jUDPWrapper: A Lightweight Approach to Access the OMNeT++/INET UDP Functionality from Java

Henning Puttnies, Peter Danielis, Leonard Thiele, Dirk Timmermann
University of Rostock, Germany
Motivation

- Evaluation of networks during design: simulation, testbed, math analysis
- Our idea of an enhanced evaluation methodology:
  1. Java simulation models using OMNeT++
  2. Derive a Java prototype implementation
- Java Extensions for OMNeT++ (JEO) exist
- Next logical step: provide a socket-based API for Java simulation models
- Abstraction layer between app. layer models und INETs UDP module
  ➔ Ease simulating as well as the derivation of prototype implementation
## Related Work

- **Wanted:** Framework to simulate Java application layer models

<table>
<thead>
<tr>
<th>Name</th>
<th>Java Simulation Models</th>
<th>Access to Simulation Time</th>
<th>Still Under Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS-3 + LXC* + JVM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(*Linux Containers)</td>
<td>(only app. layer)</td>
<td>(only with modified Linux kernel, limited precision)</td>
<td></td>
</tr>
<tr>
<td>NS-3 + DCE*</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(*Direct Code Execution)</td>
<td>(C++ only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNSS*</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>(*Fast Network Simulation Setup)</td>
<td>(needs a Java simulator/emulator)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JNS/ JNetworkSim/ Jprowler/ Java Simulator</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>
jUDPWrapper – Basic Concept

- **Design targets:**
  - Fit optimally into the OMNeT++/INET ecosystem
  - No modifications of the OMNeT++/INET code
  - Lightweight ➔ keep track with new OMNeT++/INET versions easily

- **DatagramSocket and InetAddress:**
  Same API as *java.net.DatagramSocket* and *java.net.InetAddress*
**jUDPWrapper – Accessing Message Fields of a Custom Data Type from Java: e.g., L3Address**

- **String as type for domain conversion**
- **`getField()`**: can access any field of a message
- **`setField()`**: only works for standard types (e.g., `int`, `double`, `bool`)
  - how to access a field of a custom type?
- Our approach: utilize a special syntax in the `*.msg` file
  - link the `setFied()` method to the corresponding string constructor

---

C++ Code

```
cObject.setField("destAddr", ...)
```

Java Code

```
SimkernelUNI.cObject_setField()
```

```
Java_org_omnetpp_simkernel_SimkernelUNI_cObject_1setField()
```

```
omnetpp_cObject_setField()
```

```
cClassDescriptor.setFieldValueAsString()
```

```
UDPSendCommandDescriptor.setFieldValueAsString()
```

```
UDPCtrlInfo.msg
```

```
UDPSendCommand
L3Address destAddr
int destPort
L3Address srcAddr
int interfaceId
```

```
L3Address.h/.cc
```

```
L3Address
uint64 hi
uint64 lo
L3Address()
L3Address(const char *str)
L3Address(const IPv4Address& addr)
```

---

10/09/2018 OMNet++ Community Summit 2018
jUDPWrapper – Accessing Message Fields of a Custom Data Type from Java: e.g., L3Address

//Original *.msg file
class UDPSendCommand extends UDPControlInfo {
    L3Address destAddr;
    int destPort = -1;
    //...
}

//Modified *.msg file
class UDPSendCommand extends UDPControlInfo {
    L3Address destAddr @editable @fromstring(inet::L3Address($));
    int destPort = -1;
    //...
}

//Original *.cc file
bool UDPSendCommandDescriptor::setFieldValueAsString(/*...*/ const{
    //...
    switch (field) {
        case 1: pp->setDestPort(string2long( value)); return true;
    //...
    }}

//Modified *.cc file
bool UDPSendCommandDescriptor::setFieldValueAsString(/*...*/ const {
    //...
    switch (field) {
        case 0: pp->setDestAddr(inet::L3Address( value)); return true;
        case 1: pp->setDestPort(string2long( value)); return true;
    //...
    }
jUDP Wrapper – UDP Example Networks

- Four example networks used for evaluation
- Show interoperability and performance of Java and C++ application layer simulation models
- Every network: 2 *StandardHosts* connected Ethernet switch (not shown for the sake of simplicity)
- jUDPBasicApp and jUDPEchoApp equivalent to their INET counterparts
  ➜ Evaluation of performance is feasible
**jUDPWrapper – Performance Evaluation 1**

- Execution time grows linearly with the # packets ➔ expected behaviour
- **OMNeT++ 5.0:**
  - 10 packets: C++ (0.0008s) approx. one order of magnitude faster than Java (0.0097s)
  - 100,000 packets: C++ (6.89s) approx. twice as fast as Java (14.5s)
jUDPWrapper – Performance Evaluation 2

- **OMNeT++ 5.4:**
  - 10 packets: C++ (0.0013s) approx. 3 times faster than Java (0.0036s)
  - 100,000 packets: C++ (6.04s) approx. twice as fast as Java (14.39s)
- Remarkable: intermediate performance of mixed language setups
  ➔ Use existing C++ modules from INET to work with your Java modules

![Graph showing execution time vs. number of packets for different configurations of OMNeT++ 5.4.](image-url)
Conclusion and Outlook

- jUDPWrapper: simple + socket-based interface to INET's UDP functionality
- Generic approach to access message fields that have a custom data type
  ➔ Serves as example of how to access INET modules from Java
- Different example applications for custom Java simulation models
- Evaluation: OMNeT++ 5.0/INET 3.4.0 and OMNeT++ 5.4/INET 3.6.4.
  ➔ Provide the Java Extensions for OMNeT++ 5.4
- Performance: C++ approx. twice as fast as Java simulation models
  - Valid for long simulation runs and release mode
  - Speedup reduced in debug mode or if a mixed language setup is used
- Entire system is publicly available
  ➔ Everyone can retry the performance measurements
- Interesting for future work: Wrapper for INET’s TCP functionality

1 [https://bwsyncandshare.kit.edu/dl/fi8R6skmuBPh6UfXHWzcgBxt/.zip](https://bwsyncandshare.kit.edu/dl/fi8R6skmuBPh6UfXHWzcgBxt/.zip)
Thank you for your attention. Questions?