CS355 Web Technologies

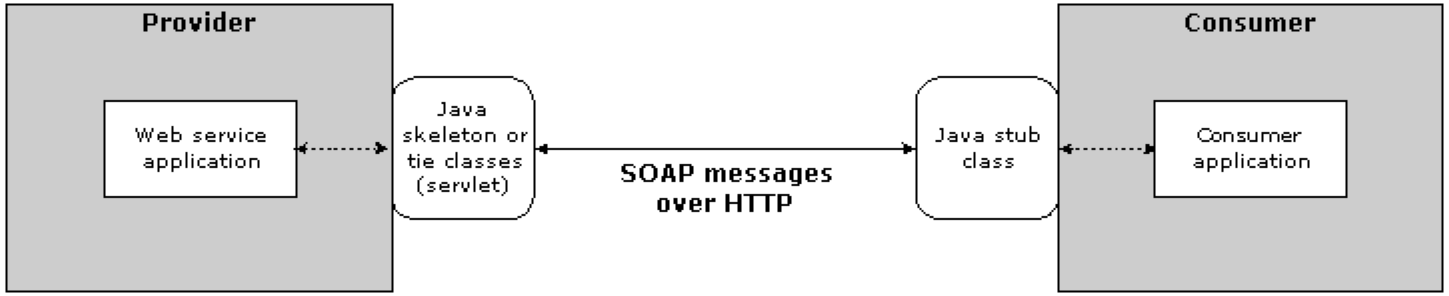
Dr. Ismail Hababeh

German-Jordanian University

Lecture 26

Web Service Design and Development

- Top-Down design



  - Step 1: starts up with a Java interface file or WSDL XML file to generate web service implementation skeleton and stub without existing web service implementation.

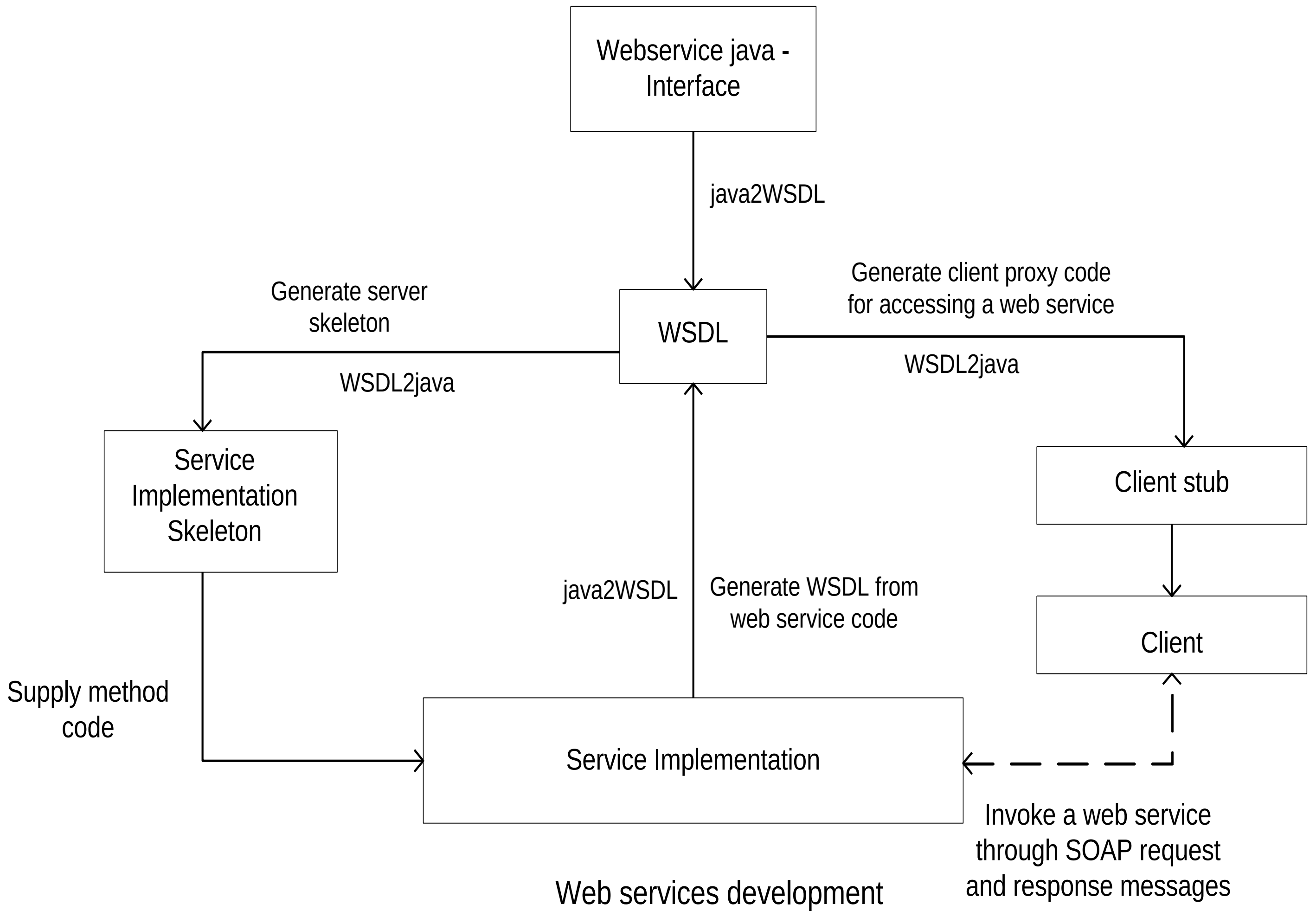
Stub: Java classes that handle SOAP message communication between a consumer application and a Web Service.

