CSE 446 / CSE598
Software Integration and Engineering

Unit 1
Service Standards and Service Development

Lecture 1-3
Advanced Service Development

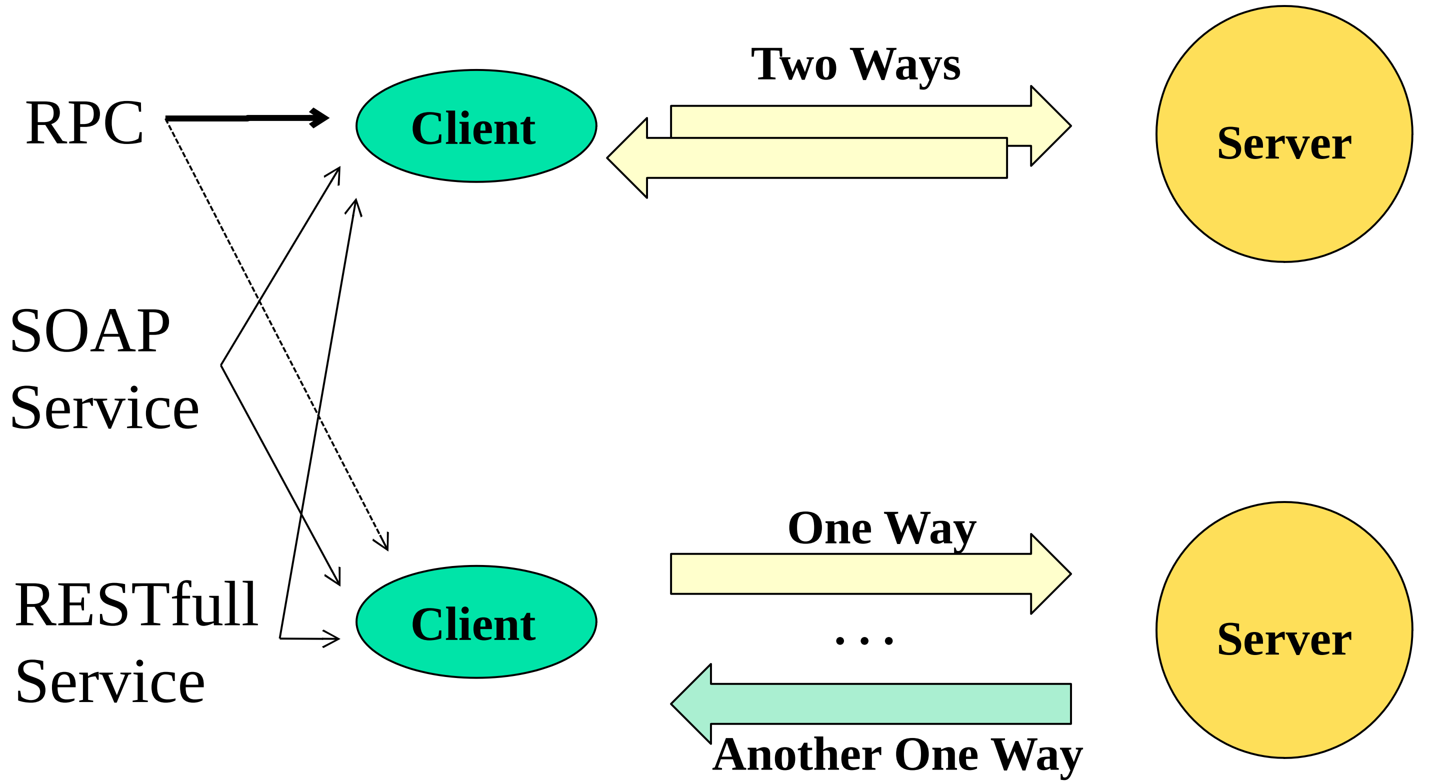
Dr. Yinong Chen
https://myasucourses.asu.edu/

Lecture Outlines

- Models of Distributed Computing
- Channels for Communication and Interfacing
  - One-way
  - Request-Reply
  - Duplex
- Bindings
  - For WSDL-SOAP services
  - For RESTful Services
  - For .Net Remoting
- Behaviors and service behaviors
  - Instancing
  - Concurrency

Models of Distributed Computing

- Remote Procedure Call (RPC) – Typically, Tightly Coupled/ Synchronous Communication
- Remote Invocation with Message Exchange – Intended for Loosely Coupled/Asynchronous Communication

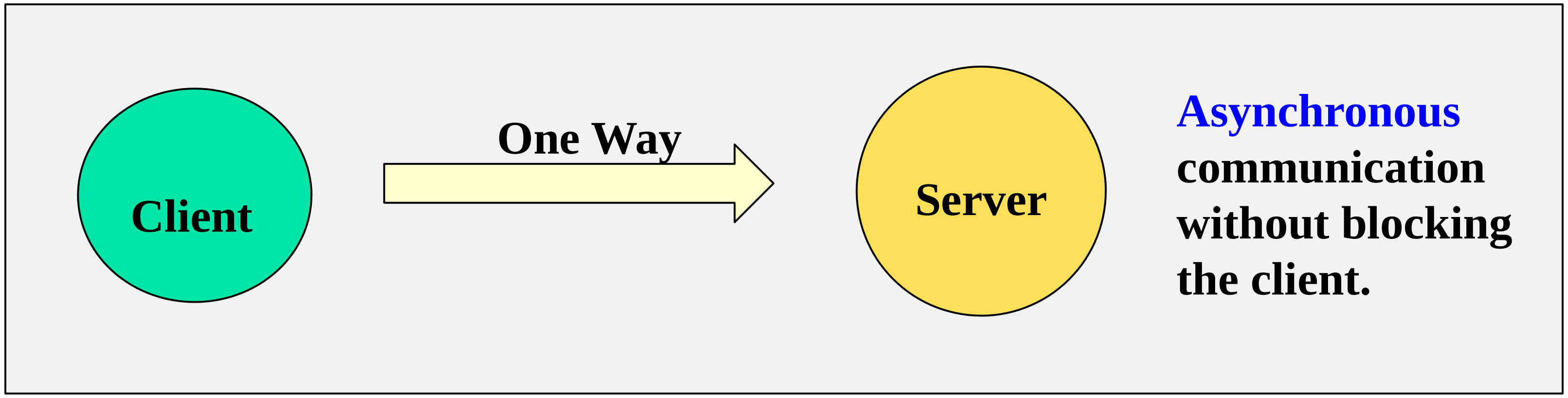


Synchronous communication: Have to wait for the response, even for *void* return type. It may hold the client for a long time.

Asynchronous communication without blocking the client.

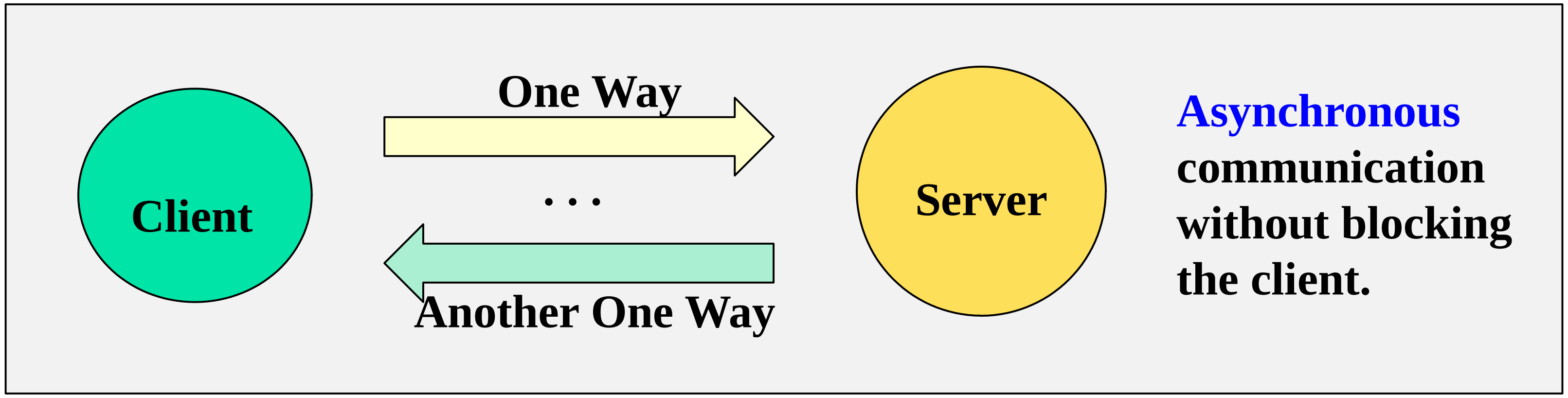
How to Communicate Asynchronously (0)

If no return result is needed, easy:
void function(types parameters);

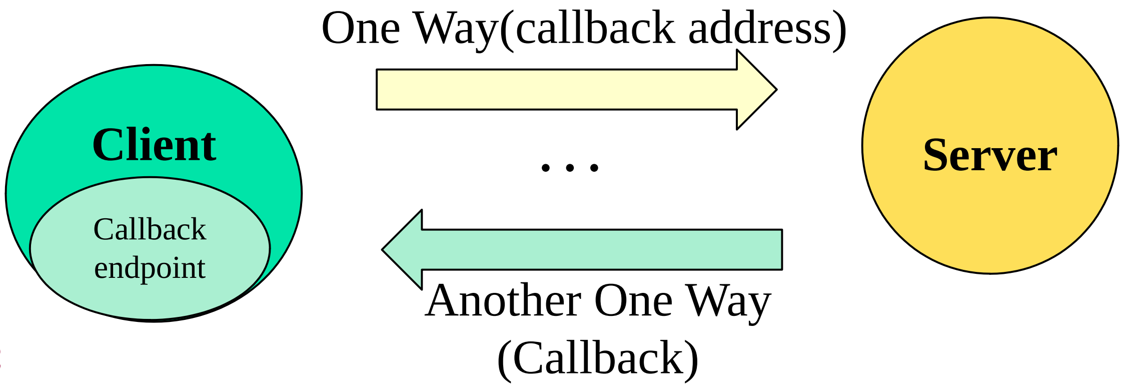


How do we do asynchronous communication requiring return results (two-way asynchronous communication)?

