Internet Multimedia Conferencing: What now?

Jonathan Rosenberg
Bell Laboratories
VoN Spring 99
Whats next?

• Service Creation
  – SIP CGI
  – Call Processing Language

• Features
  – Caller handling directives
  – Call control
Service Creation

- SIP can do lots of neat services
  - call redirect web
  - web IVR
  - email call logs
  - follow-me
  - distributed home line emulation

- How to program these services?
Who Creates Services?

- Lots of people
  - service providers
  - third parties
  - end users

- Security Requirements
  - End users shouldn’t access resources, crash servers
  - CGI model dangerous!
Some Requirements

• Rapid Development
  – easy services = easy program
• Rapid Deployment
  – Servers stay up
  – No code rebuilds
• Cross Platform
  – Vendor, possibly protocol

• Easy Learning
  – Little protocol expertise
  – Basic programming expertise
How’s it done on the web?

- CGI = Common Gateway Interface
  - Popular web dynamic content tool
  - Request arrives at server
  - server spawns process
  - request info sent to process
  - process returns web page
SIP CGI

• SIP and HTTP are close cousins
• SIP CGI nearly same
  – Allow for “persistent” scripts
  – Allow script to proxy and return responses

Benefits
  – ANY programming language
  – Loose coupling with server
    • separate process
    • IPC by file descriptors
    • no rebuilds or out of service
  – Little SIP understanding needed
Example perl Script

- Provides “call forward unconditional service”
- Uses database to store forwarding lists
- Returns error if user not in database
- only 19 lines of perl!!

```perl
#!/usr/bin/perl -w

use DB_File;

sub fail {
  my($status, $reason) = @_;    print "SIP/2.0 $status $reason\n\n";
  exit 0;
}

tie %addresses, 'DB_File', 'addresses.db'  or fail("500", "Address database failure");

$to = $ENV{'HTTP_TO'};

if (! defined( $to )) {
  fail("400", "Missing Recipient");
}

$destination = $addresses{$to};

if (! defined( $destination )) {
  fail("404", "No such user");
}

print "CGI-PROXY-REQUEST-TO $destination SIP/2.0\n";
print "CGI-Reexecute-On: never\n";

untie %addresses; # Close db file
```
Another solution

- CGI is unlimited in flexibility
  - ideal for service providers, third parties
  - not ideal for end users

- Want restrictions for end users
  - limited access to resources
  - limited services
  - guarantees on correctness
  - bounds on execution time
  - automated verification
Call Processing Language

- A scripting language to describe call services
- Language properties guarantee safety
- Based on XML
- Hand or tool authoring
- Example: “call forward on busy/no-answer”
- Under development in IETF

```xml
<call>
  <location url="sip:jones@pc.ex.com">
    <proxy timeout="8s">
      <busy>
        <location url="sip:jones@vmail.ex.com" merge="clear" id="voicemail">
          <proxy />
        </location>
      </busy>
      <noanswer>
        <link ref="voicemail" />
      </noanswer>
    </proxy>
  </location>
</call>
```
How is it used?

- User authors script
  - tools or by hand
- User uploads script to server
  - SIP, HTTP, FTP, email
- Someone calls user
- Server executes script
- Script rings phone at work
- Call accepted
Who controls the service?

- Three parties interested in a call
  - caller
  - callee
  - administrators
- CPL and CGI used for callee/administrator logic
- How about caller logic?
SIP Caller Preferences

- SIP Extensions for specifying caller preferences
- Preferences carried in INVITE setup message
- Preferences for
  - reaching callee at home or work
  - fax, video, audio call
  - mobile or landline
  - secretary or voicemail
  - priority locations

Preference: video
Additional Preferences

• Caller can also direct server features
  – proxy or redirect
  – reach all, reach first
  – call queue

• Challenge
  – Merging all these preferences
  – Feature interaction
  – Differentiator!
New SIP Features

- SIP’s current features are mostly for call initiation
- Need features for post-call setup
  - Transfer
  - Multi-party
  - Dial-in bridges

Requirements
- Features can be implemented anywhere
  - end systems alone
  - central servers
- Don’t specify feature - specify primitive
- Primitive example
  - Add call leg
Not your ordinary transfer...

- Internet is not PSTN
- End systems can be smart
- Give them information
  - All parties know what service is invoked
- Give them a choice
  - Any party can refuse or accept service
- Give them security
  - Parties can be sure of identity of others involved in service
- Dumb endpoints can ignore these
  - don’t tell user about service
  - accept service
  - ignore security
E2E Call Transfer

• A transfers B to C
• What's new?
  – C knows A did transfer
  – C can accept/reject transfer
  – B knows it's being transferred to C
  – B can accept/reject transfer
Net-Provided Transfer

- Same service - A transfers B to C
- Proxy steps in
- A does same thing
- Service looks like traditional transfer now!
Conclusion

- CGI and CPL for programming services
- Caller preferences
- Call control