# Designing SOAP Services

http://www.pluralsight.com/

# Outline

- SOAP Service Design Styles
- Understanding the WS-* Stack
- Exposing Control Interfaces over SOAP
- Creating Fast SOAP Services
- Maximizing SOAP Service Interoperability
- Designing Secure, Interoperable SOAP Services

# SOAP Service Design Styles

- **Fire and Forget**
  - One-way MEP– handy for eventing
  - No faults returned
- **Remote Procedure Calls**
  - Request/Response MEP
  - Use SOAP Faults to return failure
  - Scale is important
- **Out of process object interaction**
  - May use Duplex MEP
  - Session frequently used
  - May use Hosted Workflow to scale

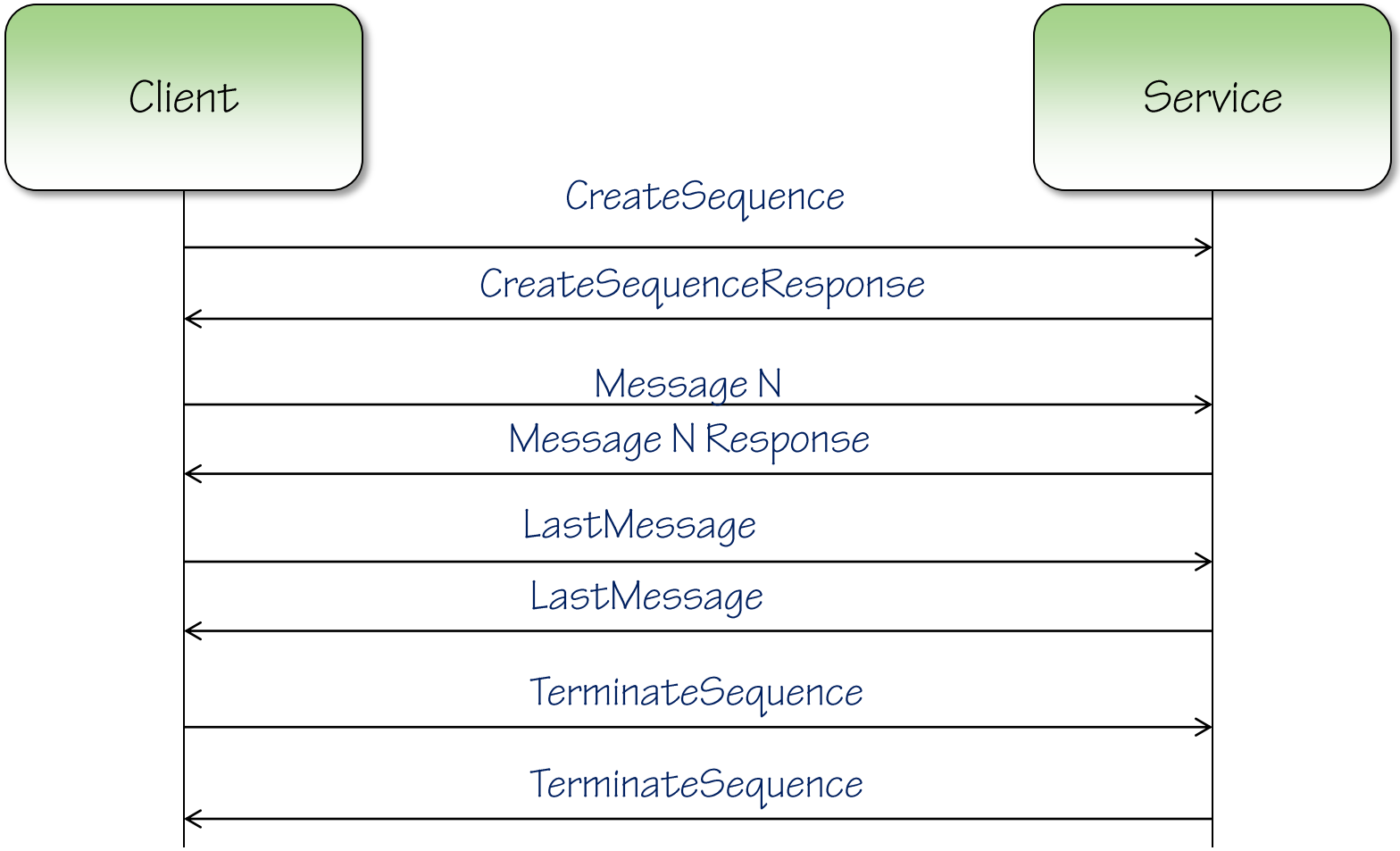
