



Project
GlassFish



JavaOne

Adding Telephony to Java™ Technology-Based Enterprise Applications

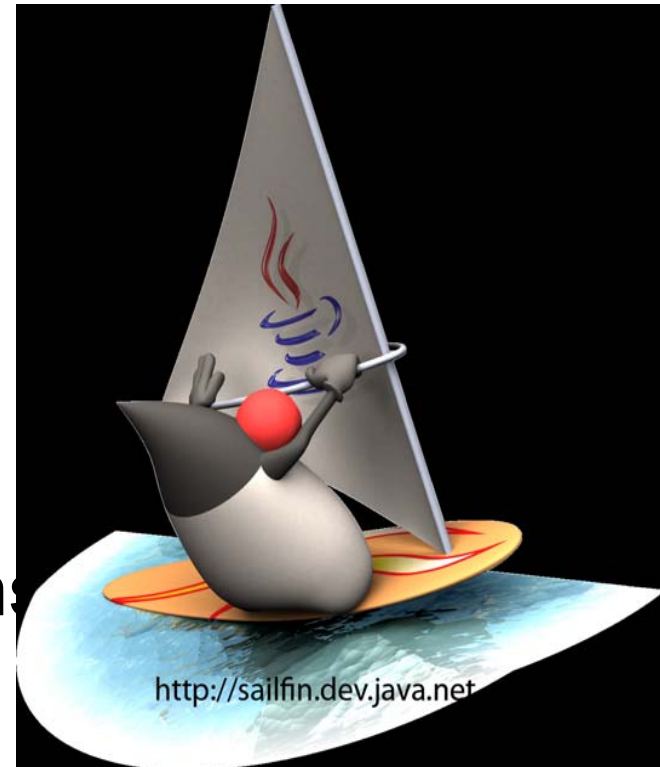
Jonathan Kaplan & Sreeram Duvur

Researcher/Architect
Sun Microsystems, Inc.
<http://glassfish.dev.java.net/>

TS-4919

Project SailFin

- Based on Ericsson's contribution of SIP Servlet technology to GlassFish™ community
- Foundation technology for many mobile services
- Also enables media interactions to be added to Enterprise Web applications



Adding Telephony to JavaTM Enterprise Applications

Learn about **SIP Servlets** and how to use them to develop **voice-enabled** enterprise applications.

Agenda

SIP Technology Overview

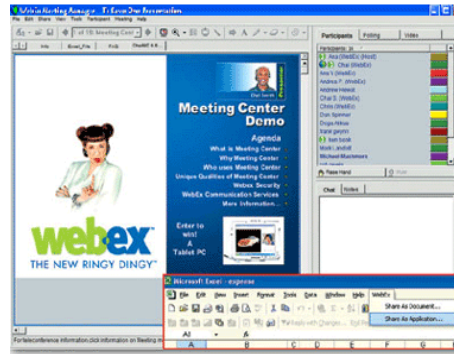
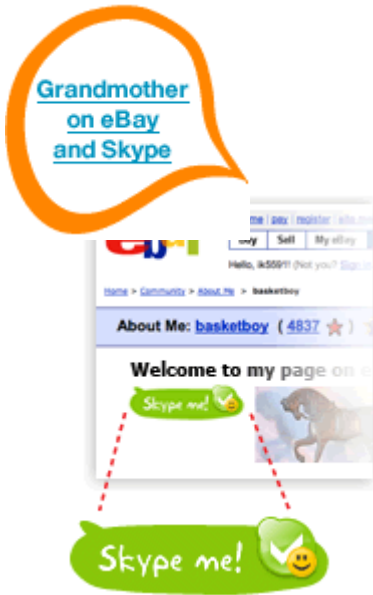
Using SIP Servlets

Simple Example (Click-to-Dial)

Enterprise Example (Conference Manager)

Scalability and Reliability

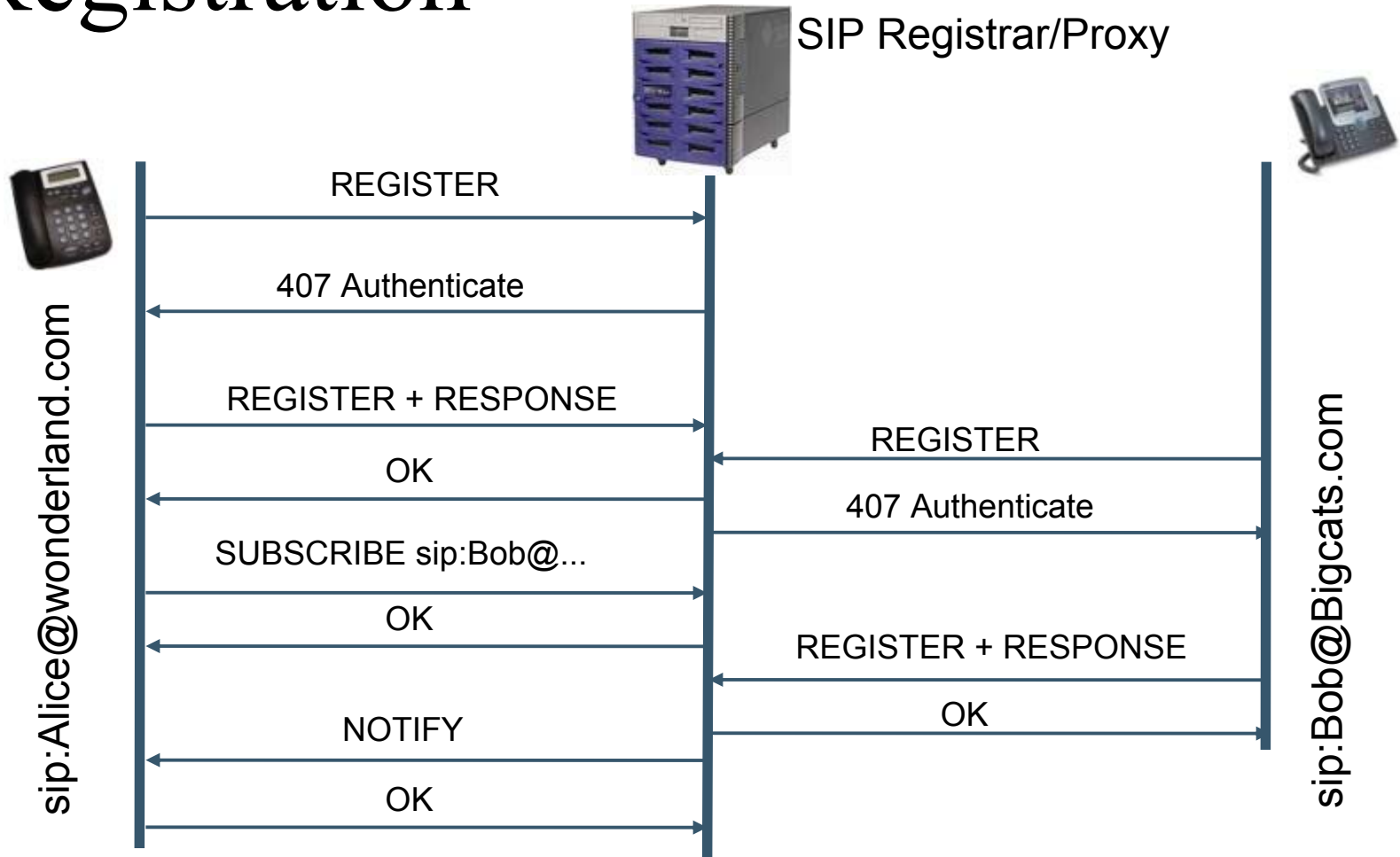
Can You Do This on the Java Platform?



