Dependability Assessment of an OGSA Compliant Middleware Implementation by Fault Injection

Nik Looker and Jie Xu
Department of Computer Science,
University of Durham, DH1 3LE, UK
{n.e.looker, jie.xu} @ durham.ac.uk

http://www.ogsa-fit.org
• What is Fault Injection?
  • Simply put, you insert faults into a system in someway in an attempt to cause an error

• Why?
  • Accelerate the incidence of failures within the system
  • Exercise seldom used paths within the code
  • Basically we are trying to compliment standard testing methods and assess the dependability of a product
A Few Questions

• What are we testing?
  • OGSA Middleware
    • Integrity of Data Returned by Calls
    • Fault Tolerance of Middleware

• Why are we testing this?
  • Detect Defects in Middleware Design
  • Dependability Comparisons Between Products

• How are we testing?
  • Slightly modified version of network level fault injection
    (we’ll get to how in a minute)
• There are a number of different F/I techniques

• Based around Network Level F/I
  • Capturing, corrupting and retransmitting network packets

• There are a number of advantages
  • Simple to implement
  • Relatively non-invasive (no app code modifications)
  • Easy to monitor results
Problems With OGSA

• OGSA implements signing of SOAP messages
  • Signing not only identifies the message as coming from a particular context but also ensures that it isn’t changed in transit

• OGSA implements encryption
  • Since the message body is encrypted, changing anything but a random byte is impossible to do.
  • Changing a byte in the message will cause the packet to fail decryption (this is an important area but other research going on).

• How to overcome this?
  • Modify messages before signing and encryption.
Fault Model

- Fault Model for OGSA Middleware
  - Physical Faults
  - Software Faults
  - Resource Management Faults
  - Communication Faults
  - Life-Cycle Faults

- Failure Modes for OGSA Middleware
  - Crash of a service
  - Crash of a hosting web server (or the host itself)
  - Hang of a service
  - Corruption of data into middleware
  - Corruption of data out of middleware
  - Duplication of messages
  - Omission of messages
  - Delay of messages
The Fault Injector

• **Fault Injector Framework**
  • Logging Function (XML format for easy analysis)
    • Detects and logs faults
    • Logs injected fault packets
  • Two pass injection mechanism:
    • First to run trigger.
    • Second to inject the fault (only runs if fault needs to be injected).

• **User Script**
  • Three classes: Results, Trigger, Injector

• Hook code in SOAP Library
Discarding Packets Experiment Replicated on GT3

- This caused a problem in the Axis library.
- If the packet from the server to the client is discarded in an RPC, the socket will pend indefinitely waiting for a connection. This effectively hangs the client.
- This is effectively the same as a server crashing after receiving a request but crashing before sending a response (one of the areas in our fault model).

Timing Latency Introduced By OGSA-FIT

- We have derived a timing model for the injection of faults based on our tools
- Experiment injected faults into the system and was timed using duration timers
- Results showed that major latencies were network transfers and XML parsing
Summary

• We have a prototype set of tools that are integrated with GT3

• Our Tools
  • Are relatively non-invasive to the system
  • Are GUI based
  • Allow automatic generation of scripts from WSDL
  • As well as testing middleware our tools can be used to test general components

• Our initial experiments discovered problems with GT3

• In the future we hope to further enhance our method with a greater level of automatic script generation and results analysis

• More information at http://www.ogsa-fit.org/