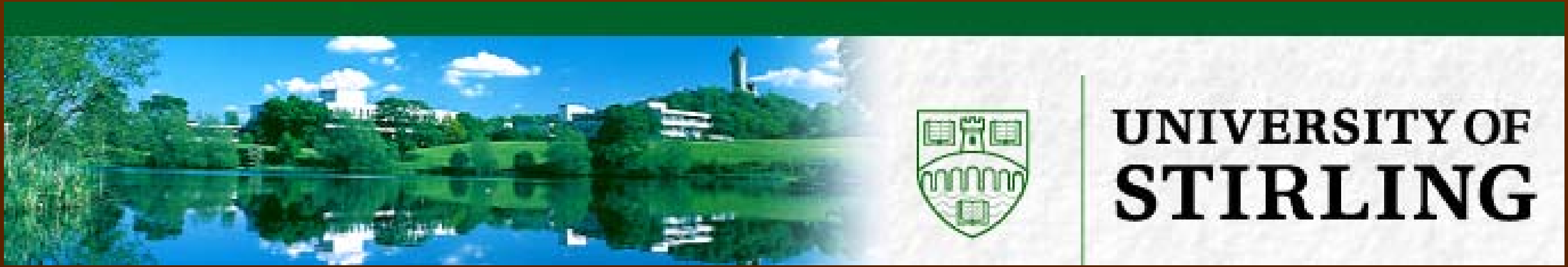
New Voice Services

© Kenneth J. Turner

(with SIP material © Mario Kolberg)

10th June 2003



Overview

- SIP (Session Initiation Protocol)
  - SIP Overview
  - SDP Protocol
  - SIP Services
  - SIP Service Research
- VoiceXML (Voice Extended Markup Language)
  - VoiceXML Overview
  - VoiceXML Elements
  - VoiceXML Forms
  - VoiceXML Service Research
  - VoiceXML Demonstration

SIP Overview

Goals of SIP

- maintains multimedia sessions
- modifies existing multimedia sessions:
  - changing participants
  - changing media
- other capabilities:
  - event notification
  - presence and instant messaging
- deals with signalling, not data transfer:
  - independent of session type
  - conveys session description

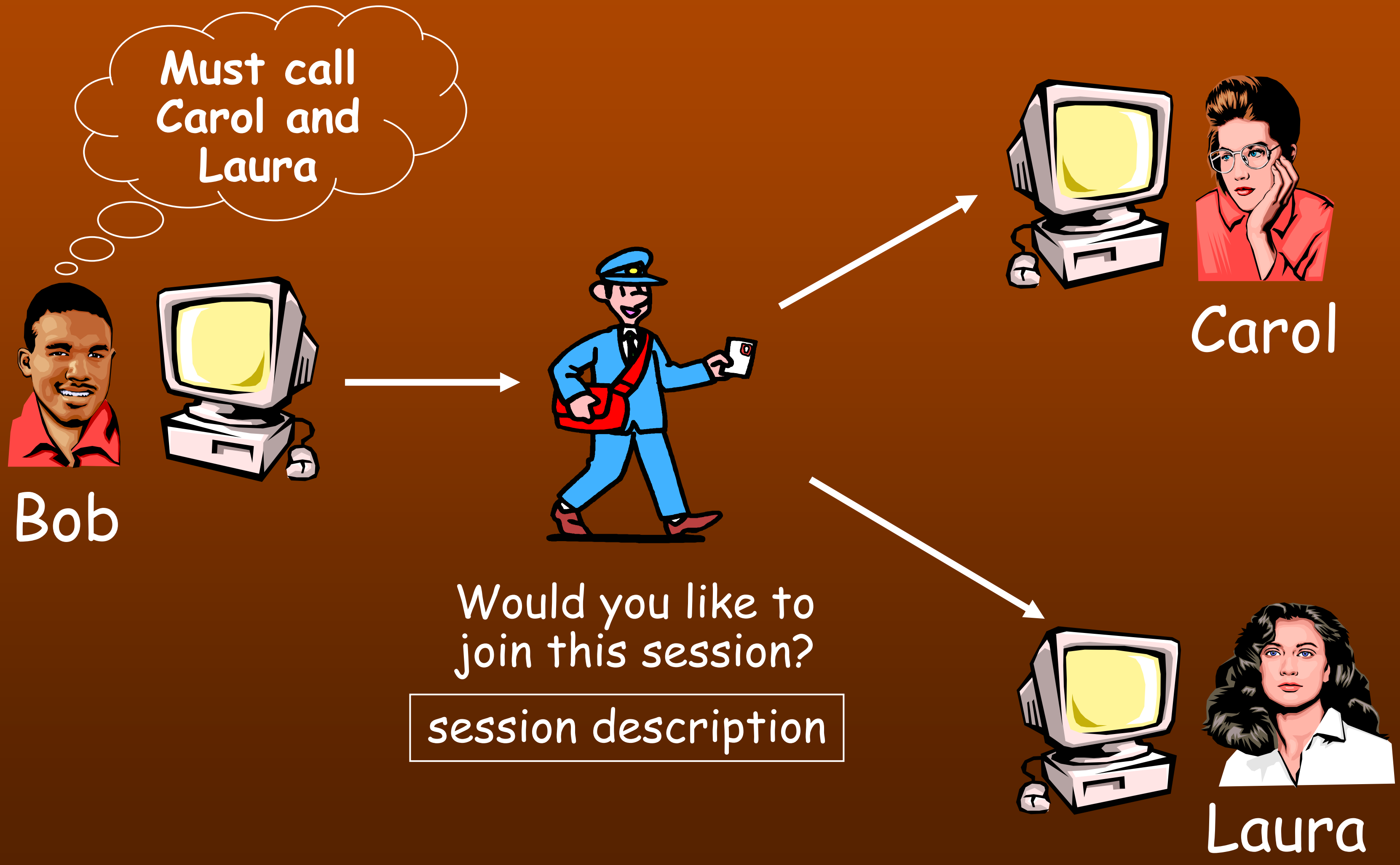
SIP Characteristics

- follows Internet philosophy:
  - small protocol building blocks
  - functionality in endpoints
  - open, neutral standards
  - multi-platform implementations
  - scalable and extensible
- traditional telephony differs:
  - heavy-weight, well-regulated standards
  - controlled by a small number of telcos
  - intelligence in the network nodes
  - restricted implementations

Commercial Aspects of SIP

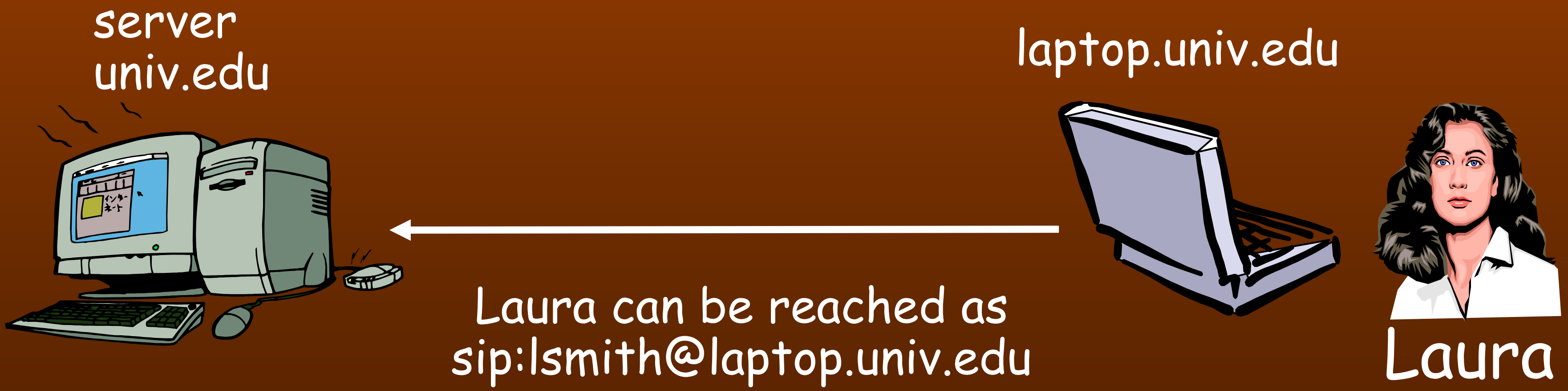
- intended for multimedia sessions
- main uptake seems to be for VoIP (Voice over Internet Protocol):
  - Internet telephony
  - SIP phones
  - 3G mobile communication
- in direct competition with the established H.323 protocol stack
- attractive due to its flexibility, e.g. Presence and IM (Instant Messaging)

A SIP Session


Must call Carol and Laura
Bob
Carol
Would you like to join this session?
session description
Laura

User Mobility

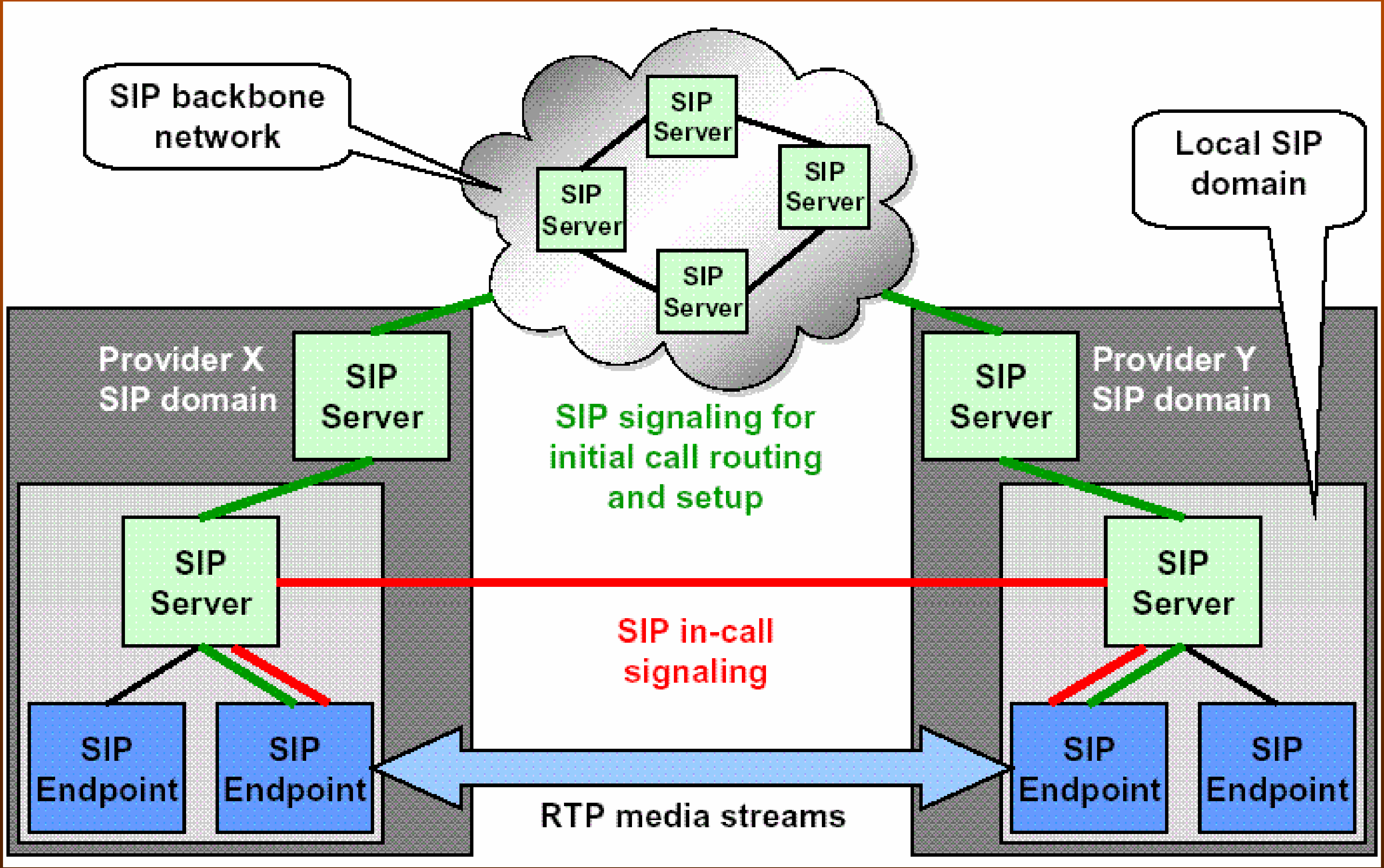
- users may be reached at different times on different systems, perhaps simultaneously
- users therefore register with SIP, and are contacted via a SIP server