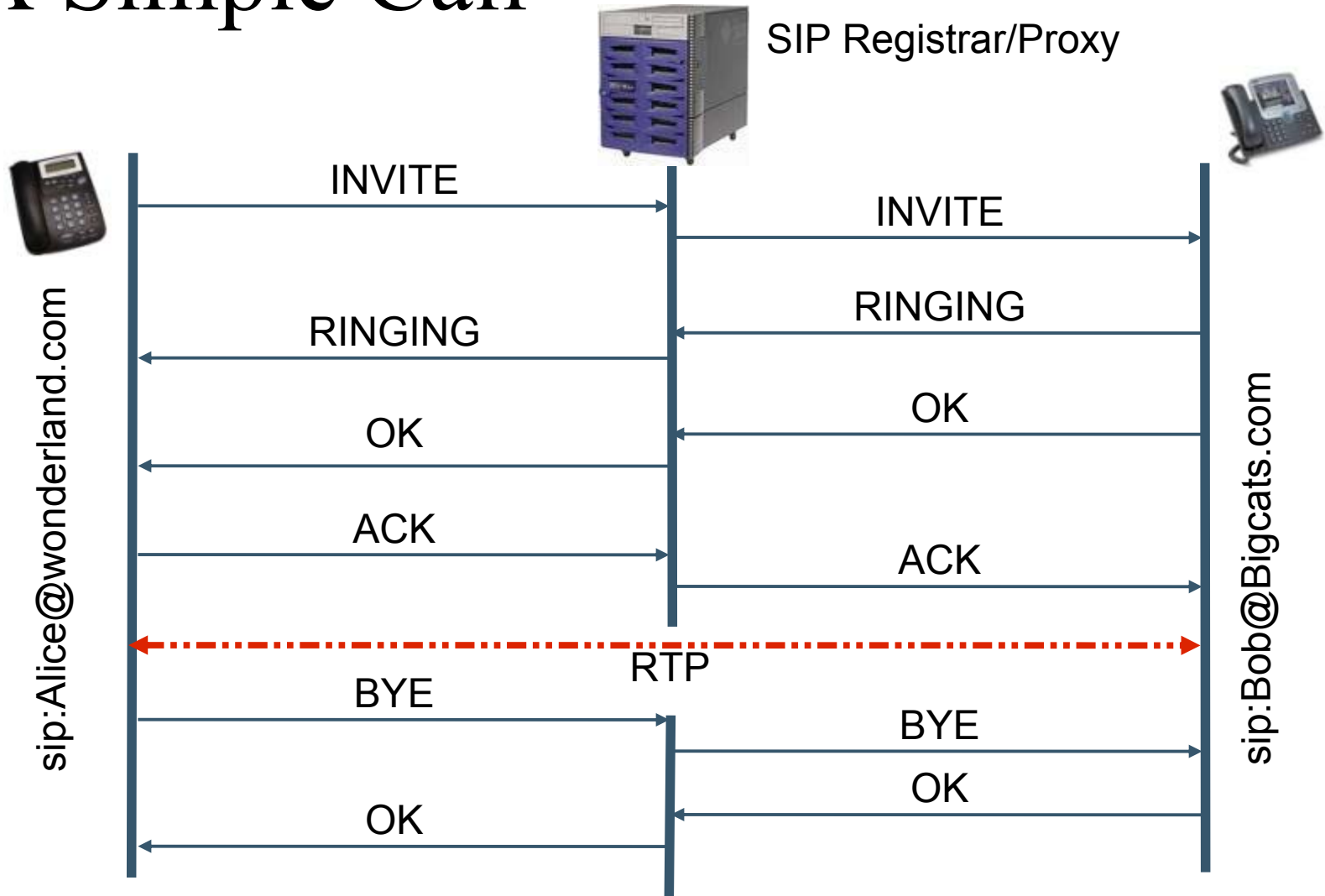
SIP Technology Makes It Possible

- **Session Initiation Protocol**
 - Signaling protocol for Internet multimedia
 - Only responsible for setting up communications
 - Defined by RFCs 3261(sip), 4566(sdp), 3550(rtp)
- **Similar to HTTP**
 - Text-based request and response
 - Shared status codes (200 OK, 404 Not Found)
 - State data stored in session (SipSession)
- **Different from HTTP**
 - Asynchronous
 - Multiple servlets can receive the same message

User Registration



A Simple Call



Agenda

SIP Technology Overview

Using SIP Servlets

Simple Example (Click-to-Dial)

Enterprise Example (Conference Manager)

Scalability and Reliability

SIP Servlets API

- SIP Servlets process SIP message
 - Based on the Generic Servlet model
 - Defined by Java Specification Request (JSR) 116
- Defines high-level objects
 - Message, Request, and Response
 - SIPApplicationSession, SipSession, SipFactory
 - Timers
- Work together with HTTP Servlets
 - HTTP Servlets can initiate calls
 - Send an HTTP URL in a SIP Redirect
 - Share session data

Writing SIP Applications

- Application contains one or more SIP Servlets to handle SIP protocol messages
- Packaged and deployed as SIP Archive
 - .sar is similar to .war web archive format
 - sip.xml: SIP-specific deployment descriptor
- SIP Servlets have different methods for handling each type of SIP message
 - e.g., servlet must contain `doInvite()` method to process an INVITE message
- No context roots!

SIP Application Composition

- A message may be seen by multiple SIP Applications simultaneously
 - e.g., CallFilter and CallForward
 - Both applications express interest in INVITE Message

```
<sip-app>
  <servlet>
    <servlet-name>CallFilter
    <servlet-class>com.acme.CallFilter
    <servlet-mapping>
      <servlet-name>CallFilter
      <pattern>
        <equal><var>request.method
        <value>INVITE
```

```
<sip-app>
  <servlet>
    <servlet-name>CallForward
    <servlet-class>com.acme.CallForward
    <servlet-mapping>
      <servlet-name>CallForward
      <pattern>
        <equal><var>request.method
        <value>INVITE
```

INVITE message is seen by both applications. Order is important!