Web Service Design and Development

- Step 2: develop (implement) the web service on server side and deploy the web service on the server.
- Step 3: develop the client stub on the client side based on the web service stub to allow access the deployed web service on the server.

- Note: If we start a web service design with a new WSDL, which is called “green field” design, a Java interface will be generated from the WSDL and the rest of development procedure will be same .

Web Service Design and Development


Webservice java - Interface
java2WSDL
WSDL
Generate server skeleton
WSDL2java
Generate client proxy code for accessing a web service
WSDL2java
Service Implementation Skeleton
Client stub
Client
java2WSDL
Generate WSDL from web service code
Supply method code
Service Implementation
Invoke a web service through SOAP request and response messages
Web services development

The Connection Model of Web Service

- Interactions between Web Services
  - Web service invocation is either synchronous or asynchronous.
  - The synchronous interaction is default interaction in Web service operations.
  - In Synchronous interaction, a client sends a Web service request and stops his operations while waiting for a response.
  - In Asynchronous interaction, a client sends a Web service request and continue his operations while waiting for a response.
  - An Asynchronous interaction should be used If the service takes large time to complete or if there is no backward channel available for the response to come back synchronously such as e-mail response to a HTTP request.

Synchronous Vs. Asynchronous Interactions

- There are 4 portType of operations in WSDL interface definitions for WSDL interaction (in, out, in/out, out/in) and can be divided into two groups:
  A. Asynchronous message-based operations (one-way interaction)
  - Asynchronous message-based operation is a single message passing operation, which can be one-way outgoing (request) notification and one-way incoming (response) notification.
  - The message may be a signal, a notification, or data to be processed by the target of the message.

Synchronous Vs. Asynchronous Interactions

B. Synchronous RPC-based operations (two-way interaction)

– The two-way request/response operation is a RPC-based operation with combination of incoming and outgoing message.

