

ECharts for SIP Servlets: a state-machine programming environment for VoIP services

Thomas M. Smith & Gregory W. Bond
AT&T Labs - Research
{tsmith,bond}@research.att.com



Outline

- ECharts Overview
- ECharts for SIP Servlets
- Future Work

ECharts History

- In 1999 work began on VoIP application server.
- Initially, features coded in “raw” Java
 - Error prone - asynchronous nature of telecom features invited races and unexpected states
 - Hard to understand
- Ended up hand-coding state machines.
- This was so time-consuming and repetitive, a **higher level language** was created to specify the **state machine logic**

ECharts History (cont'd)

Similarities to **UML Statecharts** (standardized by OMG):

- Graphical syntax
- Useful abstractions
 - Embedded machines
 - Concurrent machines
 - Fork/join transitions

ECharts History (cont'd)

- Extensions to UML Statecharts:
 - **Textual** and graphical **syntax** - easy to code
 - Selection of transition priority rules to enforce determinism
 - New, safe, modular, communications mechanisms within and between machines
 - **Support for re-use**
 - Support for **dynamically created machines**
 - Relies on host language

SIP Servlet API

- Created via Java Community Process (JCP)
- Version 1.0 is specified in **JSR 116**
- Version 1.1 under development in JSR 289
- Extension of generic servlet API
- Classes extend `javax.servlet.sip.SipServlet`, and **implement callback methods** (such as `doInvite`) to specify service logic
- At least four commercial implementations

Our Programming Model

- **Application programmer specifies FSM logic**, coded in ECharts.
- We provide a JSR-116 compliant SIP servlet, which creates the specified FSM and dispatches received messages to appropriate ports.
- These messages drive FSM logic.
- Java host language allows **full use of SIP Servlet API**.
- Well-suited for back-to-back user agents (B2BUA).

Benefits of SIP/ECharts

- **Centralization of application logic**
- ECharts port abstraction
- Embedded state machines
 - State machine re-use
 - Overriding default behavior
- Support for non-SIP events
 - Timed transitions
 - Other protocols (e.g., HTTP)
- Message class hierarchy
- Automatic termination handling
- Support for application composition

Port Abstraction

- An ECharts port maps to a SipSession object, so developer can associate a **symbolic name** with a particular SipSession

```
public void doResponse(SipServletResponse resp) {  
    if( resp.getSession().getId() == callerSessionId ) {  
        ...  
    } else if ...  
}
```

```
transition AWAIT_BYE_RESPONSE - caller ? Response -> END;
```



Support for embedded machines

- Support for embedded machines provides a straightforward method to **re-use machine logic**, as well as **override default behavior**.
- Well-documented transition priority rules determine which transition will fire.
- Example : Use B2BUA to set up a call to media server, but do not pass "180 RINGING" message from media server to caller.

```
// default behavior is to propagate messages between
// caller and mediaServer ports
//
state SETUP_CALL : B2BUA(box, caller, mediaServer);

// for this case : do nothing
transition SETUP_CALL - mediaServer ? ProvisionalResponse180
-> SETUP_CALL.DEEP_HISTORY;
```

Non-SIP events - Timed Transitions

- **Transitions can be based on fixed delays** as well as messages on ports
- Uses ServletTimer mechanism from SipServlet API, but with much less programmer overhead

```
// create timer
timerService = getServletContext().getAttribute(TIMER_SERVICE);
timerService.createTimer(appSession, timeoutMsec, false, info);
...
// elsewhere, implement timer listener
public void timeout(ServletTimer timer) {
    ...
}
```

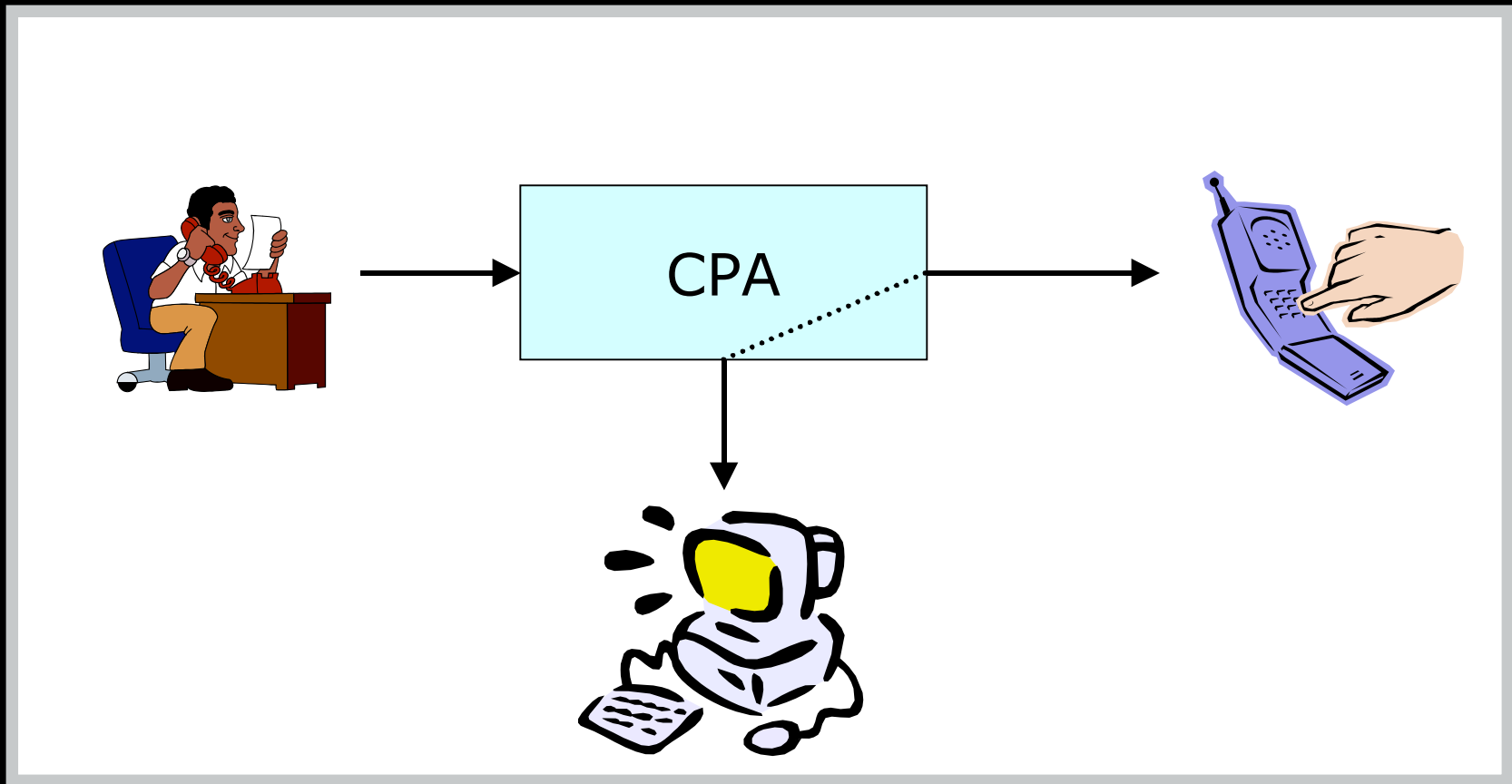
```
transition RINGING - delay(timeoutMsec) -> NO_ANSWER;
```