Session Management

- Every established dialog results in a SIPSession
 - Tracks one point-to-point communication session
 - Similar to HttpSession
 - Can be explicitly invalidated or timed out
- SIPApplicationSession
 - Groups many protocol sessions
 - A SIP application can have many simultaneous sessions
 - Unique to each active invocation of application
 - Reclaimed by explicit invalidation or timer
 - All child sessions must also be ready for reclamation

JSR 289: SIP Servlets 1.1

- Convergence: combine Java Platform, Enterprise Edition (Java EE platform) and SIP
- Java EE platform services in SIP applications
 - Dependency injection
 - Invoke Web Services
 - Transactions, Enterprise JavaBeans™ (EJB™) technology, and Java Persistence API
 - Java EE platform Connectors and Message Driven Beans associated with SIP Sessions
 - Co-packaging
- SIP services in Java EE platform applications
 - Initiate calls from servlets and EJBs
 - Share state between SIP and web sessions

Converged Sessions

- HttpSession and SIPSession objects are interlinked

```
HttpSession httpSession=req.getSession();
```

```
ConvergedHttpSession convSession =  
    (ConvergedHttpSession) httpSession;
```

```
SipApplicationSession appSession =  
    convSession.getApplicationSession();
```

```
Iterator<SipSession> sipSessions =  
    appSession.getSessions("SIP");
```

Agenda

SIP Technology Overview
Using SIP Servlets

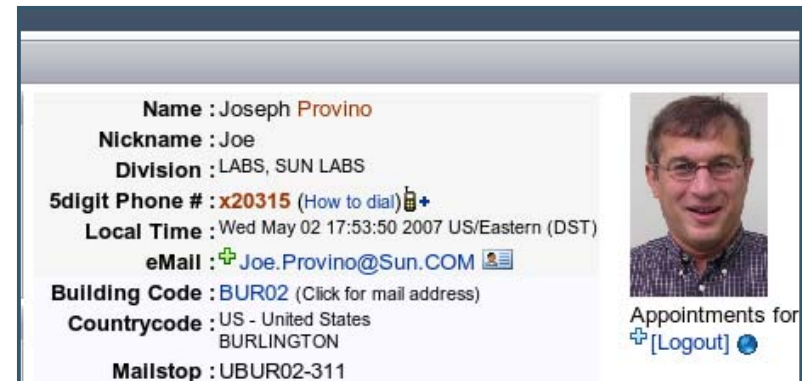
Simple Example (Click-to-Dial)




Enterprise Example (Conference Manager)


Scalability and Reliability



Click-to-Dial

- Launch and register SIP softphone
- Click on a registered attendee to place a call
 - Calls your phone first
 - When you answer, calls the other person
 - When they answer, starts the call

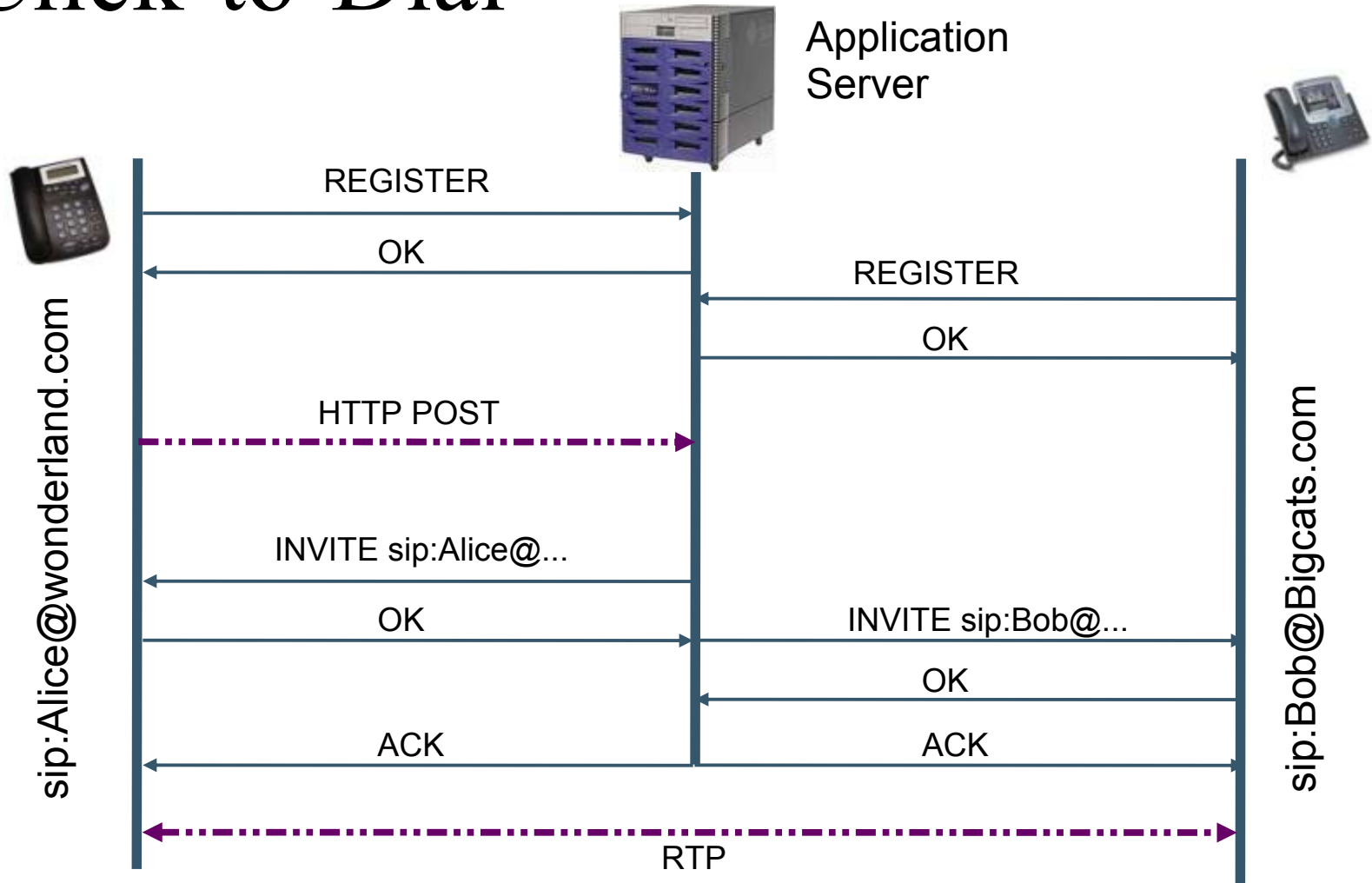


Name : Joseph Provino
Nickname : Joe
Division : LABS, SUN LABS
5digit Phone # : x20315 (How to dial) +
Local Time : Wed May 02 17:53:50 2007 US/Eastern (DST)
eMail :  Joe.Provino@Sun.COM 
Building Code : BUR02 (Click for mail address)
Countrycode : US - United States
BURLINGTON
Mailstop : UBUR02-311



Appointments for
 [Logout] 

Click-to-Dial



Registrar SIP Servlet

```
public class RegistrarServlet extends SipServlet {
    protected void doRegister(SipServletRequest req) {
        // get the data model from the servlet context
        Model m = (Model) servletCtx.getAttribute("model");

        // find Alice's object in the database
        Person a = m.getPerson(req.getFrom());

        // store the updated phone number for this person
        a.setTelephone(req.getFrom().getURI());
        m.updatePerson(a);

        // send an OK response
        SipServletResponse resp = req.createResponse(SC_OK);
        resp.send();
    }
}
```

PlaceCall HTTP Servlet

```
@Resource private SipFactory sipFactory;

protected void processRequest(HttpServletRequest req,
                             HttpServletResponse resp) {

    // get registered SIP addresses from database
    Person alice = model.getPerson(request.getSession());
    Address a = alice.getTelephone();
    Address b = getAddress(request.getAttribute("to"));

    // create a new INVITE request to Alice
    SipServletRequest req =
        sipFactory.createRequest(appSession, "INVITE", b, a);

    // store Bob's address in the sip session
    req.getSession().setAttribute("addr", b);

    // send the request
    req.send();
}
```

