A Critique of JCSP Networking

Kevin Chalmers, Jon Kerridge and Imed Romdhani

School of Computing
Napier University
Breakdown

- Introduction
- Object Serialization in Java
- Current JCSP Implementation
- Problems
- Towards a Better Solution
- Conclusion
Introduction

• Numerous network aware CSP inspired frameworks
  – JCSP, pony, CSP.NET, C++CSP, PyCSP

• Majority based on T9000 virtual channel model
  – Links and channels multiplexing over one another

• None interact
  – No reason why not

• Performance usually judged on task parallelisation
  – I’m interested in the communication abstraction
Introduction

- Mobile devices
- Mobile processes (agents)
- Ubiquitous computing
  - Environment of autonomous interacting devices
  - Complex
Object Serialization in Java

- Functionality
- Example – Integer object
Object Serialization in Java

• Java object ↔ bytes

• Requirements
  – Serializable or Externalizable interface
  – ObjectInputStream and ObjectOutputStream

• Class description and object data sent
  – Class description includes inheritance information

• Control signals

• Use of references in the stream
  – Aliasing is a problem
Object Serialization in Java

- **Integer object**
  - Extends **Number**
  - Wraps 32-bit value in an object
Current JCSP Implementation

- High level architecture
- Virtual channel
- Message hierarchy
Current JCSP Implementation

- Virtual channel
  - `NetChannelOutput` to `NetChannelInput` (via the `LinkTx` and `LinkRx`)
  - At least five processes required
Current JCSP Implementation

```
Message
#destIndex : long
#sourceIndex : long
#dest/CNLabel : string

Channel Message
Message.BounceMessage
Message.PingMessage
Message.PingReplyMessage
ConnectionMessage

ChannelMessage.Data
#data : object
#acknowledged : bool

ChannelMessage.Ack
ChannelMessage.WriteRejected

ConnectionMessage.Open
<data : object

ConnectionMessage.Close
<data : object

ConnectionMessage.Ping
<data : object

ConnectionMessage.Pong
<data : object
```
Problems

- Resource usage
- Complexity
- Message cost
- (Java) objects only
- Performance
- High priority Link processes
- Exception handling
- Lack of protocol
Problem – Resource Usage

• Numerous processes in operation
• Start up
  – LoopbackLink (2), LinkServer, LinkManager, EventProcess
• Extra set up
  – First NetChannelOutput creates handler (CNS requires one)
  – CNSService process
• Five processes created and destroyed during Link creation
• Subsequent operations
  – Each Link to a Node requires two processes (CNS Link)
  – Each NetChannelInput requires one process (one to CNS)
• PDA limited to 400 threads
  – Standard initialised Node uses 11 (including main)
Problem - Complexity

- Subjective
- Difficult to extend functionality
- Difficult to extract functionality
- Difficult to modify functionality
- Premise is simple
  - Crossbar between Links and channel ends
- Implementation complicated
Problem - Complexity
Problem – Message Cost

• Object messages are expensive
  – Serialization of message + serialization of sent object

• Object streams reset after each send
  – Internal pool of messages
  – Aliasing on stream
  – Class information sent each time

• Deal of sent information
  – Type, destination, source and possibly data
Problem – Message Cost

