Remote Procedure Call & The Broker Pattern

CHANDAN R. RUPAKHETI

WEEK 3-3

Context

A distributed system can take advantage of the computing power of multiple CPUs or a cluster of low-cost computers

Certain software may only be available on specific computers

Parts of the software may have to run on different network segments due to security considerations

Some services may be provided by business partners and may only be accessed over the Internet

Problem

How can you structure a distributed system so that application developers don't have to concern themselves with the details of remote communication?

How to achieve interoperability between distributed software components?

Solution – Broker Pattern

Separate client and server by inserting an intermediary called a **broker**

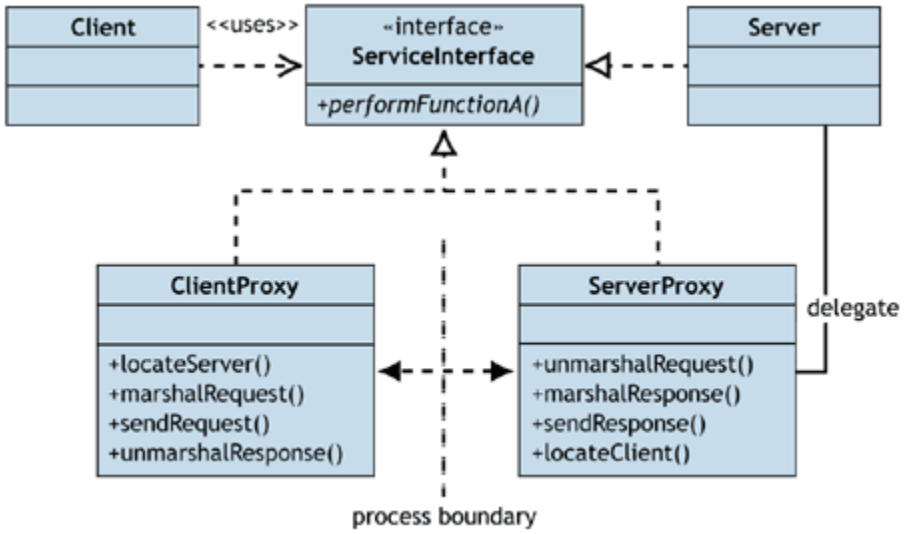
Use the ***Broker*** pattern to hide the implementation details of remote service invocation

Without Broker 1/3

Structure with no distribution

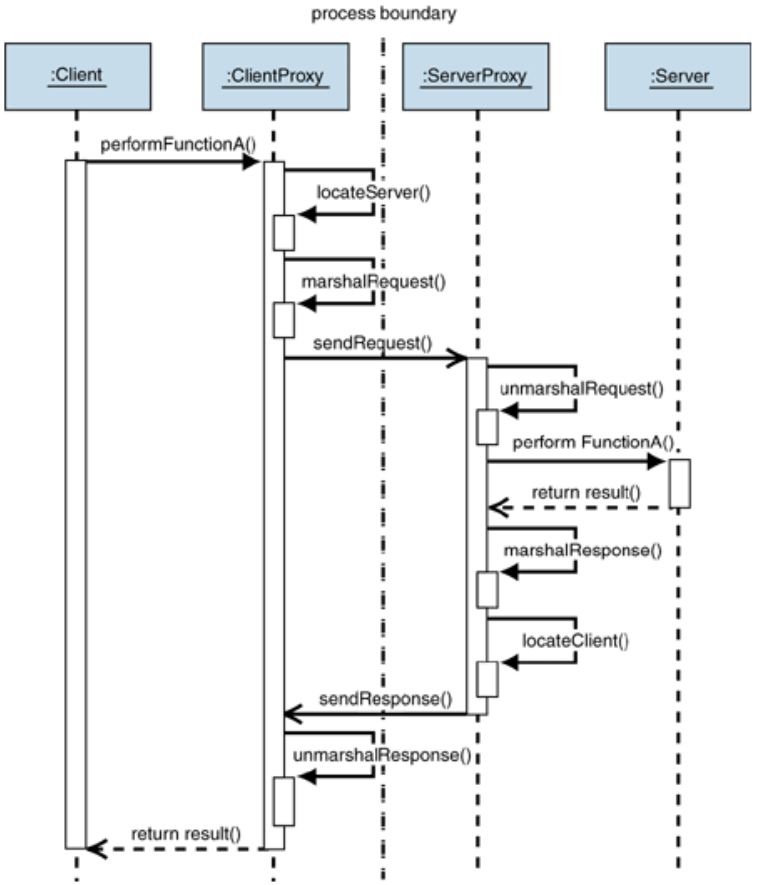
http://msdn.microsoft.com/en-us/library/ff648096.aspx