HandleCall: SIP Servlet

```
protected void doSuccessResponse(SipServletResponse resp) {  
  
    // retrieve the addresses from the session  
    Address a = (Address) session.getRemoteParty();  
    Address b = (Address) session.getAttribute("addr");  
  
    // send an invite to Bob  
    SipServletRequest invite =  
        sf.createRequest(appSession, "INVITE", a, b);  
    invite.setContent(resp.getContent(), "application/sdp");  
  
    // link the sessions  
    session.setAttribute("Linked", invite.getSession());  
    invite.getSession().setAttribute("Linked", session);  
  
    invite.send();  
}
```



DEMO

Click-to-Dial

Agenda

SIP Technology Overview

Using SIP Servlets

Simple Example (Click-to-Dial)

Enterprise Example (Conference Manager)

Scalability and Reliability

Sun Labs Conference Manager

- Integrates VoIP-based telephony
- Implemented on top of SailFin prototype
 - Converged SIP and web application
 - **Demonstrates possibilities of JSR 289**
- Uses Sun Labs Voice Bridge
 - Pure Java technology RTP Mixing
 - Advanced audio features
- Based on research into the most common problems in distributed meetings

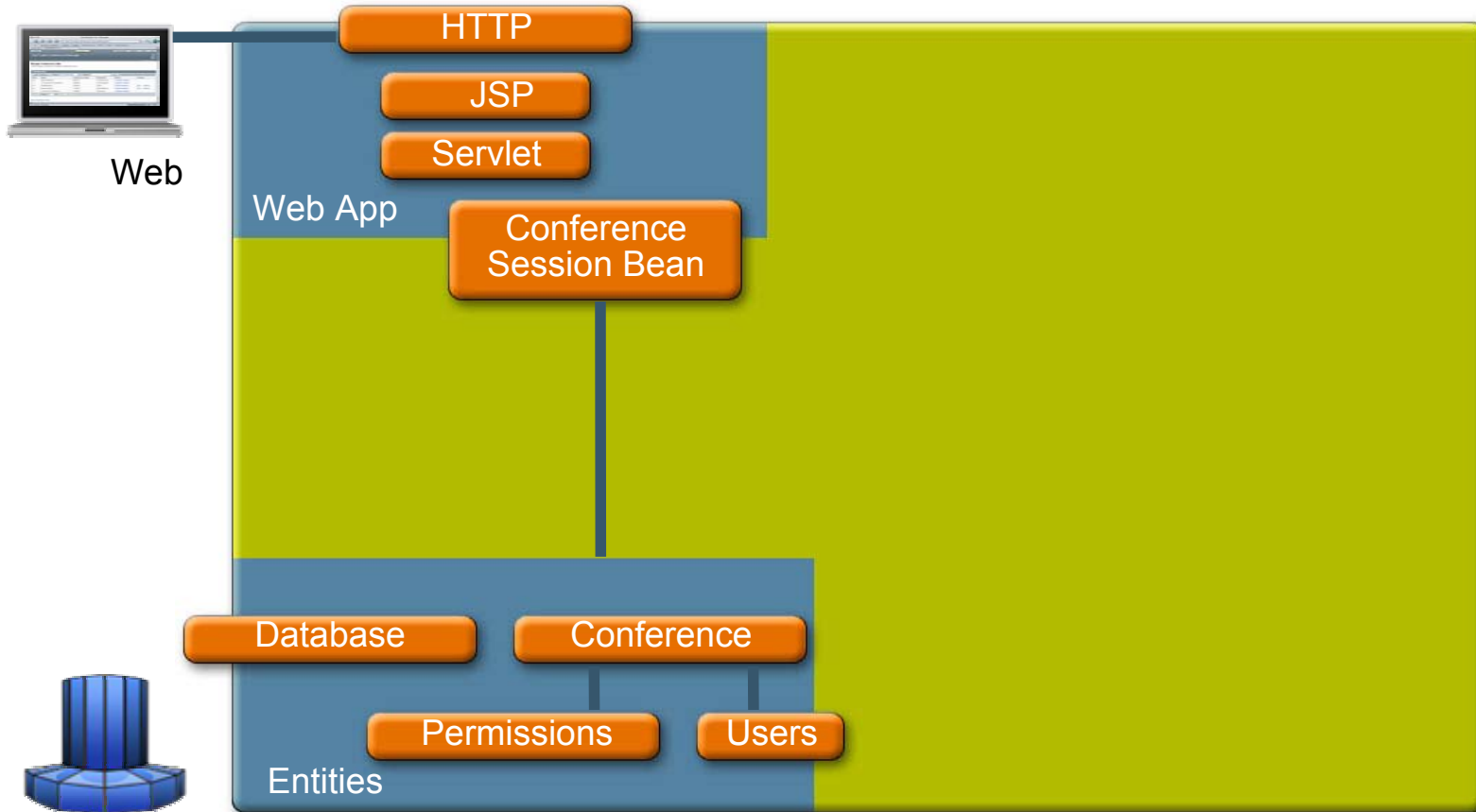


DEMO

Conference Manager

Architecture

Traditional Web App



JSP™ = JavaServer Pages™

Integrating a Media Gateway

- Runs as a separate process
- Choose strategy based on interface to gateway
- SIP-based gateway
 - Back-to-back user agent
 - e.g., MSML, MSCML
- Other gateways
 - Java EE platform Connector, Message-Driven EJB technology
 - e.g., MGCP, **Sun Labs Voice Bridge**
- **Coming Soon: JSR 309**