How to Communicate Asynchronously (1)

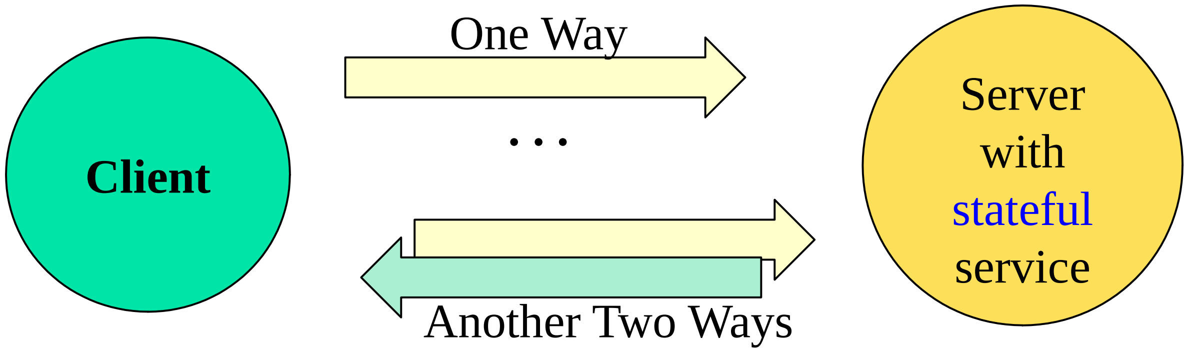


- The sever calls back
  - Pass the callback address when the client calls the service
  - The service calls the client using the callback address



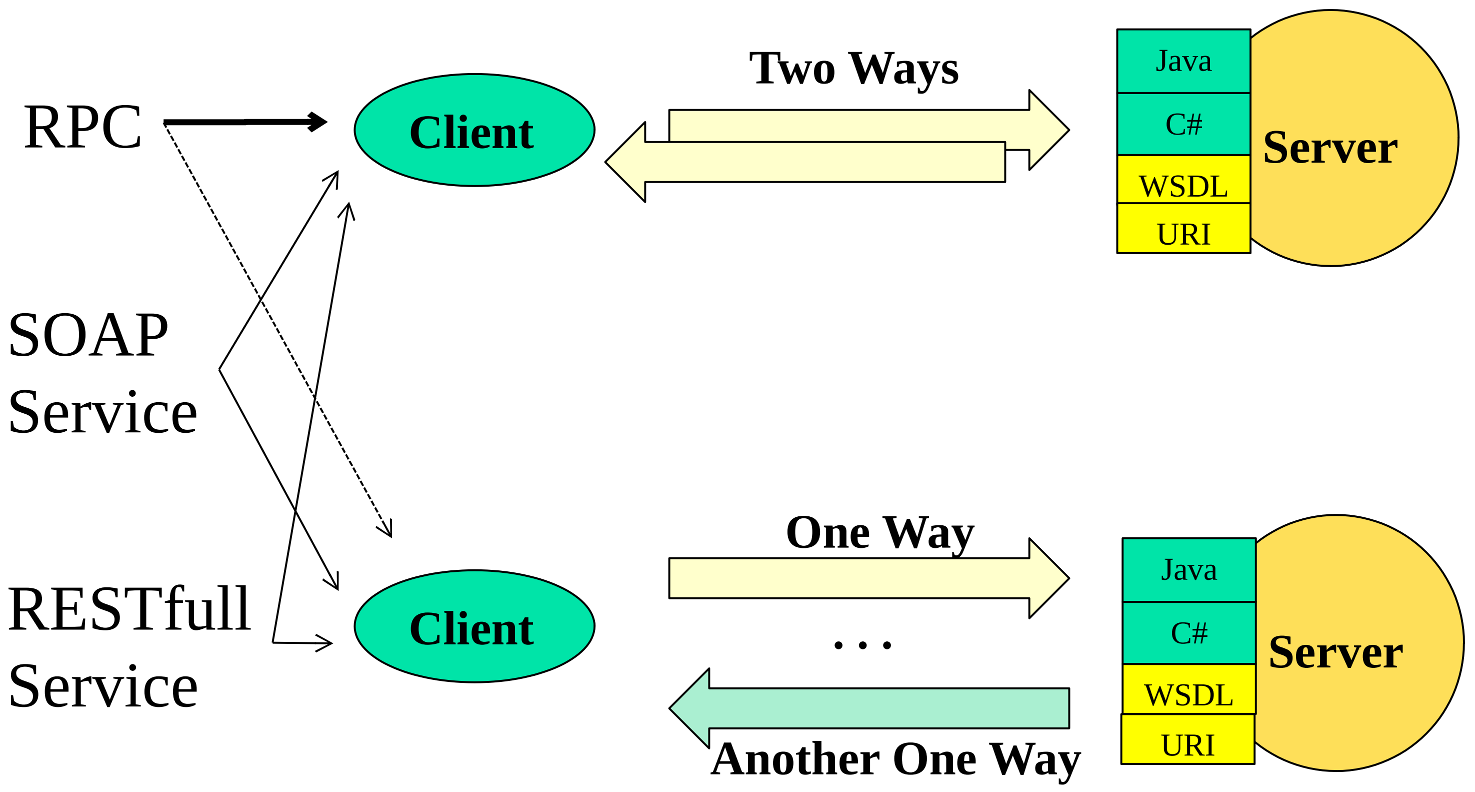
How to Communicate Asynchronously (2)

- The sever provides a stateful service and keeps the data for another call from the client;
  - Without state, the service would have to block-wait the second call to deliver the result, which is not acceptable to the server
- The second call can be triggered by an event from the server, or the client's polling a state
  - The client can make the second call with some delay.