SIP Elements

- inspired by HTTP client-server approach:
  - requests and responses
  - clients, servers, proxies
  - plain-text messages
  - URIs (Universal Resource Identifiers)
- SIP entities:
  - user agents
  - proxy servers
  - redirect servers
  - registrars
  - (location servers)

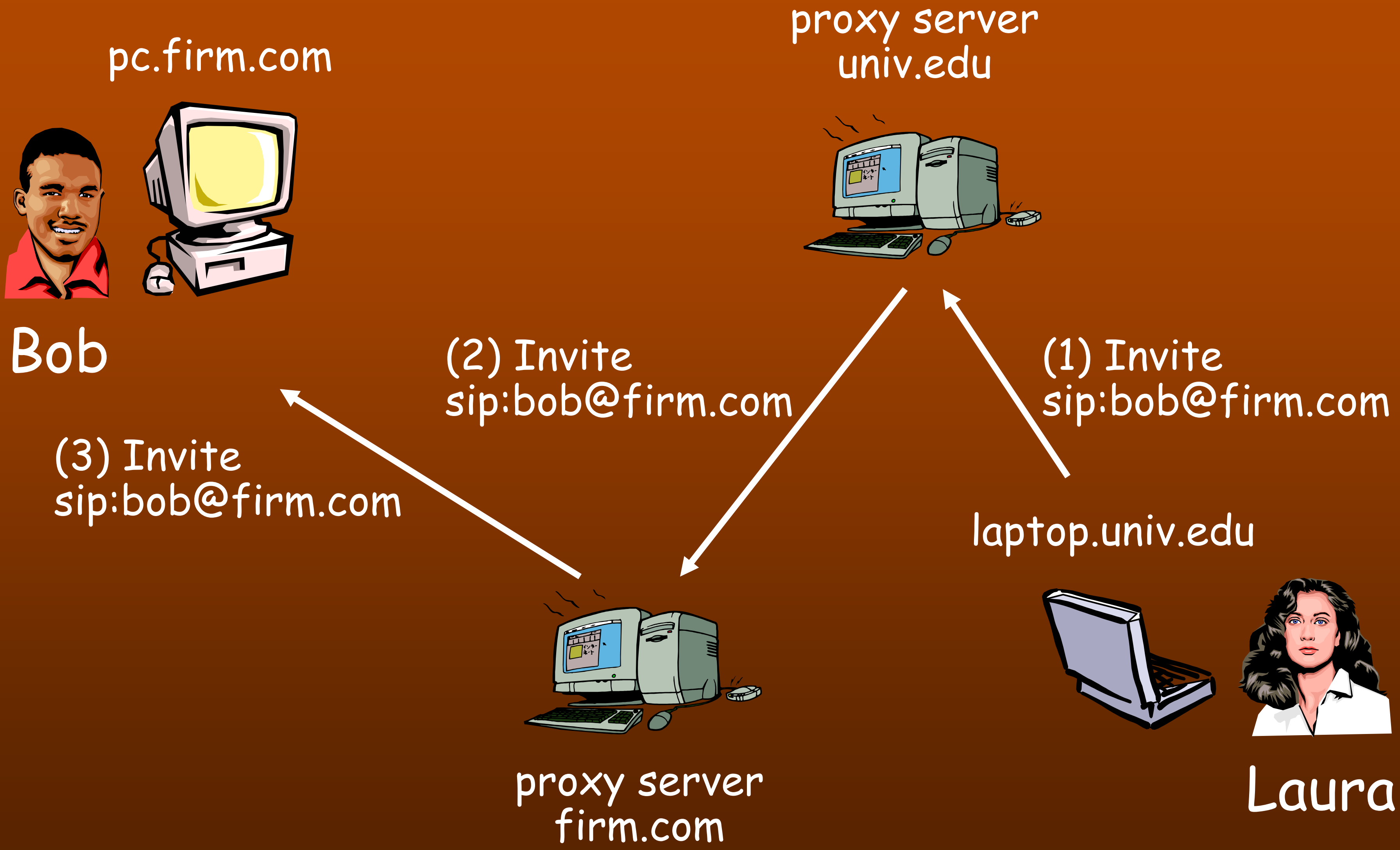
Overall SIP Organisation



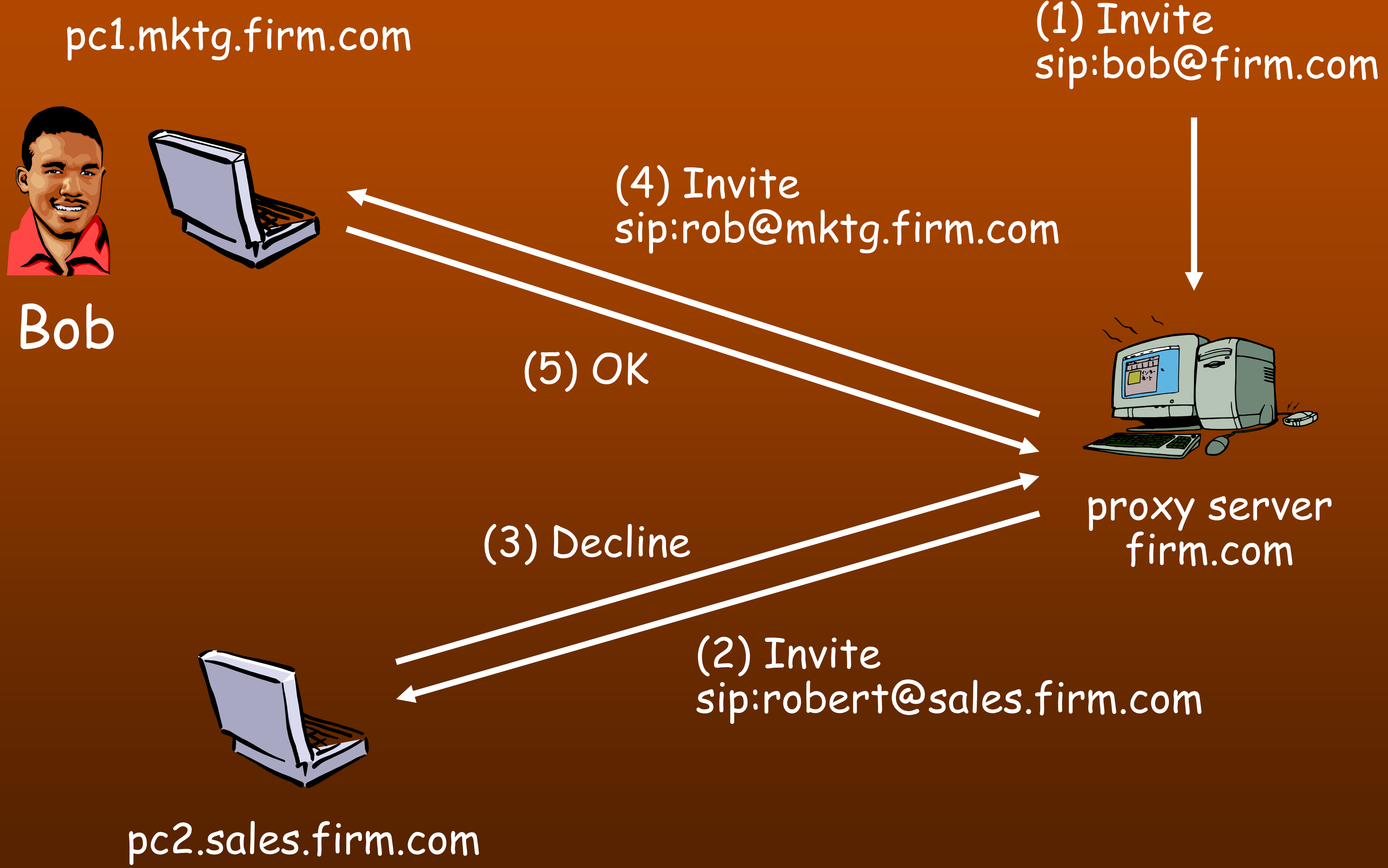
User Agent

- the entity that supports the SIP user:
  - handles incoming/outgoing session requests
  - User Agent Client (UAC) — user side
  - User Agent Server (UAS) — protocol side
- user interface:
  - special computer program (softphone)
  - SIP phone
  - SIP-enabled device (e.g. appliance, PDA)
  - might not interact with user directly (e.g. automated announcement)