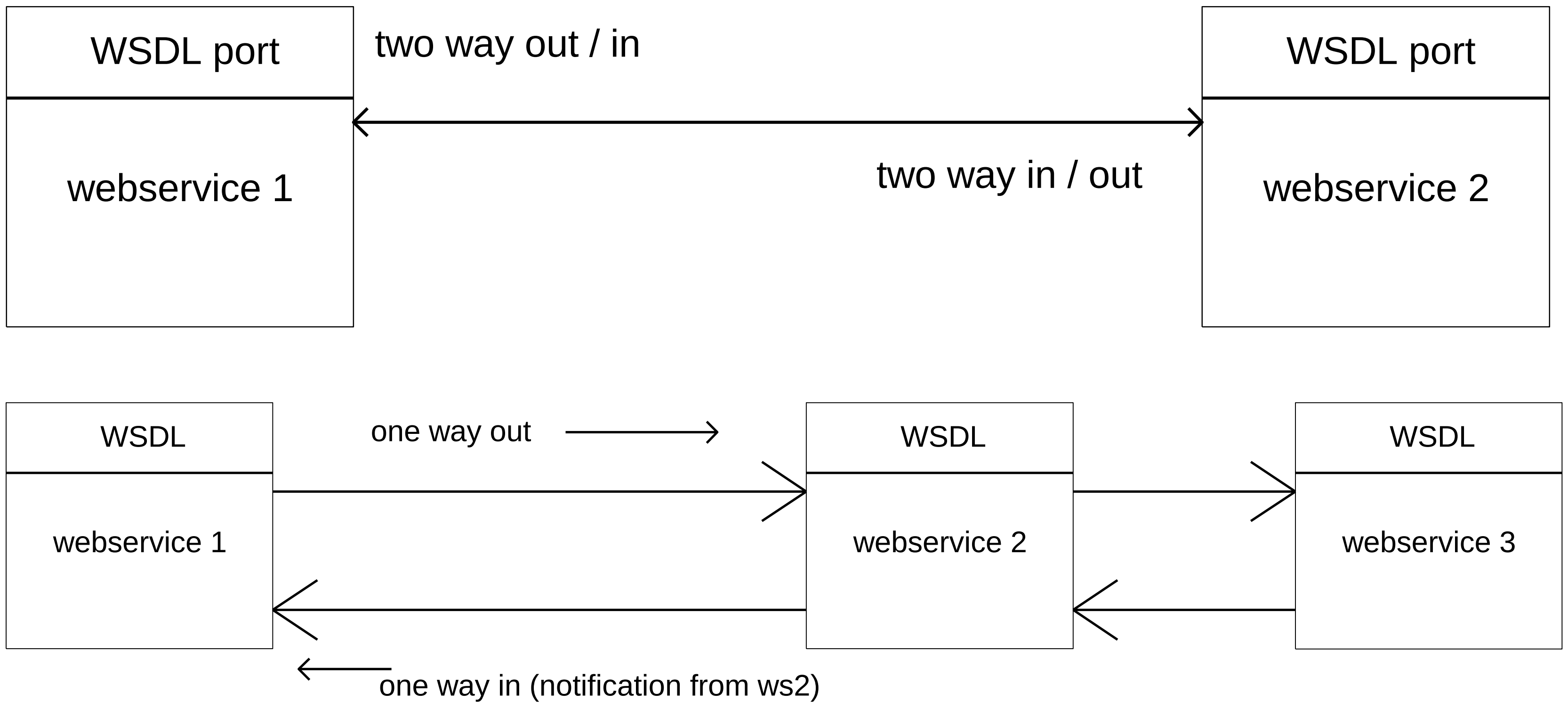
- The source WSDL sends out an outgoing SOAP (request) message and expect to get an incoming (response) SOAP message right away in the out/in mode.

- The target WSDL gets incoming (request) message and responds an outgoing (response) message in the in/out mode. Obviously, they are synchronous operations.

Synchronous Vs. Asynchronous Interactions

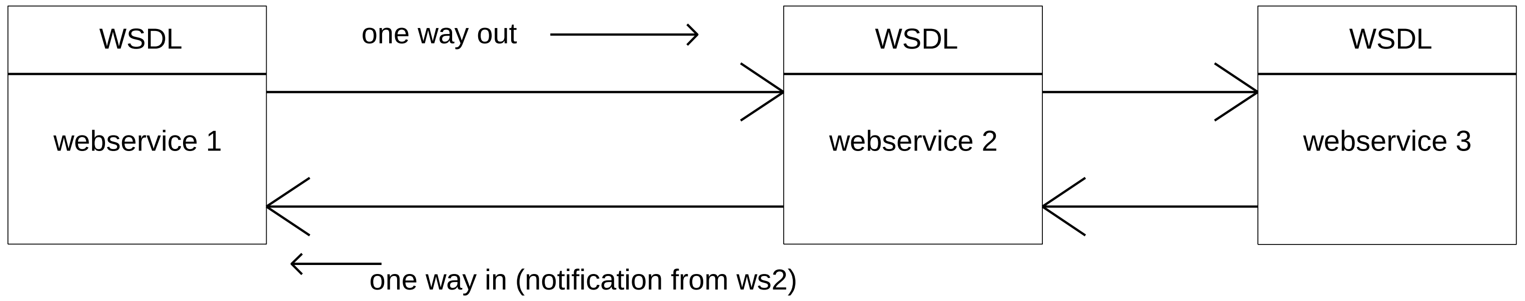
- The basic web service connection is established between two endpoints of Web services.
- The source webservice1 has an outgoing message port which connects the same type of incoming message port of target webservice2.
- There are (2-way channels) between webservice1 and webservice2. The connection model between source and target services is a two separate unrelated one-time channels (not permanent).

Synchronous Vs. Asynchronous Interactions



Synchronous Vs. Asynchronous Interactions

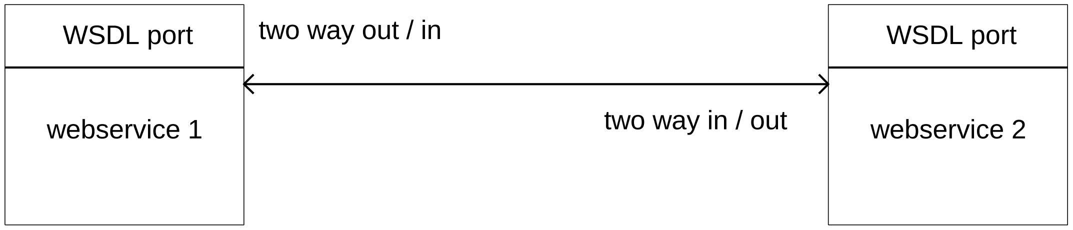
- The webservice2 may process data according to the message got from webservice1 or process the message itself and then forward it to third Web service.



