



Xuejun Chen
BMW Group
Forschung und Technik
Munich, Germany



Evolving Communication Mechanisms of the OSGi Framework

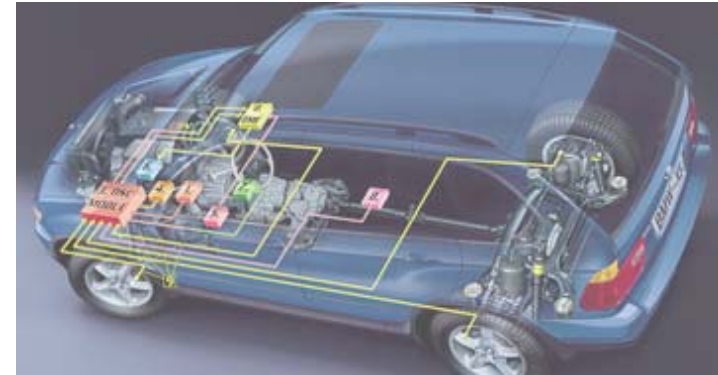


Overview

- Motivation
 - Dynamic distribution of applications in vehicle
 - OSGi is being used in vehicles.
- Current solution to location transparency
 - Java RMI is used.
 - Performance problem
- New solution to location transparency
 - Automatically switching invocations from remote to local and vice versa
- Conclusion

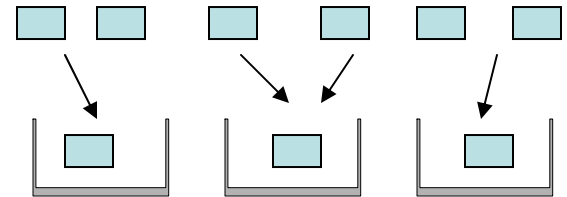
Motivation (1)

- Vehicle software systems
 - Controllers and sensors
 - Telematics systems
 - Management systems
- Distributed systems in vehicles
 - Today, approx. 70 ECUs in a BMW vehicle
 - In the future, maybe only several high performance ECUs in vehicles



Motivation (2)

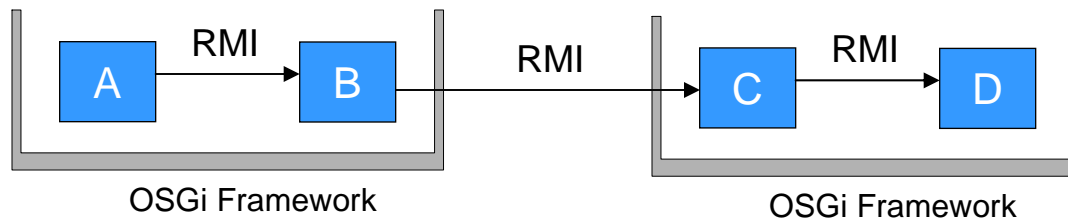
- Requirements of future vehicle software
 - Optimized system performance
 - Dynamic distribution of applications
 - Location transparency



- OSGi is being used in vehicle software
 - However, OSGi does not provide support for location transparent communication.

Current solution to location transparency

- Java RMI is used for location transparent communication.



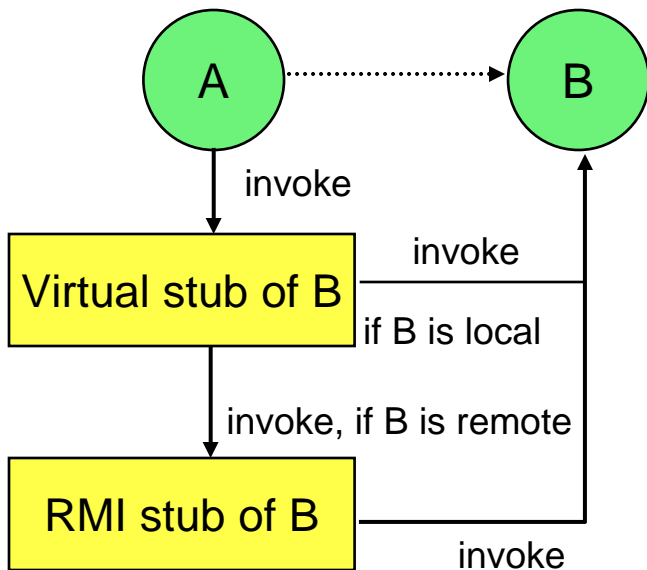
- System performance is degraded because RMI is used for local communication
 - Java RMI's serialization takes at least 25% of the costs of a remote method invocation.
 - The costs of serialization rise with growing object structures.

New solution to location transparency

- Virtual stub based solution to location transparency
- A virtual stub is a local object for a client component, and forwards the invocations from the client to the server.
- A virtual stub is dynamically loaded and updated by a system service.
- The virtual stub can automatically switch invocations from remote to local and vice versa.
- The virtual stub is used like a normal Java RMI stub.

