Junction: A Decentralized Platform for Ad Hoc Social and Mobile Applications

Ben Dodson, Monica Lam, Chanh Nguyen, Te-Yuan Huang
Motivation
Motivation

- **Ad Hoc**
  - Bring together devices with no previous contact

- **Mobile**
  - Leverage the personalized computing power in our pockets

- **Social**
  - Bring multiple users together for real-time applications

Requirements of multi-party, ad hoc activities

- 1. Device discovery
- 2. Code deployment
- 3. Communication
Demo: weTube
Ad Hoc Activities

- Multimedia sharing
  - Bring your personality with you (your music, your photos, ...)
- Multiplayer games
  - Public / private screens for new types of play
  - No single application server
- Secure web transactions
Consumer-Friendly Secure Web Transactions

Snap2Pass
Challenge-response web auth.

Snap2Pay
One-time use credit cards
Contributions

Make developing decentralized, multi-party activities easy.

- Activity Director: Join activities with a single click
  - Junction URIs for easy sharing

- Quick invitations for nearby activities
  - QR codes, Bluetooth beacon

- Cross-platform programming
  - Activity script for resolving platform-specific code

- Switchboard: A universal, app-agnostic messaging service
  - Separate resource provisioning from app development
Architecture

Web application

Junction activity
Sharing Activities

- Activity sessions are represented as URIs.
- The URI allows a participant to look up the activity script as well as join the activity.

junction://prpl.stanford.edu/126d95e0fbce11de8a?role=player
Activity Script

- Cross-platform programming via an activity script
  - Defines the roles of an activity
    - Also specifies platforms and codebases for each role
  - Defines unique identifier for activity, as well as a friendly name
  - Represented as JSON
weHoldEm's Activity Script

{ ad: "edu.stanford.prpl.poker",
  friendlyName: "weHold'Em",
  roles: { "player": {platforms: 
    {android: {package: "edu.stanford.prpl.poker",
      url: http://prpl.stanford.edu/poker/downloads/poker.apk}}}, 
    {web: { url: "http://prpl.stanford.edu/poker/play" }}}
  "table": {platforms: 
    { web: {url: "http://prpl.stanford.edu/poker/table" }},
  "dealer": {platforms: 
Sharing Activities :: Activity Director

- Director, like a web browser, enables “click-and-run” for activities
  - Handles junction:// URIs in a universal way
  - Active usage: Phone joins nearby activity
  - Passive usage: Push application to settop box
Sharing Activities :: Invitations

- Use the Activity Director to share invitations in useful ways
  - QR Code: Secure, visual channel, supported on different platforms
  - SMS: Invite mobile friends at a distance
  - Email, IM, Blog, etc. (like web links)
  - Activity developers expose sharing to users; Junction library makes this easy.

- Director accepts any invitation and launches the appropriate app
Switchboard and Client Implementation

- Switchboard built on standard, out-of-the-box XMPP
  - Uses Multi-User Chat extension
    - All Junction operations containable in MUC
  - Works on most standard XMPP implementations
  - Use BOSH for HTTP connections
- Three client platforms currently supported
  - Javascript (uses StropheJS XMPP library)
  - Desktop JAVA (uses Smack)
  - Android (JAVA library augmented with platform-specific hooks)
- Fairly easy to add new platforms
  - Many existing XMPP client libraries
Switchboard Service

**Advantages**
- Scalability
- Separation of concerns
- Choice of privacy and economic models
- High-level abstraction enables optimization (OpenFlow)
- Open API

**Deployment models**
- Application-dedicated
- Personal or institutional
- Peer-to-peer
- Vicinity
- Network operator
SFNet – Multicast Delegation
# Junction Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Lines of Code Per Role</th>
<th>Dev. Days</th>
<th>Ad Hoc</th>
<th>Platform-spanning</th>
<th>Personal Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Social apps: Communication</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weMeet</td>
<td>200</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>weClick</td>
<td>Instructor: 140, Player: 120</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Multimedia: Quick and fun collaboration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weTube</td>
<td>Player: 450, Phone: 600</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>weTunes</td>
<td>Jukebox: 520, Remote: 420</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Games: Scalable, distributed applications</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weBluff</td>
<td>1500</td>
<td>3</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weHold’Em</td>
<td>Player: 800, table: 750, dealer: 1700</td>
<td>30</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td><strong>Personal apps: use the phone and the PC together</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Snap2Pass</td>
<td>Host: 1600, web: 120, phone: 400</td>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Snap2Web</td>
<td>Browser: 320, phone: 400</td>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

- Device-spanning activities
- Ease of use: “Click-and-run” for ad hoc activities
- Ease of development: Multiparty activities
- Ease of deployment: Separate app development from infrastructure concerns with a switchboard
PocketSchool Interactive Learning Adhoc Network

- Story comprehension
- Encourage learning through competition.
- Paul Kim, Sch. of Educ., Sunnyvale School District, Feb 2010
Ad-Hoc Game Between Phones

Start Activity
Invite by SMS

Accept
Download software
Join activity
Hello Junction (Javascript)

/*[ Javascript includes and HTML gui omitted. ]*/
/* Activity Script */
var spec = /* JSON activity script */;

/* Actor */
var receiver = { roles: [ "receiver" ],
onMessageReceived:
    function(msg) {
        $('#inbound').append(msg.text);
    }
};

/* Binding */
var jm = JunctionMaker.getInstance("prpl.stanford.edu");
var jx = jm.newJunction(spec, receiver);

/* Invitation URI */
var invite_qr = jx.getInvitationQR("sender");
$('invitationQR').attr('src',invite_qr);
Applications can expose different mechanisms for sharing activities
Can also utilize the Director for exposing all possible methods supported by the platform

// Show a QR code in Javascript
var imgUrl = jx.getInvitationQR("player");
$('#qr_img').attr(src, imgUrl);

// find an activity by QR in Android
AndroidJunctionMaker
  .getInstance()
  .findActivityByScan(this);

// General-purpose invite in Android
AndroidJunctionMaker
  .getInstance()
  .inviteActor(this, jx, "player");
Motivation

- Mobile phones as identity device
  - Store files, bookmarks, shared secrets
  - Stronger privacy than any cloud service
  - Always connected
Sharing Activities :: Casting

- Phones are actively used by users
- Other devices (servers, settop boxes) not as much
- For them, Director passively accepts invitations