- This message-based operation is typical asynchronous communication style that a client of such web service does not need to wait any reply back from the web service it requests and just simply continue its own control flow.

Synchronous Vs. Asynchronous Interactions

- RPC-based request/response message exchange takes place between two endpoints of web services where webservice1 has an out/in port and webservice2 has an in/out port.

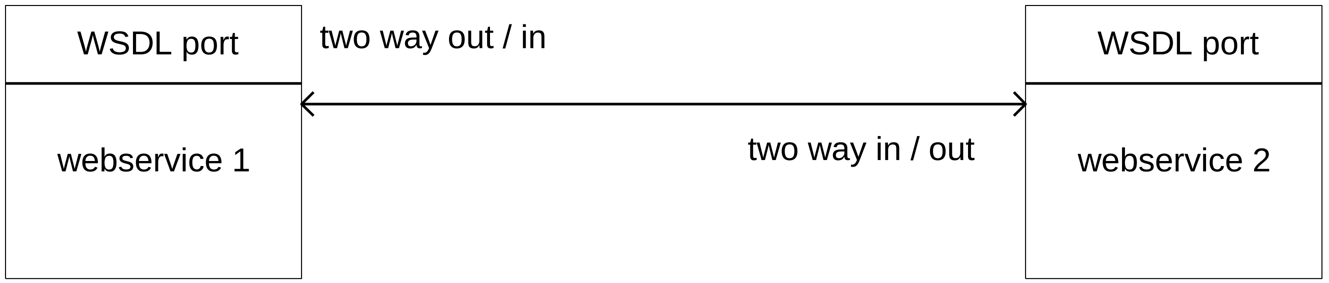


- webservice1 can send a SOAP request to the in-port of webservice2 and webservice2 responds a SOAP message via its out-port to the in-port of webservice1.

- The sequence of incoming message before outgoing messages in the definition of portType in WSDL specifies an in/out portType operation which is a service provider operation.

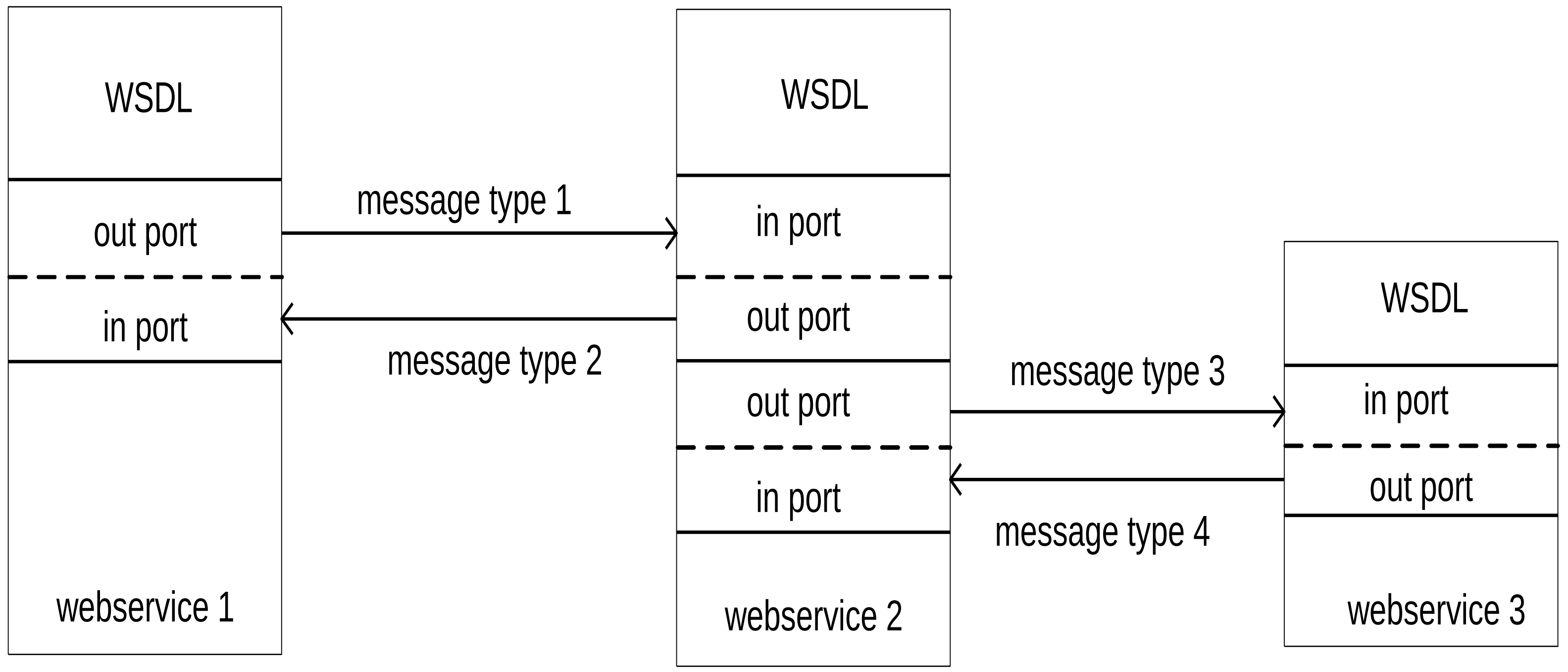
Synchronous Vs. Asynchronous Interactions

