distributed Erlang
FreeSWITCH
Data
Receive
Command?
Execute
Event?
Events
Send

```
ei_x_new_with_version(&rbuf);
status = ei_xreceive_msg_tmo(listener->sockfd, &msg, &buf, 10);
switch (status) {
case ERL_TICK:
        break;
case ERL_MSG:
        switch (msg.msgtype) {
        case ERL_SEND:
                if (handle_msg(listener, &msg, &buf, &rbuf)) {
                        return;
                }
                break;
        case ERL_REG_SEND:
                if (handle_msg(listener, &msg, &buf, &rbuf)) {
                        return;
                }
                break;
        case ERL_EXIT:
                handle_exit(listener, &msg.from);
                break;
        default:
                break;
        }
        break;
case ERL_ERROR:
         break;
default:
          break;
}
check_log_queue(listener);
check_event_queue(listener);
if (check_attached_sessions(listener) != SWITCH_STATUS_SUCCESS) {
      return;
}
```

```
Self = self(),
spawn(fun() ->
        {'bgapi', Node} ! {'bgapi', Cmd, Args},
        receive
            {'error', Reason} ->
                Self ! {'api', {'error', Reason}};
            {'ok', JobID} ->
                Self ! {'api', {'ok', JobID}},
                receive
                    {'bgok', JobID, Reply} ->
                        Self ! {'bgok', JobID, Reply};
                    {'bgerror', JobID, Reply} ->
                        Self ! {'bgerror', JobID, Reply}
                end
        after ?TIMEOUT ->
                Self ! {'api', 'timeout'}
        end
  end),
receive
    {'api', X} -> X
end
```