Java EE Platform Connector Interfaces

```
public interface BridgeConnection {
    // place a new call and return a call id
    public String placeCall(CallParticipant participant);

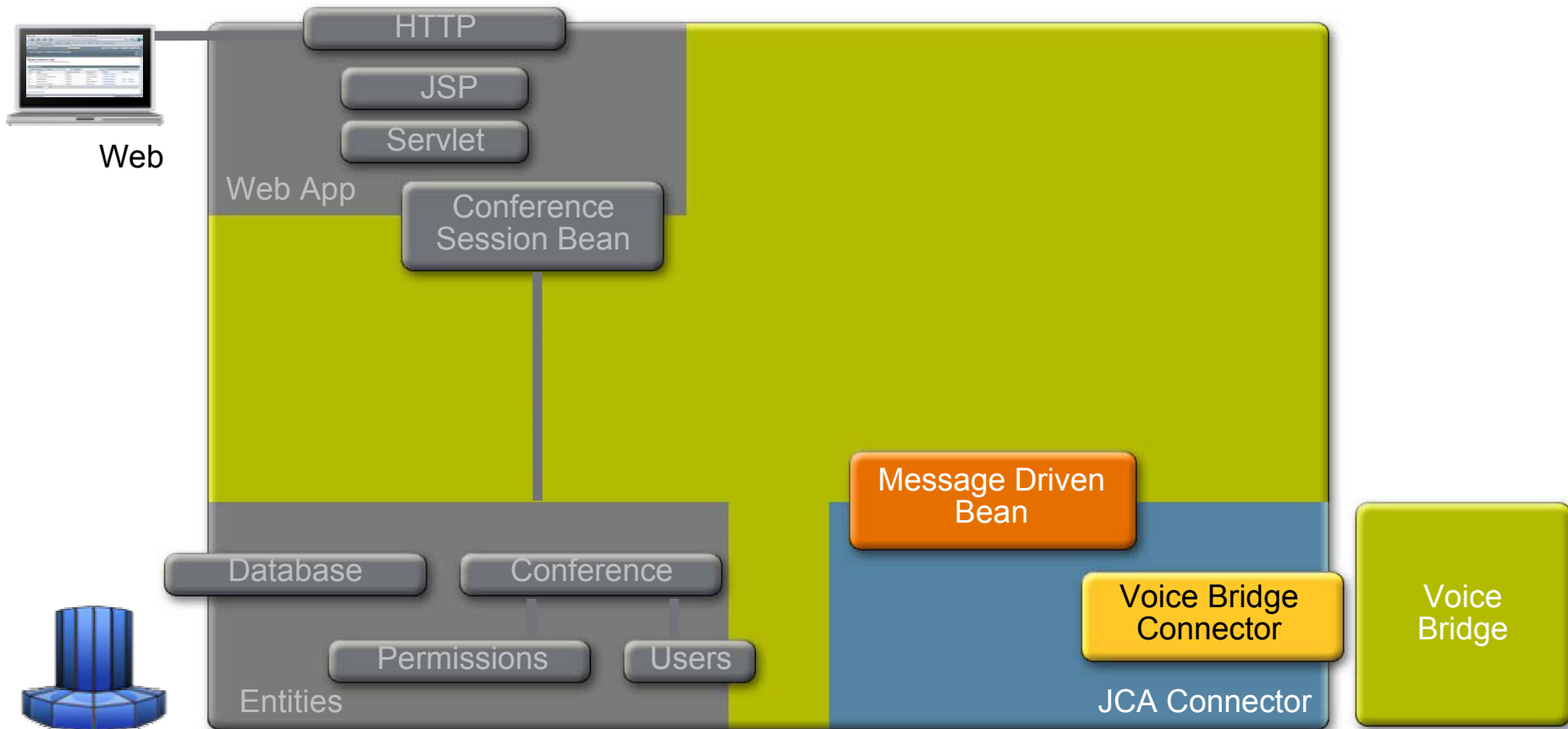
    // mute or unmute the given call
    public void setMute(String callId, boolean isMuted)
    ...

    // get the next message from the bridge
    public BridgeMessage nextMessage();
}

public interface BridgeMessageListener {
    // called when a message is received from the bridge
    public void onMessage(BridgeMessage message);
}
```

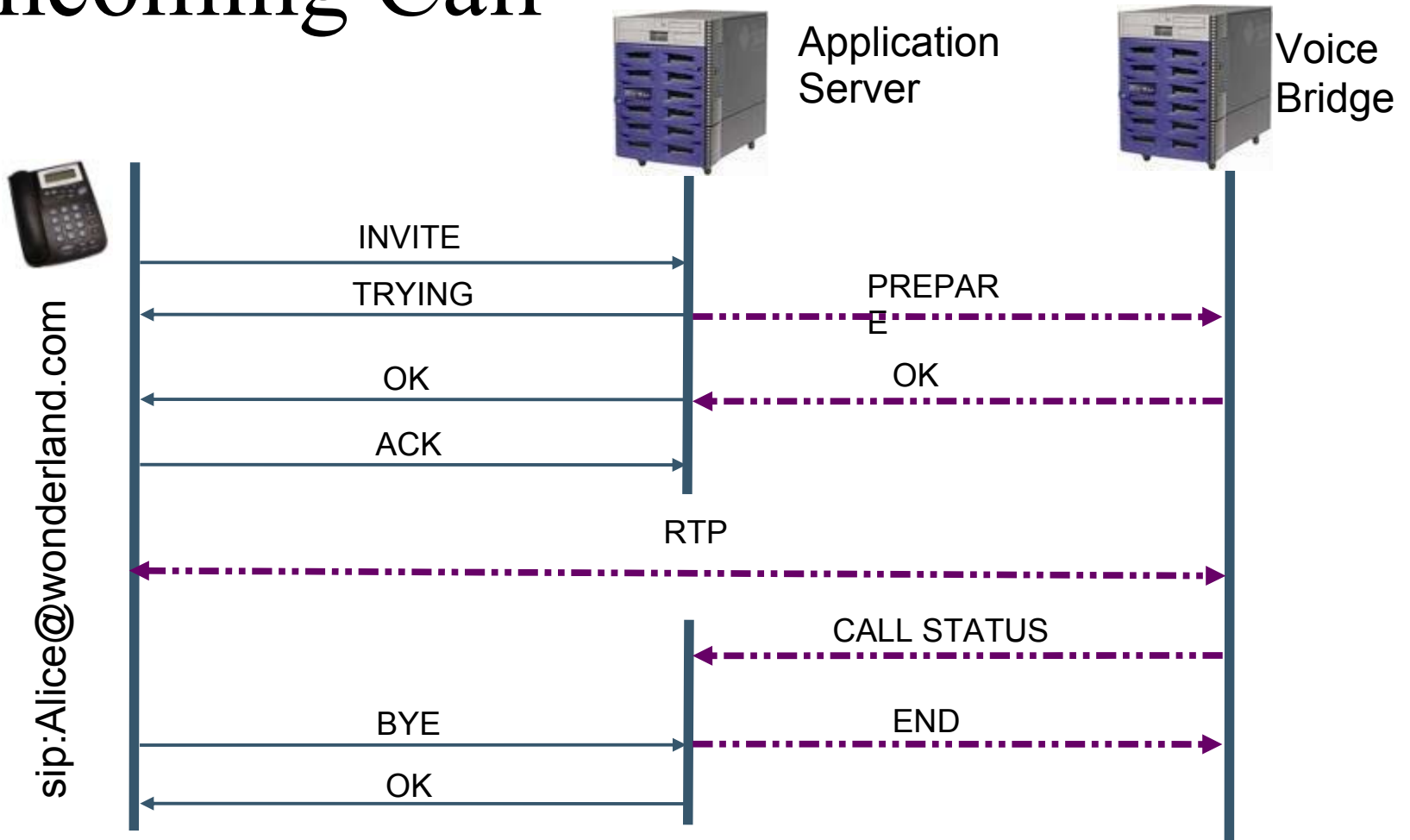
Architecture

Java EE Platform Connector



JCA = J2EE™ Connector Architecture

Incoming Call



SIP Servlet for Incoming Call

```
protected void doInvite(SipServletRequest req) {
    // notify that we are trying
    SipServletResponse response =
        req.createResponse(SipServletResponse.SC_TRYING);
    response.send();

    // place the call in the bridge
    BridgeConnection bc = bcf.getConnection();
    String callId = bc.placeCall(cp);

    // setup the session
    SipApplicationSession s = req.getApplicationSession();
    Map callMap = (Map) s.getAttribute("callMap");
    callMap.put(callId, req.getSession());
    req.getSession().setAttribute("req", req);
}
```

Managing Conference State

- SipSession
 - One per call
 - Maintain user-specific state
 - e.g., name, speaking, mute
- SipApplicationSession
 - One per conference
 - Maintain conference-wide state
 - e.g., user list, voting state