- The sequence of outgoing message before incoming messages in the definition of portType in WSDL specifies an out/in portType operation which is a client service request operation.



- RPC-based style service is a typical synchronous communication style service that a client must wait till the result of invoked method of Web service is received.

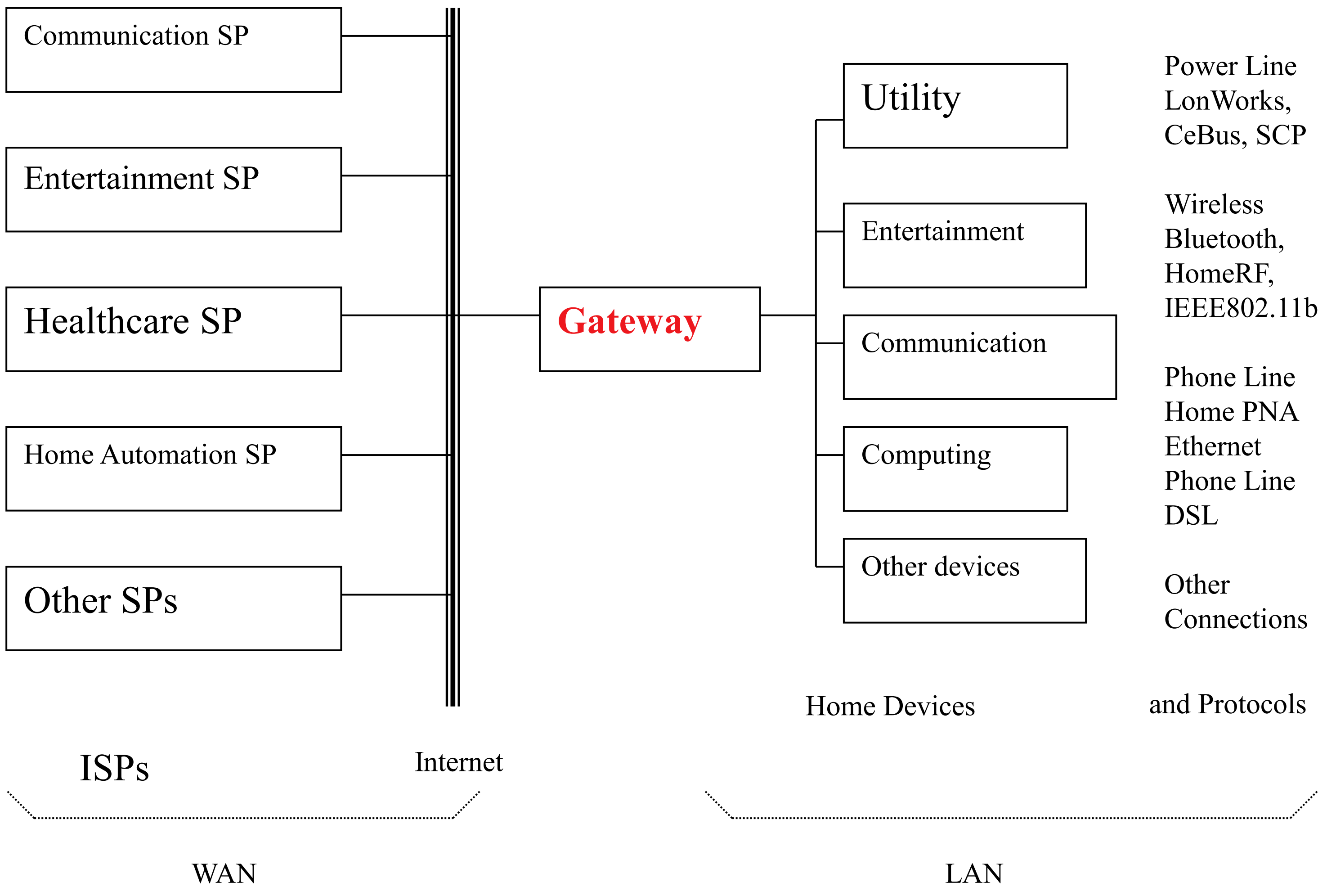
Synchronous Vs. Asynchronous Interactions