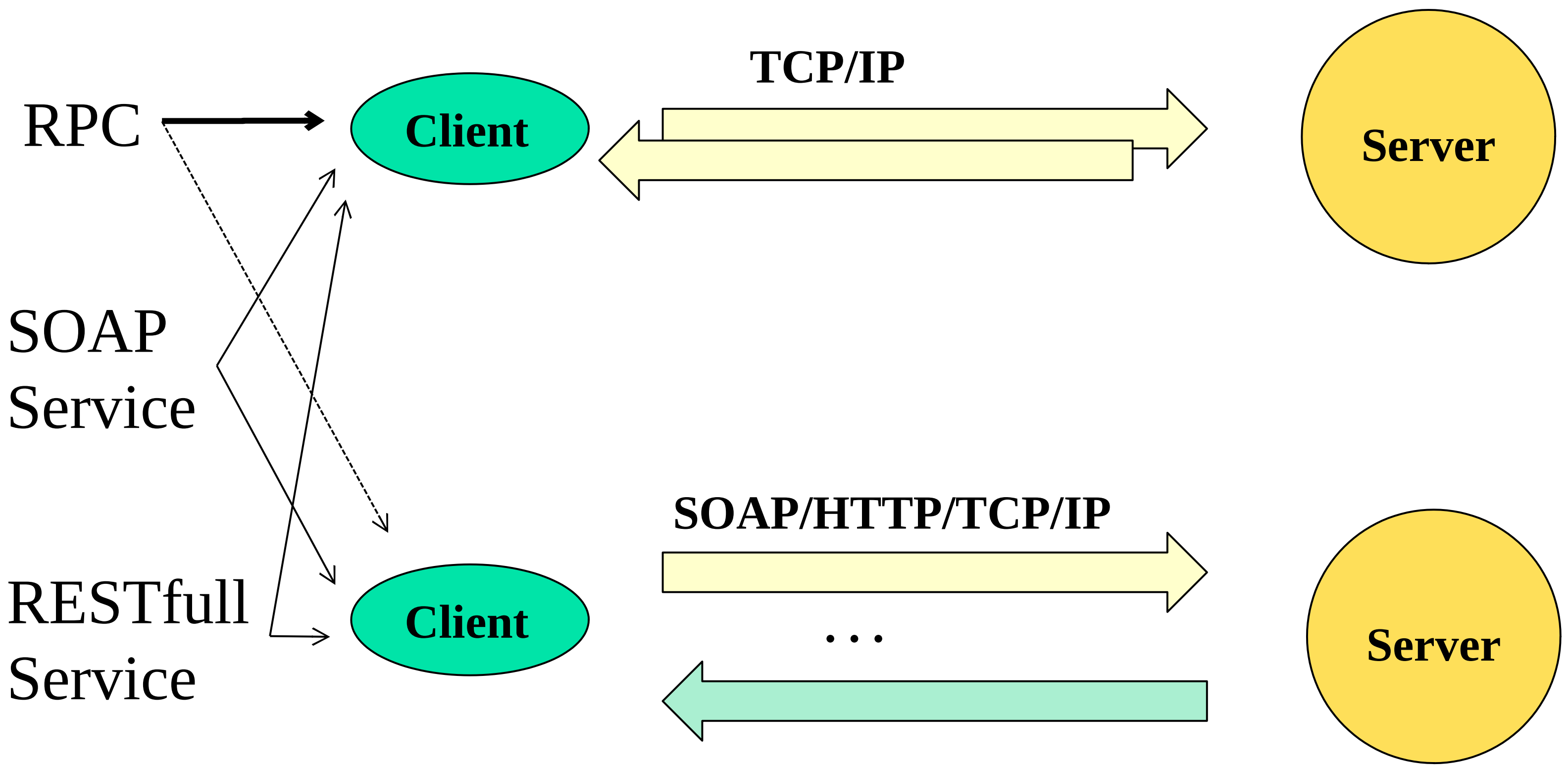
Interfacing

- Remote Procedure Call (RPC) – Typically: Language-based
- Remote Invocation with Message Exchange – Typically: WSDL/URI



Communication Protocols

- Remote Procedure Call (RPC) – Typically: TCP/IP Channel
- Remote Invocation with Message Exchange – SOAP/HTTP

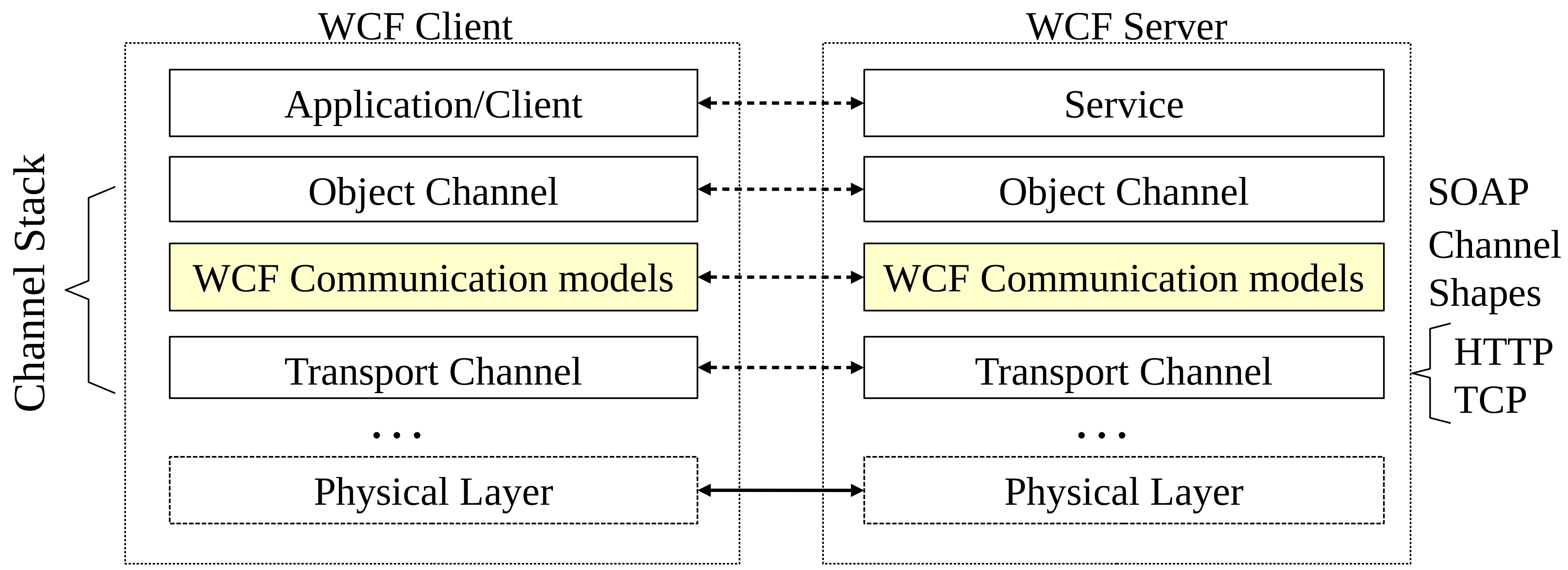


WCF Protocol Channel Stack

http://msdn.microsoft.com/en-us/library/ms729840.aspx

- A channel provides a programming model for sending and receiving messages between a client and a server.
- WCF channel stack is a layered communication stack with one or more channels that process messages:
  - SOAP (Simple Object Access Protocol)
  - WCF channel shapes
  - The transport channel is responsible for adapting the channel stack to the underlying transport, for example, TCP, HTTP, SMTP and other types of transport protocols.

WCF Communication Models and Protocol Channel Stack

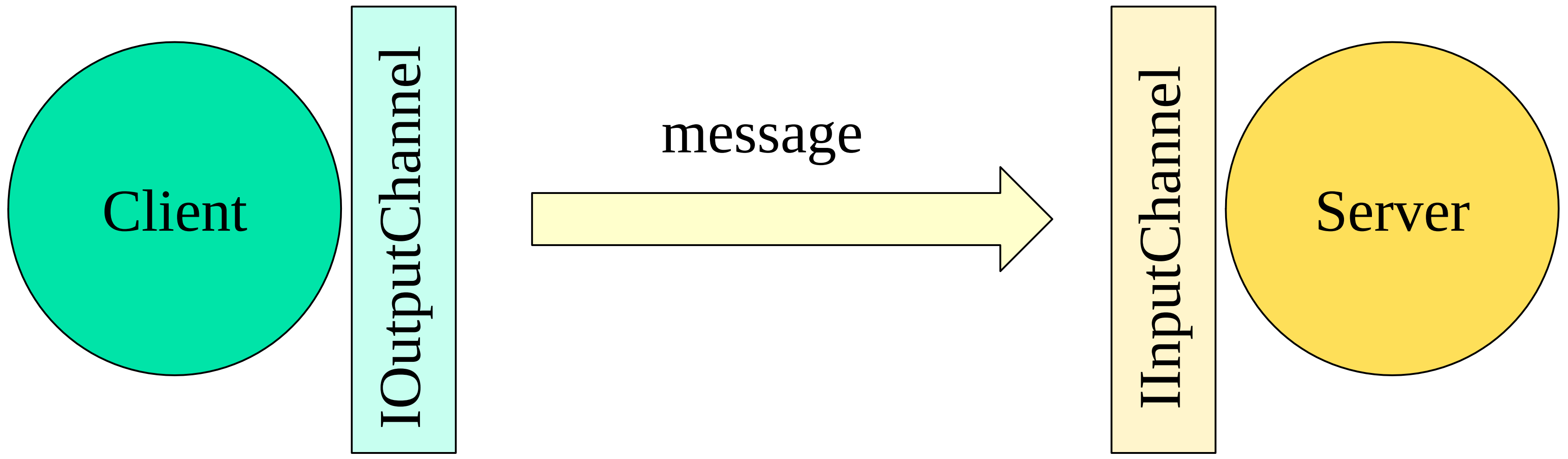


WCF Communication Models

- WCF supports three communication models
  - Request-reply (Two-way synchronous communication)
  - One-way (asynchronous communication without returning result)
  - Duplex (asynchronous communication with returning result)
- To support these models, without changing the remote services, a set of channel shapes are implemented:
  - IInputChannel
  - IOutputChannel
  - IDuplexChannel
  - IRequestChannel
  - IReplyChannel

