1. Many-to-Many Invocation:
A New Object Oriented Paradigm to build Software Infrastructures
for Ad Hoc Collaborative Systems

Hans-Peter Bischof  Alan Kaminsky
hpb@cs.rit.edu     ark@cs.rit.edu

Rochester, NY, USA
November/17/2002
2. Current Infrastructure

2.1. Current Hardware Infrastructure
2.2. Administration

- ip addresses
- router in place
- routing tables
- ... and more
2.3. Current Software Infrastructure

Router

129.21.30.26
nfs server

web server
129.21.30.99

print server
ie

Router

129.21.30.26
nfs server

mount

web/nfs
2.4. Typical Communication Pattern

- one to one: ip addresses and ports
- one to many: broadcast/multicast
- many to a many: multicast and channels
2.5. Typical Ways to Communicate

- Sending data
- Invoking methods
2.6. A typical Client-Server Architecture
2.7. A typical Client-Server Architecture Problem

- If the server goes away, the whole application dies, even though the clients can communicate directly.
- Helpful: No central servers!
3. Different Kind of Infrastructure

3.1. Different Kind of Hardware Infrastructure
3.2. Different Kind of Environment: Ad Hoc Network

- Play at the theater is a little boring
- Meetings: play backgammon/poker/chatting
- NFL Superbowl XXXVIII
3.3. A typical Serverless Architecture
3.4. Different Kind of Software Infrastructure

- To simplify programming
  - Object oriented abstraction of many-to-many communication
  - Broadcast method invocations: M2MI

- To simplify deployment
  - No proxy compilers, codebase servers, activation daemons, ...
  - Automatic proxy synthesis

- To simplify operation and administration
  - No network addresses, ad hoc routing protocols, ...
4. M2MI: A New Paradigm for Ad Hoc Collaborative Systems

- M2MI provides an object-oriented method call abstraction based on broadcasting.
- M2MI-based systems do not require central server
- M2MI-based systems do not require network administration
- M2MI simplifies system deployment by eliminating the need for always-on application servers
- M2MI is well-suited for an ad hoc networking environment where central servers may not be available
4.1. References

- Omnihandle: Refers to all objects that implement an interface

- Multihandle: Refers to a group of objects that implement an interface

- Unihandle: Refers to one object that implements an interface
4.2. Using Omnihandles

- Export remote objects
  
  ```java
  M2MI.export (a, Foo.class);
  ```

- Get an omnihandle
  
  ```java
  Foo allFoos = (Foo) M2MI.getOmnihandle (Foo.class);
  ```

- Invoke a method on the omnihandle
  
  ```java
  allFoos.y();
  ```
4.3. Using Multihandles

- Get a multihandle
  
  ```java
  Foo someFoos = (Foo)M2MI.getMultihandle(Foo.class);
  ```

- Attach objects
  
  ```java
  someFoos.attach(a);
  ```

- Invoke a method on the multihandle
  
  ```java
  someFoos.y();
  ```
4.4. Using Unihandles

- Export remote object and get unihandle
  
  ```java
  Foo b_Foo = (Foo)M2MI.getUnihandle(b, Foo.class);
  ```

- Invoke a method on the unihandle
  
  ```java
  b_Foo.y();
  ```
4.5. Characteristics of M2MI Invocations

- M2MI is an object oriented abstraction of many-to-many communication.

- Semantics of M2MI:
  - Methods may have arguments
  - Objects passed by copy (object serialization)
  - Handles give pass-by-reference
  - M2mi methods can not return a value.
  - M2mi methods can not throw an exception
  - Parameters are passed as pass-by-value.
  - Method calls are non-blocking
5. M2MI Based Application

5.1. The Idea for a Chat Application

```java
allChats.setMessage("Hello there");
```

```java
allChats.setMessage("Hi y'all");
```

```java
allChats.setMessage("Greetings folks");
```
5.2. The Chat Interface

```java
public interface Chat {
    public void putMessage(String line);
}
```
5.3. Chat Object Source Code

```java
public class ChatObject implements Chat {
    private String myUserName;
    private Chat allChats;

    public ChatObject (String theUserName) {
        myUserName = theUserName;
        M2MI.export(this, Chat.class);
        allChats = (Chat)M2MI.getOmnihandle(Chat.class);
    }

    public void send(String line) {
        allChats.putMessage(myUserName + ' ' + line);
    }

    public void putMessage(String line) {
        myChatFrame.addLineToLog(line);
    }

    public static void main (String args []) {
        String input;
        ChatObject aChatObject = new ChatObject(args[0]);
        while ((input = readFromTerminal()) != null)
            aChatObject.send(input);
    }
}
```
6. Service Discovery

- Service discovery today can done for example with JINI
- Requires a server
- How can a service discovery work in an ad hoc network?
- There are no sever in an ad hoc network.
6.1. The Idea

```
printDiscovery.request (theClient);

theClient.report (c_Printer, "C");

theClient.report (a_Printer, "A");

theClient.report (b_Printer, "B");

c_Printer.print (theDocument);
```
6.2. The Interfaces

```java
public interface PrintDiscovery {
    public void request(PrintClient client);
}

public interface PrintClient {
    public void report(PrintService printer, String name);
}

public interface PrintService {
    public void print(Document doc);
}
```
6.3. Other M2MI Applications

- Conversations
  Conversations in quiet spaces, conversations in noisy spaces, . . .

- Groupware
  Presentations, whiteboard, note taking, file sharing, document authoring, calendar scheduling, . . .

- Sensor networks
  Video surveillance, medical monitoring, battlefield intelligence, . . .

- Middleware frameworks
  - Shared tuple spaces, . . .

- Multiplayer Games!
7. M2MI Architecture

7.1. Layers

- Application Layer
- Invocation Layer
- Messaging Layer
- Data Link Layer
- Physical Layer

\{ M2MI \\
\{ M2MP \\
\{ Ethernet, 802.11, Bluetooth, \ldots \}
7.2. Software Architecture
8. Status

- Initial version of M2MI written in Java
- Tested on desktop hosts
- Some performance and throughput measurements done
- Uses UDP/IP for transport
- Another version uses Ethernet raw sockets for transport
- Several M2MI-based collaborative applications developed Chat, IM, whiteboard, calendar, file sharing, tuple space
8.1. In Progress

- M2MI monitoring API
  - Observe and debug M2MI invocations flowing through the network

- M2MI security
  - Confidentiality, participant authentication, service authentication
  - Serverless techniques: Zero knowledge proofs, . . .
  - Elliptic curve based techniques
8.2. Future Plans

- Go small and wireless
  - Port M2MI and M2MP to small mobile devices
  - Test with wireless networking

- Push M2MP into the kernel

- Develop lots of M2MI-based applications in a variety of domains

- Devise reusable design patterns and class libraries for M2MI-based collaborative applications
9. Questions

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>,</th>
<th>-</th>
<th>.</th>
<th>/</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>:</td>
<td>;</td>
<td>&lt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>@</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td>M</td>
<td>N</td>
<td>O</td>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>S</td>
<td>T</td>
<td>U</td>
<td>V</td>
<td>W</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>\</td>
<td>]</td>
<td>^</td>
<td>_</td>
<td>.</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td>k</td>
<td>l</td>
<td>m</td>
<td>n</td>
<td>o</td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>u</td>
<td>v</td>
<td>w</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>
10. Contact Information

The Anhinga Project
http://www.cs.rit.edu/~anHINGA/

Hans-Peter Bischof
http://www.cs.rit.edu/~hpB/
hpB@cs.rit.edu

Alan Kaminsky
http://www.cs.rit.edu/~ARK/
ark@cs.rit.edu