# Understanding the WS-* Stack

- **Some of WS-* is about additional headers:**
  - WS-Addressing
  - WS-AtomicTransaction
  - WS-Security
- **Some is about out of band communications:**
  - WS-MetadataExchange
  - WS-CoordinationContext
- **Some is about choreography:**
  - WS-ReliableMessaging (aka WS-RM)
  - WS-Discovery
  - WS-Trust
  - WS-SecureConversation
  - WS-Federation

# What is a choreography?

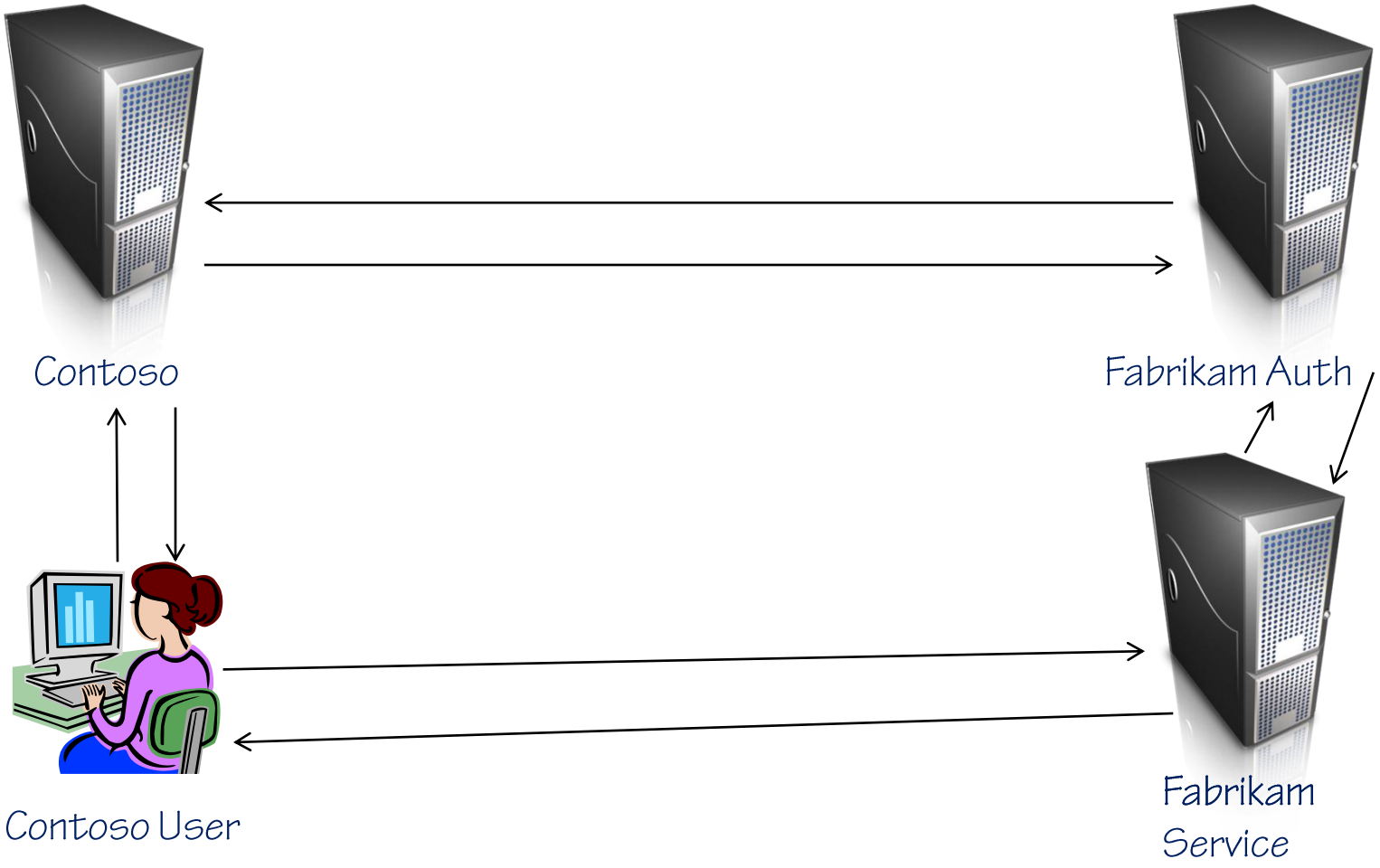
- **A choreography is a pre-arranged set of motions.**
- **These motions involve messages and take time.**
- **Questions:**
  - Can you afford the messages?
  - Can you amortize the cost of the messages by incurring the cost infrequently?
  - Answer this soon