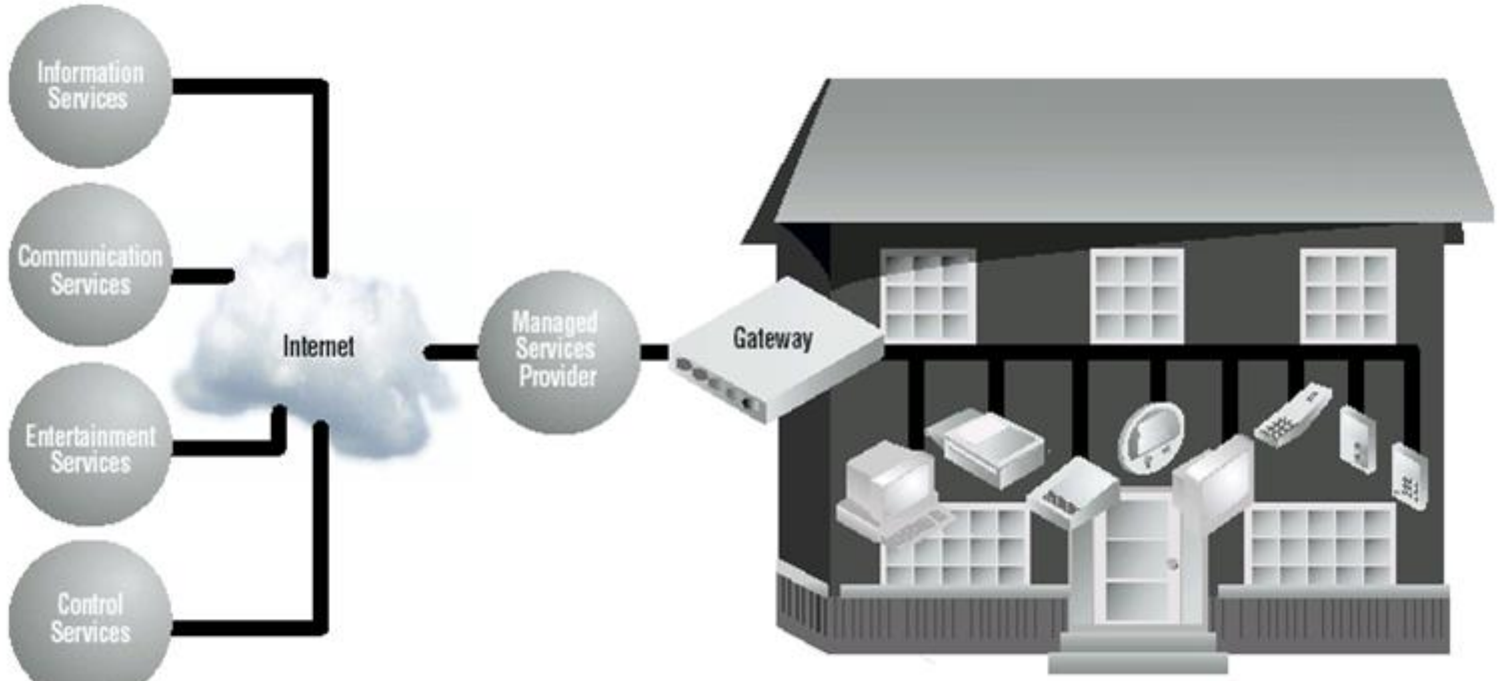
OSGi Concepts

- Open Service Gateway initiative (OSGi) provides dynamic functionality to Java that makes Java the premier environment for software integration and thus for development.
- Delivery of multiple services over wide-area networks to local networks and devices.
- The focus of OSGi is on services