Virtual stub

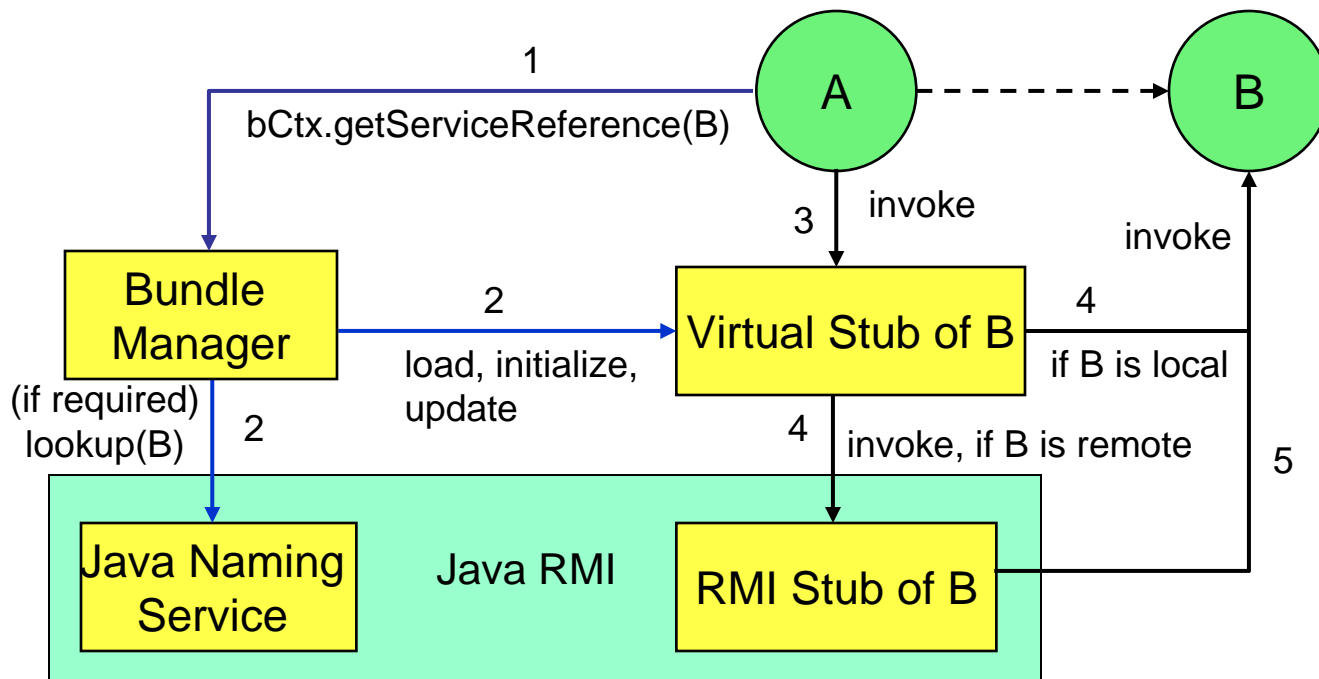
- Components communicate with each other by using virtual stubs.



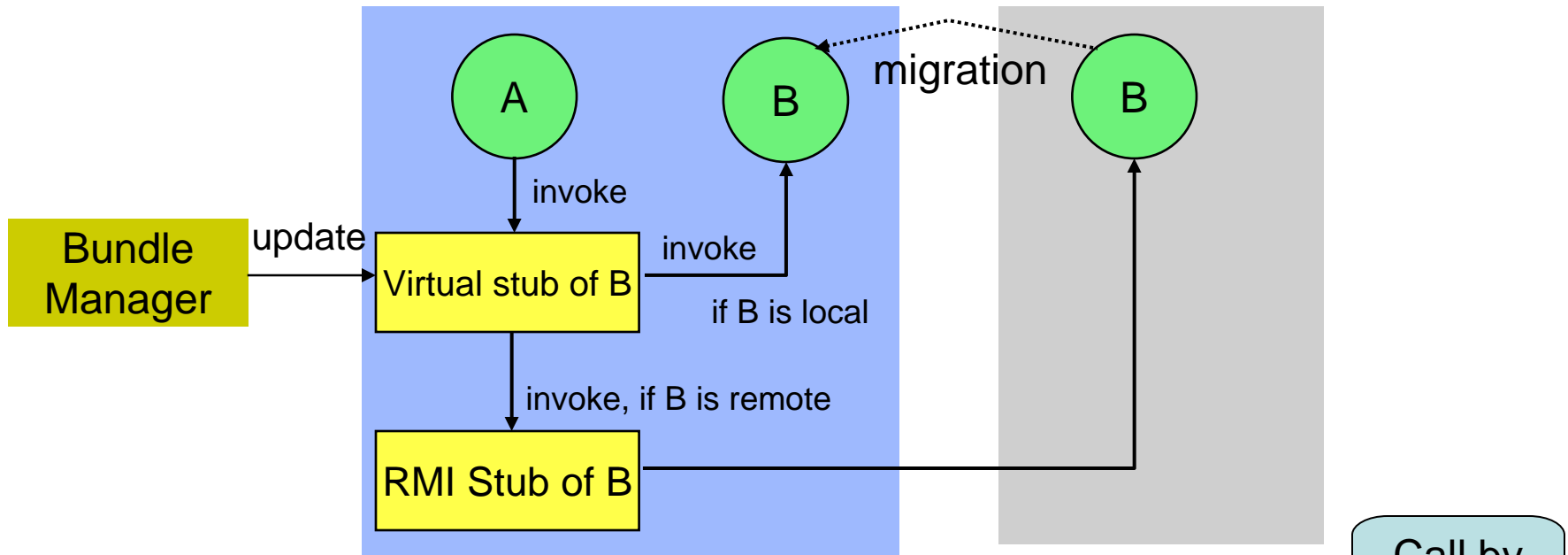
```
TestInterface v_Stub;
v_Stub = (TestInterface)
componentContext.lookup("TestServer_B");
try {
    info = v_Stub.testMethod("a test call");
}
```