Termination Handling

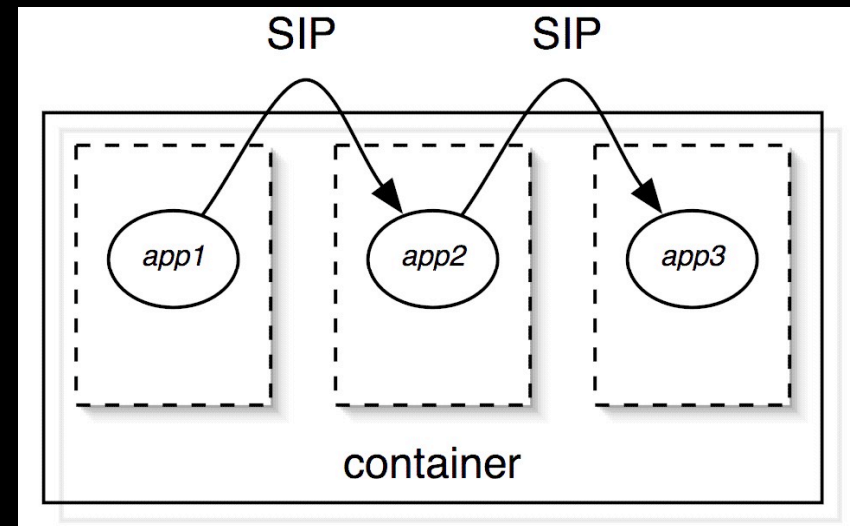
- **Automatic call teardown** is triggered by one of these conditions:
 - FSM transitions to terminal state
 - Received BYE/CANCEL with no matching transitions
- **Example:** consider an application which connects a called party to a media server (for authentication, etc.). Caller hangs up. For default handling (tear down all legs, regardless of state), *no code is required*.

Example: Called party authentication



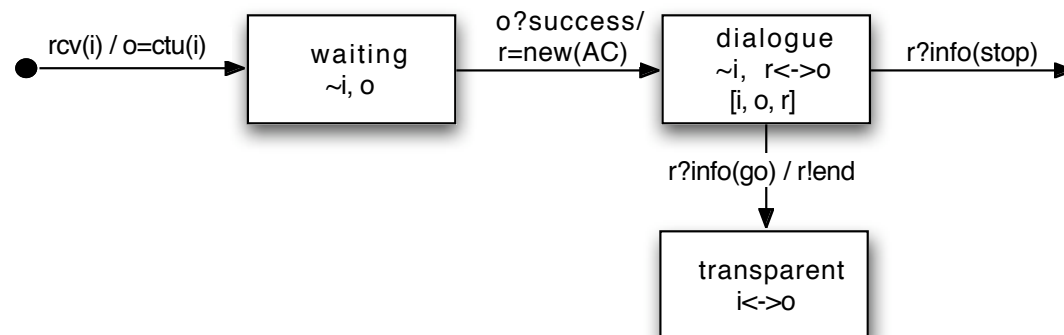
Support for Application Composition

- Packaging features as separate servlet applications allows for easier development, re-use, maintenance and analysis
- JSR 289 will address this topic explicitly when it is released
- We have created an **adaptation layer for JSR 116 containers** that mimics the proposed JSR 289 implementation.



Coming Soon: **StratoSIP**

- Domain-specific feature programming language providing **SIP signaling abstractions** and **distributed media control**
- Will be implemented in ECharts for SIP Servlets
- Active research area for 2007



Summary

- **ECharts for SIP Servlets** provides a mechanism for simplifying the development and maintenance of complex SIP applications.
- This technology is available as **open source software**.
- Visit **echarts.org** for more information.