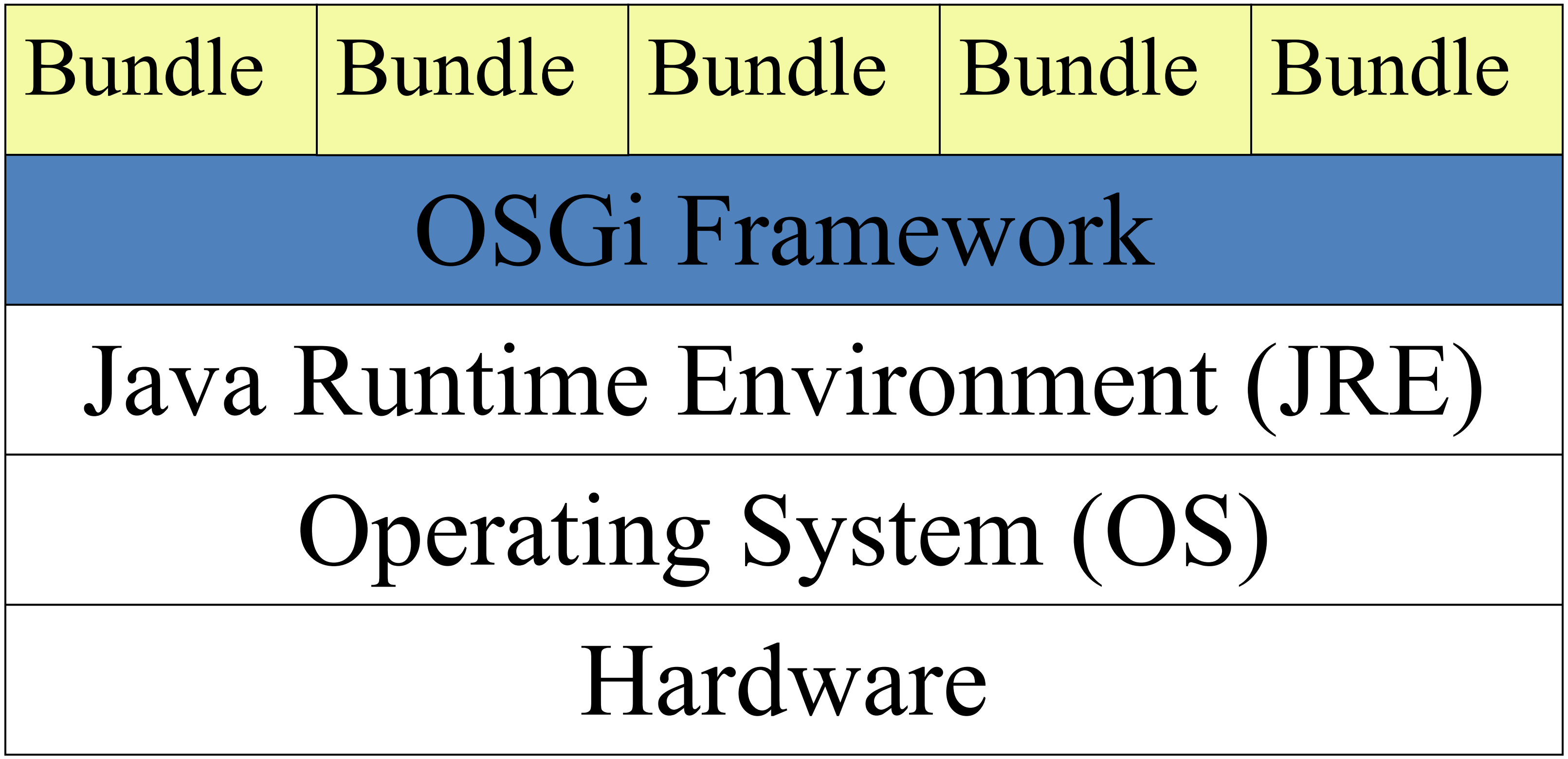
OSGi Service Oriented Computing



Internet of Things Based on OSGi



OSGi Architecture



Java Runtime Environment

- JRE is a software layer that runs on top of a computer’s operating system and provides the class libraries and other resources that a specific Java program needs to run.
- JRE is the runtime portion of Java software, which is all you need to run it in your Web browser.
- The JRE consists of the Java Virtual Machine (JVM), Java platform core classes, and supporting Java platform libraries.

OSGi Bundles

- A bundle is a JAR file that contains class files and resources such as images, HTML pages, and other data files.

- Bundle = interface + implementation + resources

  Interface: what the service does

  Implementation: how to perform the service

  Resources: images, HTML pages, and other data files

- Services are delivered as bundles

Bundles Development Model - Example

- Resources

```
META-INF/MANIFEST.MF

player/service/DigitalPlayer.class

player/impl/mp3/Activator.class

player/impl/mp3/MP3Player.class

player/skin/regular.html

player/media/mp3/sample.mp3
```

- Interface

```
//imports
package player.service;
public interface DigitalPlayer
{
 public void play( InputStram musicStream);
}
```

- Implementation

```
public class MP3Player implements
      DigitalPlayer
{  public void play (InputStream inStream) {
     … …}
}
```

Bundle Manifest file - Example

```
Bundle-Name: clock
Bundle-Description: The Clock service
Bundle-Vendor: Sun Microsystems, Inc.
Bundle-Version: 1.0
Bundle-DocURL: http://java.sun.com/products/embeddedserver
Bundle-ContactAddress: jes-comments@sun.com
Bundle-Activator: clock.ClockServiceImpl
Import-Package: org.osgi.service.http, javax.servlet, javax.servlet.http
Export-Package: clock
Export-Service: clock.ClockService
```

Bundles Connections Model

- Bundles are protected from each other : Code within one bundle cannot refer to classes inside another bundle, nor instantiate them or invoke their methods
- Bundles connect to each other through the framework and its Export/Import – Package declarations in its manifest file

OSGi Framework

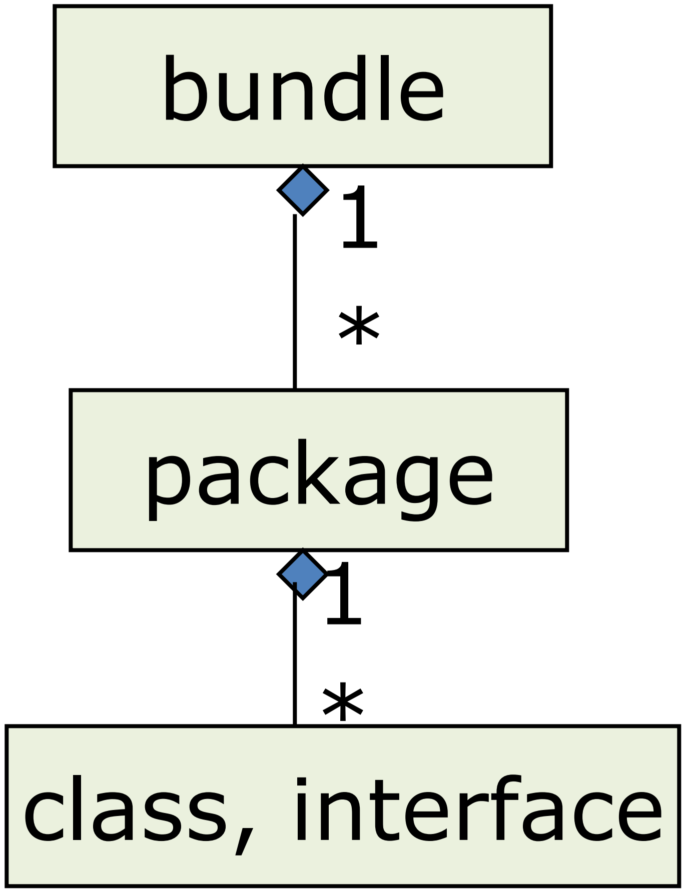
- OSGi framework provides a hosting environment for bundles. The services provided by OSGi are:
  - Managing the life cycle of bundles
  - Resolving interdependencies among bundles and making classes and resources available from a bundle
  - Maintaining a registry of services
  - Firing events and notifying listeners when:
    - A bundle's state changes
    - A service is registered/unregistered
    - The framework is launched/raises an error

Building Applications with Bundles

- Bundles are the building blocks of applications.
- Common operations are factored out and programmed into a set of basic modules
- More sophisticated and high-level bundles rely on the basic modules to get their work done.
- Import, export packages, and register services

Bundle packages and classes

- Individual classes or interfaces cannot exported/imported.
- The package is the smallest unit for exported/imported.
- There is no peer-to-peer export/import
- Connections are made through the framework



Register and Obtain Services

- If a bundle contains some services, it usually registers (publishes) the services with the service registry maintained by the framework when the bundle is created.
- Once a service is registered, another bundle can look it up and consume the service.

Handling the Dynamic Service Dependency

- *What will happen if bundle A depends on bundle B and B unregisters its service at run time?*
- The framework will broadcast an event of B’s intention, and A needs to do the proper cleanup to ensure it no longer calls B’s service.
- B’s un-registration is not completed until all interested parties have been notified and they have finished any action in response to the pending event of B being unregistered.

Stopping and Uninstalling a Bundle

- The framework calls the bundle activator’s stop method with the following tasks:
  1. Unregister the service provided by the bundle. During this process, an event is broadcast to notify interested listeners that the service is being unregistered
  2. Release any services in use by the bundle
  3. Remove any event listeners added by the bundle
  4. Move the bundle back to the *resolved state*
  5. Broadcast another event to notify listeners that the bundle has been stopped