Integrating virtual stub into OSGi

- Implementing a bundle manager that registers the references of the local bundles, and loads, updates a required virtual stub for a client bundle.



Automatically switching invocations from remote to local and vice versa



The method invocation in the virtual stub of B:

```
if (local)           /* A and B are local.*/  
    str = (String)ref.testMethod(clone(info));  
else                 /* A and B are remote.*/  
    str = (String)ref.testMethod(info);
```

Call by value

Measurement results of local/remote invocations

Test case: component *A* calls a method of component *B*, which returns a string.

Runtime environment	Middleware	Time (ms)
Two components run on a desktop PC. i.e. <i>A</i> and <i>B</i> are local.	Java RMI	1.0114
	Virtual Stub	0.0623
Component <i>A</i> runs on a desktop, component <i>B</i> runs on a laptop, i.e. <i>A</i> and <i>B</i> are remote.	Java RMI	2.7913
	Virtual Stub	2.8356

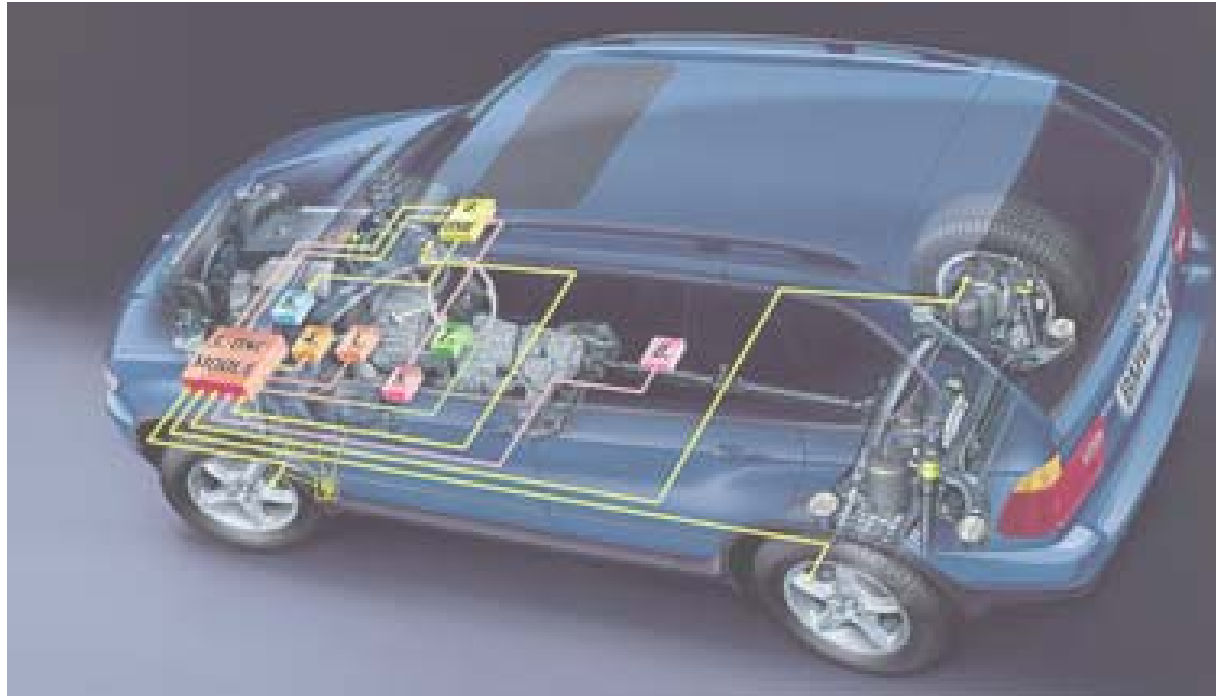
- When both components are local, the system performance of using virtual stub is significantly higher than using Java RMI.
- Overhead of virtual stub is only 0.0443 ms.

The measurement results are cited from [Chen2002].

Conclusion

- By using virtual stub, communication between components is location transparent, and at the same time, the system performance is not degraded.
- For more detailed information, please see the following article:

[Chen2002] Chen, Xuejun. Extending RMI to Support Dynamic Reconfiguration of Distributed Systems. In: Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS 2002), pages 401 – 408, July, 2002.



Thank you for your attention!

eMail: xuejun.chen@bmw.de