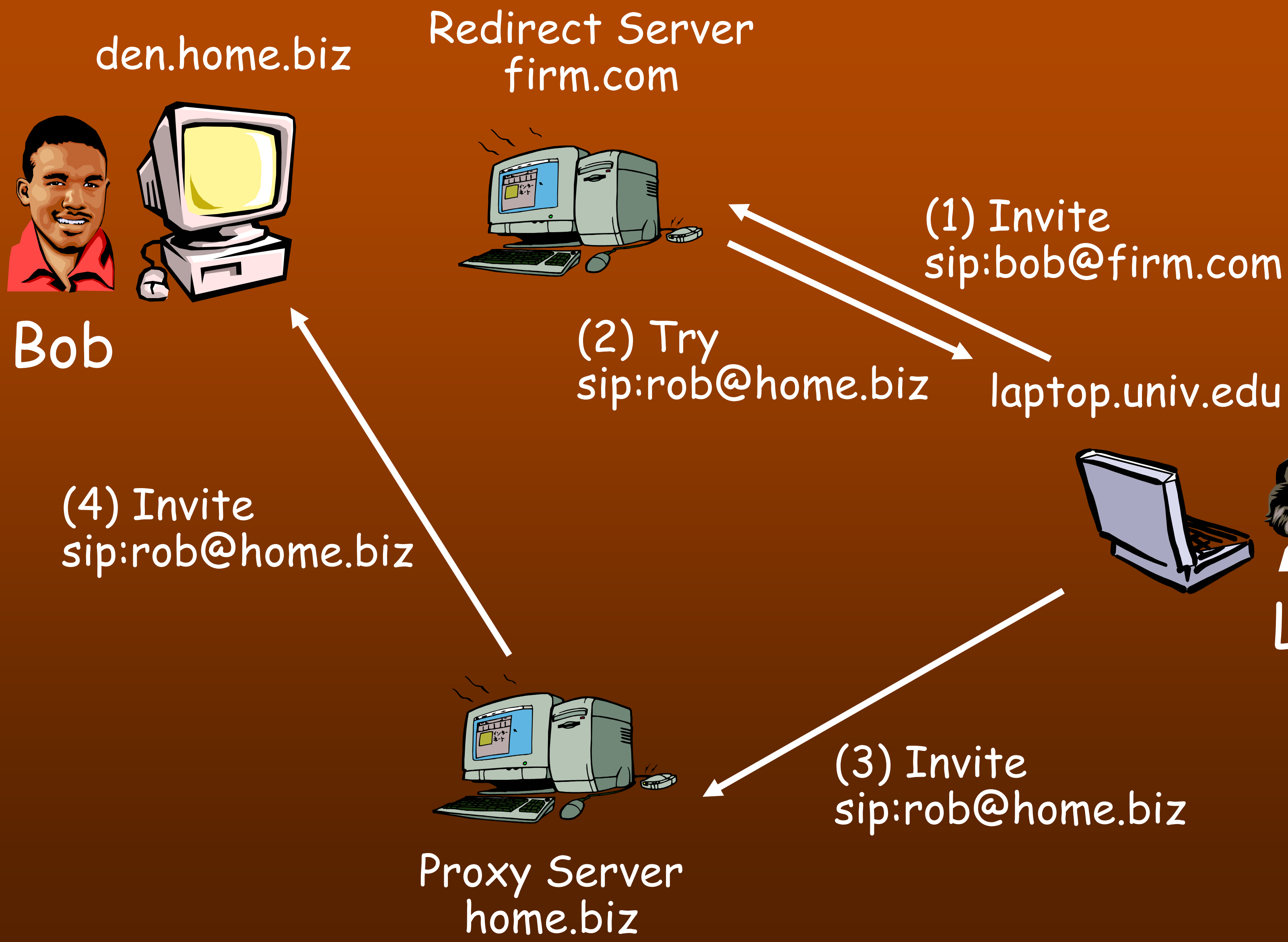
Proxy Server



Forking Proxy Server



Redirect Server



Registrar, Location Server

- Registrar:
  - accepts registrations
  - usually co-located with proxy server or redirect server
- Location Server:
  - strictly, outside SIP
  - stores and returns possible locations for users
  - uses registrars or other databases
  - might use LDAP (Lightweight Directory Access Protocol)

SIP Protocol

SIP Requests

- six request ‘methods’ form the core of SIP:
  - INVITE
  - ACK
  - CANCEL
  - BYE
  - REGISTER
  - OPTIONS
- other methods include functions for:
  - event notification
  - Quality of Service notification
  - messaging

Request Format

- a request has the form:
  - request line
  - header lines
  - blank line
  - message body (if any)
- the request line gives the method, URI and protocol version:

```
INVITE sip:bob@firm.com SIP/2.0
```

SIP Responses

- a request triggers one or more responses
- response codes are numeric, with a natural language explanation:

1XX	information	(provisional)
2XX	success	(final)
3XX	redirection	(final)
4XX	client error	(final)
5XX	server error	(final)
6XX	global failure	(final)

Response Format

- a response has the form:
  - status line
  - header lines
  - blank line
  - message body (if any)
- the status line gives the protocol version, status code and explanation:

```
SIP/2.0  180  Ringing
```

- provisional responses are not sent reliably
- final responses are sent reliably

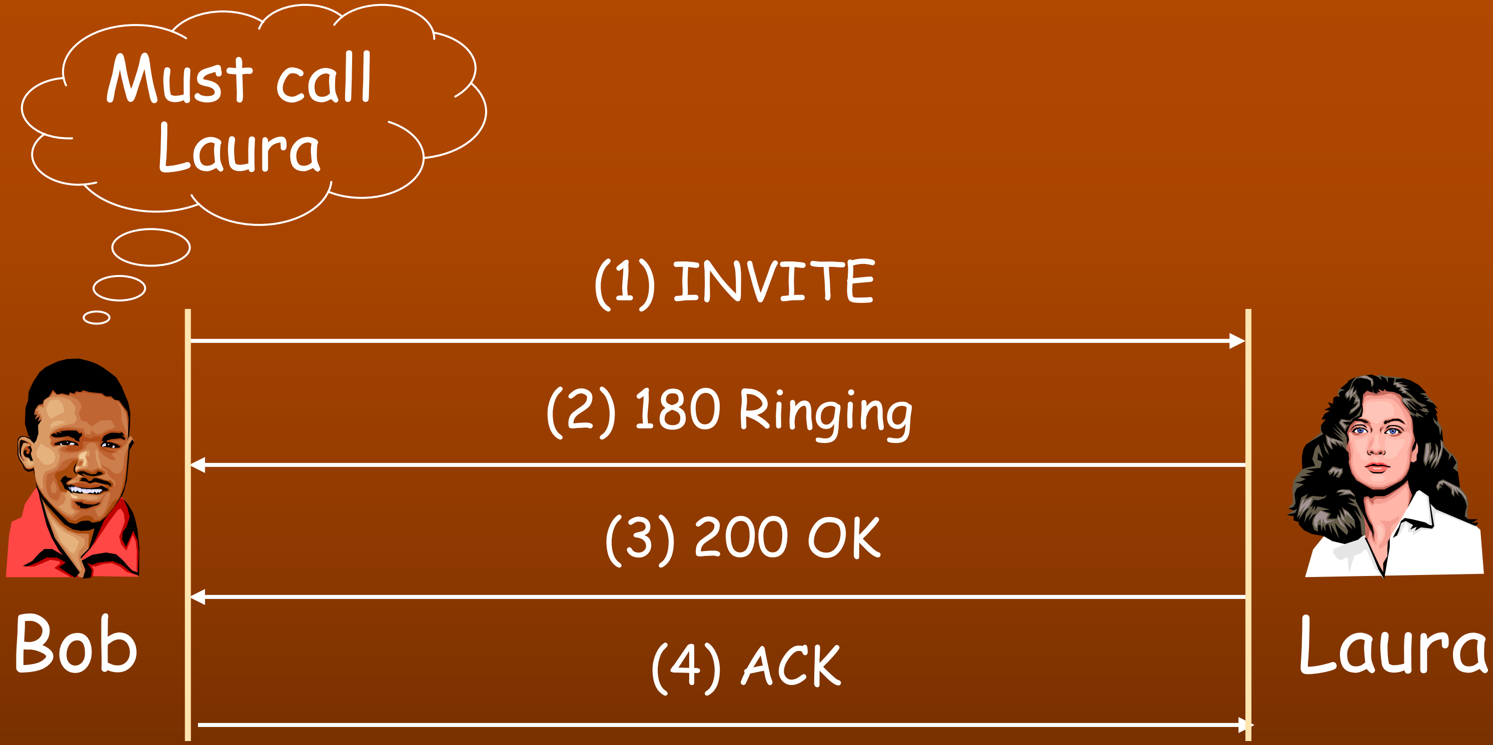
SIP Headers

- a header line has the form:

  name : value

- examples:
  - Call-Id identifies a session uniquely
  - Contact provides a direct URI for the user
  - CSeq gives the command sequence for ordering
  - From gives a URI for the session initiator
  - To gives a URI for the session invitee
  - Via records addresses of proxies on the route

INVITE, ACK

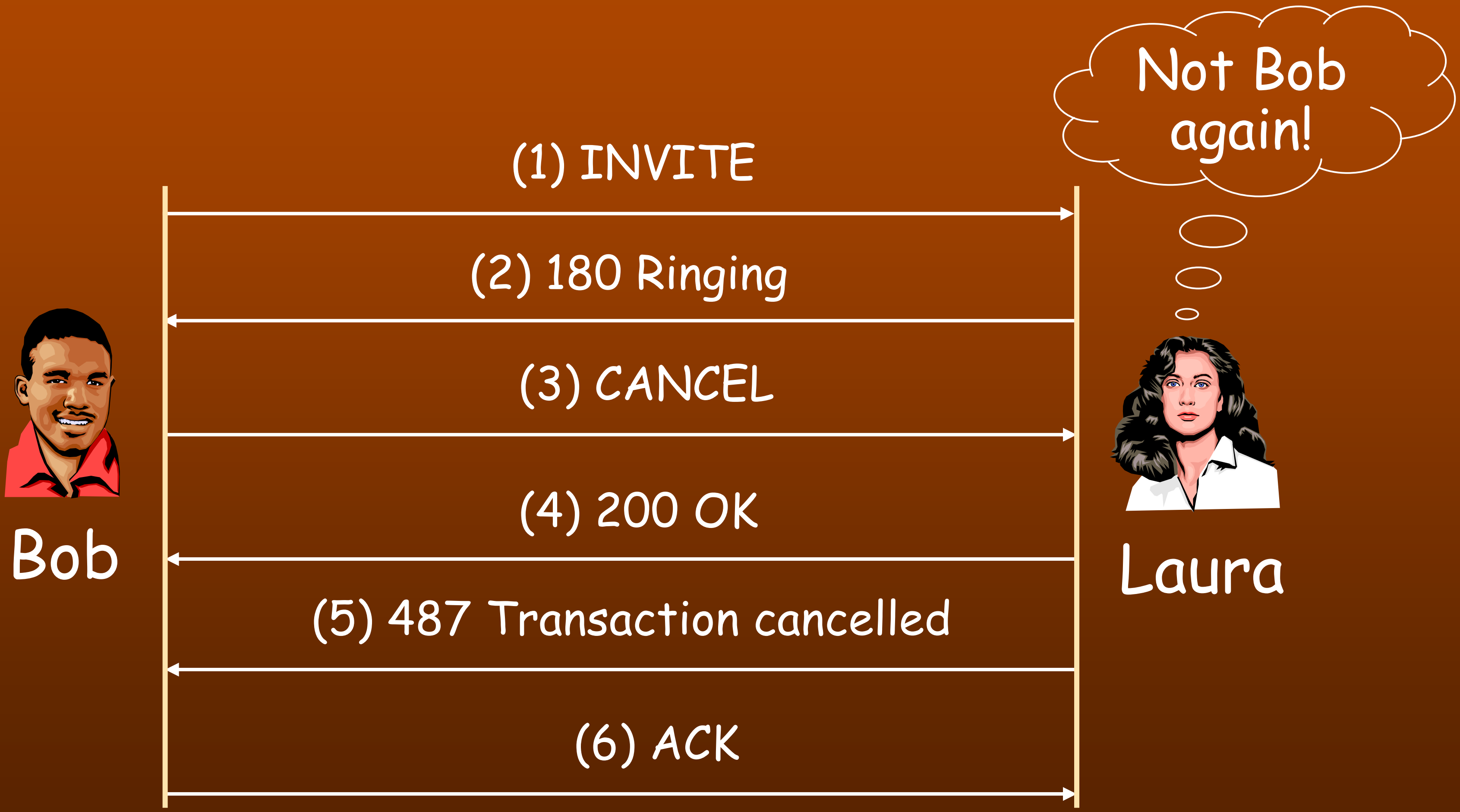


session description:

```
o=Bob 28908 42807 IN IP4 131.160.1.112
s=We have to talk
c=IN IP4 131.160.1.112
t=0 0
m=audio 49170 RTP/AVP 0
```

