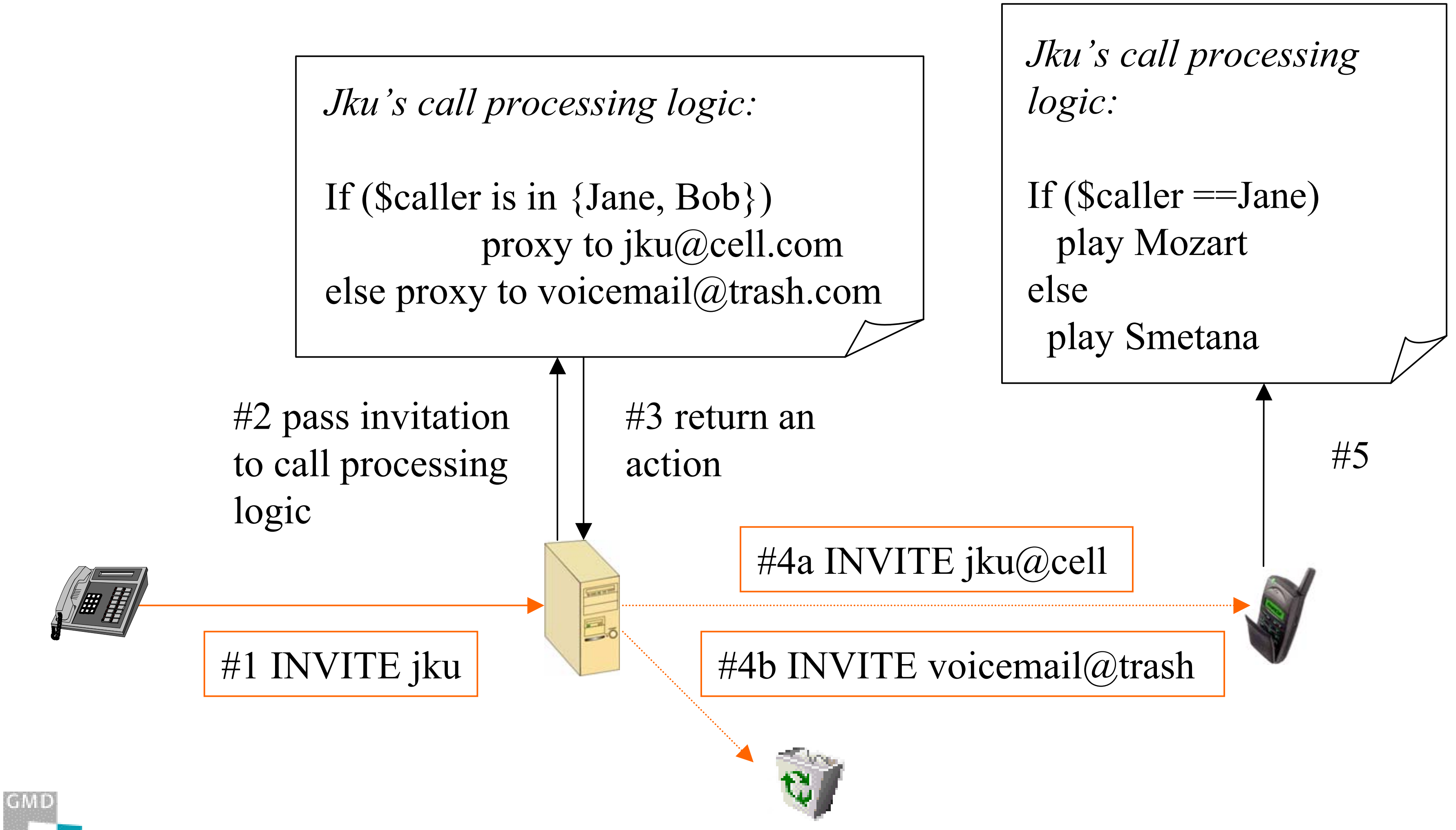
Programming SIP (Sisalem et.al)

