ConnectJS: A cross mobile platform actor library for multi-networked mobile applications

Elisa Gonzalez Boix, Christophe Scholliers, Nicolas Larrea, Wolfgang De Meuter

AGERE! 2015
Mobile Wireless Applications

- Applications running on mobile ad hoc networks

- Wireless communication with volatile connections
- Spontaneous interactions with little infrastructure
Ambient-oriented Programming in AmbientTalk

- Programs are event loops that communicate **asynchronously**.
- Far references **tolerate** network failures.

```plaintext
deftype InstantMessenger;
  whenever: InstantMessenger discovered: {
    |messenger|
    def future := messenger<-getName();
    when: future becomes: {
      |name|
      buddyList.put(name, messenger);
      display("Added buddy:" + name);
    }
  }
  export: chat as: InstantMessenger;
```

<table>
<thead>
<tr>
<th>'local' object</th>
<th>'remote' object</th>
</tr>
</thead>
<tbody>
<tr>
<td>buddyList</td>
<td>chat</td>
</tr>
<tr>
<td>messenger</td>
<td>getName()</td>
</tr>
<tr>
<td>? future</td>
<td>resolve(name)</td>
</tr>
<tr>
<td>message queue</td>
<td>event loop</td>
</tr>
</tbody>
</table>

AmbientTalk

AmbientTalk
Cloud-based Applications

CLIENTS

HTML
JS
CSS

application servers
geocode service
partner service
media storage

search engine
content site
social content

CRM service

Google

PhoneGap

titanium

Twitter
Rich Mobile Applications
Programming Rich Mobile Applications

How to ease the development of mobile applications that communicate over multiple networking interfaces?

How to alleviate the pyramid of doom?
When the web meets mobile computing …

• makes it possible to create mobile applications using a single code base (i.e. JavaScript) instead of device-specific languages

and run natively on multiple platforms
but what about mobile wireless applications?
Cross-platform ambient-oriented library

AmbientJS

Titanium

OS

aMobileApplication.js

Uniform Distributed API

UI API

Phone API

other

JavaScript - Java/ObjectiveC Bridge

Native Application

iOS

Android
var Ambient = require('js/ambient/ambient');
var buddyList = {};

function initializeMessenger(name) {
    var remoteInterface = Ambient.createObject({
        "getName" : function () { return myName; },
        "talkTo" : function (msg) { displayMessage(msg);}
    });

    Ambient.exportAs(remoteInterface, "MESSENGER");
    Ambient.wheneverDiscovered("MESSENGER", function(reference){
        var msg = Ambient.createMessage("getName",[]);
        var future = reference.asyncSend(msg, "twoway");
        future.whenBecomes(function(reply) {
            buddyList[reply] = reference;
        });
    });

    Ambient.online();
}

function sendMessage(text, buddyName){
    var msg = Ambient.createMessage('talkTo', [myName + ": " + text]);
    buddyList[buddyName].asyncSend(msg);
}
How to reconcile P2P and client-server styles?
Network-transparent References

- Far references that abstract over underlying networking technology and provide at most once delivery guaranties.
Network-transparent References (NTRs)

- Far references that abstract over underlying networking technology and provide at most once delivery guarantees.
Network-transparent References (NTRs) 2

- Designates a **single** unique object over multiple networking interfaces

Unified mailbox, dequeing places a message to be sent via a networking interface

ddequeing tags the message with a sequence number and sends it

Service-side of the NTR checks takes care of duplicates and lost messages
How to alleviate the Pyramid of Doom?

```javascript
varUrls = [
  'http://api.openweathermap.org/data/2.5/weather?q=Pittsburg',
  'http://api.events.org/data/2.5/date?q=25-10-2014'];

function recommend(weatherinfo, events) {
  ...
};

httpGet(Urls[0]).then(function(weatherinfo) {
  httpGet(Urls[1]).then(function(events) {
    recommend(weatherinfo, events);
  })
});
```

Future Combinators

\[
\text{Future.of} (\text{recommend}) \\
\quad .ap(\text{httpGet}(\text{Urls}[0]))
\quad .ap(\text{httpGet}(\text{Urls}[1]))
\]
ConnectJS

• Cross-platform ambient-oriented library which explores:
  
  • **network transparent references** for seamlessly communication over the cloud or an infrastructureless mobile networking technology.
  
  • **future combinators** for structuring asynchronous code.

http://soft.vub.ac.be/ambientJS/