Experiences in Deploying Services within the Axis Container

Beytullah Yildiz
Indiana University
byildiz@indiana.edu

ICIW'06 February , 2006
Guadeloupe, French Caribbean

Outline

- Introduction
  - Web Service
  - Handler
  - Container
- Experiences and Suggestions
- Conclusion

Web Service

- Provides a distributed computing environment to solve interoperability issues.
- An effort for seamless communication.
- Leverages XML.
- Has support from a large community.
- Many specifications.

Handler

- Services facilitate incremental addition of capabilities, called handler or filter.
- An endpoint’s capability is enhanced without the need for making changes to the application.
- A handler may need peers in both client and service.
- Several handlers could be cascaded together to comprise a handler chain.
- Handler sequence needs to be defined.

Container

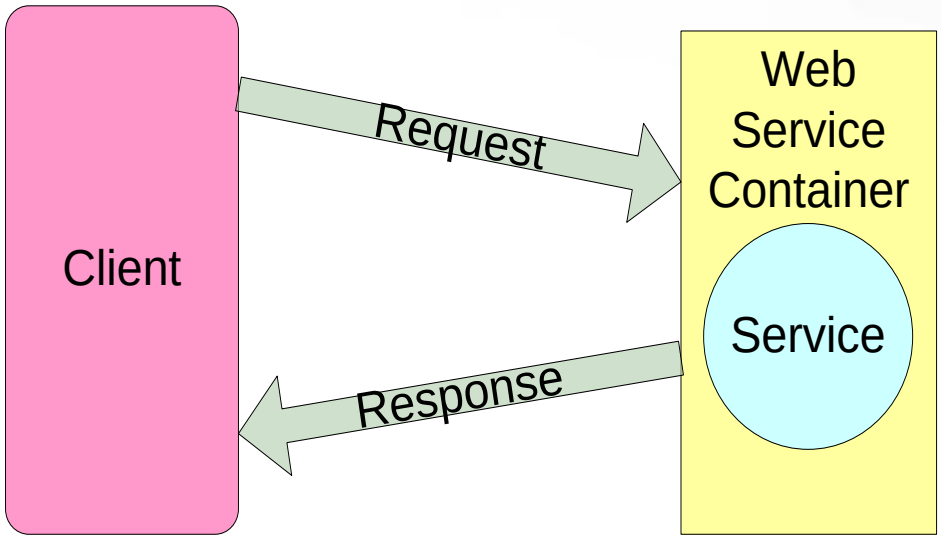
- A Web Service is typically hosted within a Web Service container.
- There are several choices for the containers depending on the platform and the language.
- Most dominant Web Service container is the open-source Apache Axis.
- We enumerate the problems, limitations and our solutions.
- We also have a set of recommendations that would make a more flexible container.

Axis I (Version 1.2)

- Java base Web Service container.
- Plethora of applications
- We describe our experiences in deploying Web Services, specifically WS-ReliableMessaging and WS-Eventing.
- Problems
  - Based on the request-response paradigm.
  - Does not support message injection.
  - Does not have the ability to gracefully terminate processing within the handler chain.
  - Handler chain has a static configuration.

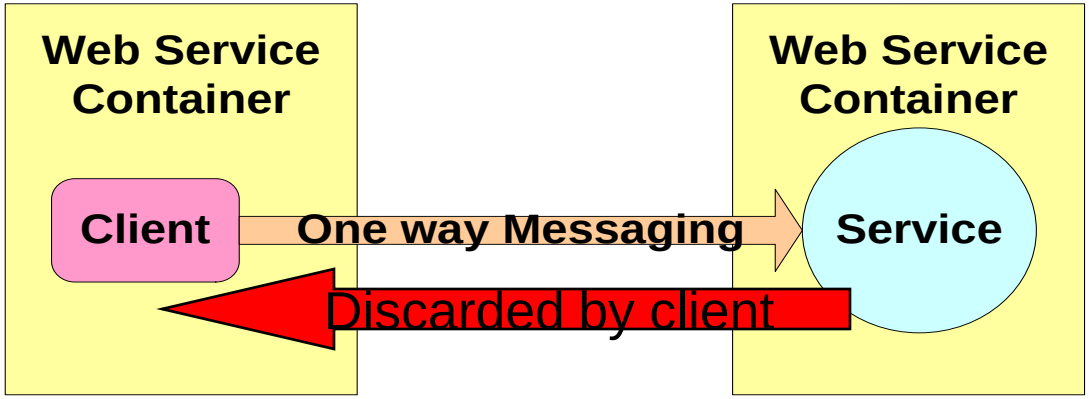
Axis II (Version 1.2)

- Every message is considered to be a request which should have its accompanying response within a pre-defined period of time.
- Does not support one-way and asynchronous messaging.