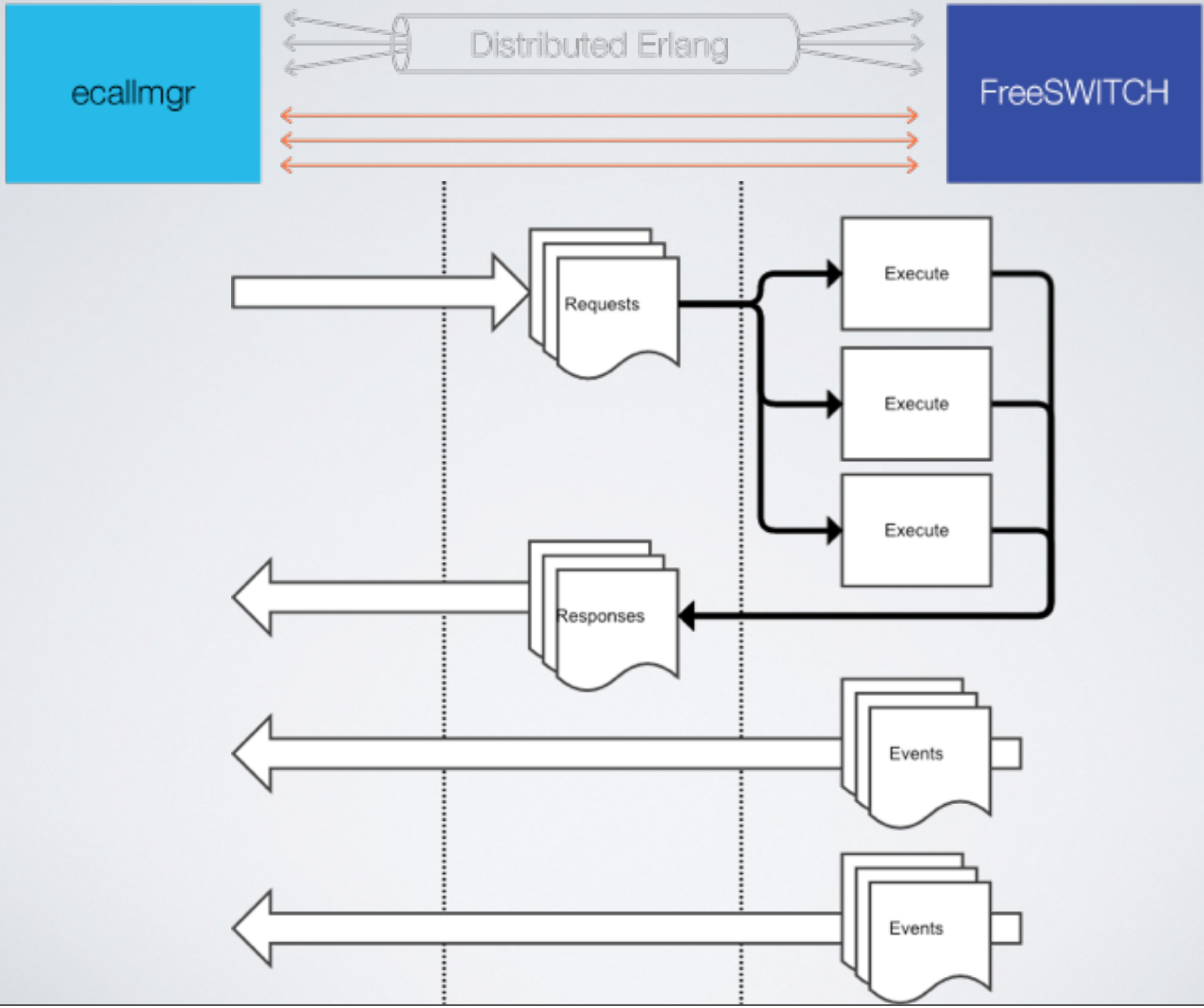
What's the problem

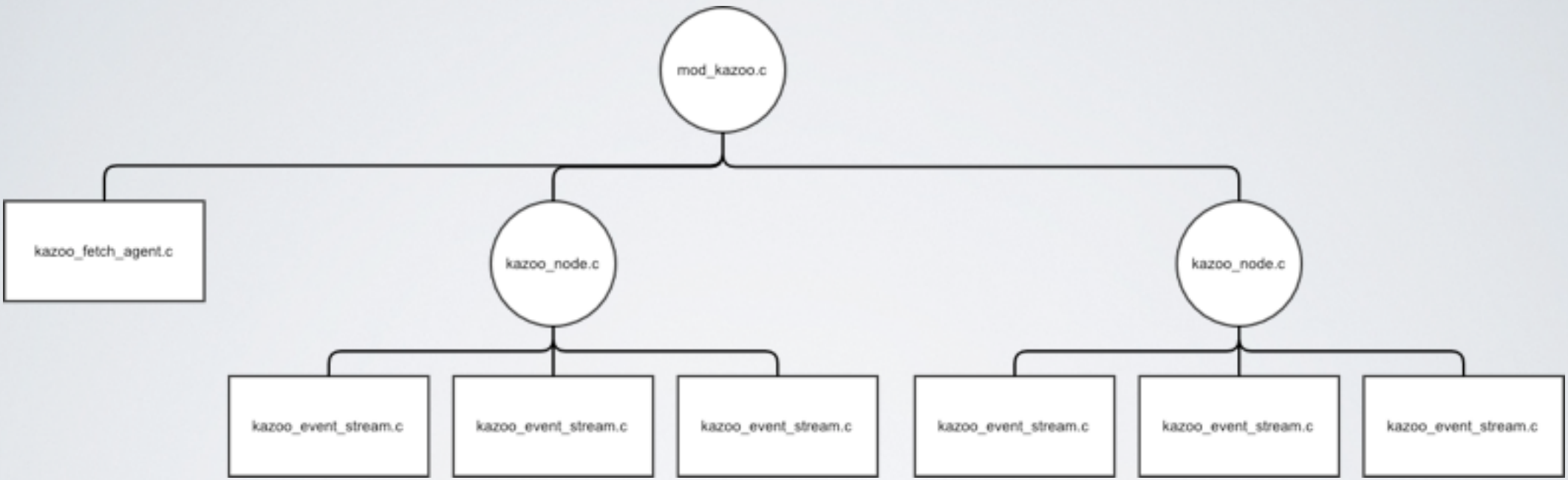
v2.0

If plan "A" didn't work
the alphabet still has 25 more letters

v3.0

If at first you don't succeed,
get a bigger hammer


ecallmgr
Distributed Erlang
FreeSWITCH
Requests
Execute
Execute
Execute
Responses
Events
Events