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TC_OBJECT</td>
<td>TC_CLASSDESC</td>
<td>Name length (32)</td>
<td>o</td>
<td>r</td>
<td>g</td>
<td>.</td>
<td>j</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>s</td>
<td>p</td>
<td>.</td>
<td>n</td>
<td>e</td>
<td>t</td>
<td>.</td>
<td>C</td>
<td>h</td>
<td>a</td>
</tr>
<tr>
<td>20</td>
<td>n</td>
<td>n</td>
<td>e</td>
<td>I</td>
<td>M</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
<td>g</td>
</tr>
<tr>
<td>30</td>
<td>e</td>
<td>$</td>
<td>D</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>Class Serialization Identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Flags</td>
<td>Variable count (2)</td>
<td>Z (boolean)</td>
<td>Name length (12)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>a</td>
<td>c</td>
<td>k</td>
<td>n</td>
<td>o</td>
<td>w</td>
<td>l</td>
<td>e</td>
<td>d</td>
<td>g</td>
</tr>
<tr>
<td>60</td>
<td>e</td>
<td>d</td>
<td>L (Object)</td>
<td>Name length (4)</td>
<td>d</td>
<td>a</td>
<td>t</td>
<td>a</td>
<td>TC_STRING</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Name length (18)</td>
<td>L</td>
<td>j</td>
<td>a</td>
<td>v</td>
<td>a</td>
<td>/</td>
<td>l</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>n</td>
<td>g</td>
<td>/</td>
<td>O</td>
<td>b</td>
<td>j</td>
<td>e</td>
<td>c</td>
<td>t</td>
<td>:</td>
</tr>
<tr>
<td>90</td>
<td>TC_ENDBLOCKDATA</td>
<td>TC_CLASSDESC</td>
<td>Name length (27)</td>
<td>o</td>
<td>r</td>
<td>g</td>
<td>.</td>
<td>j</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>s</td>
<td>p</td>
<td>.</td>
<td>n</td>
<td>e</td>
<td>t</td>
<td>.</td>
<td>C</td>
<td>h</td>
<td>a</td>
</tr>
<tr>
<td>110</td>
<td>n</td>
<td>e</td>
<td>I</td>
<td>M</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>e</td>
<td></td>
<td>Class Serialization Identifier</td>
<td>Flags</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>Variable count (0)</td>
<td>TC_ENDBLOCKDATA</td>
<td>TC_CLASSDESC</td>
<td>Name length (20)</td>
<td>o</td>
<td>r</td>
<td>g</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>j</td>
<td>c</td>
<td>s</td>
<td>p</td>
<td>n</td>
<td>e</td>
<td>t</td>
<td>.</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
<td>g</td>
<td>e</td>
<td>Class Serialization Identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td></td>
<td>Flags</td>
<td>Variable count (3)</td>
<td>J (long)</td>
<td>Name length (9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>d</td>
<td>e</td>
<td>s</td>
<td>t</td>
<td>i</td>
<td>n</td>
<td>d</td>
<td>e</td>
<td>x</td>
<td>J (long)</td>
</tr>
<tr>
<td>180</td>
<td>Name length (11)</td>
<td>s</td>
<td>o</td>
<td>u</td>
<td>r</td>
<td>c</td>
<td>e</td>
<td>l</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td>d</td>
<td>e</td>
<td>x</td>
<td>L (Object)</td>
<td>Name length (12)</td>
<td>d</td>
<td>e</td>
<td>s</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>V</td>
<td>C</td>
<td>N</td>
<td>L</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>l</td>
<td>TC_STRING</td>
<td>Name length (18)</td>
</tr>
<tr>
<td>210</td>
<td>L</td>
<td>j</td>
<td>a</td>
<td>v</td>
<td>a</td>
<td>/</td>
<td>l</td>
<td>a</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>g</td>
<td>/</td>
<td>S</td>
<td>t</td>
<td>r</td>
<td>i</td>
<td>n</td>
<td>g</td>
<td>:</td>
<td>TC_ENDBLOCKDATA</td>
</tr>
<tr>
<td>230</td>
<td>TC_BASE</td>
<td>destIndex</td>
<td>destVCNLabel</td>
<td>acknowledged</td>
<td>dataIndex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NAPIER UNIVERSITY
EDINBURGH
### Problem – Message Cost

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TC_OBJECT</strong></td>
<td><strong>TC_CLASSDESC</strong></td>
<td>Name length (31)</td>
<td>o</td>
<td>r</td>
<td>g</td>
<td>.</td>
<td>j</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>s</td>
<td>p</td>
<td>n</td>
<td>e</td>
<td>t</td>
<td>.</td>
<td>C</td>
<td>h</td>
<td>a</td>
</tr>
<tr>
<td>20</td>
<td>n</td>
<td>n</td>
<td>e</td>
<td>l</td>
<td>M</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
</tr>
<tr>
<td>30</td>
<td>e</td>
<td>S</td>
<td>A</td>
<td>c</td>
<td>k</td>
<td>Class Serialization Identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Flags</td>
<td>Variable count (0)</td>
<td><strong>TC_ENDBLOCKDATA</strong></td>
<td><strong>TC_CLASSDESC</strong></td>
<td>Name length (27)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>o</td>
<td>r</td>
<td>g</td>
<td>.</td>
<td>j</td>
<td>c</td>
<td>s</td>
<td>p</td>
<td>.</td>
</tr>
<tr>
<td>60</td>
<td>e</td>
<td>t</td>
<td>.</td>
<td>C</td>
<td>n</td>
<td>a</td>
<td>n</td>
<td>n</td>
<td>e</td>
</tr>
<tr>
<td>70</td>
<td>M</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
<td>g</td>
<td>e</td>
<td>Class Serialization Identifier</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td>Flags</td>
<td>Variable count (0)</td>
<td><strong>TC_ENDBLOCKDATA</strong></td>
<td><strong>TC_CLASSDESC</strong></td>
<td>Name length (27)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>Name length (20)</td>
<td>d</td>
<td>e</td>
<td>s</td>
<td>p</td>
<td>.</td>
<td>j</td>
<td>c</td>
<td>s</td>
</tr>
<tr>
<td>100</td>
<td>n</td>
<td>e</td>
<td>t</td>
<td>M</td>
<td>e</td>
<td>s</td>
<td>s</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>g</td>
<td>e</td>
<td>Class Serialization Identifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>Flags</td>
<td>Variable Count (3)</td>
<td>J (Long)</td>
<td>Name length (9)</td>
<td>d</td>
<td>e</td>
<td>s</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>l</td>
<td>n</td>
<td>d</td>
<td>e</td>
<td>x</td>
<td>J (Long)</td>
<td>Name length (11)</td>
<td>s</td>
<td>o</td>
</tr>
<tr>
<td>140</td>
<td>u</td>
<td>r</td>
<td>c</td>
<td>e</td>
<td>i</td>
<td>n</td>
<td>d</td>
<td>e</td>
<td>x</td>
</tr>
<tr>
<td>150</td>
<td>Name length (12)</td>
<td>d</td>
<td>e</td>
<td>s</td>
<td>t</td>
<td>V</td>
<td>C</td>
<td>N</td>
<td>L</td>
</tr>
<tr>
<td>160</td>
<td>a</td>
<td>b</td>
<td>e</td>
<td>l</td>
<td><strong>TC_STRING</strong></td>
<td>Name length (18)</td>
<td>L</td>
<td>j</td>
<td>a</td>
</tr>
<tr>
<td>170</td>
<td>v</td>
<td>a</td>
<td>/</td>
<td>l</td>
<td>a</td>
<td>n</td>
<td>g</td>
<td>/</td>
<td>s</td>
</tr>
<tr>
<td>180</td>
<td>r</td>
<td>i</td>
<td>n</td>
<td>g</td>
<td>:</td>
<td><strong>TC_ENDBLOCKDATA</strong></td>
<td><strong>TC_BASE</strong></td>
<td>destIndex</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem – (Java) Objects Only

