ABSTRACTIONS FOR PROGRAMMING SIP BACK-TO-BACK USER AGENTS

INTRODUCING . . . StratoSIP: SIP at a very high level

Pamela Zave Gregory W. Bond Eric Cheung Thomas M. Smith

AT&T Laboratories—Research Florham Park, New Jersey
RECORD VOICE MAIL AND DO NOT DISTURB AT RUNTIME

Alice says that call is not urgent; DND sends "busy" upstream

Alice says that call is urgent

Bob answers ringing phone
StratoSIP supports this programming style, which is . . .

**COMPOSITIONAL FEATURE PROGRAMMING**

Each feature is an independent program.

Each feature works correctly by itself or composed with any combination of other features.

Feature interactions are predictable; they can be managed without changing the features themselves.

For example, by adjusting feature order, which is data in the feature container.

at the top level, each feature program is a finite-state machine

a program has autonomy: when it has a function to perform, it performs it without external help

a program has transparency: when it has no function to perform, it is unobservable

a program is context-independent: it does not refer to other features

THIS STYLE WAS INTRODUCED AND PROVEN SUCCESSFUL BY THE "DISTRIBUTED FEATURE COMPOSITION" (DFC) ARCHITECTURE
PROGRAMS ARE BACK-TO-BACK USER AGENTS . . .

. . . which means that they are the endpoints of SIP dialogs.

However, contrary to some popular opinions about B2BUAs . . .

- they can run in user endpoints as well as application servers in the network
- they are easy to program (because of StratoSIP)
- they preserve the end-to-end behavior of SIP signaling, except when the program's purpose is to modify it

PROGRAMS ARE ALSO SIP SERVLETS

JSR 289 SIP Servlet Container

- DFC application router
- SIP Servlet Applications
- StratoSIP programs
- StratoSIP runtime

assembles graph structures of programs and signaling channels
implementation in ECharts and Java

all components of the runtime environment are available as open-source software
all operations appear atomic—rcv includes receiving an initial Invite, sending 100 Trying, and sending 183 to establish the dialog

dialog variable

indicates success in reaching the desired party

in simple cases succeeded becomes true because of 200 OK, but it can also be an abstraction of many other SIP messages

choice of operations tells the router how to route the outgoing Invite, without mentioning other features

Cancel, Bye, or failure response to initial Invite, plus acknowledgments
media control . . . is performed automatically from the declarative state annotations . . . works in all contexts and scenarios, starting from any state . . . composes correctly with media control in other feature programs
Customer calls Trainee, who is being trained by Supervisor.

**NotMonitored:**
- $c \leftrightarrow t$

**Monitored:**
- $c \leftrightarrow cc$, $t \leftrightarrow ct$, $s \leftrightarrow cs$
  - $c \leftrightarrow t$, $s \leftrightarrow t$

**Postmortem:**
- $t \leftrightarrow s$

- $rcv(c) / t = ctu(c)$
- $ended(s) / ended(cc); ended(ct); ended(cs)$
- $rcv(s) / cc = new(conf); ct = new(conf); cs = new(conf)$
- $ended(c) / ended(cc); ended(ct); ended(cs)$
- $ended(s) / ended(cc); ended(ct); ended(cs)$
- Outgoing dialogs to conference bridge
- Incoming dialog from Supervisor

Supervisor hangs up

Customer hangs up

Signaling (as opposed to media) linkages
ATTENDED TRANSFER AND TRAINEE MONITORING

1. Customer (C) 
   T's AT: Talking 
   T's TM: Not Monitored 
   Trainee (T)

2. T's AT: Talking 
   T's TM: Monitored 
   Supervisor (S)

3. T's AT: Consulting 
   T's TM: Monitored 
   Expert (E)

4. T's AT: Transferred 
   T's TM: Post Mortem
COMPOSITIONAL MEDIA CONTROL

THIS IS THE CORE TECHNOLOGY OF StratoSIP

- implemented using the "third-party call control" style of SIP
- completely general with respect to feature functions
- compositional semantics has been specified formally in temporal logic
- implementation has been verified using model-checking
- based on 9 years of research
- turns a very difficult kind of programming into something safe and easy, accessible to (for example) Web programmers

WE DO NOT YET KNOW WHETHER IT IS COMPATIBLE WITH MEDIA CONTROL USING Refer MESSAGES, Replaces AND Join HEADERS
<table>
<thead>
<tr>
<th>What version of SIP must endpoints implement?</th>
<th>Compositional Media Control</th>
<th>Refer/Replaces/Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic SIP (RFC 3261)</td>
<td>RFC 3261 plus 3 extension RFCs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Can feature programs be located in either endpoints or servers?</th>
<th>Compositional Media Control</th>
<th>Refer/Replaces/Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>YES</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>When independently-developed features are used together, what must be done so that the composed features control media correctly?</th>
<th>Compositional Media Control</th>
<th>Refer/Replaces/Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>nothing</td>
<td>rewrite each feature program with extra cases to coordinate with the actions of the other</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How many end-to-end signaling messages are required for the sample scenario?</th>
<th>Compositional Media Control</th>
<th>Refer/Replaces/Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>81</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Are there any other inefficiencies?</th>
<th>Compositional Media Control</th>
<th>Refer/Replaces/Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>in some scenarios there are signaling hairpins (not media hairpins!)</td>
<td>we think this can be fixed</td>
<td>NO</td>
</tr>
</tbody>
</table>
**ONGOING WORK**

**IMPLEMENTATION**

- finishing implementation
- testing with JSR 289 SIP Servlet containers (OCCAS, SailFin, Mobicents)
- hope to release as open-source code

**SUPPORT FOR CONVERGED SERVICES**

- we now have automatically-generated Web interfaces to feature and routing data
- automatically generate Web interfaces for feature events (commands and status reports)
- explore deeper levels of convergence

**FUTURE WORK**

**LANGUAGE EXTENSIONS**

- support for an arbitrary number of dialogs
- separate control of multimedia channels

**SIP INTEGRATION**

- handle a few additional requests introduced in SIP extensions (straightforward)
- look for safe ways to handle Refer/Replaces/Join