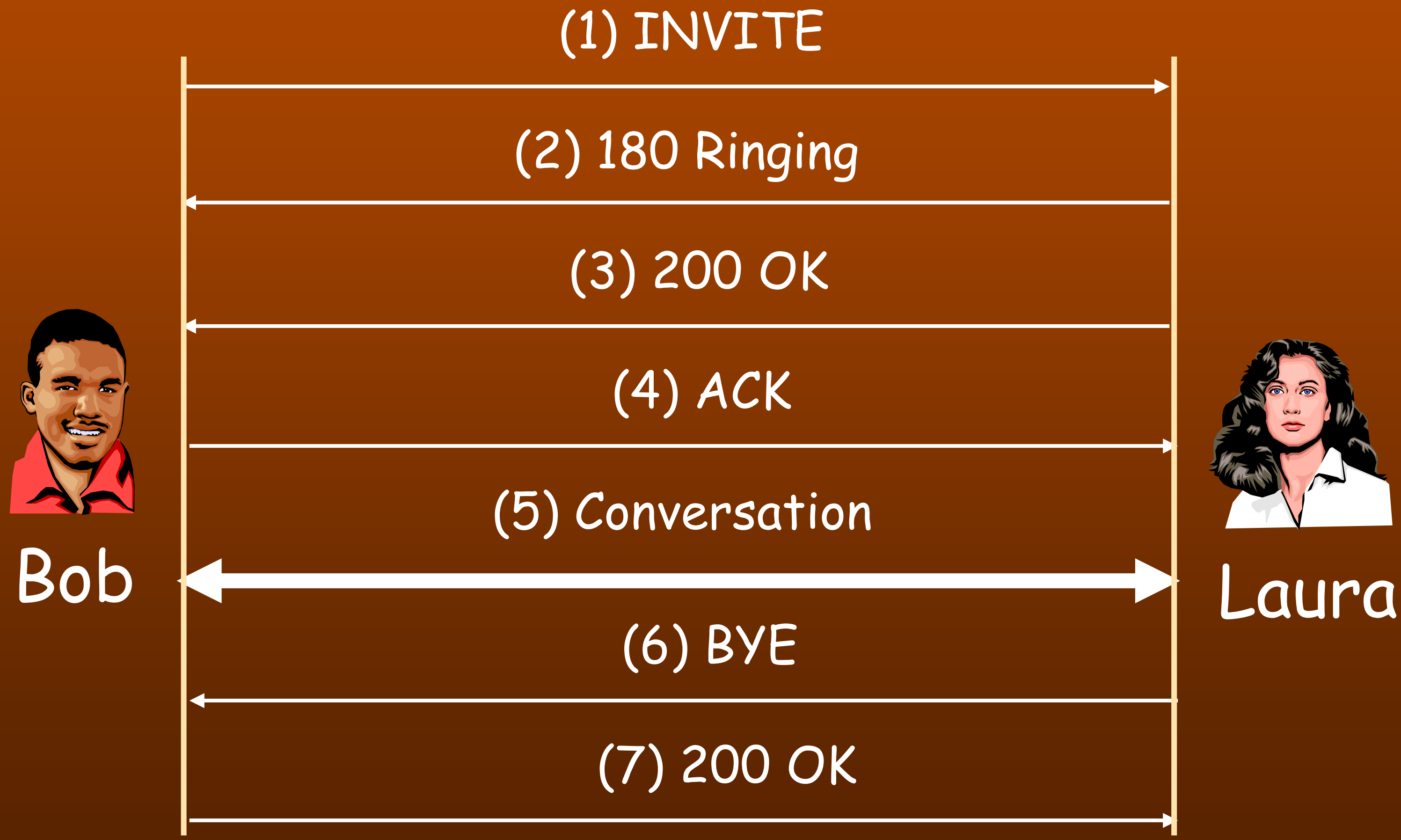
CANCEL

- cancel a request:



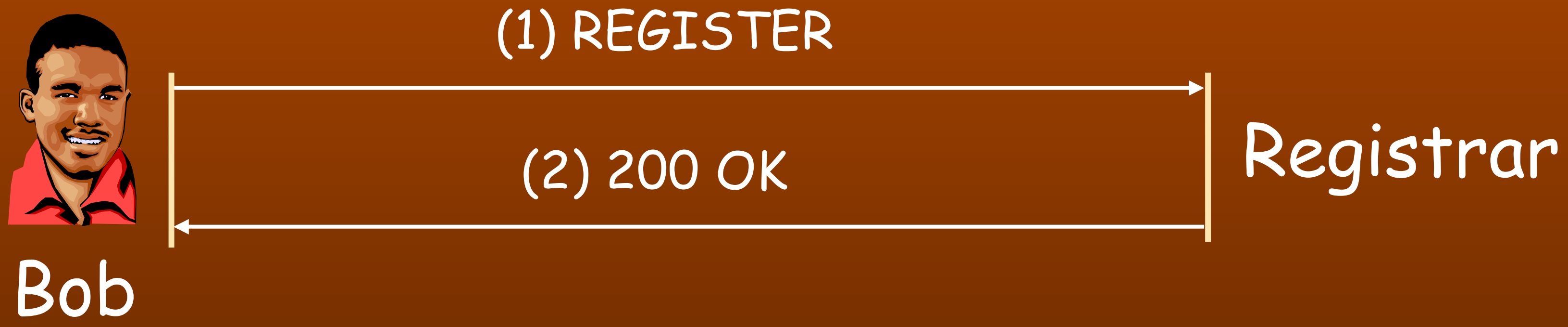
BYE

- close a session:



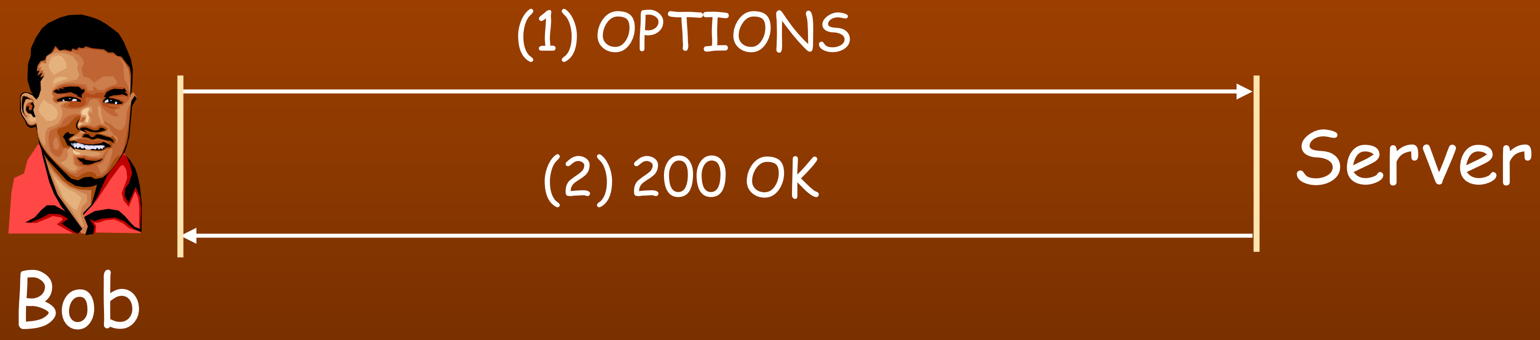
REGISTER

- register user location (and script):

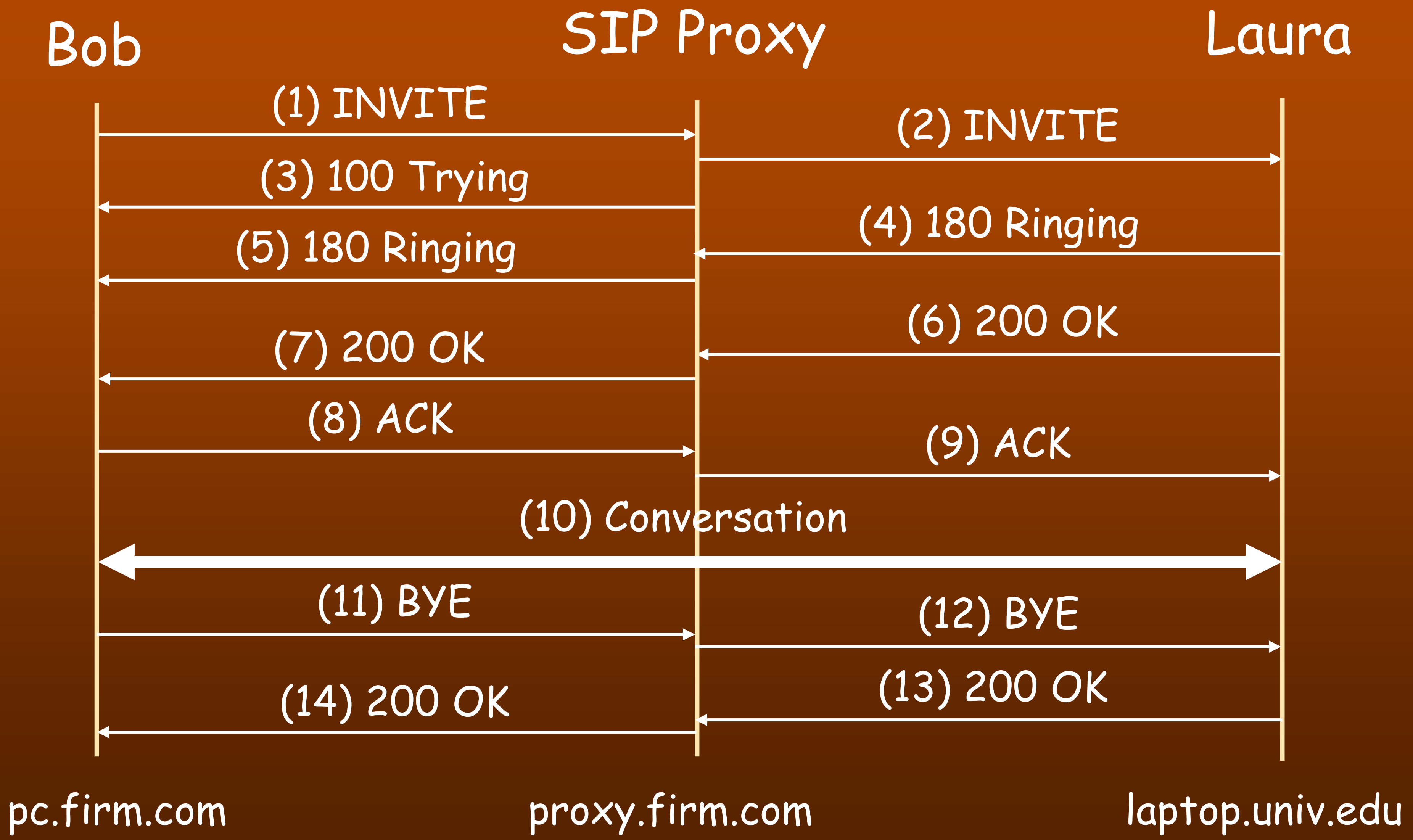


OPTIONS

- query SIP capabilities:



A Complete Session



Session Description

- SDP (Session Description Protocol) is carried by SIP as data on INVITE/Response
- SDP describes a session and its media:
  - session subject
  - times the session is active
  - connection information
  - origin address and port number
  - session media
- SDP is descriptive, not really a protocol

SDP example

- origin (name, Ids, Internet address):
  o=Bob 28908 42807 IN IP4 131.160.1.112
- subject:
  s=We have to talk
- connection (Internet address):
  c=IN IP4 131.160.1.112
- times (start, finish):
  t=0 0
- media (kind, port, protocol, profile):
  m=audio 49170 RTP/AVP 0

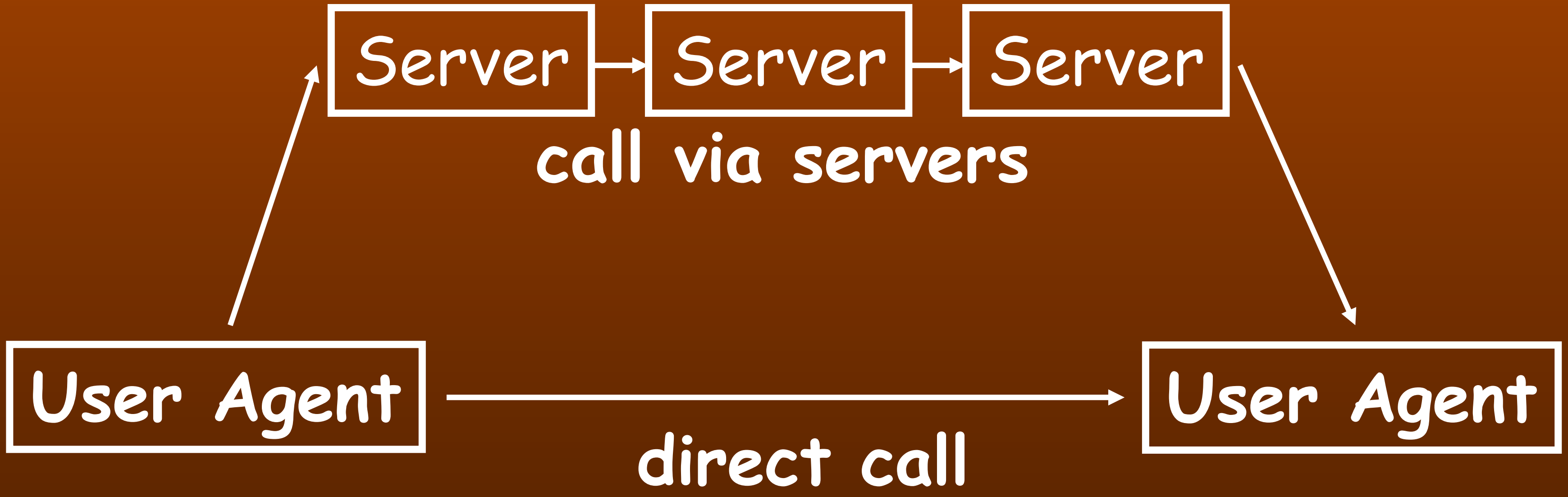
SIP Services

Services in SIP

- services provide additional functionality
- services can be more advanced than PSTN:
  - extra signalling information available
  - can be integrated with web and databases
  - endpoints can negotiate
- several locations possible for services:
  - User Agents (e.g. 'intelligent phones')
  - Proxy Servers
  - Redirect Servers

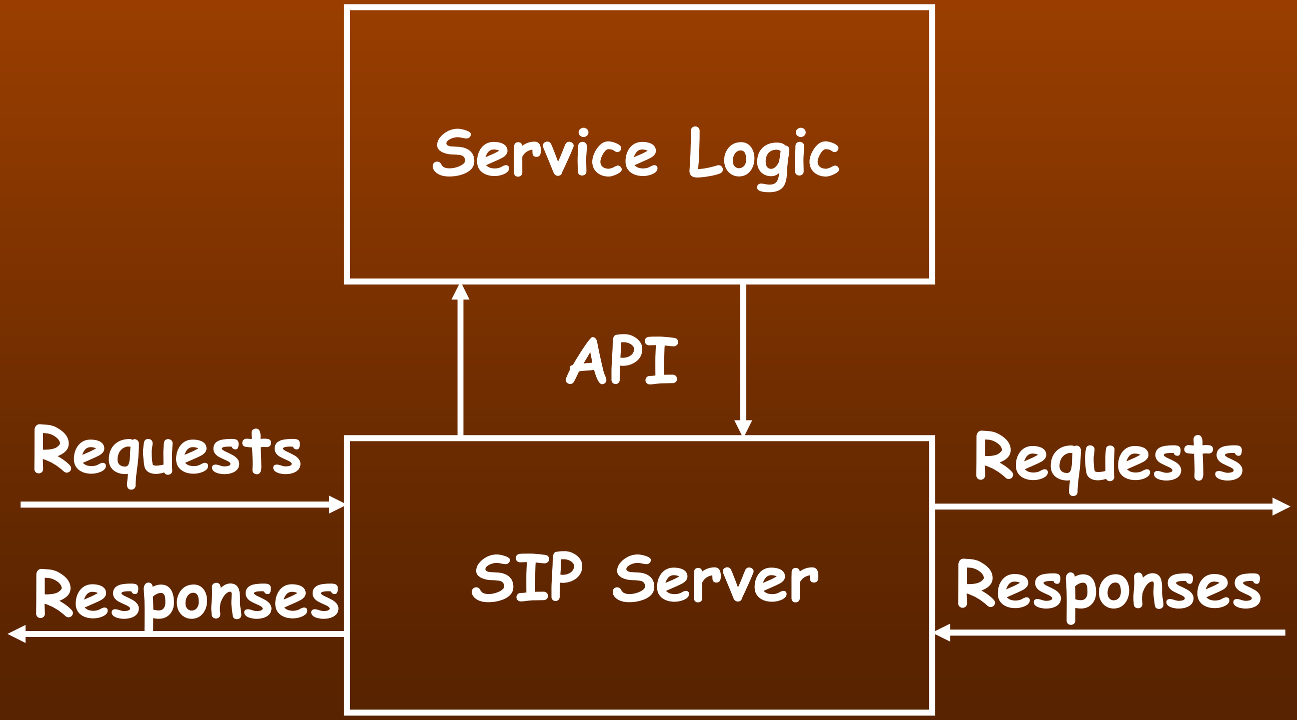
User Agent State

- servers cannot know user agent state:
  - 'busy', etc. can be decided only by the user
  - signalling may bypass servers:



Service Languages

- CPL (Call Processing Language)
- SIP CGI (Common Gateway Interface):
- SIP Servlets:

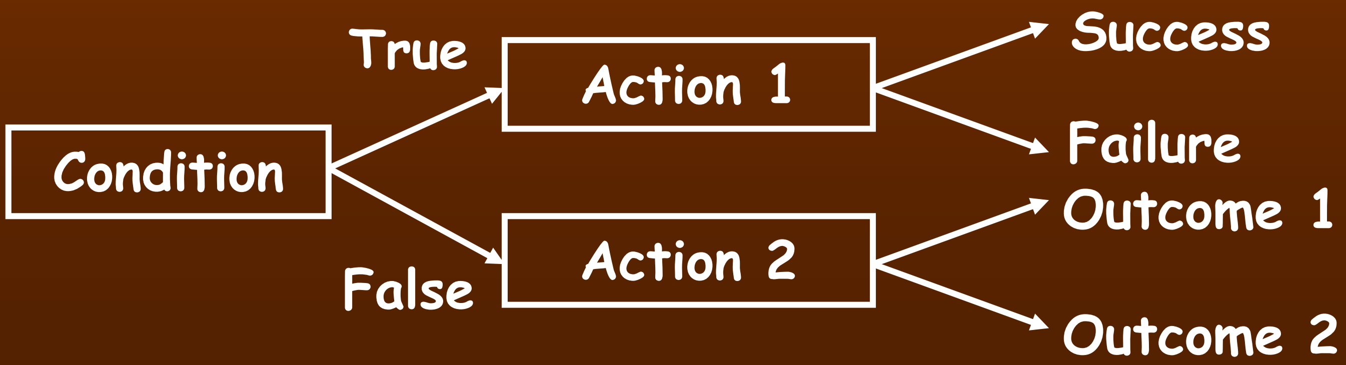


SIP CGI

- call scripting:
  - patterned after HTTP CGI
  - links server and program (C, Java, Perl, TCL, …)
  - CGI program functionality unrestricted
- full access to message headers and bodies:
  - can change fields of a message
  - can generate requests and responses
  - can use any resources (e.g. databases)
  - can be state-full, i.e. maintain session state
  - can use cookies
  - powerful, but risky (e.g. delays, resources)

SIP CPL

- defines XML-based scripts to filter calls:
  - textual (or some graphical equivalent)
  - no loops, so guaranteed to terminate
  - restricted processing, so safe to execute
- deployment:
  - typically uploaded to server using REGISTER
  - one script per user for INVITE handling



Some CPL Tags

- Signalling Actions:
  - proxy
  - redirect
  - reject
- Decisions:
  - address-switch
  - priority-switch
  - string-switch
  - time-switch
- Location Modifiers:
  - location
  - lookup
- Subprograms:
  - sub (call)
  - subaction (define)
- Other actions:
  - log
  - mail

CPL Example

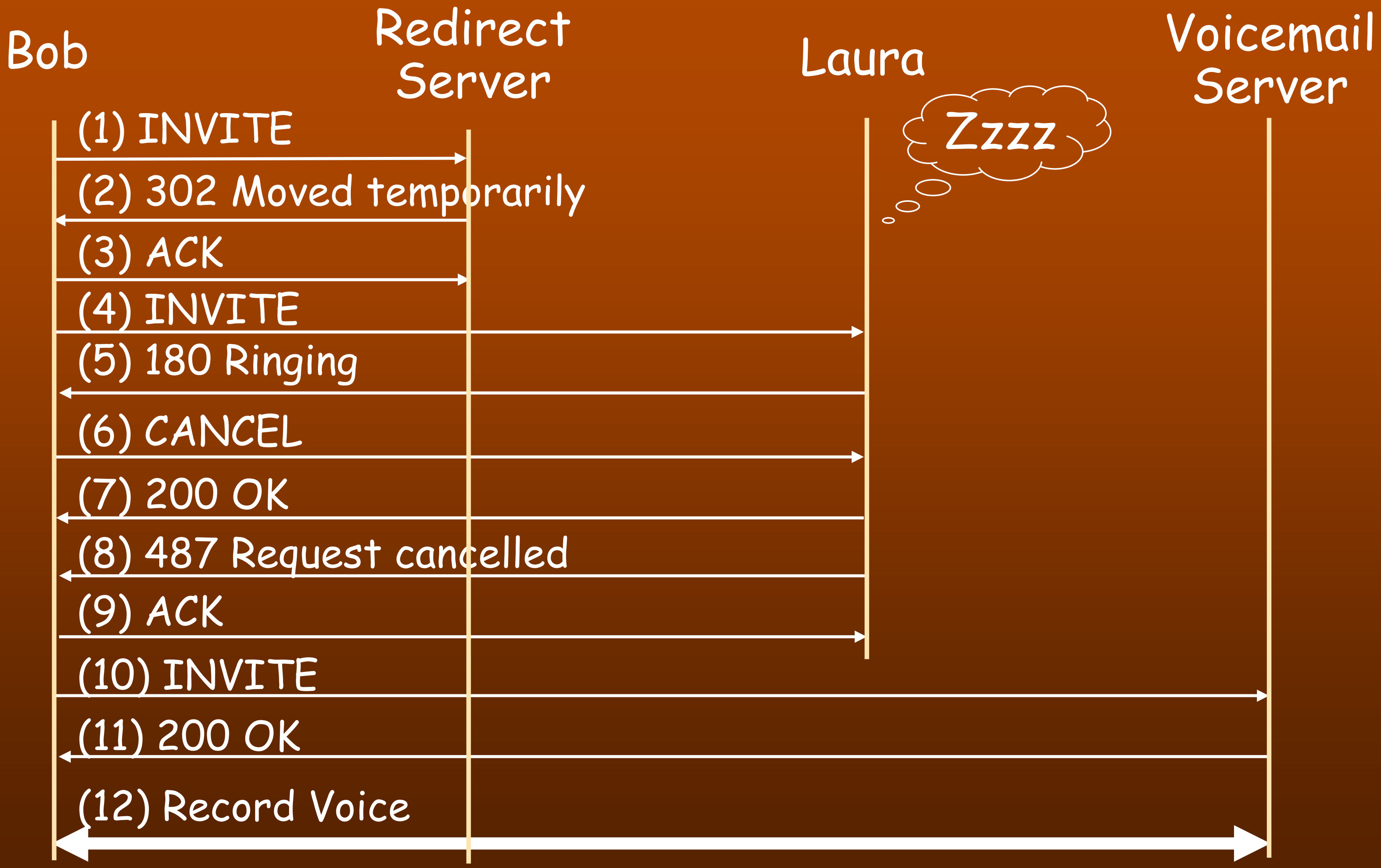
- closing tags are omitted below:

```
<incoming>
 <address-switch field="origin" subfield="host">
  <address subdomain-of="com">
   <location url="sip:bob@firm.com">
    <proxy timeout="10">
     <busy>
      <sub ref="busyMessage"/>
     <noanswer>
      <sub ref="recordMessage"/>
 <otherwise>
  <sub ref="errorMessage"/>
```

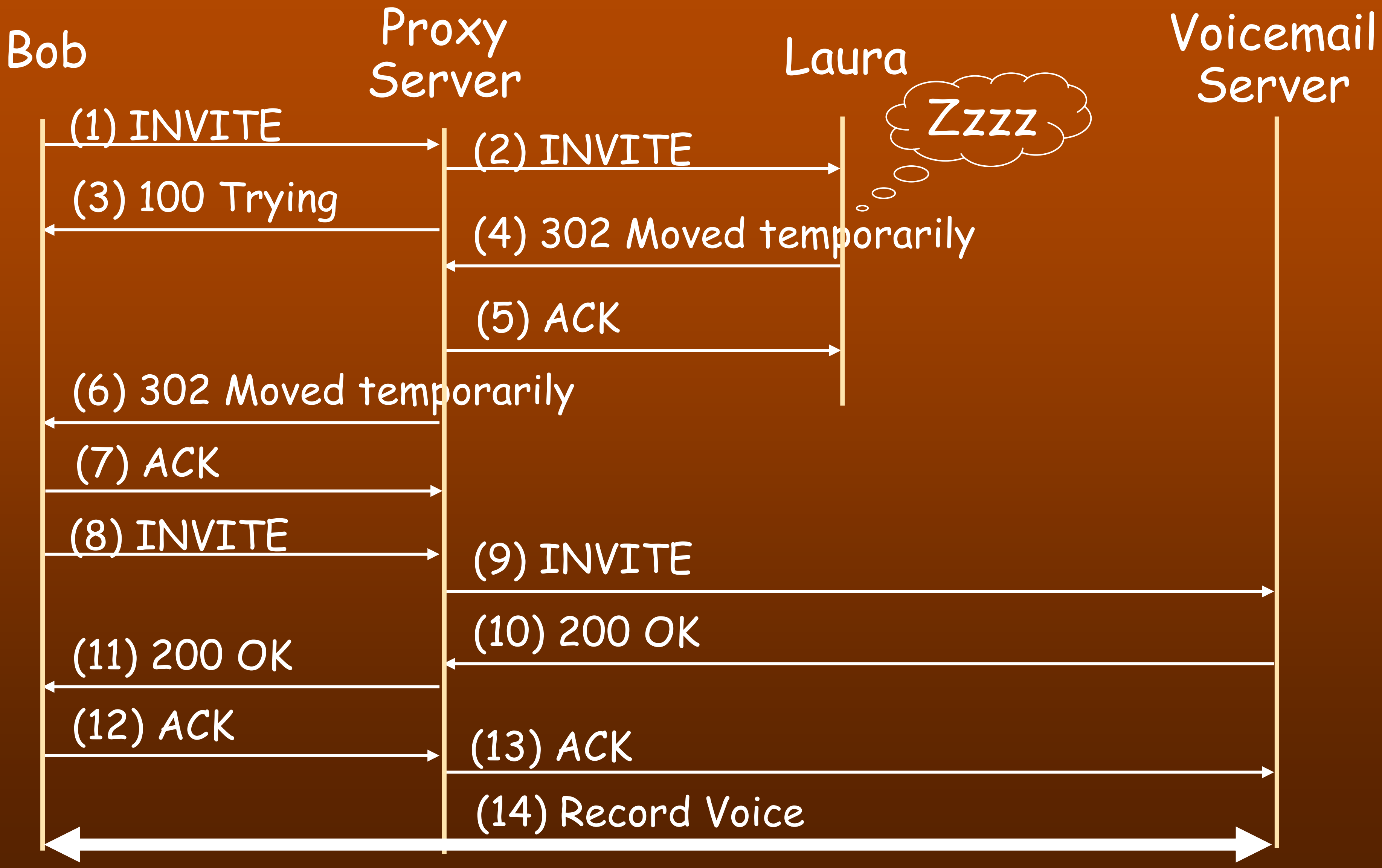
CFDA (Call Forward Don't Answer)

- User Agent Client:
  - manual control from user interface
  - calls must be redirected, not proxied by server
- User Agent Server:
  - local user agent controls the call
  - logic can be easily updated by the user
- Proxy Server:
  - no special logic in user agents
  - server must be updated for new logic