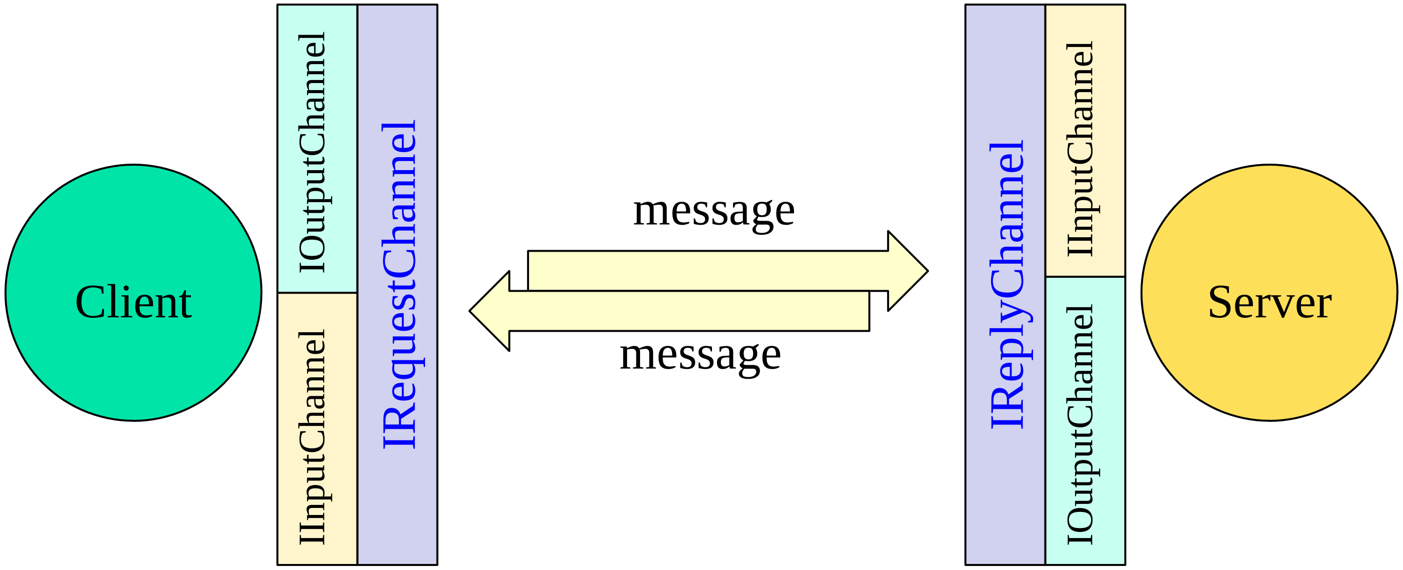
WCF One-Way Communication Model

- One-way communication model is used in the situation where the client sends a message to the server without requesting a return value
- The channel shapes IOutputChannel and IInputChannel are used to implement the one-way communication:



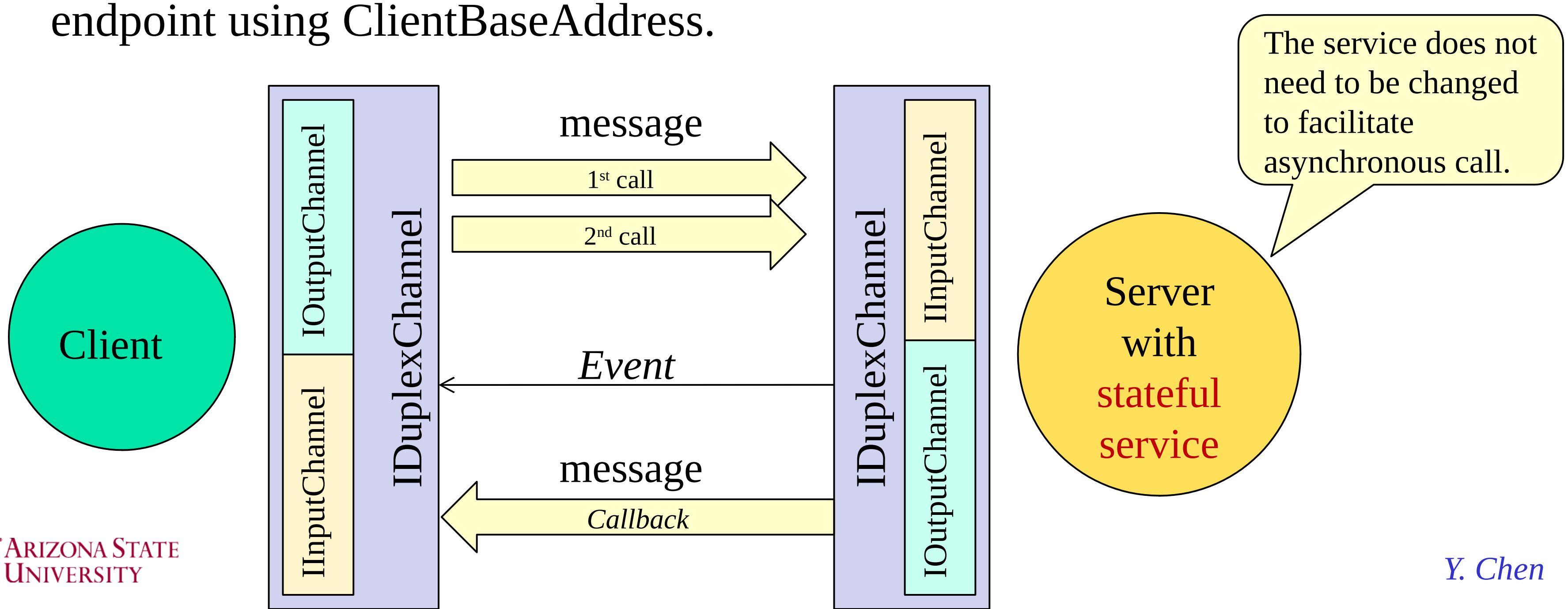
WCF Request-Reply Model (Default)

- Request-reply communication model combines two one-way channel shapes into IRequestChannel and IReplyChannel
- The model is used for synchronous communications:
  1. Client sends a request and block-waits for response,
  2. Server processes the request and sends response back to client.

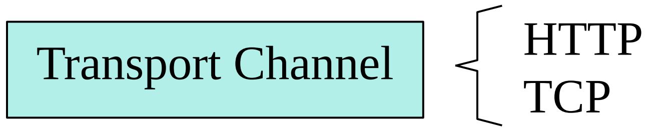


WCF Duplex Communication Model

- Duplex communication model uses two one-way channel shapes to compose a new shape called IDuplexChannel
- This model facilities asynchronous exchange. Two options:
  - The client makes two calls. Make the second call after receiving an event.
  - Client sends a request to server, and the service handles the request. It may take a long time to complete/calculate the result. After completion, the service calls back to provide results. Client need to create a public URI as a callback endpoint using ClientBaseAddress.

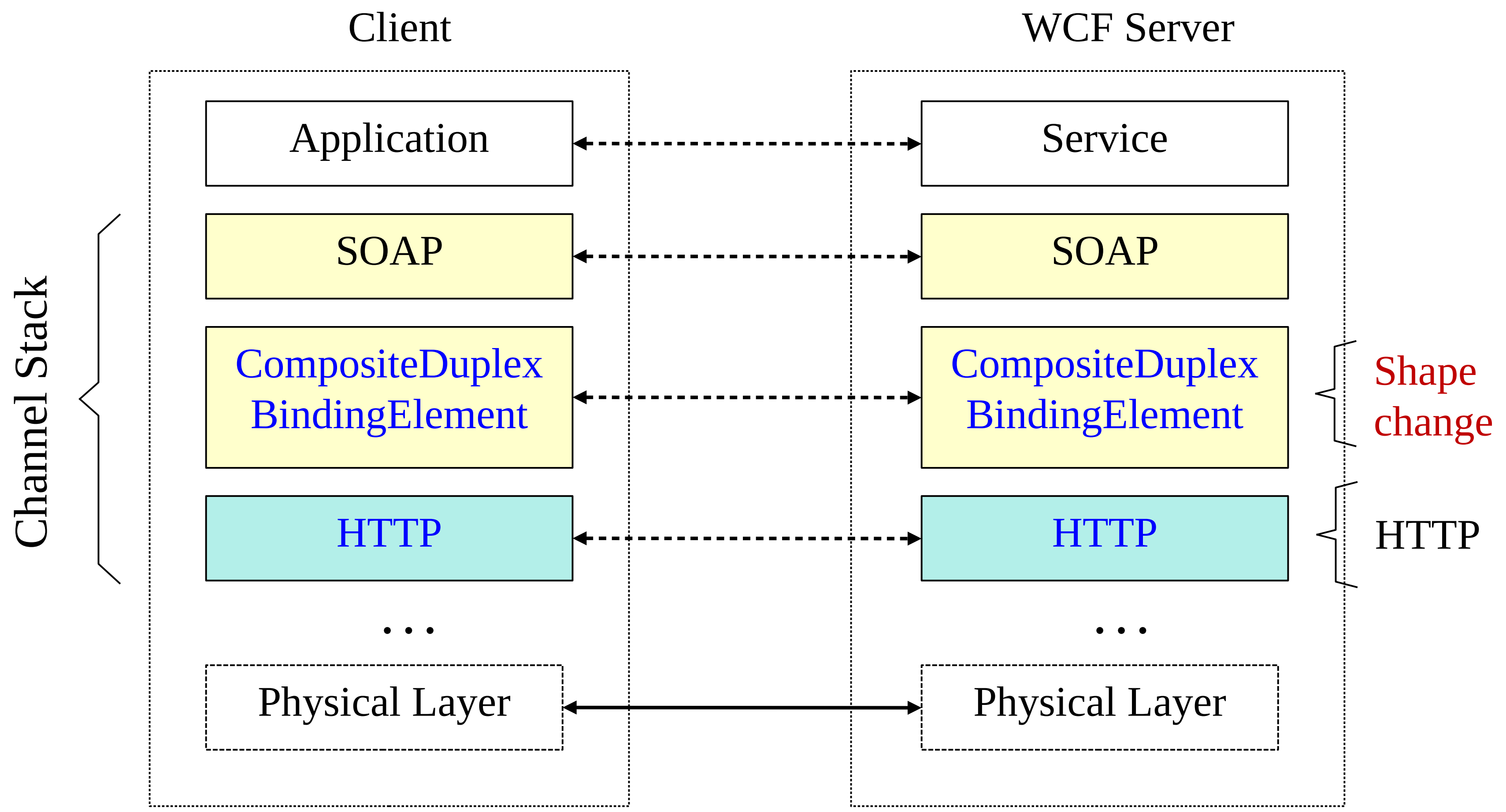


Transport Channel HTTP and TCP



- HTTP and TCP are the main transport channels
- HTTP automatically supports request-reply model
- TCP supports one-way and duplex
- In order to do one-way and duplex on HTTP, a shape-changing layer is needed, which involves a number of classes for specification and binding:
  - OperationContractAttribute Class
  - OneWayBindingElement class for one-way
  - CompositeDuplexBindingElement class for duplex