# Choreography: WS-ReliableMessaging

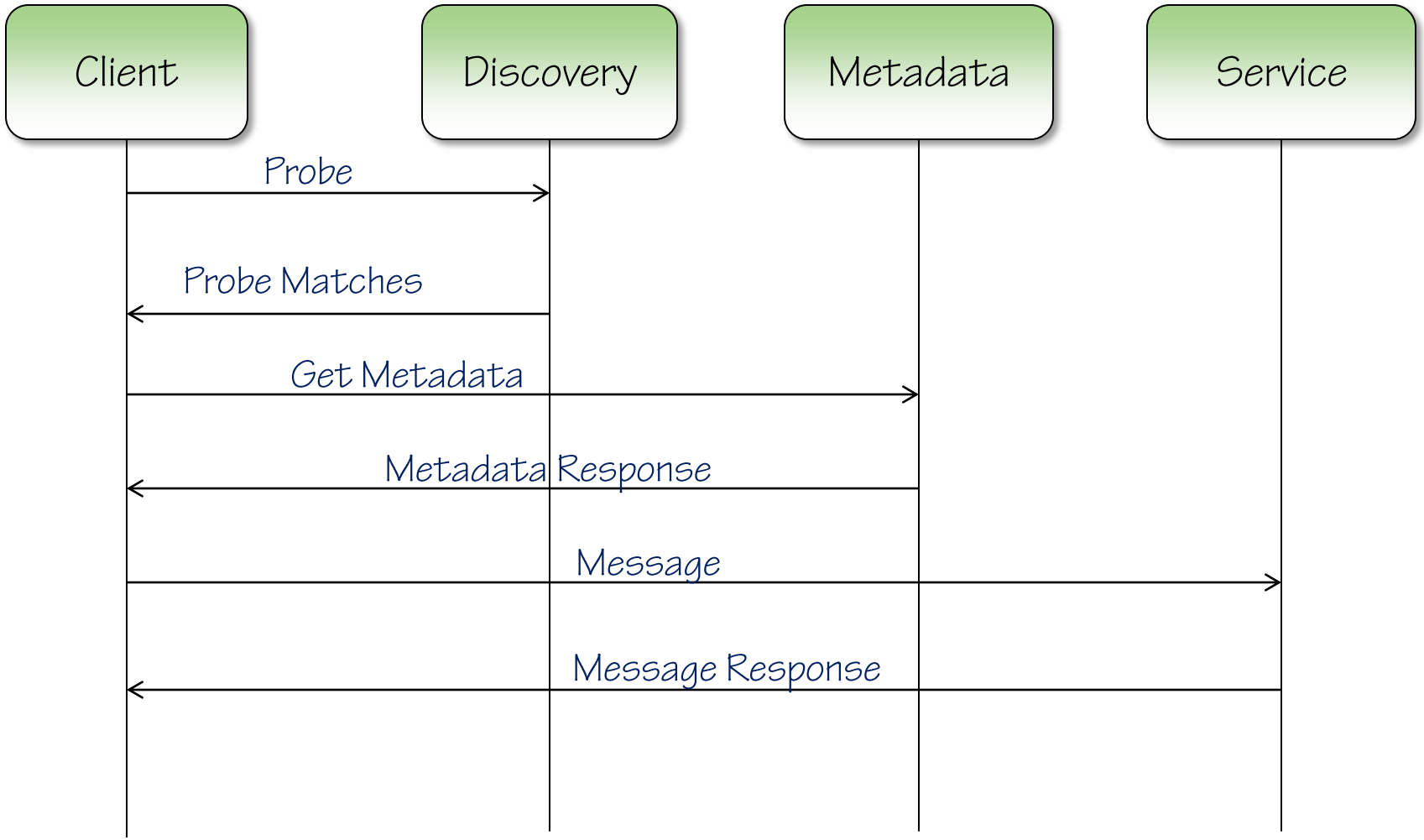


# Choreography: WS-Trust/WS-SecureConversation

# Choreography: WS-Federation



# Choreography: WS-Discovery



# Exposing Control Interfaces over SOAP

- **Control interface: An interface that allows one to change the behavior of a running object.**
- **Use Singleton instance**
- **Prefer secure binding**
  - NetTcpBinding, NetNamedPipeBinding, WSHttpBinding
- **Limit number of concurrent callers**
- **If object exposes 'events', use Duplex binding**
  - NetTcpBinding, NetNamedPipeBinding, WSDualHttpBinding

# Creating Fast SOAP Services

- **Fast: High throughput, low latency**
- **Preferred format: Binary serialization**
  - Available through NetTcpBinding, NetNamedPipeBinding, CustomBinding
- **For request/response, server should allow**
  - Transport level security. Skip message level.
  - For reliable delivery, use reliable transports (TCP, HTTP)
  - Not use duplex
  - Load balance against connections
- **Client should**
  - Cache client proxies
  - Catch exceptions indicating that the service has closed the connection and re-open the connection

# Designing for fast one-way operations

```
[OperationContract(IsOneWay = true)]
void DoWork(WorkItem work);
```

- **void return type**
- **No FaultContract**
- **IsOneWay set to true**
- **Advantage:**
  - Client perceives message as fast: no need to wait for processing.
  - Server side can scale
- **If using this across a service, use per-call instancing, load balance calls.**
- **Do not use session**

# Maximizing SOAP Service Interoperability

- **Most important: Limit protocol choice to HTTP**