Without Broker 2/3

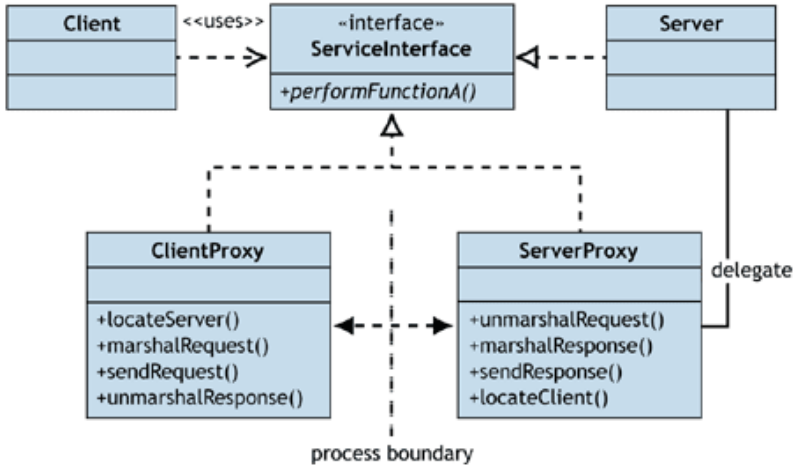


Structure with distribution

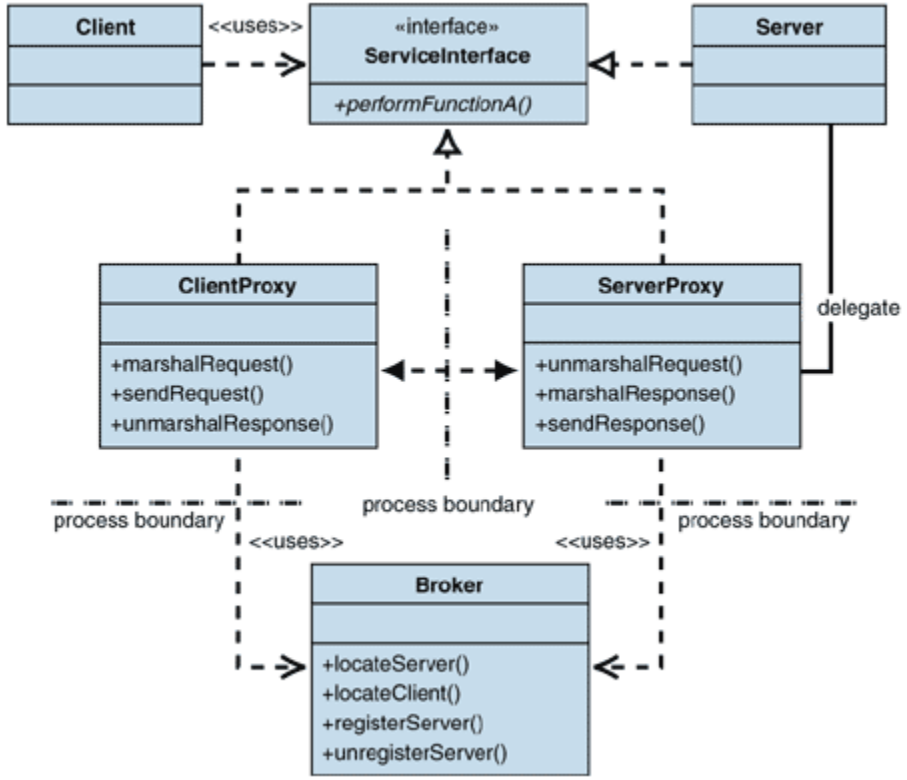
Without Broker 3/3



With Broker 1/3

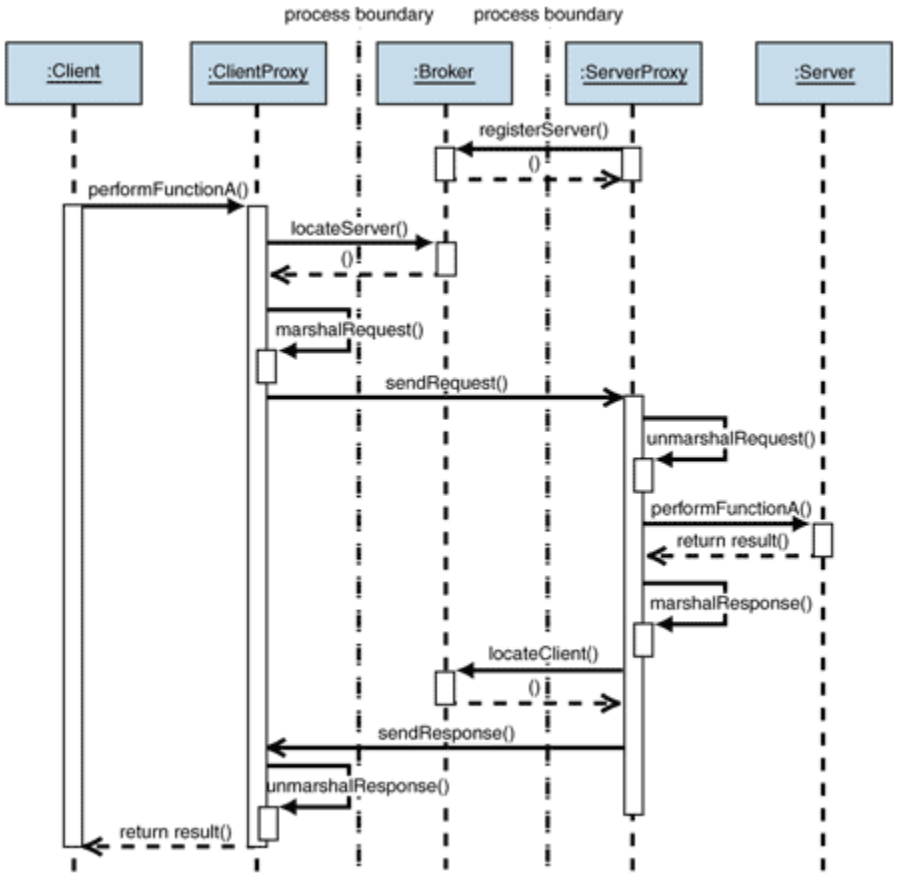


Structure without broker



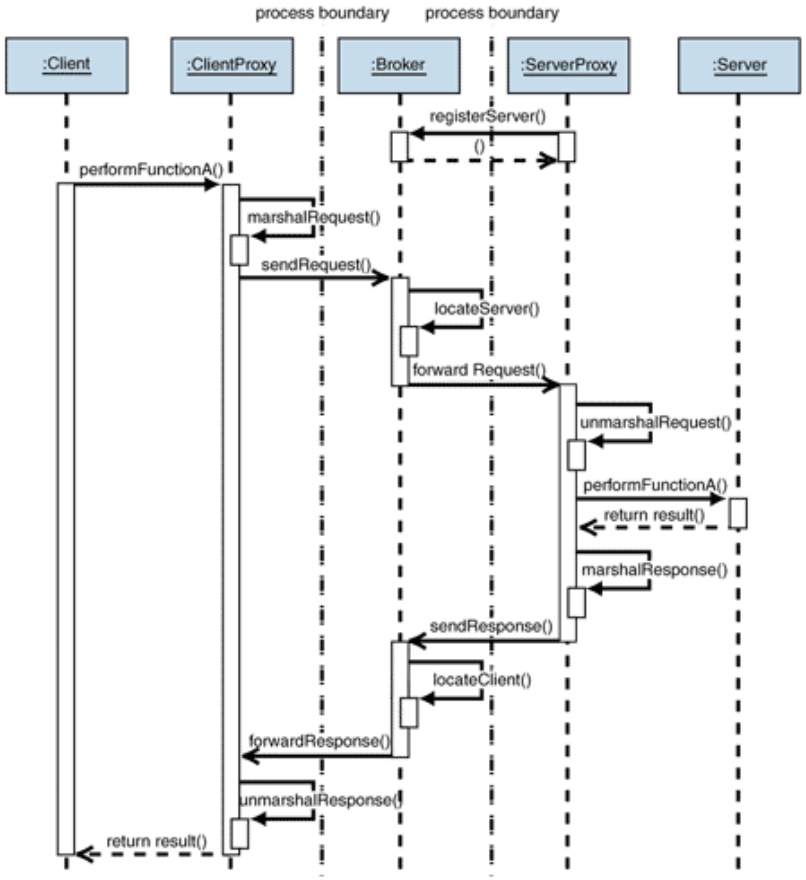
Structure with broker

With Broker 2/3



Broker behavior with server look-up

With Broker 3/3



Behavior of Broker serving as intermediary

Limitations of Broker

- Added layer of indirection, hence, latency
- May be a communication bottleneck
- Single point of failure, hence, target for attacks
- Adds up-front complexity
- May be difficult to test

Next ...

Next Week

- Shared Data, Peer-to-Peer

Things Due

- Assignment 4 - 11:55 pm on Friday