Message-Driven Bean for Call Status

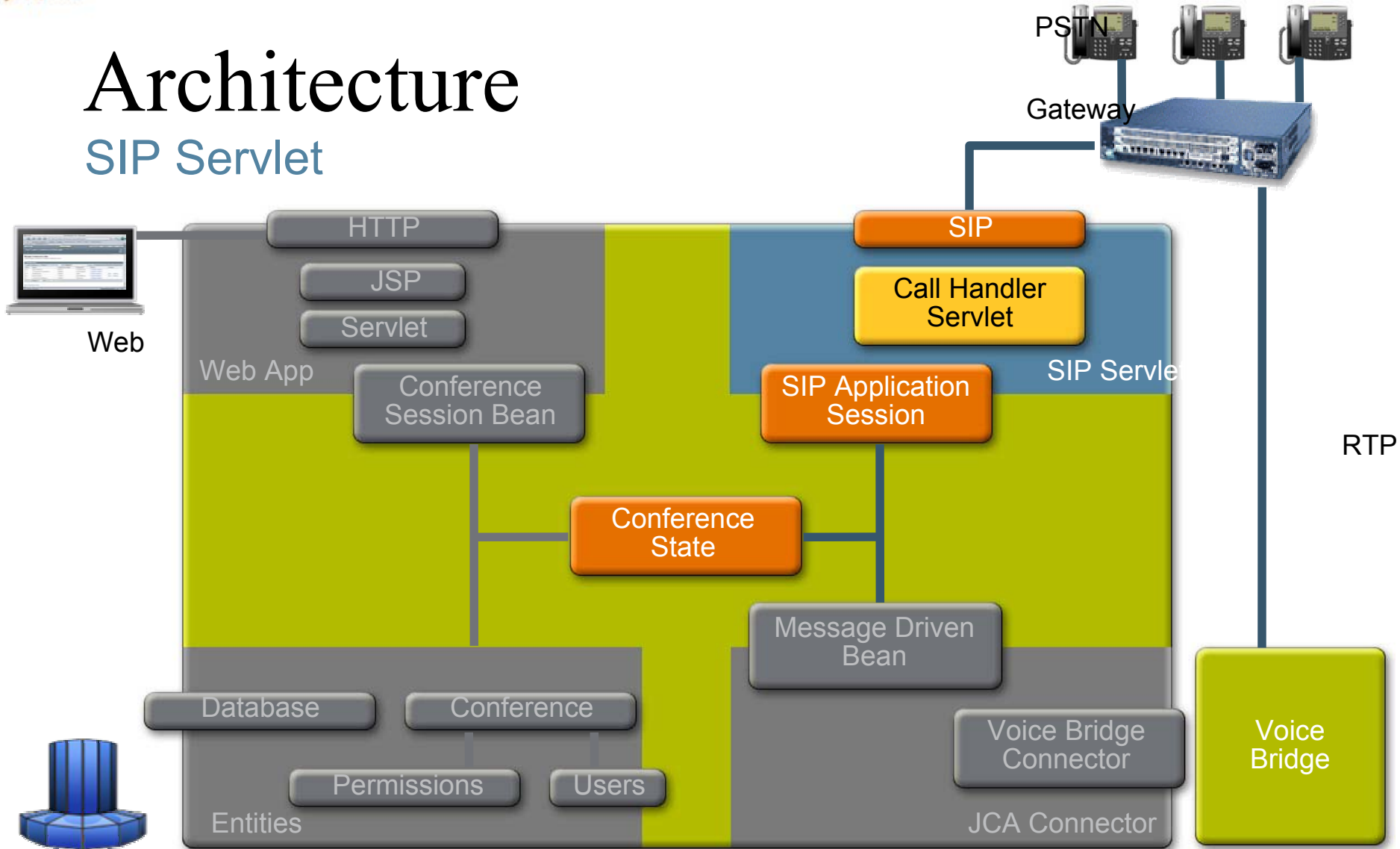
```
@MessageDriven
public class StatusBean implements BridgeMessageListener {
    @EJB private CallHandlerLocal callHandler;
    @Resource private SipSessionsUtil ssu;

    public void onMessage(BridgeMessage message) {
        // get the call from the session
        Call call = getCall(ssu, message.getCallId());

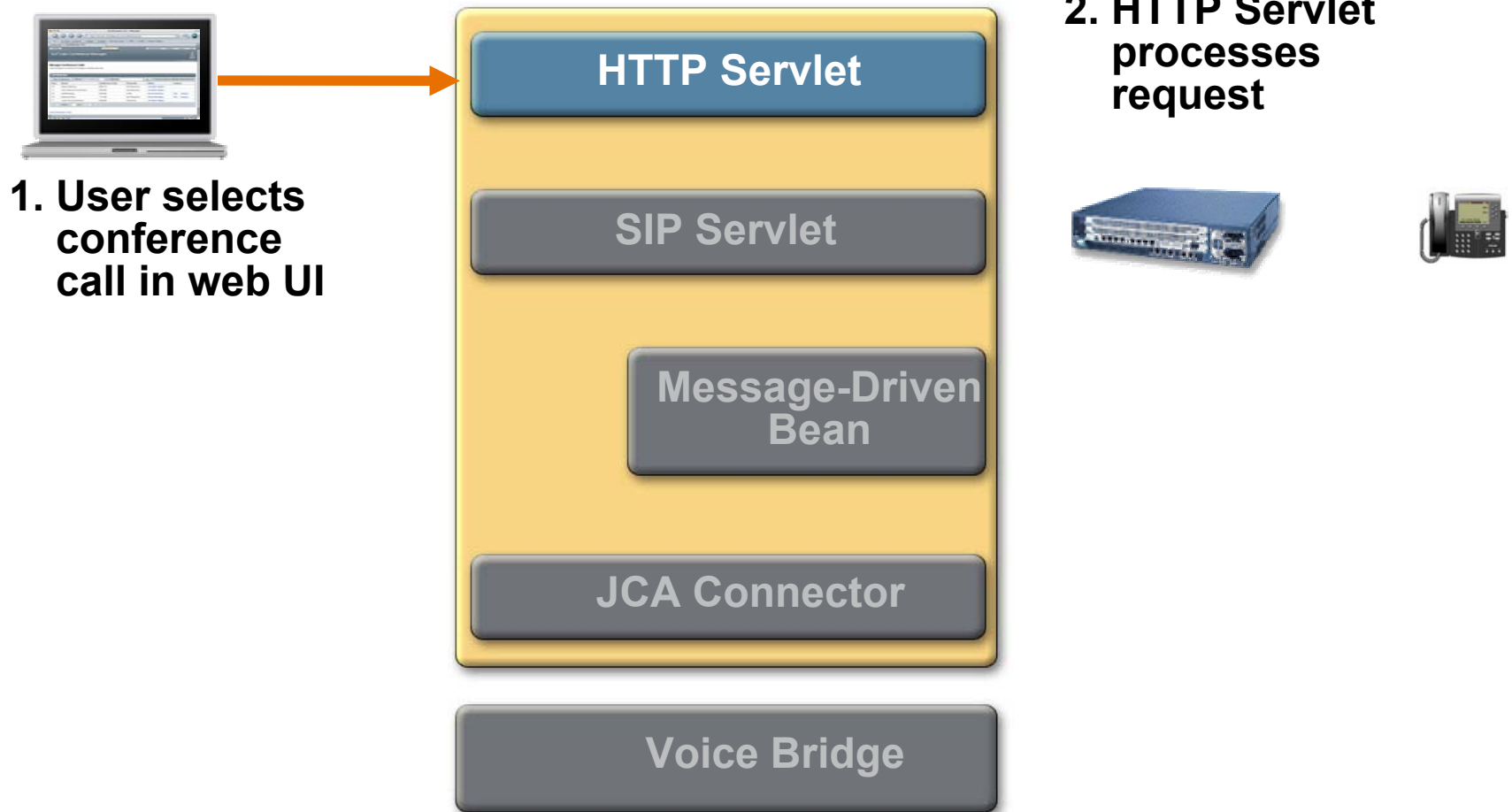
        // handle status updates
        switch (message.getCode()) {
            case STARTEDSPEAKING:
                callHandler.speaking(call, true);
                break;
            case ENDED:
                callHandler.callEnded(call);
                ...
        }
    }
}
```