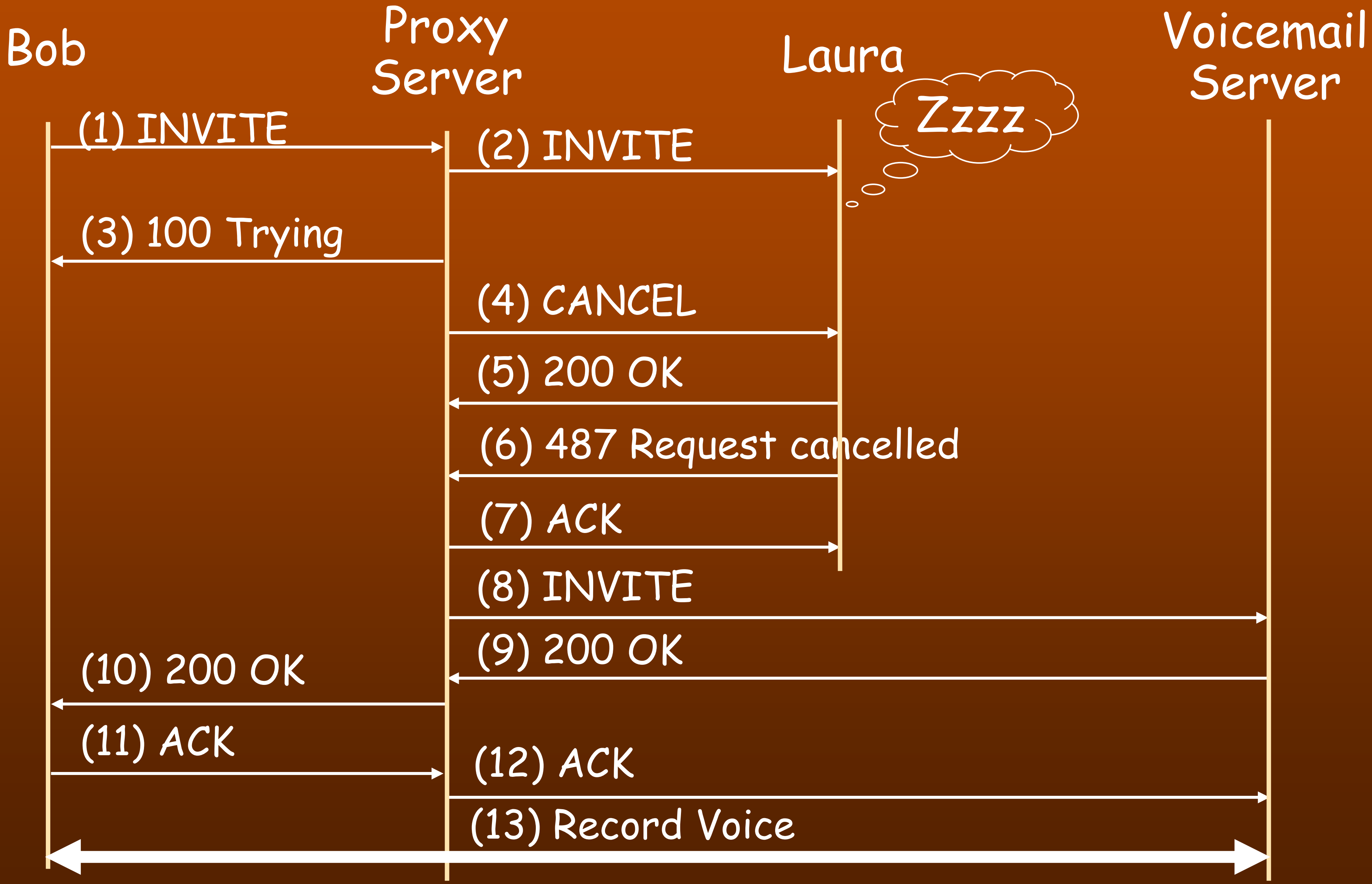
CFDA – User Agent Client



CFDA – User Agent Server



CFDA – Proxy Server



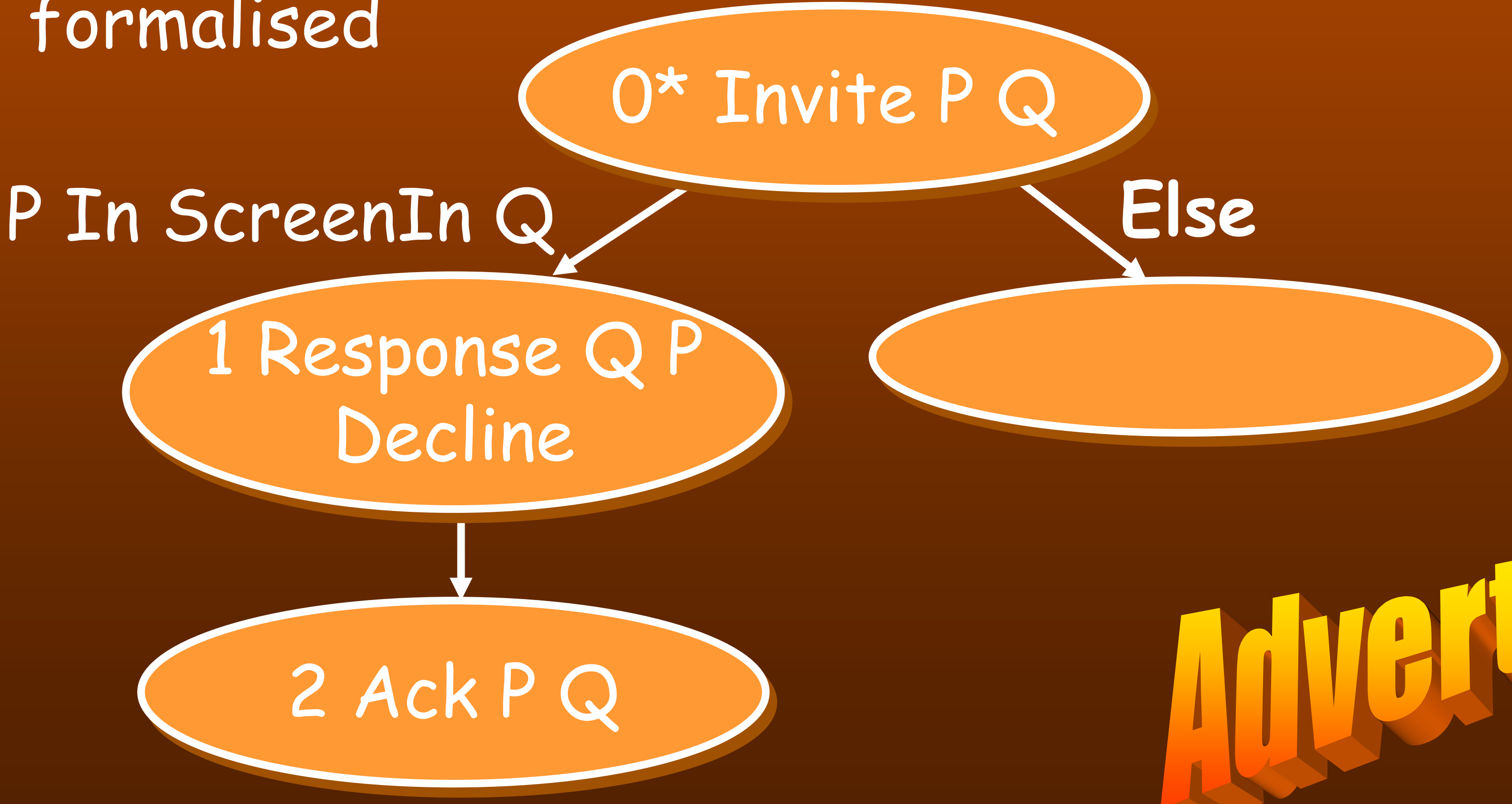
SIP Service Research

Research Questions

- SIP services:
  - what might they be?
  - how do they differ from voice services?
  - where should they be located?
  - should services differ in agents and servers?
- SIP features:
  - what is a necessary and sufficient set?
  - how should they be composed?
  - how do they differ from voice features?
  - what feature interactions might arise?
  - how to detect and resolve interactions?

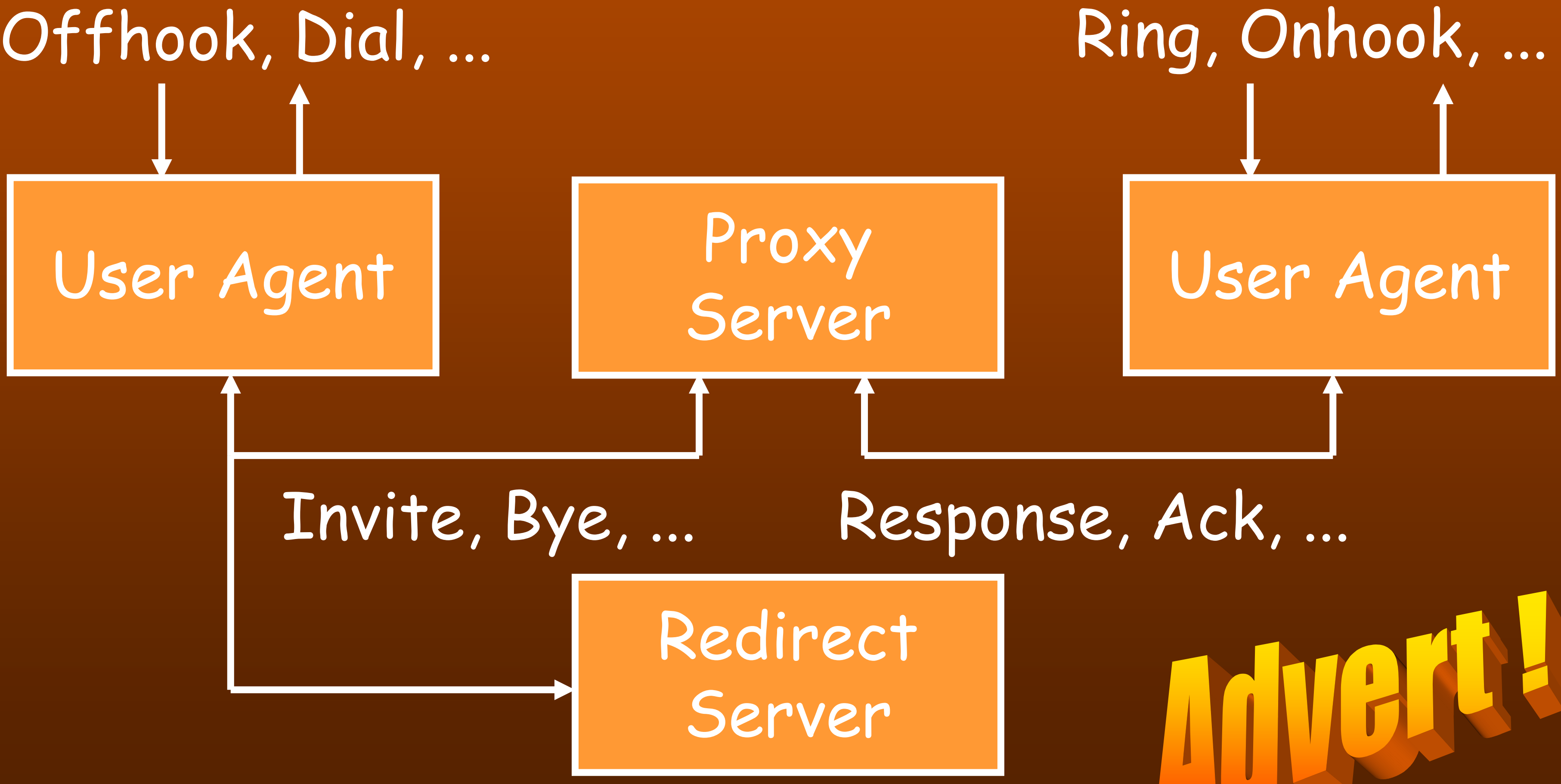
CRESS Service Description

- CRESS (Chisel Representation Employing Systematic Specification):
  - graphical
  - formalised



CRESS Service Architecture

- service/protocol mapping:



SIP Sample References

Printed Material

- J Lennox *et al*: 'Common Gateway Interface for SIP', RFC 3050
- J Lennox and H Schulzrinne: 'CPL: A Language for User Control of Internet Telephony Services', Internet Draft
- J Rosenberg *et al*: 'Session Initiation Protocol', RFC 3261
- H Sinnreich and A Johnson: *Internet Communications using SIP*, John Wiley
- SIP Annual Conferences: January, Paris

Web Pages

- Columbia SIP: www.cs.columbia.edu/sip
- NIST Parser: www.antd.nist.gov/proj/iptel
- SER (SIP Express Router) : www.iptel.org/ser
- SIP Center: www.sipcenter.com
- SIPComm: www.sipcomm.com
- SIPHON (SIP Phone) : sourceforge.net/projects/siphon
- VOCAL (Vovida Open Communication Application Library): www.vovida.org

SIP … Some Coffee

VoiceXML Overview

VoiceXML Goals

- natural speech input and output
- mainly telephony, but not limited to this
- linked to common technologies such as databases and the Internet
- acceptable audio interface:
    - more natural than 'press button 2 for sales'
    - accessible to the partially sighted
    - good also for mobile users
    - audio wrapping for legacy applications ('screen-scraping')

