Tracing CORBA applications using interceptors

Zoltán Mann
Supervisor: Dr. Károly Kondorosi

Budapest University of Technology and Economics
Department of Control Engineering and Information Technology

Contents

1. Motivation
2. Tracing in general
3. Interceptors
4. Tracing using interceptors
5. Conclusion

Motivation

1. Technical development:

   - Distributed, heterogeneous environment
   - Object-oriented integration → CORBA

Motivation

2. Applications:

- e-Business
- Embedded systems

Motivation

3. Growing competition:

- Functional and reliability requirements
- Time-to-market pressure

Contents

Tracing vs. debugging

≠

Aims of tracing *(use cases)*

1. Checking correct behaviour
2. Locating bugs
3. Better understanding of how the system works
4. Monitoring crucial applications
5. Automatic documentation extraction
6. Performance analysis, identifying bottlenecks

Current tracing mechanisms

Current solutions are bound to particular programming languages.

Problems with distributed systems:

- Heterogeneity
- Collecting information from different namespaces
- Synchronization, lack of global clock etc.

Current tracing mechanisms

State of the art in distributed systems:

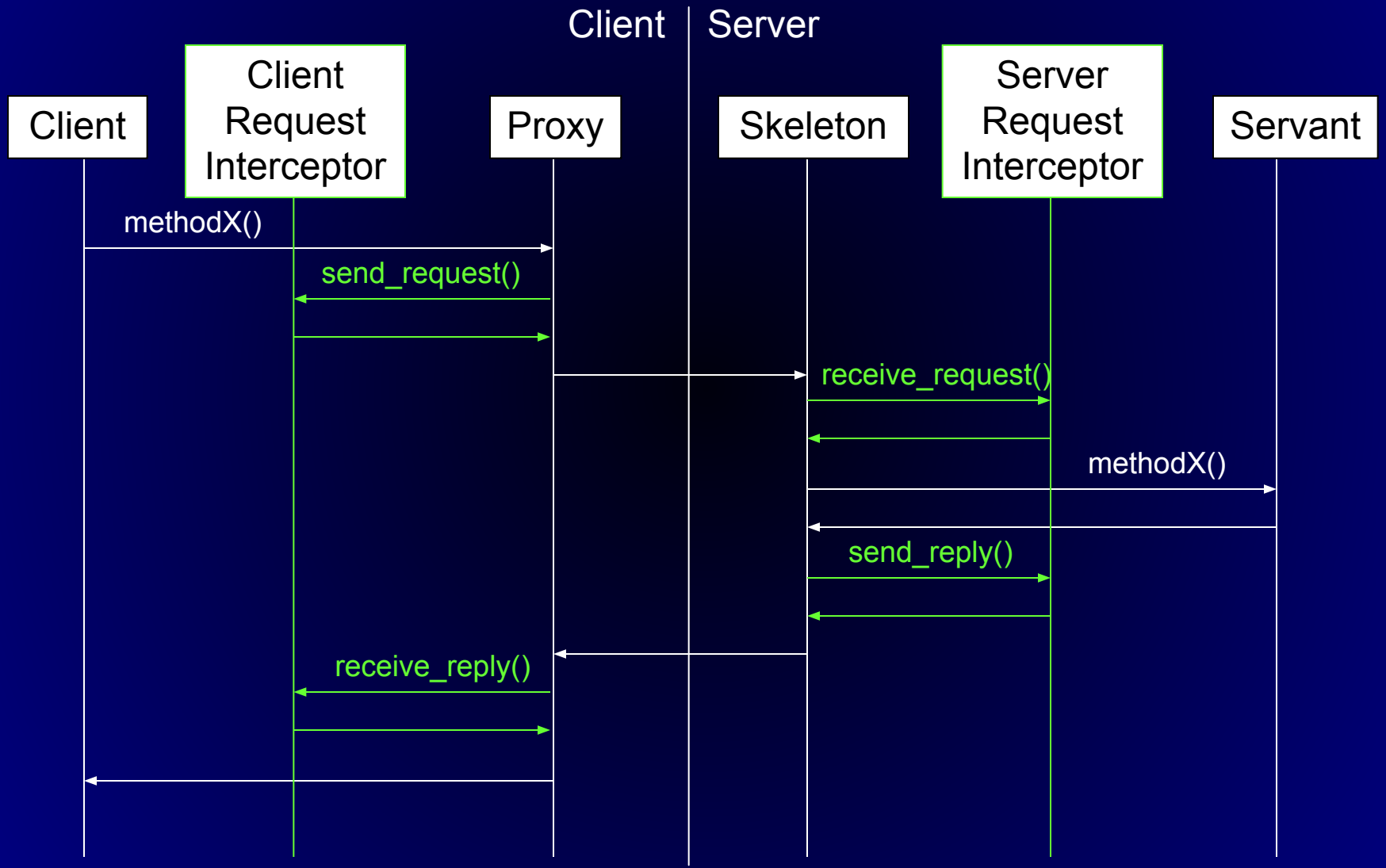
+ Central tracer
– Manual instrumentation
  – Extra programming work
  – Prone to errors

Contents

CORBA interceptor

- An object implementing the interceptor interface
- extending the functionality of the ORB
- using callback methods
- without actually modifying the ORB