• Channel will only take objects
  – ChannelMessage.Data only takes object data

• Workarounds
  – Auto boxing, convert into byte array first

• Still a Java object
  – Serialization overhead
  – No inter-framework communication
Problem - Performance

- Different test classes developed
  - Varying in complexity (unique objects to object references)
  - Internal object arrays (Integer and Double)
  - Varying in size $n$ (0 to 100)
- TestObject4
  - Large and fairly complex
- Size
  - ($n = 0$) $\rightarrow$ 326 bytes
  - ($n = 1$) $\rightarrow$ 500 bytes
  - ($n > 1$) $\rightarrow$ 500 + 68($n - 1$) bytes
- Complexity
  - Object references: $10 + 8n$
  - Unique objects: $10 + 4n$
- Roundtrip time – PDA to PC
Problem - Performance

Time ms

Size of Sent Object (Bytes)

Normal Streams
Buffered Streams

NAPIER UNIVERSITY
EDINBURGH
Problem - Performance

- Buffered Streams
- Networked Channels

Time ms

Size of Sent Object (Bytes)
Problem – High Priority Links

- Link processes given maximum priority in the JVM
  - Depends on JVM implementation
- Computation must wait for I/O to complete
  - Within obvious I/O performance and multi-core properties
- Adding Links and channels increases I/O requirements
  - And thus reduces computation resources
- JCSP networking aimed at high computation to low communication ratio systems
Problem – High Priority Links

![Graph showing network channel receive and unacked channel receive over time and size of sent object (bytes).]
Problem – High Priority Links

- CommsTime on PDA
Problem – Exception Handling

• Poor exception handling
  – write() may block if Link to input end fails
  – Exceptions thrown internally but not propagated
  – If lucky might get a LinkLostException
  – If luckier might get a ChannelDataRejectedException
    •Rejectable channels now deprecated
  – Use LinkLostEventChannel to detect Link failure
    • NetChannelOutput not guarded
Problem – Lack of Protocol

• No interaction between frameworks
  – Even between different versions of Java

• Difficult to add new communication primitives
  – Can use underlying network channels (extra resources)
  – Numerous places to add functionality

• Data should be extracted from communication message
  – Serialization (again)
Ongoing Work and Conclusions

- JCSP Networking 2.0
  - Features
- Conclusions
Towards a Better Solution

• JCSP Networking 2.0
  – See Fringe session

• Features
  – Lower resource usage
    • Input channels are now lightweight
  – Reduced message size
    • 9 bytes standard, 13 for data messages
  – Better performance and scalability
  – No reliance on serialization
  – Data conversion extracted to channel layer
Towards a Better Solution

• Features (continued)
  – Simpler (layered) model
    • Layers only understand certain message types
  – Easier to extend
  – Networked barrier
  – Properties exposed
    • Link priority, buffer size
  – Error handling improved
  – Well defined protocol
  – Verified model
    • Using Spin (mobile channels)
    • FDR? – mobile channels in CSP
Conclusions

• JCSP Networking had some problems
  – Resource usage
  – Performance – serialization
  – Interoperability – serialization
  – Configuration and extension
  – Exception handling
  – All reflect badly for JCSP outside parallel computing usage

• These can be overcome
  – New JCSP Networking implementation
  – New protocol

• Future work – framework interaction?
  – JCSP, pony, C++CSP, PyCSP, CSP.NET
Questions?