VoiceXML Characteristics

- consolidates earlier efforts on IVR (Interactive Voice Response)
- includes sophisticated voice technology:
  - natural-sounding TTS (Text To Speech)
  - effective grammar-driven speech recognition
- integrates well with databases, the Web, scripting languages, ...
- uses familiar imperative and event-driven programming language concepts
- uses popular XML approach

Commercial Aspects

- strong interest in IVR services:
  - callers want to talk, not press buttons
  - disability legislation may force the use of IVR
  - answering phones costs US firms $30bn p.a.
  - IVR typically 10% of manual system cost
- IVR has been widely adopted:
  - call centres
  - travel bookings and timetables
  - interactive banking
  - voicemail and dictation
  - voice-controlled devices
- VoiceXML Forum involves many key players

VoiceXML Model

- VoiceXML mainly focuses on fields:
  - filled by speech input following some grammar
  - progressively filled, may not be all or in order
  - a variable holds the value of each field
  - field variables can be used in expressions and sent to a database or script
- as well as conventional program flow, VoiceXML may also be driven by events:
  - events are hierarchical
  - events can be caused by user input, program conditions or explicitly

VoiceXML Levels

- platform:
  - the infrastructure to support VoiceXML
  - contains interpreters, speech recogniser, speech synthesiser, event dispatcher, …
  - some functions and variables vary by platform
- application:
  - the overall set of scripts
  - documents hold VoiceXML, HTML, scripts, …
  - the root may contain common definitions
  - explicit program control switches documents
- form (or menu): groups fields
- field: the unit of user input

VoiceXML Elements

XML Basis

- VoiceXML is an application of XML:
  - standard DTD, though company variants exist
  - follows the usual XML structure:

```
<?xml version="1.0"?>
<vxml version="2.0">
  <form>
    <block>
      Hi there!
    </block>
  </form>
</vxml>
```

Variables and Expressions

- (JavaScript) variables are scalars/arrays:
  - boolean
  - numeric
  - property (platform variable)
  - string
- used as follows:

```
<assign name="area" expr="base * height"/>
<property name="timeout" value="5"/>
<value expr="'Hello'  + planetoid"/>
```

- expressions use JavaScript operators and functions (e.g. getDate, indexOf, round)

Branches

- unconditional branches to forms:

```
<goto expr="baseURI + '#' + item"/>
```

- conditions, but no **case/switch** (&gt; is '>'):

```
<if cond="salary &gt; 100000">
  rich
<elseif cond="salary &gt; 50000"/>
  comfortable
<else/>
  poor
</if>
```

- leave a VoiceXML script with <**exit**/>

Subprograms

- no loops, but can iterate over an array:

```
<foreach item="s" array="scores">
  Score <value expr="s"/> points
</foreach>
```

- can call a form like a subroutine:

```
<subdialog name="userEntry" src= "login.vxml">
  <filled> … </filled>
</subdialog>
```

- the result is an event or value list:

```
<return namelist="username password"/>
```

- data may be sent to a URI (e.g. a script):

```
<submit next="order.jsp" namelist="make type"/>
```

Audio

- speech output can be given explicitly:

```
<audio>good morning</audio>
```

- speech mark-up can be included for emphasis, pronunciation, pauses, etc.
- speech synthesis is reasonable quality, but pre-recorded speech is used commercially:

```
<audio src="msg23.wav">good morning</audio>
```

- a prompt may be subject to the input attempt count and/or a condition:

```
<prompt count="3" cond="answer != 42"/>
```

- a <reprompt> usually repeats the last prompt (actually, the first enabled prompt)

Events

- event names may be:
  - standardised (e.g. cancel, noinput)
  - platform-specific (e.g. for certain failures)
  - script-defined (e.g. weather.precipitation.rain)
- events are generated and intercepted:

```
<throw event="boiler.pressure.high"/>
<catch event="boiler"> … </catch>
```

- if there is no handler specific to the event, a more general handler can deal with it
- common events (e.g. error, help):
  - have default platform handlers
  - have convenience syntax (e.g. **<error>**, **<help>**)

VoiceXML Forms

Grammars

- field input is defined by a grammar:
  - field options (e.g. a list of products)
  - built-in (e.g. boolean, date, number, phone)
  - ABNF (Augmented Backus-Naur Form), SRGF (Speech Recognition Grammar Format), ...
- if field options are given, <enumerate/> can be used to list them
- grammar-directed recognition yields event:
  - filled (input matched, field variable set)
  - noinput (no input heard)
  - nomatch (input does not match grammar)

Fields

- a field has:
  - a named variable to hold its value
  - an optional type or grammar
  - sub-elements like prompts and event handlers
- a field is ignored if it already has a value
- example to convert net to gross cost:

```
<field name="price" type="currency">
  <prompt> State the net cost </prompt>
  <filled> Gross <value expr="price*1.1"/> </filled>
  <noinput> Nothing heard! <reprompt/> </noinput>
</field>
```

Forms

- an application may have several documents
- a document may have one or more forms
- a form groups fields, and may include generic code (e.g. assignments, handlers)
- a branch may occur to a document or field
- example to read a product and cost:

```
<form id="order">
  <field name="product"> ... </field>
  <field name="price" type="currency"> ... </field>
</form>
```

Other VoiceXML Capabilities

- VoiceXML can be generated dynamically by scripts (cf. dynamic HTML)
- JavaScript can be arbitrarily mixed with VoiceXML, e.g. for complex calculations
- mixed-initiative forms allow several fields to be filled in any order
- VoiceXML can activate phone connections (cf. 'click to dial')
- VoiceXML can record and replay speech input (cf. voicemail)

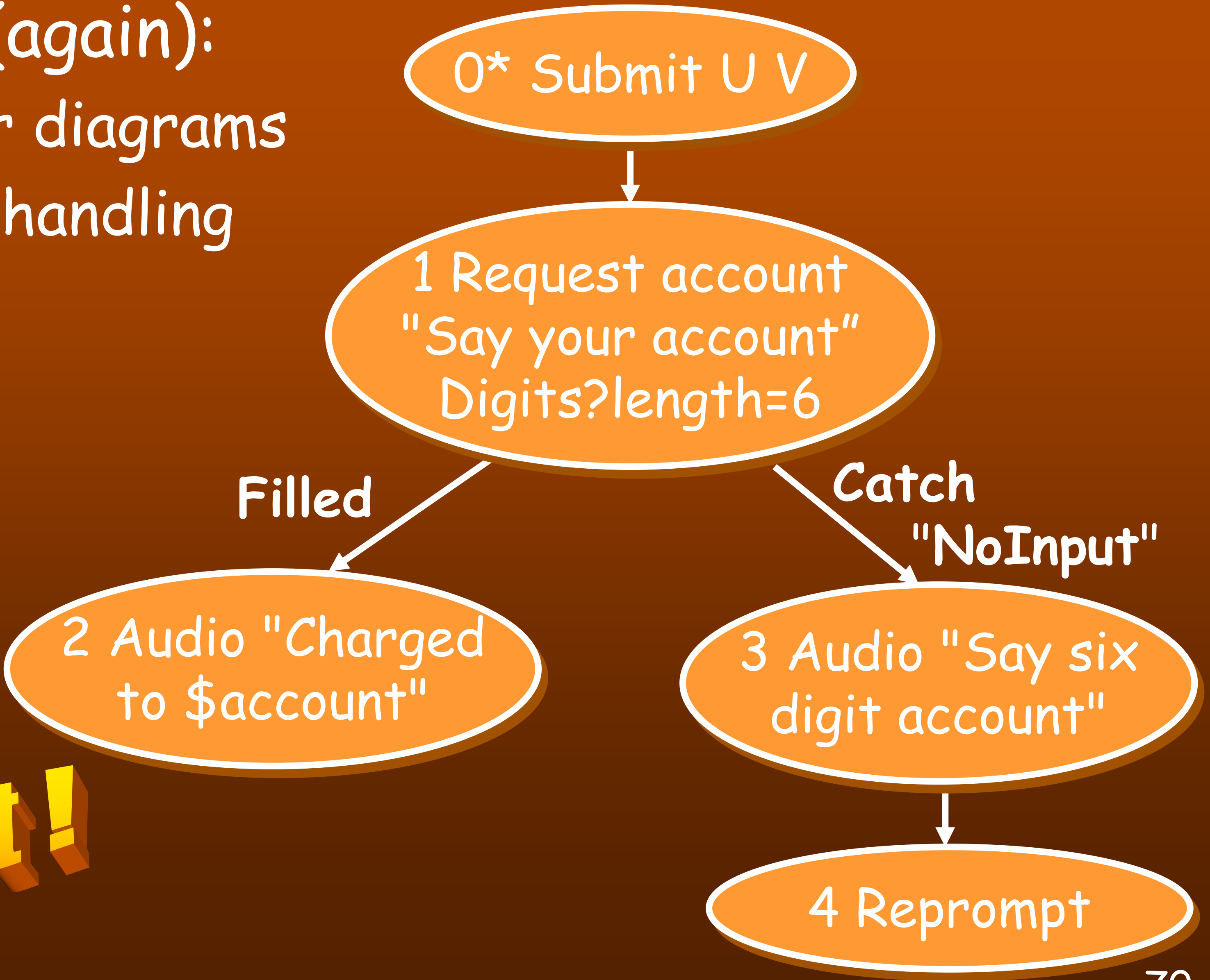
VoiceXML Service Research

Research Questions

- VoiceXML services:
  - what might they be?
  - can they be separated from their embedding (databases, scripts, Web)?
  - how to analyse services for correctness and completeness?
- VoiceXML features:
  - do features have a meaning in VoiceXML?
  - how should they be composed?
  - what feature interactions might arise?
  - how to detect and resolve interactions?

CRESS Service Description

- CRESS (again):
  - similar diagrams
  - event handling


0* Submit U V
1 Request account "Say your account" Digits?length=6
Filled
Catch "NoInput"
2 Audio "Charged to $account"
3 Audio "Say six digit account"
4 Reprompt

Advert !

VoiceXML Sample References

Printed Material

- C Sharma and J Kunins: *VoiceXML*, Wiley
- VoiceXML Forum: *VoiceXML Standard*, Version 2.0
- S Weinschenk and D T Barker: *Designing Effective Speech Interfaces*, Wiley

Web Pages

- BeVocal Café: developers.bevocal.com
- Covigo Studio: www.covigo.com
- Elvira (Extensible LSD VoiceXML Interpreter for Dialog Applic.): www.fi.muni.cz/lsd/elvira
- Nuance V-Builder: www.nuance.com
- Open VXI: fife.speech.cs.cmu.edu/openvxi
- Public VoiceXML: www.publicvoicexml.org
- TellMe Studio: studio.tellme.com
- VoiceXML Forum: www.voicexml.org
- VoiceXML Review: www.voicexmlreview.org
- Voxeo Community: www.voxeo.com

VoiceXML … Show and Tell