Callback methods



Definitions & implementations

- First definition: CORBA 2.3
- Incompatible implementations
- September 1998: OMG RFP
- December 1999: Joint Submission
- March 2000: CORBA 3.0 Working Draft

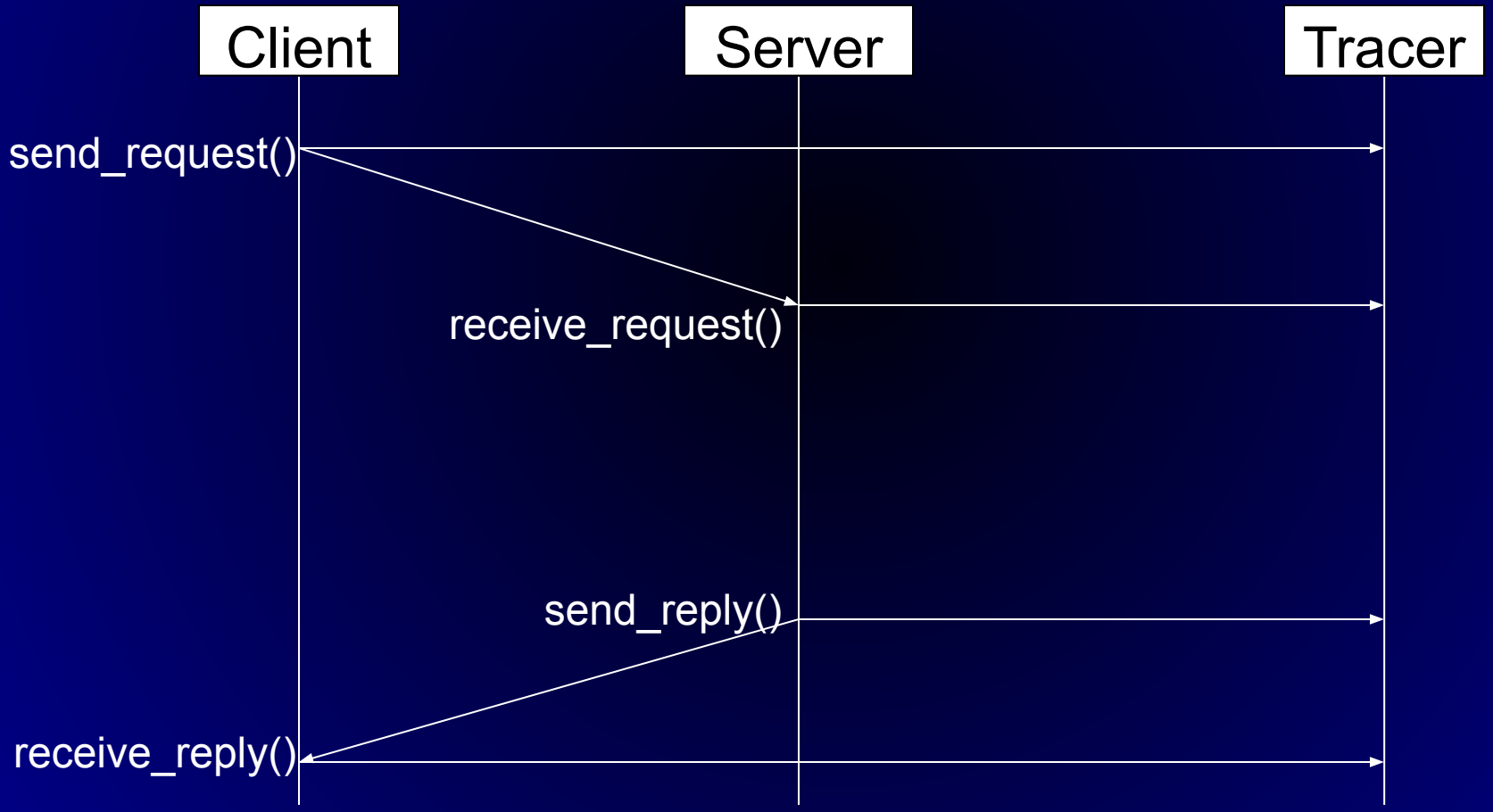
- The used implementation: TAO 1.1 and 1.1.9 beta

Other possible solutions

- Smart proxies
- Servant managers
- Pluggable protocols
- ORB event handlers
- …

Contents

Tracing using interceptors



The resulting architecture

- ✔ Automatic instrumentation
- ✔ One single line of initialization code per name space
- ✔ With the standard registration mechanism of interceptors, even that could be avoided
- ✔ Until then: a slight modification of the ORB
- ✔ Open system

Overhead

- In interactive mode: ~ 500 % communication overhead
- In local mode: ~ 15 %

Contents

1. Motivation
2. Tracing in general
3. Interceptors
4. Tracing using interceptors
5. Conclusion

Results

- ✔ A tracing architecture satisfying the previously defined requirements
- ✔ A tool for documenting and interactive tracing of CORBA applications
- ✔ The solution works in a distributed and heterogeneous environment
- ✔ Prototype of a future product

Future plans

- Improving interoperability
- Extending the architecture for other middleware systems, such as DCOM
- Improving user interface, with the inclusion of possible users
- Making the tracer persistent
- …

Tracing CORBA applications using interceptors

Zoltán Mann
Supervisor: Dr. Károly Kondorosi

Budapest University of Technology and Economics
Department of Control Engineering and Information Technology