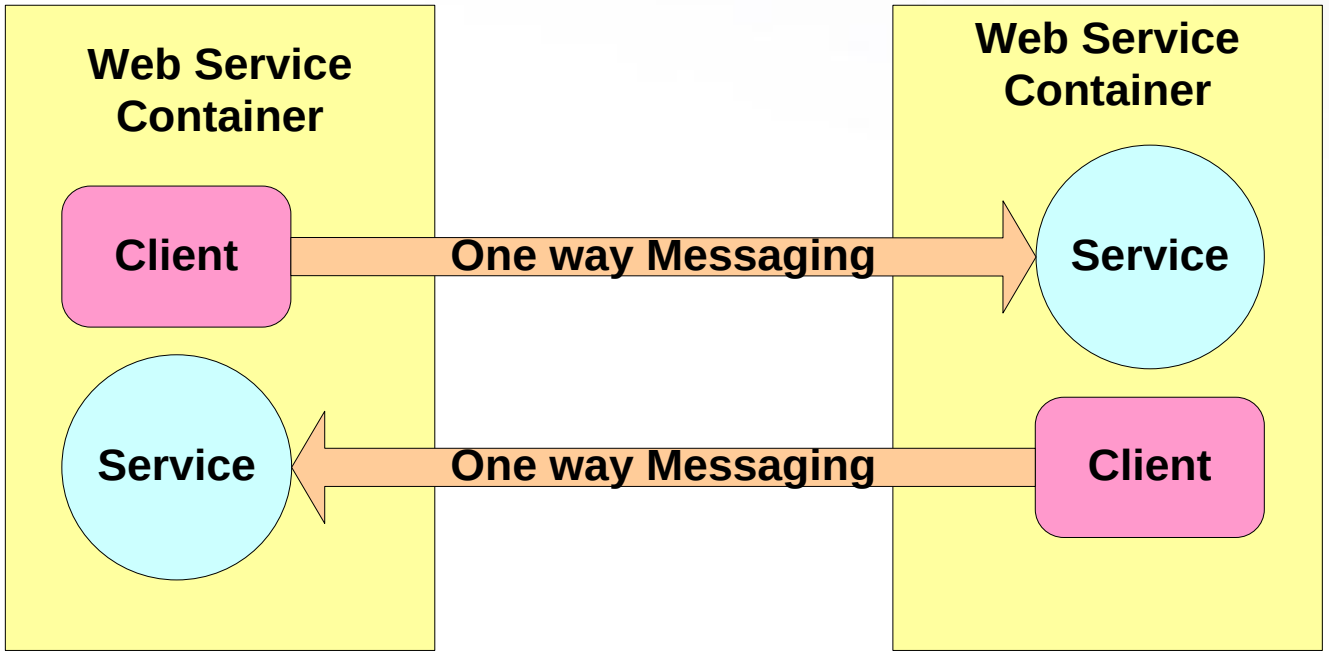
One-way Messaging

- An example; informing an entity about an event, Notification and Acknowledgement.
- Can be accomplished by discarding the dummy response messages.
- Responding back should be optional.
- Can help to build asynchronous messaging.



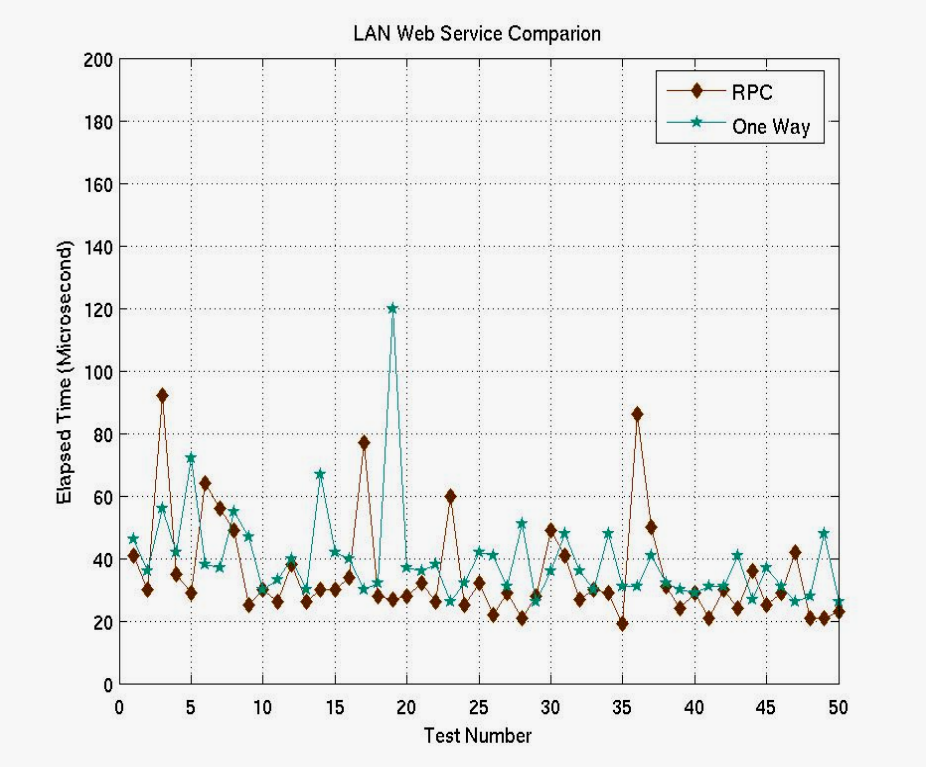
Asynchronous Messaging

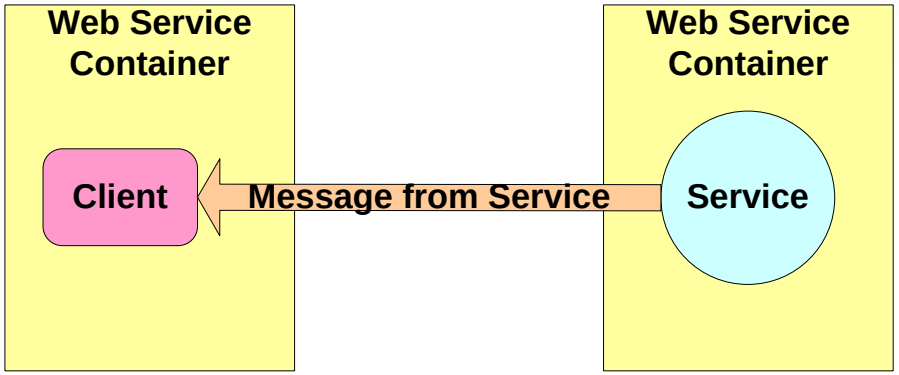
- There exist many scenarios where this is needed
  - Bundling acknowledgments.
- Services are addressable while clients are not.
- Service is not able to send subsequent response messages after a certain amount of time.



Improvement in Messaging

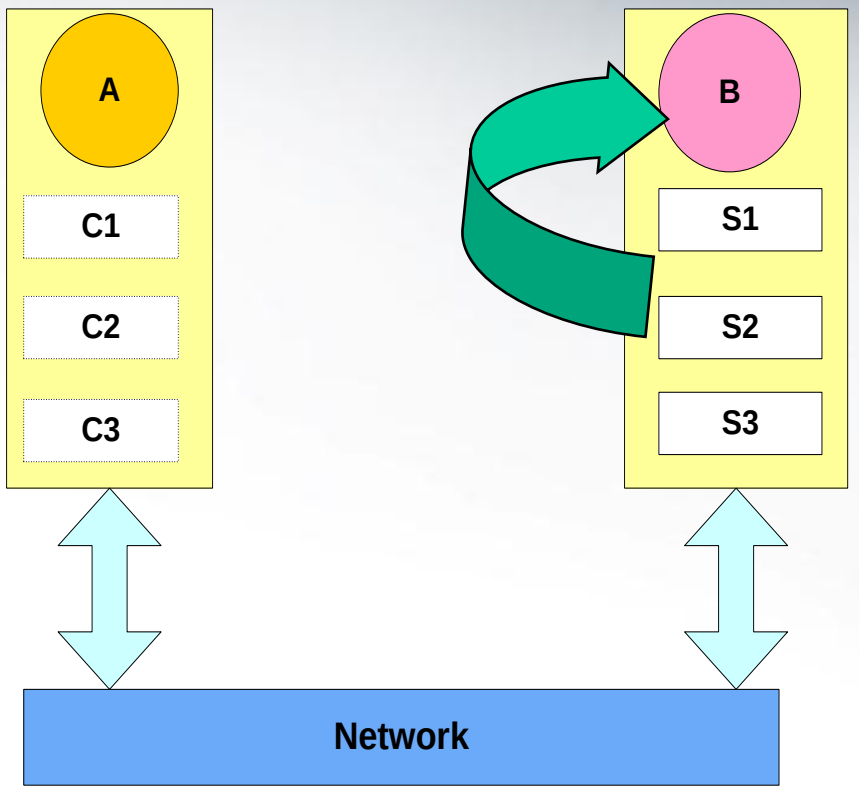
- Utilizing one way messaging costs more.
- Overhead comes from a new “service call” initiation.
- Messaging can be improved by adding message initiation capability on the service side.