Architecture

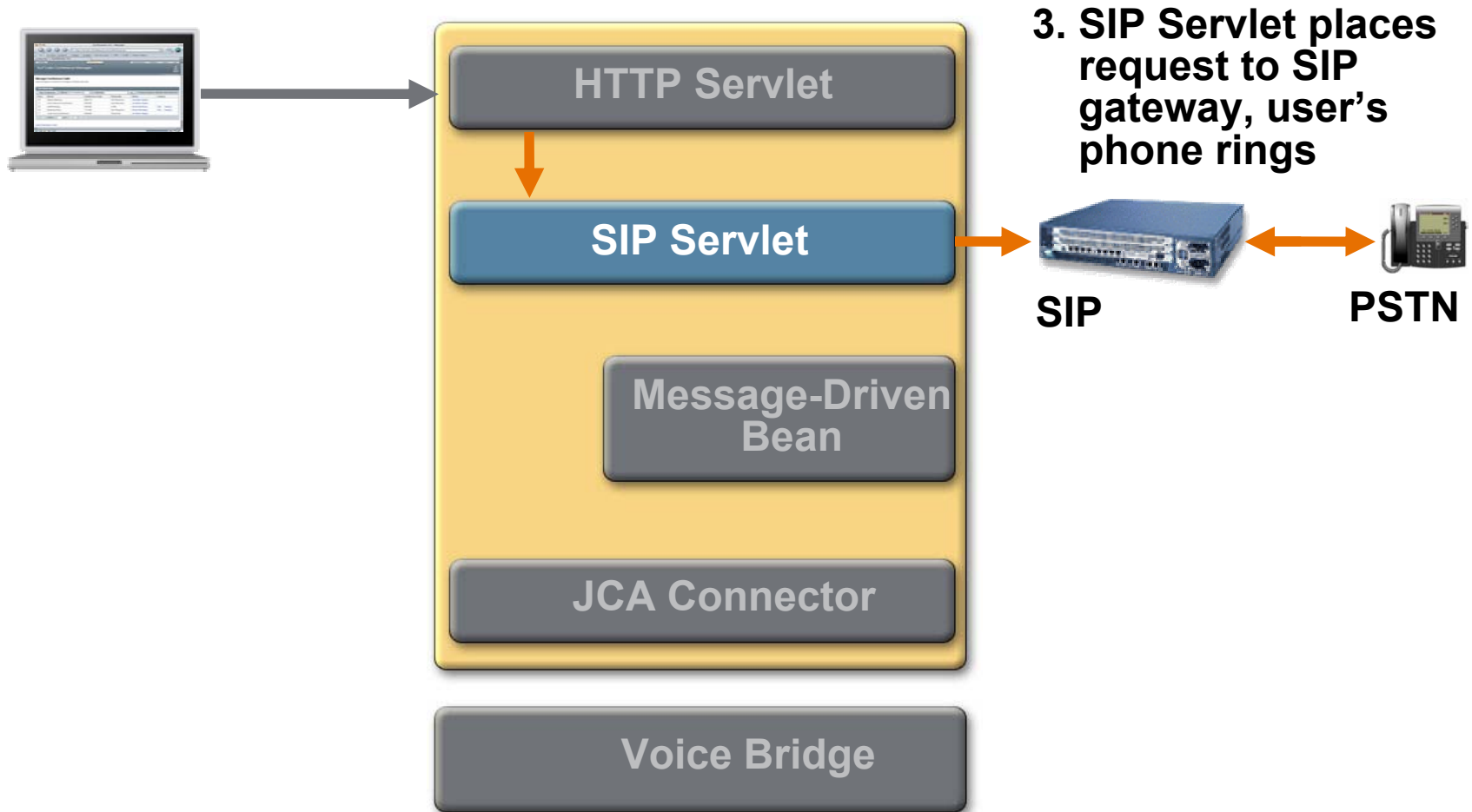
SIP Servlet



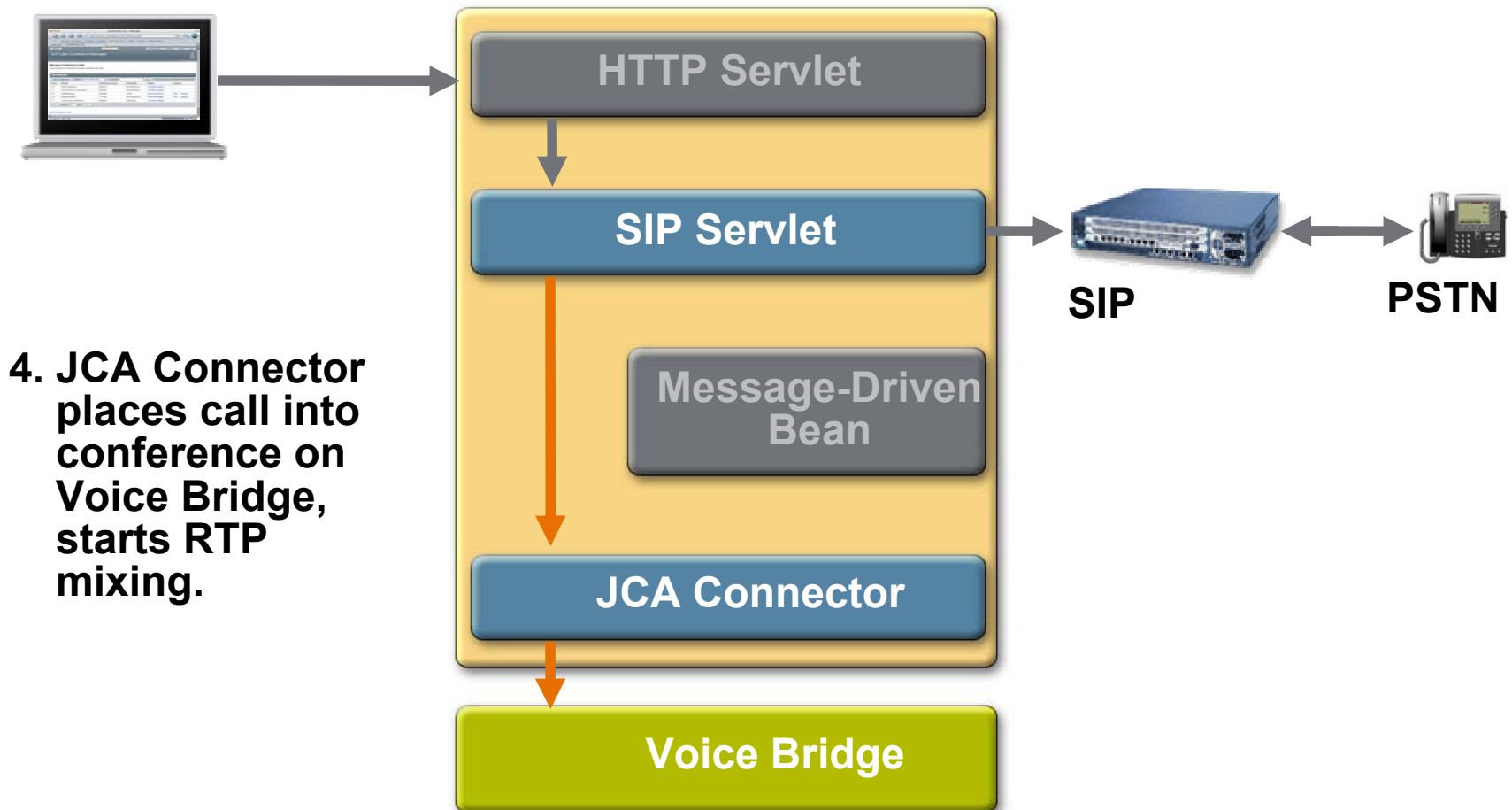
Putting It All Together: Placing a Call



Putting It All Together: Placing a Call

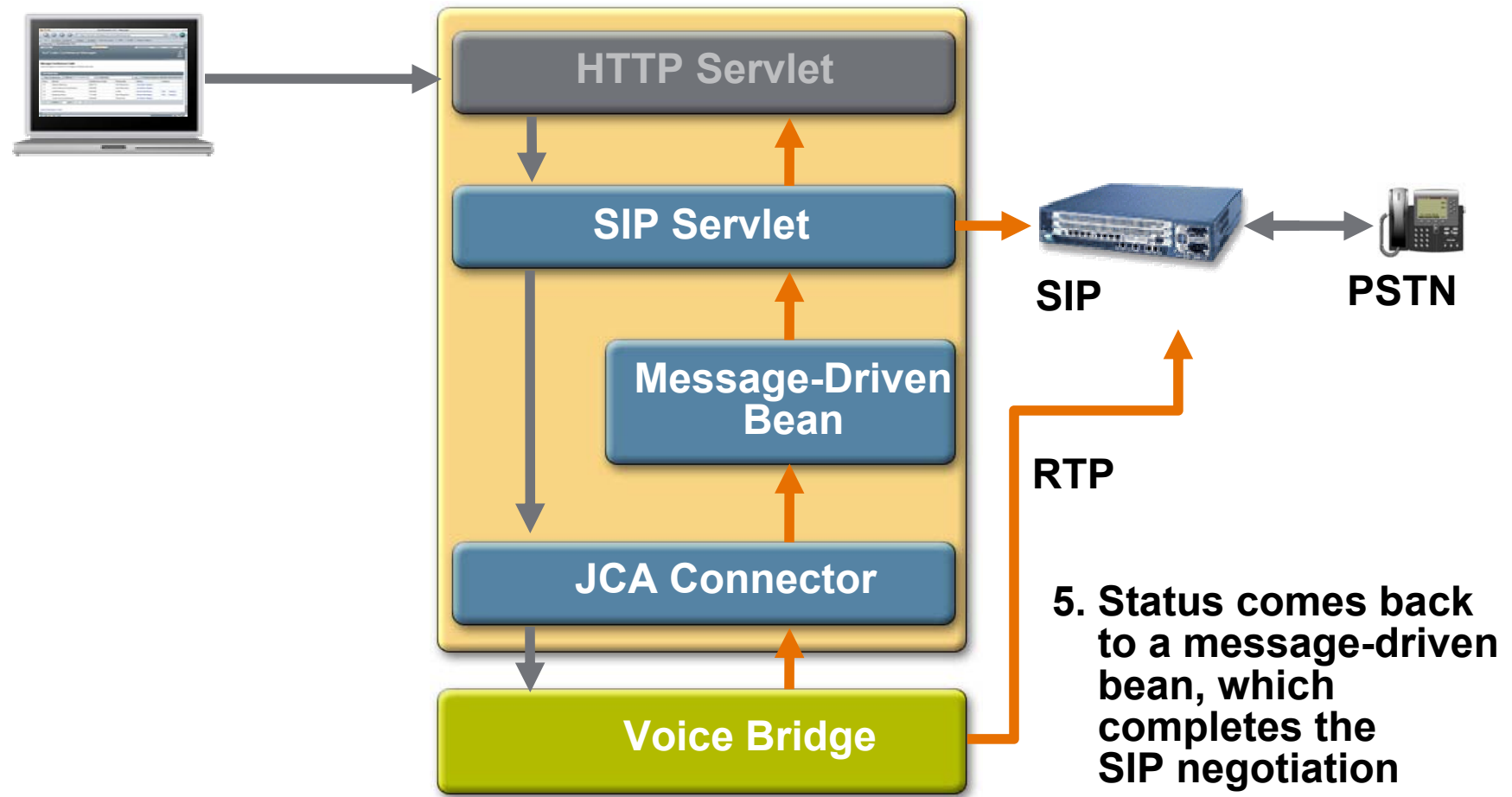


Putting It All Together: Placing a Call



4. JCA Connector places call into conference on Voice Bridge, starts RTP mixing.

Putting It All Together: Placing a Call



Agenda

SIP Technology Overview

Using SIP Servlets

Simple Example (Click-to-Dial)

Enterprise Example (Conference Manager)

Scalability and Reliability

Scalability

- SIP Load Balancing works on header fields
 - Session-Id stored in Via header
 - Call-Id
 - From and To headers
 - Other policies like actual server load
- Usually sticky SIPSessions
- Complexities for conference calls
 - Conference calls with thousands of callers
 - Callers are scattered across the world
 - For performance reasons good to direct all calls in a conference call to same server instance in cluster

Scalability

- Too many callers in a conference
 - Centralized audio mixing can get expensive
 - User's voice quality experience can degrade
 - Solution: hierarchical mixing with careful handling of timestamps
- Participants are from scattered geographies
 - Aggregate users based on source IP address
 - Local audio mixing and upstream transmission
 - Receive remote mixed stream and match with locally mixed audio

Summary

- JSR 289 brings communications to enterprise Java applications
- Java EE platform enables development of powerful communications applications
 - From click-to-dial to multi-user collaboration
 - Add multimedia to existing applications
- Just the beginning
 - Need a better multi-user collaboration frameworks

For More Information

- Project SailFin
<http://sailfin.dev.java.net>
- GlassFish™ Project
<http://glassfish.dev.java.net>
- SIP Servlets 1.1
<http://jcp.org/en/jsr/detail?id=289>
- Conference Manager
<http://research.sun.com/projects/mc/confmgr.html>



Q&A

Jonathan Kaplan & Sreeram Duvur



Project
GlassFish



JavaOne

Adding Telephony to Java™ Technology-Based Enterprise Applications

Jonathan Kaplan & Sreeram Duvur

Researcher/Architect
Sun Microsystems, Inc.
<http://glassfish.dev.java.net/>

TS-4919