mod_kazoo.c
kazoo_fetch_agent.c
kazoo_node.c
kazoo_node.c
kazoo_event_stream.c
kazoo_event_stream.c
kazoo_event_stream.c
kazoo_event_stream.c
kazoo_event_stream.c
kazoo_event_stream.c

```
if (switch_queue_trypop(ei_node->send_msgs, &pop) == SWITCH_STATUS_SUCCESS) {
        ei_send_msg_t *send_msg = (ei_send_msg_t *) pop;
        ei_helper_send(ei_node, &send_msg->pid, &send_msg->buf);
        ei_x_free(&send_msg->buf);
        switch_safe_free(send_msg);
}
status = ei_xreceive_msg_tmo(ei_node->nodefd, &received_msg->msg, &received_msg->buf, 5);
switch (status) {
case ERL_TICK:
        break;
case ERL_MSG:
        if (switch_queue_trypush(ei_node->received_msgs, received_msg) != SWITCH_STATUS_SUCCESS) {
                ei_x_free(&received_msg->buf);
                switch_safe_free(received_msg);
        }
        received_msg = NULL;
        break;
case ERL_ERROR:
        switch (erl_errno) {
        case ETIMEDOUT:
        case EAGAIN:
                break;
        case EMSGSIZE:
                switch_clear_flag(ei_node, LFLAG_RUNNING);
                break;
        case EIO:
                switch_clear_flag(ei_node, LFLAG_RUNNING);
                break;
        default:
                if (status < 0) {
                        switch_clear_flag(ei_node, LFLAG_RUNNING);
                }
                break;
        }
        break;
default:
        if (status < 0) {
                switch_clear_flag(ei_node, LFLAG_RUNNING);
        }
        break;
}
```

```
gen_server:cast({'mod_kazoo', Node}, {'sendevent', EventName, Headers})
```

```
gen_server:call({'mod_kazoo', Node}, {'api', Cmd, Args}, Timeout)
```

```
if (switch_queue_pop_timeout(event_stream->queue, &pop, 500000) ==
SWITCH_STATUS_SUCCESS) {
        switch_event_t *event = (switch_event_t *) pop;

        if (event_stream->socket) {
                ei_x_buff ebuf;
                char high, low;
                switch_size_t size = 1;

                ei_x_new_with_version(&ebuf);
                ei_encode_switch_event(&ebuf, event);

                high = ebuf.index >> 8;
                low = ebuf.index & 0xFF;

                switch_socket_send(event_stream->socket, &high, &size);
                switch_socket_send(event_stream->socket, &low, &size);

                size = (switch_size_t)ebuf.index;
                switch_socket_send(event_stream->socket, ebuf.buff, &size);

                ei_x_free(&ebuf);
        }

        switch_event_destroy(&event);
}
```

**Nagle-less (TCP_NODELAY)*

```
gen_tcp:connect(IP, Port, [{'mode', 'binary'}, {'packet', 2}])
```

How to C-node

Step one – Listen

ei_connect_xinit

Step Two – Connect

ei_accept_tmo

Step Three – Communicate

ei_xreceive_msg

```
ei_decode_version(buf->buff, &buf->index, &version);

/* is_tuple(Buff) */
ei_get_type(buf->buff, &buf->index, &type, &size);
if (type != ERL_SMALL_TUPLE_EXT) {
        return SWITCH_STATUS_GENERR;
}

/* {_, _, _} = Buf */
ei_decode_tuple_header(buf->buff, &buf->index, &arity);
if (arity != 3) {
        return SWITCH_STATUS_GENERR;
};

/* {'$gen_call', _, _} = Buf */
if (ei_decode_atom_safe(buf->buff, &buf->index, atom)
        || strncmp(atom, "$gen_call", 9)) {
        return SWITCH_STATUS_GENERR;
}

/* {_, Sender, _}=Buff, is_tuple(Sender) */
ei_get_type(buf->buff, &buf->index, &type, &size);
if (type != ERL_SMALL_TUPLE_EXT) {
        return SWITCH_STATUS_GENERR;
}

/* {_, _}=Sender */
ei_decode_tuple_header(buf->buff, &buf->index, &arity);
if (arity != 2) {
        return SWITCH_STATUS_GENERR;
}
```

```
/* {Pid, Ref}=Sender */
if (ei_decode_pid(buf->buff, &buf->index, &send_msg->pid)
        || ei_decode_ref(buf->buff, &buf->index, &ref)) {
        return SWITCH_STATUS_GENERR;
}

/* {_, _, Request}=Buff, is_tuple(Request) */
ei_get_type(buf->buff, &buf->index, &type, &size);
if (type != ERL_SMALL_TUPLE_EXT) {
        return SWITCH_STATUS_GENERR;
}

/* {_, _}=Request */
ei_decode_tuple_header(buf->buff, &buf->index, &arity);
if (arity != 2) {
        return SWITCH_STATUS_GENERR;
}

/* {is_auth, _}=Request */
if (ei_decode_atom_safe(buf->buff, &buf->index, atom)
        || strncmp(atom, "is_auth", MAXATOMLEN)) {
        return SWITCH_STATUS_GENERR;
}
```