Stopping Message Propagation

- A scenario: acknowledgment in reliable messaging.
- A very crucial performance issue if the endpoint gets a huge amount of acknowledgements.
- The current architecture does not allow us to stop message propagation gracefully.
- Causes network overheads; Exception propagates back through handler structure and back to the client.
- The completed tasks are rolled back if an exception is thrown.

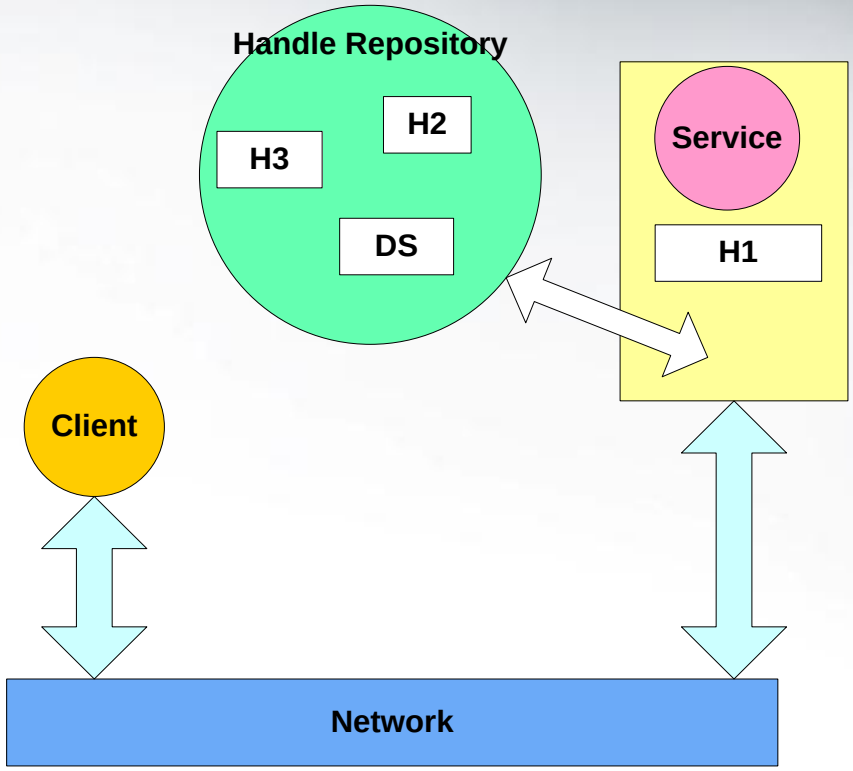


Flexible Handler I

- A processing order within a chain is important.
- There exists two options; static and dynamic.
- Currently, containers utilize the static approach.
  - Axis 2 is dynamic.
- The chain is setup when a service is being deployed.
- A new handler can not be added, just as an old one cannot be removed from the chain after deployment.
- A static structure is generally easy to implement, but harder to customize.

Flexible handler II

- Handler chains should be customizable on the fly.
- A Web Service needs to have the ability to select its handlers from the pool of handlers.
  - A digital signature handler
- A handler may need to be removed on the fly too.
  - A parameter increment Handler



Conclusion

- We suggest that containers should support one way and asynchronous messaging.
- There should be a mechanism to ensure that a message can be gracefully stopped while traversing though the handler chain.
- Handler chain should be able to deploy the handlers dynamically.