RobustTrav: NAT Optimisation for the RobustCooperation Suite

Christoph Beckmann, Tom Gross, Ferdinand Kastl
Human-Computer Interaction Group
University of Bamberg
96045 Bamberg, Germany

<firstname>.<lastname>(at)uni-bamberg.de

Introduction

Cooperative systems
- Supporting remote teams
- Based on distributed software architectures

Internet Protocol version 4 (IPv4)
- Technological backbone with multifarious strengths
- Entails challenges such as limited addressing (e.g., notifications passing firewalls and routers)

Network Access Translation (NAT) mechanisms
- Building private networks with routing of inbound and outbound traffic
- Port mapping allows direct client-to-client connections as well as server-initiated push notifications—but hard to configure and not always allowed

We present the RobustTrav Mechanism
- Effective and efficient NAT traversal
RobustCooperation Suite—Architecture

Support for remote teams working on artefacts

- Distributed software architecture relies on sensor-based event processing
  - Capturing, storing, and delivery of events among all components
  - Bases on the sensor-based platform Sens-ation [Gross et al. 2006]
- Provide components for
  - Communication (i.e., RCIM)
  - Coordination (i.e., RCCal)
  - Data sharing (i.e., RCSpace)
RobustCooperation Suite—Challenges

RobustTrav Mechanism—Concept

Connects clients and server within and across networks
- **Publish/subscribe** (Pub/Sub) mechanism on top of **XML-RPC**
- Leveraging on the **long polling approach** [Russel 2006]
- Implemented in the RCIM and RCCal client components as well as in the RCSensServer server component
- **Typical PubSub sequence** between RCIM, RCCal, and RCSensServer
RobustTrav Mechanism—Parameters

Parameters most significant for quality of RobustTrav mechanism identified by

- A thorough literature review, as well as our own experience with developing and deploying event notification systems for over a decade

Parameters and value ranges

- **Server parameter** specify the configuration of the server before runtime
  - Server Timeout (0–120.000ms)
- **Request parameters** specify the processing of the respective request in the server and the clients
  - Client Timeout (0–120.000ms)
  - Client Regenerate Time (0–500ms)
  - Batch Size (1–Cache Size)
  - Batch Timespan (0–10.000ms)

RobustTravOpt—Hard- and Software Setup

Series of systematic performance measurements for optimising parameters of RobustTrav mechanism

- Setup as recommended by [Biwas et al. 2011]
- **Native network host**—runs RCSensServer component
  - Mac-Book Pro, Intel Core i5 2.4 GHz with 4 GB of DDR3 RAM
  - OS X 10.6.7 and Java version 1.6.0_24
- **Simulated NAT environment**
  - Oracle’s virtualisation software Virtual Box version 4.0.10
- **Virtualised network host**—runs test scenario
  - Event generator for publishing events to the RCSensServer
  - RobustTravClient for subscribing, receiving, analysing, and logging
  - 1 core and 1024 MB RAM assigned
  - Ubuntu Linux 10.04 with Java version 1.6.0_20
RobustTravOpt—Test Case Arrangement

One test case for each parameter
- Divided value range of parameters in 10 steps (exact values in the paper)
- Lower value range values are denser to see changes better

Each test case has 10 test runs with specific test values
- Obtained typical event sending frequency from development test
  - Send an event every 30ms
  - 50 dynamically created RobustTravClient send these events
- 10.000 events per run allow occurrence of at least two timeouts

Data obtained from test cases
- Total amount of 25.000.000 events received
- Total test duration of 240 minutes

---

RobustTravOpt—Optimal Configuration

Optimised parameter values test run to confirm performance

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Min [ms]</th>
<th>Avg [ms]</th>
<th>Max [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Timeout</td>
<td>1.000ms</td>
<td>1,29</td>
<td>169,35</td>
<td>1.553,72</td>
</tr>
<tr>
<td>Client Timeout</td>
<td>1.000ms</td>
<td>1,87</td>
<td>169,53</td>
<td>1.825,93</td>
</tr>
<tr>
<td>Client Regenerate Time</td>
<td>100ms</td>
<td>2,88</td>
<td>168,01</td>
<td>10.405,89</td>
</tr>
<tr>
<td>Batch Size</td>
<td>5 events</td>
<td>1,94</td>
<td>169,10</td>
<td>10.349,52</td>
</tr>
<tr>
<td>Batch Timespan</td>
<td>0ms</td>
<td>2,19</td>
<td>203,30</td>
<td>10.523,80</td>
</tr>
<tr>
<td>Optimised parameter values test run</td>
<td>1,66</td>
<td>167,50</td>
<td>2.677,29</td>
<td></td>
</tr>
</tbody>
</table>

- Efficient minimum and average latencies
- Balancing and stable maximum latency
Related Work

NAT traversal mechanisms
• Peer-to-peer systems [Baset & Schulzrinne 2010]
• Autonomous NAT Traversal [Mueller et al. 2010]
• Universal Plug and Play [UPnP Forum 2011]
• AJAX-Push mechanism [Russel 2006]

Measuring NAT performance
• NAT/Firewall Traversal Cost Model [Biswas et al. 2011]

Groupware and awareness support
• NESSIE environment [Prinz 1999]
• GROOVE groupware [Dustdar et al. 2004]
• WGWSOA middleware [Maciel et al. 2009]
• PPPSpace [Gross et al. 2010]

Conclusion

RobustTrav
• Long polling mechanism implemented for XML-RPC
• Demonstrated its feasibility in the RobustCooperation Suite
• Remote teams profit from efficient push notifications in inhomogeneous network environments

RobustTravOpt
• Systematically optimised five most significant parameters of RobustTrav
• Lessons learned
  • Synchronised clocks within NAT environment through virtualisation
  • Minimal overhead for generating events through raw sockets
References


References (cont’d)


Results

RobustTravOpt—Server Timeout

- Optimum at 1,000ms
- Latencies varied between 1,29ms–10,7s
- Minimal latencies depend on the timespan between different client connects
- Maximal latencies plateau above 1,29ms–10,7s

Parameter Tested Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Tested Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Timeout</td>
<td>500, 1,000, 5,000, 10,000, 30,000, 60,000, 75,000, 90,000, 105,000, 120,000 [ms]</td>
</tr>
</tbody>
</table>

---

RobustTravOpt—Server Timeout

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Tested Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Timeout</td>
<td>500, 1,000, 5,000, 10,000, 30,000, 60,000, 75,000, 90,000, 105,000, 120,000 [ms]</td>
</tr>
</tbody>
</table>

---

References (cont’d)