WCF Duplex Channel Stack: Example



How Do We Write Asynchronous Service?

http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.aspx

```
[ServiceContractAttribute]
public class ChannelModelExample
{     // The client waits until a response message appears.
[OperationContractAttribute]
      public int MethodOne (int x, int y) { return x+y; }
      // The client generates asynchronous calls.
[OperationContractAttribute(IsOneWay=true)]
      public void MethodTwo (int x) { f(x); return; }
      [OperationContractAttribute(AsyncPattern = true)]
      public void MethodThree (int x, out int y)  {
            y = complexFunction(x); return }
      // The client returns as soon as an outbound message is dispatched
      // to the service; no response is generated or sent from the service.
       // Variable y will be accessed later
}
```

Request-rely by default

Synchronous, block waiting

one-way model

Duplex model

Output variable

Use void

Properties of OperationContractAttribute

Action	Gets or sets the WS-Addressing action of the request message.
AsyncPattern = true	Indicates that an operation is implemented asynchronously using a Begin<methodName> and End<methodName> method pair in a service contract.
HasProtection Level	Gets a value that indicates whether the messages for this operation must be encrypted, signed, or both.
IsInitiating	Gets or sets a value that indicates whether the method implements an operation that can initiate a session on the server (if such a session exists).
IsOneWay = true	Gets or sets a value that indicates whether or not an operation returns a reply message.
IsTerminating	Gets or sets a value that indicates whether the service operation causes the server to close the session after the reply message, if any, is sent.
Name	Gets or sets the name of the operation.
Protection Level	Gets or sets a value that specifies whether the messages of an operation must be encrypted, signed, or both.
ReplyAction	Gets or sets the value of the SOAP action for the reply message of the operation.
TypeId	When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

Two-Call Interface of Async

http://msdn.microsoft.com/en-us/library/system.servicemodel.operationcontractattribute.asyncpattern.aspx

```
[ServiceContract]
public interface IAddTwoNumbers
{     // If the asynchronous method pair appears on the client channel,
      // the client can call them asynchronously to prevent blocking.
      [OperationContract (AsyncPattern=true)]
      IAsyncResult BeginAdd(int a, int b, AsyncCallback cb, AsyncState s);

      [OperationContract]
      int EndAdd(IAsyncResult r); // 2nd call to obtain the result

      // This is a synchronous version of the BeginAdd and EndAdd pair.
      // It can be generated in the client channel code using utility toll.
      [OperationContract]
      int Add(int a, int b);
}
```

Give imple-mentation in another file

out

A full working example of asynchronous call will be given in a later lecture.

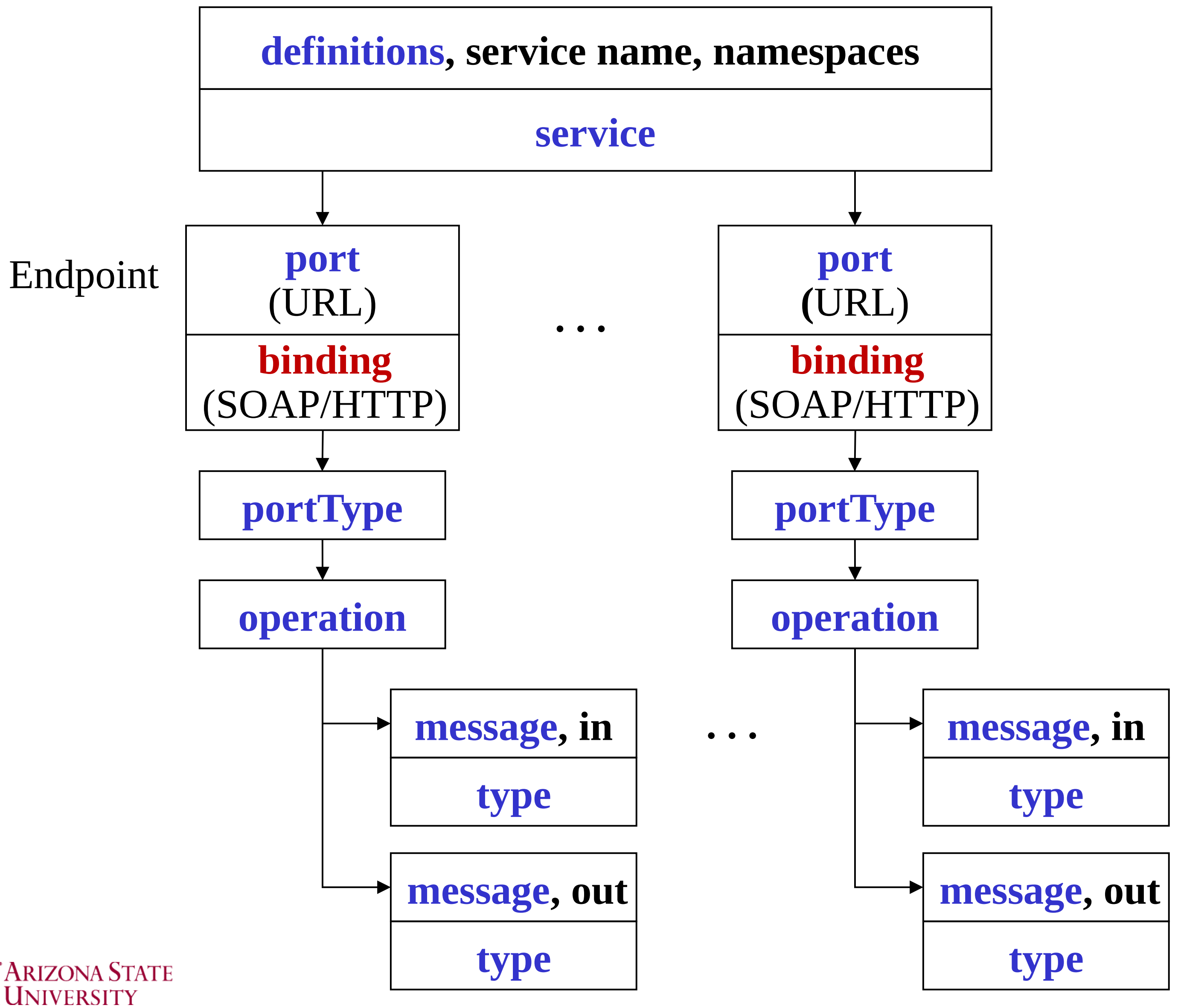
Lecture Outlines

- Models of Distributed Computing
- Channels for Communication and Interfacing
  - One-way
  - Request-Reply
  - Duplex
- Bindings
  - For WSDL-SOAP services
  - For RESTful Services
  - For .Net Remoting
- Behaviors and service behaviors
  - Instancing
  - Concurrency