Step Three – Communicate

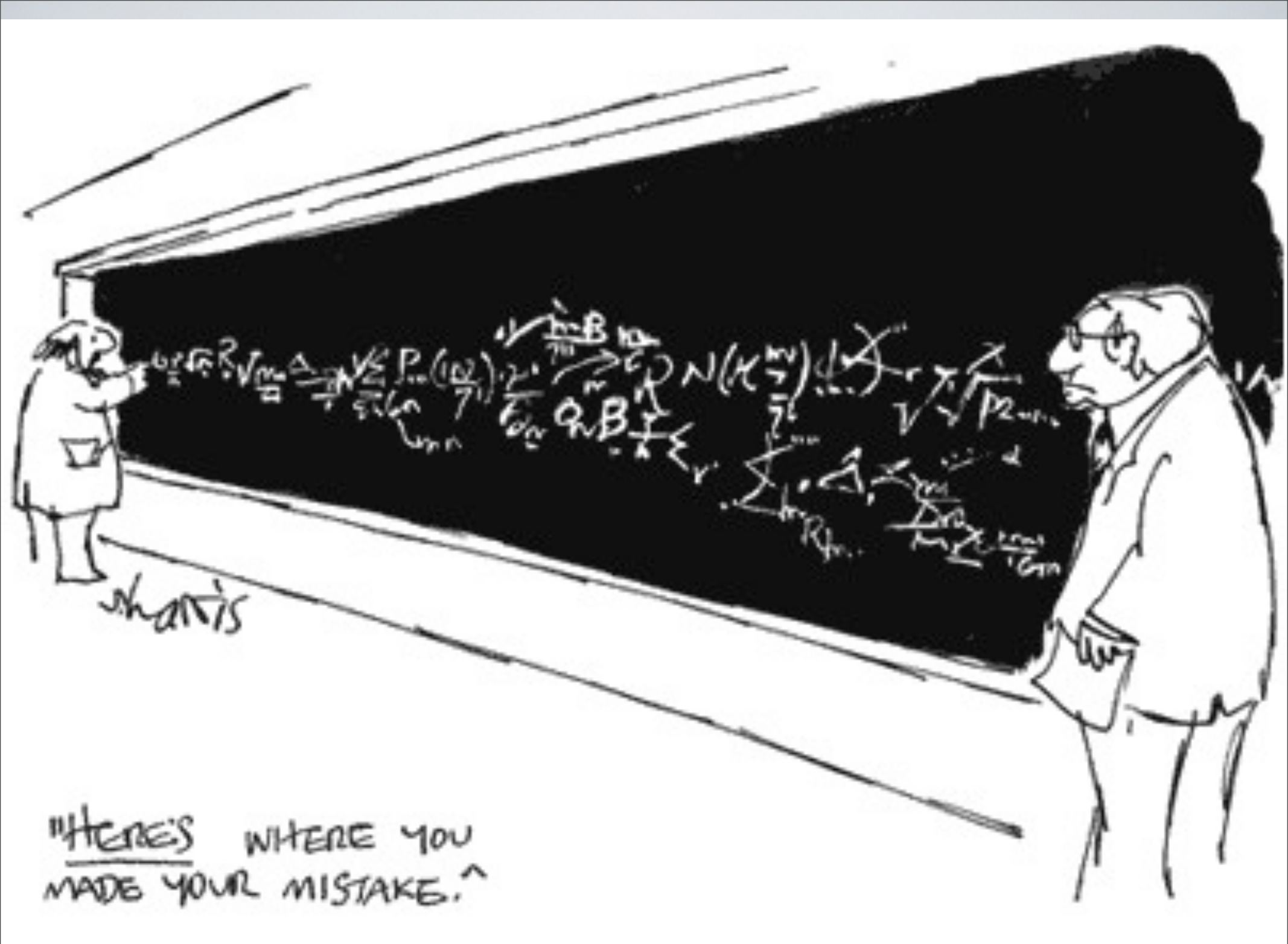
ei_send

```
ei_x_new(&send_msg->buf);

/* To ! {Tag, Reply} */
ei_x_new_with_version(&send_msg->buf);
ei_x_encode_tuple_header(&send_msg->buf, 2);
ei_x_encode_ref(&send_msg->buf, &ref);
ei_x_encode_atom(&send_msg->buf, "yes");
```