- **Security: Use transport authentication**
  - Basic authentication over SSL: Credentials go in plain text but SSL encrypts the transport.
- **Binding to use: BasicHttpBinding**
- **Avoid: Reliable Messaging, Transactions, message level security**
- **Design for:**
  - Per-call instancing
  - Avoid System.Data.DataSet
  - Keep data structures simple: Arrays, strings, DateTime, numbers, and complex data types.

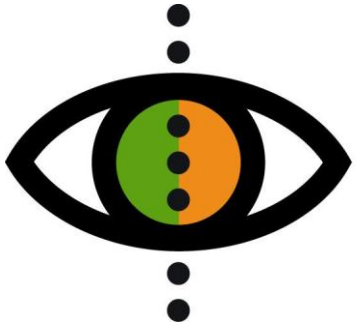
# Designing Secure, Interoperable SOAP Services

- **WS-Security/WS-Trust/WS-SecureConversation**
  - user name/password, X.509 certificate, SAML assertion, Kerberos ticket, or issued token from a third-party trust authority
- **Java**
  - Apache CXF: http://cxf.apache.org/
  - JBossWS: http://community.jboss.org/wiki/JBossWS
  - Metro: http://java.sun.com/webservices/
  - XML and WebServices Security: https://xwss.dev.java.net/
- **Various**
  - $WSO_2$ C based implementation with hooks for Java, C, C++, PHP, Perl, Python, Ruby, Spring, Jython

# Summary

- Primary design styles for services: event, RPC, and out of process object interaction
- Know the WS-* choreographies. Security, ReliableMessaging, and Discovery do involve a bootstrap process and cancel process that take time.
- Control interfaces should be designed around a small number of concurrent clients.
- For fast services, depend on WCF to WCF communications and binary serialization. Use transport security if you need to secure the conversation.
- For interoperability, HTTP is your best bet. Lately, more and more security and reliable messaging is finding its way into enterprise toolkits.

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust