Asterisk as a Media Application Server

Matt Jordan
@mattcjordan
Director of Technology, Digium
Goal:
Can we make Asterisk a generic media application server?
Scalability
Traditional Deployments
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan

- Option 1: Each Asterisk server the same
Traditional Deployment: Option 1

Alice

Kamailio 1

Kamailio 2

Asterisk PBX

Asterisk PBX

Asterisk PBX

Bob
Traditional Deployment: Option 1

- Kamailio acts as Registrar
- Round Robin routing
- Multi-tenant
- Asterisk systems share configuration
Traditional Deployment: Option 1

- Kamailio acts as Registrar
- Round Robin routing
- Multi-tenant
- Asterisk systems share configuration
Problems with Option 1

- Sharing Configuration
  - Not easily scaled
  - All systems must know all information
  - Requires careful dialplan construction (func_odbc)
[customer_one]

exten => 1000,1,NoOp()
    same => n,ConfBridge(1000,c_one_profile)
    same => n,Hangup()

[customer_two]

exten => 1000,1,NoOp()
    same => n,ConfBridge(1000,c_two_profile)
    same => n,Hangup()}
[customer]

exten => 1000,1,NoOp()
same => n,ConfBridge(1000,$\{ODBC\_CONF\_PROF($\{CALLERID(num)\})\})
same => n,Hangup()
Traditional Deployment: Option 1

Alice Calls 1000 (Conference)
Traditional Deployment: Option 1

Bob Calls 1000 (Conference)
Traditional Deployment: Option 1

Asterisk PBX

Kamailio 1

Kamailio 2

Asterisk PBX

Asterisk PBX

Asterisk PBX

Alice

Bob

::-(

Creative Innovation – Customer Satisfaction – Continual Quality Improvement
Problems with Option 1

- Sharing Configuration
  - Not easily scaled
  - All systems must know all information
  - Requires careful dialplan construction (func_odbc)

- func_odbc: Still doesn't scale well!
  - Can defer customer logic to external system
  - Cannot easily defer routing/application logic
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan

- Option 1: Each Asterisk server the same
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan

- Option 1: Each Asterisk server the same
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan

- Option 1: Each Asterisk server the same

- Option 2: Special purpose Asterisk servers
Traditional Deployment: Option 2

- Kamailio acts as Registrar
- Route based on functional purpose (with round robin amongst those)
- Multi-tenant
- Asterisk systems share configuration
Traditional Deployment: Option 2

Alice

Kamailio 1

Kamailio 2

Asterisk Conferencing

Asterisk PBX

Bob
Traditional Deployment: Option 2

Alice

Bob

Kamailio 1

Kamailio 2

Asterisk PBX

Asterisk PBX

Asterisk PBX

Asterisk Conferencing

But what if I need another Conferencing Server?
Traditional Deployment: Option 2

- Alice
- Bob
- Kamailio 1
- Kamailio 2
- Asterisk PBX
- Asterisk Conferencing
- Asterisk PBX

::-(
Traditional Deployments

- Asterisk front-ended with Kamailio
  - Kamailio acts as Registrar, provides Location
  - Asterisk provides media services

- Use Traditional Asterisk Dialplan

- Option 1: Each Asterisk server the same

- Option 2: Special purpose Asterisk servers
Problems with Option 1 and Option 2

- **Sharing Configuration**
  - Not easily scaled
  - All systems must know all information
  - Requires careful dialplan construction (func_odbc)

- **func_odbc**: Still doesn't scale well!
  - Can defer customer logic to external system
  - Cannot easily defer application logic

- **Functional systems only partially mitigate the problem**
  - Application logic still affects routing decisions
  - Impacts how easy we can scale
Problems with Tradition

- Optimal routing should not require application logic: Not Kamailio's job
Problems with Tradition

- Optimal routing should not require application logic: Not Kamailio's job
- Application Logic impacts routing
Problems with Tradition

- Optimal routing should not require application logic: Not Kamailio's job
- Application Logic impacts routing
- Ideal situation
  - Every instance of Asterisk is generic
  - Kamailio just routes based on performance
Goal:
Can we make Asterisk a generic media application server?
Remove the application logic from Asterisk
ARI: An API for building custom communications applications
ARI: A Super Quick Overview

- A REST(ful) API
  - Exposes the raw Asterisk primitives as resources
A REST(ful) API
- Exposes the raw Asterisk primitives as resources
  POST /channels/12345/answer
  DELETE /bridges/awesome_bridge
  PUT /deviceStates/my_dev/state=BUSY
ARI: A Super Quick Overview

- A REST(ful) API
  - Exposes the raw Asterisk primitives as resources
    POST /channels/12345/answer
    DELETE /bridges/awesome_bridge
    PUT /deviceStates/my_dev/state=BUSY

- JSON Events over WS
  
  ```json
  {   event: 'ChannelHangup',
      channel: { 'id': '12345',
                   'name': 'PJSIP/alice'
      }
  }
  ```

- A tiny bit of dialplan
  
  ```
  exten => _XXXX,1,Stasis(your-app)
  ```
**ARI: A Super Quick Overview**

- **Stasis dialplan app**
- **Asterisk Dialplan Execution**
- **ARI External Execution**
import ari

client = ari.connect('http://localhost:8088', 'ari_user', 's3cr3t')

conf_bridge = client.bridges.create(type='mixing,dtmf_events', bridgeId='awesome_conf')

def stasis_start_cb(channels, ev):
    channel = channels.get('channel')
    channel.answer()

    conf_bridge.play(media='sound:beep')
    conf_bridge.addChannel(channel=channel.id)

client.on_channel_event('StasisStart', stasis_start_cb)

client.run(apps='awesome_conference')
ARI: An API for building custom communications applications

(THAT'S THE DIALPLAN)
Nontraditional Deployment
Step 1: Remove the Dialplan*

* Conspiracy Theorists Rejoice
[default]

exten => _X.,1,NoOp()
same => n,Stasis(EVERYTHING)
same => n,Hangup()
Step 2: Use a Message Bus Approach
def get_app_by_exten(exten):
    """This is terrible, but demos the concept"""
    if exten == '1000':
        return exec_conference
    else:
        return default_app_exec

def stasis_start_cb(channel, ev):
    exten = ev.get('exten')
    app = get_app_by_exten(exten)
    app(channel)

    client.on_channel_event('StasisStart',
                           stasis_start_cb)
Getting a bit more extreme

Alice

Bob

Kamailio 1

Kamailio 2

SIP

Asterisk 1

Asterisk 2

ARI

App Server

App Server

Asterisk n
Getting a bit more extreme

Treat as Scalable Microservices
A philosophical taxonomy

- Keep things as simple as possible, but no simpler
- Kamailio: manage SIP
- Asterisk: manage media
- Application logic: your choice
Questions