Is it good for the company

```
struct ei_event_stream_s {
    switch_memory_pool_t *pool;
    ei_event_binding_t *bindings;
    switch_queue_t *queue;
    switch_socket_t *acceptor;
    switch_pollset_t *pollset;
    switch_pollfd_t *pollfd;
    switch_socket_t *socket;
    switch_mutex_t *socket_mutex;
    switch_bool_t connected;
    char remote_ip[25];
    uint16_t remote_port;
    char local_ip[25];
    uint16_t local_port;
    erlang_pid pid;
    uint32_t flags;
    struct ei_event_stream_s *next;
};
typedef struct ei_event_stream_s ei_event_stream_t;
```

```
switch_status_t bind_fetch_agents() {
        bind_fetch_agent(SWITCH_XML_SECTION_CONFIG, &globals.config_fetch_binding);
        bind_fetch_agent(SWITCH_XML_SECTION_DIRECTORY, &globals.directory_fetch_binding);
        bind_fetch_agent(SWITCH_XML_SECTION_DIALPLAN, &globals.dialplan_fetch_binding);
        bind_fetch_agent(SWITCH_XML_SECTION_CHATPLAN, &globals.chatplan_fetch_binding);
        bind_fetch_agent(SWITCH_XML_SECTION_CHANNELS, &globals.channels_fetch_binding);

        return SWITCH_STATUS_SUCCESS;
}
```


"HERE'S WHERE YOU MADE YOUR MISTAKE."

```c
int ei_decode_string_or_binary_limited(char *buf, int *index, int maxsize, char *dst) {
        int type, size, res;
        long len;

        ei_get_type(buf, index, &type, &size);

        if (type != ERL_STRING_EXT && type != ERL_BINARY_EXT && type != ERL_NIL_EXT) {
                return -1;
        }

        if (size > maxsize) {
                return -1;
        }

        if (type == ERL_NIL_EXT) {
                res = 0;
                dst = '\0';
        } else if (type == ERL_BINARY_EXT) {
                res = ei_decode_binary(buf, index, dst, &len);
                dst[len] = '\0'; /* binaries aren't null terminated */
        } else {
                res = ei_decode_string(buf, index, dst);
        }

        return res;
}
```

```
for (i = 0, hp = event->headers; hp; hp = hp->next, i++);

if (event->body)
        i++;

ei_x_encode_list_header(ebuf, i + 1);

if (uuid) {
        char *unique_id = switch_event_get_header(event, "unique-id");
        ei_x_encode_binary(ebuf, unique_id, strlen(unique_id));
} else {
        ei_x_encode_atom(ebuf, "undefined");
}

for (hp = event->headers; hp; hp = hp->next) {
        ei_x_encode_tuple_header(ebuf, 2);
        ei_x_encode_binary(ebuf, hp->name, strlen(hp->name));
        switch_url_decode(hp->value);
        ei_x_encode_binary(ebuf, hp->value, strlen(hp->value));
}

if (event->body) {
        ei_x_encode_tuple_header(ebuf, 2);
        ei_x_encode_binary(ebuf, "body", strlen("body"));
        ei_x_encode_binary(ebuf, event->body, strlen(event->body));
}

ei_x_encode_empty_list(ebuf);
```

github.com/2600hz

www.2600hz.com