- Examples
  - “discard all calls from Monica during my business hours”
  - “redirect authenticated friends to my cell phone, anyone else to my secretary”
  - “if busy, return my homepage and redirect to recorder”
- Users and third parties may program
- SIP follows HTTP programming model
- Mechanisms suggested in IETF: CGI, Call Processing Languahge (CPL), Servlets

Call Processing Logic Example



Jku’s call processing logic:

```
If ($caller is in {Jane, Bob})
            proxy to jku@cell.com
else proxy to voicemail@trash.com
```

Jku’s call processing logic:

```
If ($caller ==Jane)
   play Mozart
else
  play Smetana
```

#2 pass invitation to call processing logic

#3 return an action

#5

#4a INVITE jku@cell

#1 INVITE jku

#4b INVITE voicemail@trash

Where May Signaling Services Live?

- Some services have to live in the network:
  - call distribution
  - services for dial-up users without always-on IP connectivity
- Some services can be implemented in both places:
  - forward on busy
- Some services work best in end-devices:
  - distinctive ringing

Service Location Examples

Feature	End-device	Proxy	Network w/media
Distinctive Ringing	Yes	Can assist	Can assist
Visual call id	Yes	Can assist	Can assist
Call Waiting	Yes	No	Yes
CF Busy	Yes	Yes	Yes
CF No Answer	Yes	Yes	Yes
CF No Device	No	Yes	Yes
Location hiding	No	Yes	Yes
Transfer	Yes	No	Yes
Conference Bridge	Yes	No	Yes
Gateway to PSTN	Yes	No	Yes
Firewall Control	No	No	Yes
Voicemail	Yes	No	Yes

Source: H. Schulzrinne: "Industrial Strength IP Telephony"

CGI

- Follows Web-CGI. Unlike Web-CGI, SIP-CGI supports proxying and processes responses as well.
- Language-indpendent (Perl, C, ...)
- Communicates through input/output and environment variables.
- CGI programs unlimited in their power. Drawback: Buggy scripts may affect server easily.
- Token is passed between SIP server and CGI to keep state across requests and related responses.

Call Processing Language

- Special-purpose call processing language.
- May be used by both SIP and H.323 servers.
- Target scenario: users determine call processing logic executed at a server.
- Limited languages scope makes sure server’s security will not get compromised.
- Portability allows users to move CPL scripts across servers.
- Scripts may be manually written, generated using convenient GUI tools, supplied by 3rd parties, ...

CPL Example

```
<incoming>
   <address-switch field="origin" subfield="host">
        <address subdomain-of="example.com">
        <location url="sip:jones@example.com">
        <proxy timeout="10">
                <busy> <sub ref="voicemail" /> </busy>
                <noanswer> <sub ref="voicemail" /> </noanswer>
                <failure> <sub ref="voicemail" /> </failure>
        </proxy>
        </location>
    </address>
    <otherwise>
        <sub ref="voicemail" />
    </otherwise>
   </address-switch>
 </incoming>
```

- Actions may include redirection, proxy, rejection

Java Servlets

- Compromise between security and power: still a powerful generic language but security provided by Java “sand-box”.
- Well-defined API is needed. As APIs are not IETF’s business this work moved to JAIN.
- JAIN thought to be a generic API applicable to almost any signaling (SIP, H.323, PSTN, etc.)
- http://java.sun.com/products/jain/index.html

Call Processing Tradeoffs

- Generality versus security
  - multipurpose programming languages provide a huge service space
  - but also a huge vulnerability space
- Performance versus portability
  - portable languages (CPL) need to be interpreted
    - higher processing delay
  - portability needed if services deployed at multiple servers or end-devices (e.g. if stored at USIMs)
- Recommendation
  - choice of appropriate service creation mechanism depends on deployment scenario, i.e. where the service is executed and by whom the service is maintained