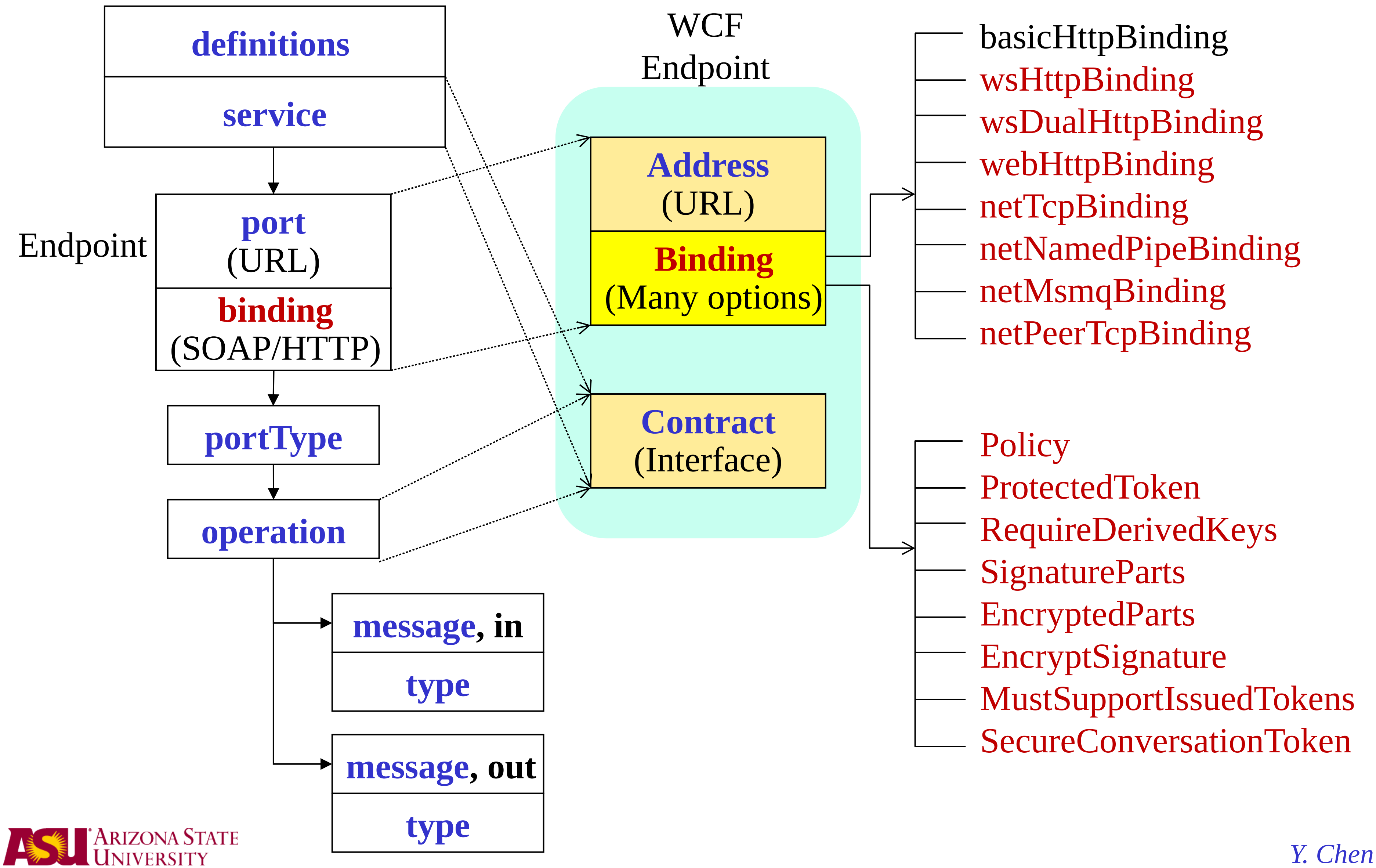
Advanced Bindings

- Bindings are pre-configured channel stacks;
- Bindings are the communication agreement between the client and service;
- A binding includes:
  - Protocols (stack) involved
  - Transport
  - Encryption and encoding, such as object to string, JSON, or to XML
- Common applications have pre-defined bindings, see next page:

Basic WSDL Document's Elements



WSDL Document's Elements and WCF Extensions



Bindings for Common Services

Binding	Description
BasicHttpBinding	A binding that is suitable for communicating with WS-Basic Profile conformant Web services, for example, ASP .NET Web services (ASMX)-based services. This binding uses HTTP as the transport and text/XML as the default message encoding.
WsHttpBinding	A secure and interoperable binding that is suitable for non-duplex service contracts, for example RESTful service. Encoding methods include XML, POX, JSON
Ws2007HttpBinding	A secure and interoperable binding that provides support for the correct versions of the Security, ReliableSession, and TransactionFlow binding elements.
WsDualHttpBinding	A secure and interoperable binding that is suitable for duplex service contracts or communication through SOAP over HTTP channel stack.

Bindings for Common Services (contd.)

Binding	Description
NetTcpBinding	A secure and optimized binding suitable for cross-machine communication between WCF applications (.Net Remoting)
NetNamedPipeBinding	A secure, reliable, optimized binding that is suitable for on-machine communication between WCF applications (.Net Remoting).
NetMsmqBinding	A queued binding that is suitable for cross-machine communication between WCF applications (.Net Remoting).
NetPeerTcpBinding	A binding that enables secure, multi-machine communication (peer).
WebHttpBinding	A binding used to configure endpoints for WCF Web services that are exposed through HTTP requests (RESTful services) instead of SOAP messages.

Binding Features

Binding	Interoperability standards	Mode of Security (Default)	Session (Default)	(Default) Transactions
BasicHttp Binding	Basic Profile 1.1	(None), Transport, Message, Mixed	None, (None)	(None), No
WSHttp Binding	WS-I	None, Transport, (Message), Mixed	(None), Transport, Reliable Session	(None), Yes
WS2007Http Binding	WS-Security, WS-Trust, WS-SecureConversation, WS-SecurityPolicy	None, Transport, (Message), Mixed	(None), Transport, Reliable Session	(None), Yes
WSDualHttp Binding	WS-I	None, (Message)	(Reliable Session)	(None), Yes

Binding Features (contd.)

Binding	Inter-operability	Mode of Security (Default)	Session (Default)	(Default) Transactions
NetTcp Binding	.NET	None, (Transport), Message, Mixed	Reliable Session, (Transport)	(None), Yes
NetNamed PipeBinding	.NET	None, (Transport)	None, (Transport)	(None), Yes
NetMsmq Binding	.NET	None, Message, (Transport), Both	(None)	(None), Yes
NetPeerTcp Binding	Peer	None, Message, (Transport), Mixed	(None)	(None), No
MsmqIntegration Binding	MSMQ	None, (Transport)	(None)	(None), Yes

Define Binding in Web.config File

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.serviceModel>
        <bindings>
            <wsHttpBinding>
                <binding name="WSHttpBinding_ICalculator“ />
            </wsHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://localhost:8000/"
    binding="wsHttpBinding"
bindingConfiguration="WSHttpBinding_ICalculator"
contract="Microsoft.ServiceModel.Samples.ICalculator"
name="WSHttpBinding_ICalculator">
            </endpoint>
        </client>
    </system.serviceModel>
</configuration>
```

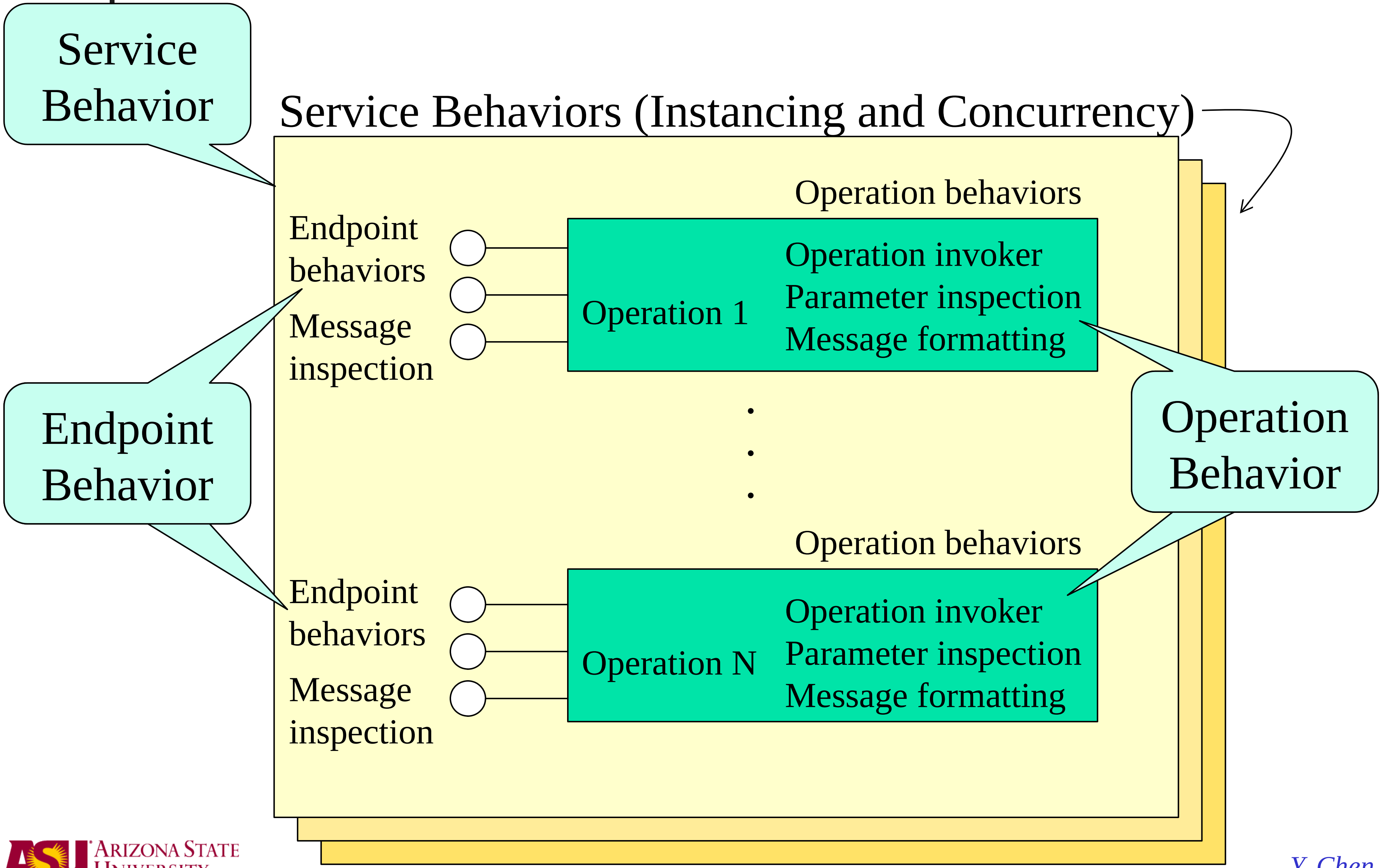
Lecture Outlines

- Models of Distributed Computing
- Channels for Communication and Interfacing
  - One-way
  - Request-Reply
  - Duplex
- Bindings
  - For WSDL-SOAP services
  - For RESTful Services
  - For .Net Remoting
- Behaviors and service behaviors
  - Instancing
  - Concurrency

Behaviors

- Behaviors here refer to the WCF classes that affect runtime operations, for example, the ServiceHost class.
- There are three levels of behaviors:
  - Service behaviors: concern instancing and overall transactions;
  - Endpoint behaviors: inspecting and taking actions on incoming and outgoing messages;
  - Operation behaviors: manipulation, serialization (convert object to string), transaction flow, and parameter handling.

Three Levels of Behaviors



Service Behaviors: Concurrency and Instancing

- WCF Service Behaviors offer two modes to control the service level behaviors:
  - InstanceContextMode
  - ConcurrencyMode
- The target of instancing mode is for the state management
- The target of concurrency mode is for increasing the number of tasks completed in the given time period to achieve the multithreading and parallel computing

State Management in ASP .Net Applications

Web applications are stateless by default. However, the following mechanisms can be used to achieve stateful applications (clients):

- View State: Save data in the browser page for multiple access of the data
- Session State: Save the data in Session[“nameIndex”] for all instances of the same browser session to repeatedly access.
- Application State: Save the data in Application[“nameIndex”] for all instances of all browser sessions to repeatedly access.

- These methods cannot be used in Web services!

State Management in WCF for Services

InstanceContextMode implements state management in WCF services. It can take one of the values

- *PerCall*: one instance is created for each incoming request. This mode creates stateless service;
- *PerSession*: one instance is created for each client session. This mode corresponds to session state at application level;
- *Single*: one instance of the service class handle all incoming requests (singleton service). This mode corresponds to application state at application level.

Design pattern

RESTful Example of Using InstanceContextMode

```
using System; using System.ServiceModel;
namespace WcfRestService4 {
    [ServiceContract]
    [AspNetCompatibilityRequirements(RequirementsMode =
    AspNetCompatibilityRequirementsMode.Allowed)]
    [ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]
    public class Service1 {
        [OperationContract]
        [WebGet(UriTemplate = "add2?x={x}&y={y}", ResponseFormat =
    WebMessageFormat.Json)]    // Add this HTTP GET attribute/directive
        public int addition(int x, int y) {
            return (x+y);
        }
    }
}
```

SVC Example of Using InstanceContextMode

http://neptune.fulton.ad.asu.edu/WSRepository/Services/Singleton/service.svc

```
using System;
using System.ServiceModel;
[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
public class Service : IService {
    int sNumber = 1000;
    public Int32 getNumber()
    {
        return sNumber;
    }
    public  Int32 takeOne()
    {
        sNumber = sNumber-1;
        return sNumber;
     }
}
```

No specific web states are defined

No specific web states are defined

```
// IService.cs
using System;
using System.ServiceModel;
[ServiceContract]
public interface Iservice  {
    [OperationContract]
    Int32 getNumber();
    [OperationContract]
    Int32 takeOne();
}
```

Deployed service address:
http://neptune.fulton.ad.asu.edu/WSRepository/Services/NumberGuess/Service.svc

Client that Accesses the Singleton Service

http://neptune.fulton.ad.asu.edu/WSRepository/Services/singletonTryIt/

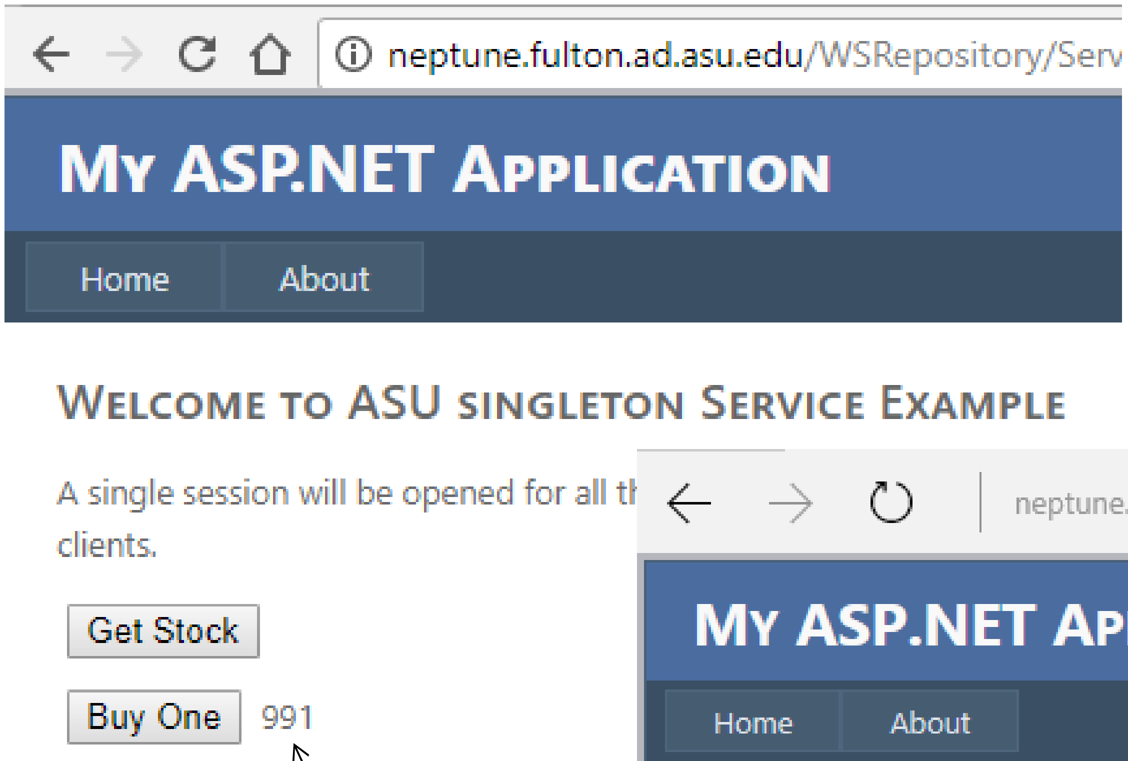
```
using System;
public partial class _Default : System.Web.UI.Page {
    protected void Page_Load(object sender, EventArgs e) {  }
    protected void btnStock_Click(object sender, EventArgs e)  {
        singletonService.ServiceClient myProxy = new
     singletonService.ServiceClient();
        Int32 sNumber = myProxy.getNumber();
        lblStock.Text = Convert.ToString(sNumber);
        myProxy.Close();
    }
    protected void btnBuy_Click(object sender, EventArgs e)  {
        singletonService.ServiceClient myProxy = new
     singletonService.ServiceClient();
        Int32 sNumber = myProxy.takeOne();
        lblRemain.Text = Convert.ToString(sNumber);
        myProxy.Close();
    }
}
```

No application states are defined

No application states are defined

TryIt from Multiple Computers

http://neptune.fulton.ad.asu.edu/WSRepository/Services/singletonTryIt/


neptune.fulton.ad.asu.edu/WSRepository/Serv

My ASP.NET Application

Home About

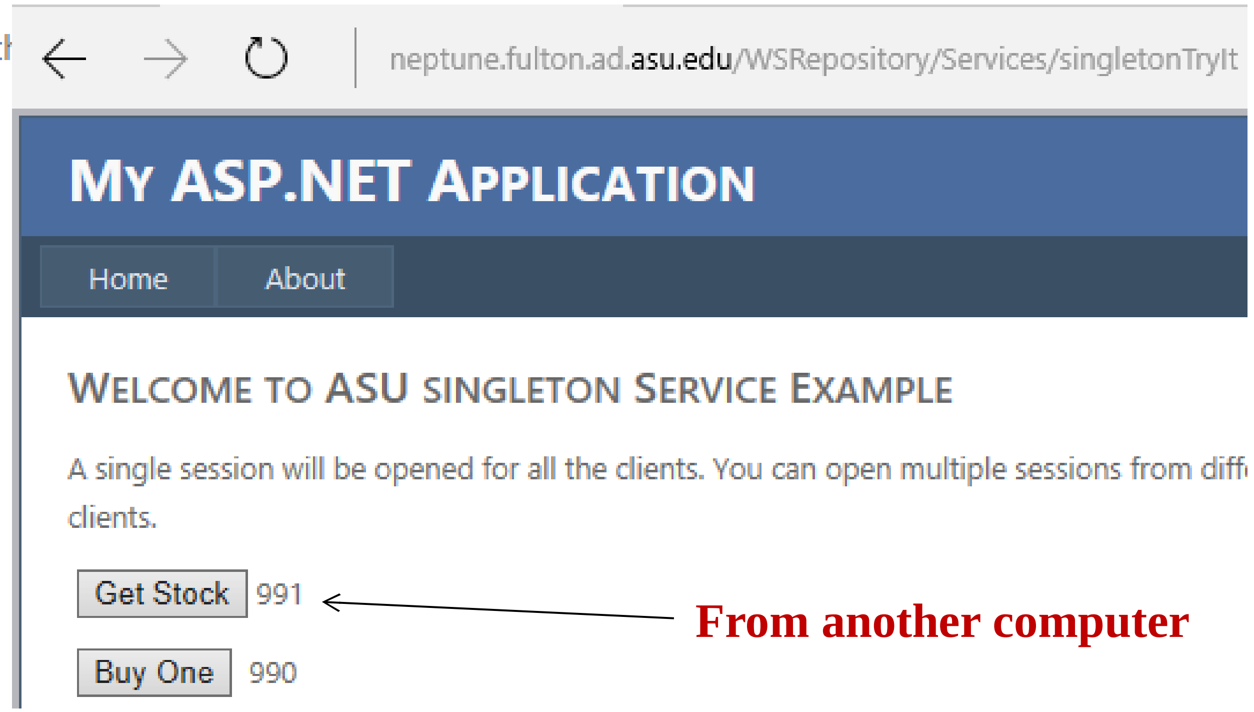
Welcome to ASU singleton Service Example

A single session will be opened for all th
clients.

Get Stock

Buy One 991

From one computer


neptune.fulton.ad.asu.edu/WSRepository/Services/singletonTryIt

My ASP.NET Application

Home About

Welcome to ASU singleton Service Example

A single session will be opened for all the clients. You can open multiple sessions from diff
clients.

Get Stock 991

Buy One 990

From another computer

Service Behaviors: Concurrency and Instancing

ConcurrencyMode: Used to control threading within one service instance. It can take one of the values:

- *Single*: Only one thread at a time can access the service instance.
- *Reentrant*: Only one thread at a time can access the service instance, but the thread can leave the instance and reenter later to continue.
- *Multiple*: multiple threads can access the service instance simultaneously. This setting implements parallel computing and requires the service class to be thread-safe (with synchronization/lock mechanisms.

Program Example: Single Thread

```
using System;
using System.ServiceModel;

[ServiceContract]
public interface IHttpFetcher {
      [OperationContract]
      string GetWebPage(string address);
}
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Single)]
class SingleCachingHttpFetcher : IHttpFetcher {
      public string GetWebPage(string address)  {
            if (this.cachedAddress == address) {
                  return this.cachedWebPage; }
            // else fetch page and save cache
      }
}
```

- This service takes URL in string and return the Web page at the URL as a string;
- Assuming cache is valid;
- How do we caching and what to cache? Text 5.5.
- Only one instance is running. Multiple request will be queued.

- How do we improve performance without queuing?

Program Example: Multiple Threads

```
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Multiple)]
class MultipleCachingHttpFetcher : IHttpFetcher {
  string cachedWebPage;
  string cachedAddress;
  readonly SlowHttpFetcher slow;
  readonly object ThisLock = new object();
  public MultipleCachingHttpFetcher() {
    this.slow = new SlowHttpFetcher();
  }
  public string GetWebPage(string address) {
    lock (this.ThisLock)
    {
      // <-- Can assume cache is valid.
      if (this.cachedAddress == address)
      {
          return this.cachedWebPage;
          // <-- Must guarantee that cache is valid because
          // the operation returns and releases the lock.
      }
```

Multiple

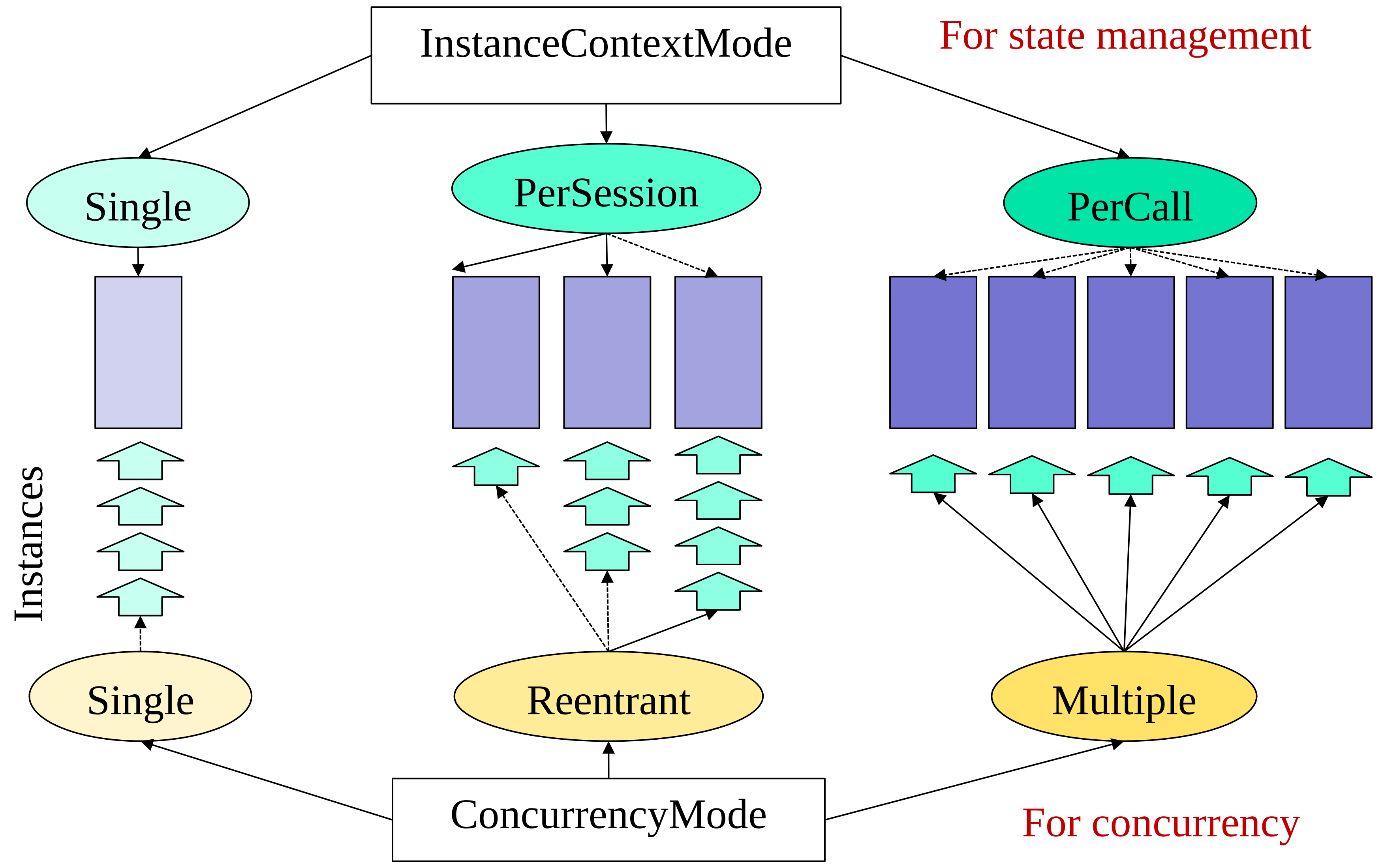
Lock, because multiple pages can access the same resource.

Program Example: Multiple Threads

```
// <-- Must guarantee that cache is valid here because
        // the operation releases the lock.
      }
      string webPage = slow.GetWebPage(address);
      lock (this.ThisLock)
      {
        // <-- Can assume cache is valid.
        // <-- Cache is no longer valid because the operation
        // changes one of the values.
        this.cachedAddress = address;
        this.cachedWebPage = webPage;
        // <-- Cache is valid again here.
        // <-- Must guarantee that cache is valid because
        // the operation releases the lock.
      }
      return webPage;
    }
  }
```

If cache is not valid, get the page, and save into cache.

InstanceContextMode vs. ConcurrencyMode



Finer Performance Management

Can we dynamically decide the values?

- Single
- Reentrant
- Multiple
- Automatically switching between the values

Cloud Computing Load balancer

Summary of the Lecture

- Models of Distributed Computing
- Communication Channels
  - One-way
  - Request-reply
  - Duplex
- Bindings
  - For WSDL services
  - For RESTful services
  - For .Net remoting
- Behaviors and service behaviors
  - Instancing for state management
  - Concurrency for multithreading
  - Cloud computing: automated management based on performance requirement