

Dynamic Web Service Deployment Using WSPeer

Andrew Harrison

Cardiff University, Wales, UK

a.b.harrison@cs.cf.ac.uk

What is WSPeer?

- An application level API written in Java for exposing applications as Web services.
- Usual Web service deployments use a container model for handling the service.
- WSPeer does not. Instead an application is notified of service related events fired by WSPeer.

The Container Model

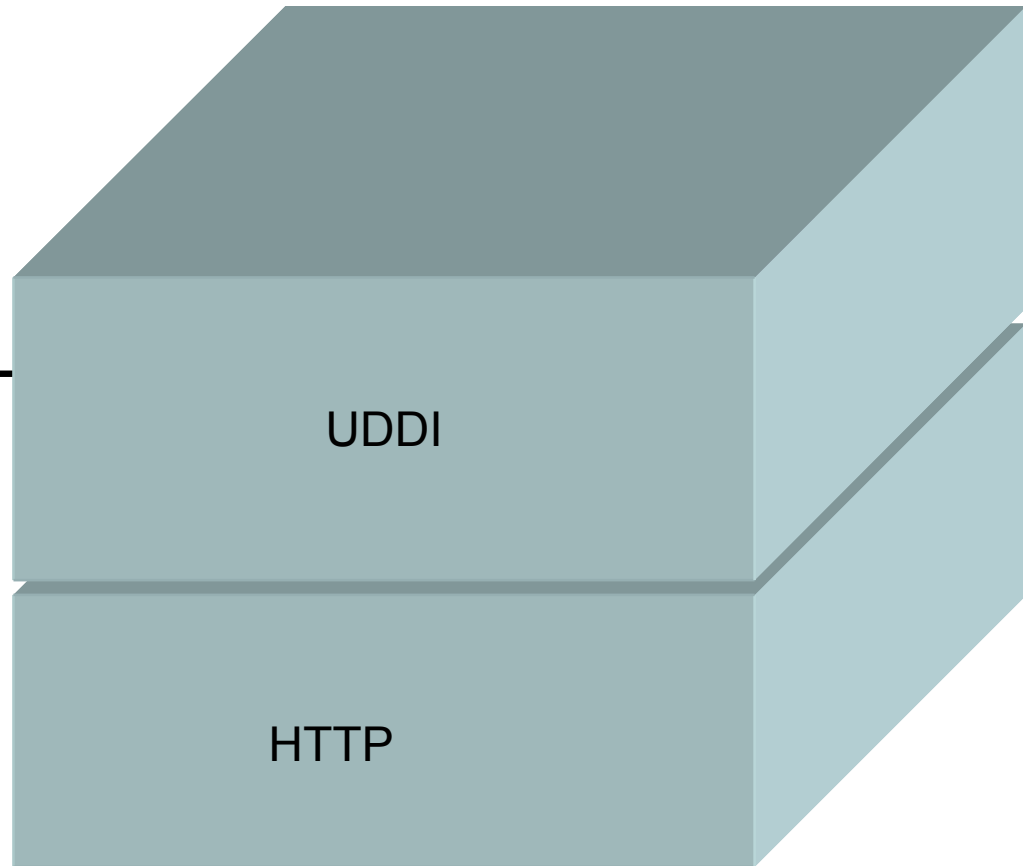
- The container model establishes an environment into which software components are deployed as services.
- The container manages the life-cycle and requests for the services.
- This model is used in client/server scenarios.
 - many services managed via one endpoint.

WSPeer

- Because of the presumption of a client/server topology, Web services have inherited the trappings of this approach:
 - HTTP for transport
 - UDDI for discovery
- From a user's point of view a Web service is just something that can process XML.

The Web Service Russian Doll

The usual
service... and
transport
layers... An XML
processor.



What Kind of XML is Processed?

Two Main XML Specifications – SOAP and WSDL

- WSDL for service description
 - enables arbitrary transports
- SOAP for message exchange
 - allows composition of protocols Via WS-
* specifications and attachments

What is a Web Service?

- Service definitions describe what an entity *can do* not what an entity *is*.
- Messages use a shared language. That language can be shared by anything that can speak the language.
- A Web service is anything that can communicate using Web service based XML documents.
- Viewing Web services as XML message processors allows us to use them more flexibly.

...What is a Web Service?

Web service does not *have* to be persistent.

Web service does not *have* to have a stable address.

Web service *can* operate across various transport protocols.

...What is a Web Service?

- Removing the container framework allows services to be deployed while remaining in the application environment that deployed them.
 - Allows services to respond more dynamically to the requirements of the application.
 - Allows current protocols to be used in a less client/server centric manner.
 - Allows new discovery and transport protocols.

WSPeer

- Acts as a complete interface to both hosting and invoking Web services.
- Can be used in 'standard' client/server architectures using technologies such as UDDI and HTTP as well as other architectures such as P2P networks.

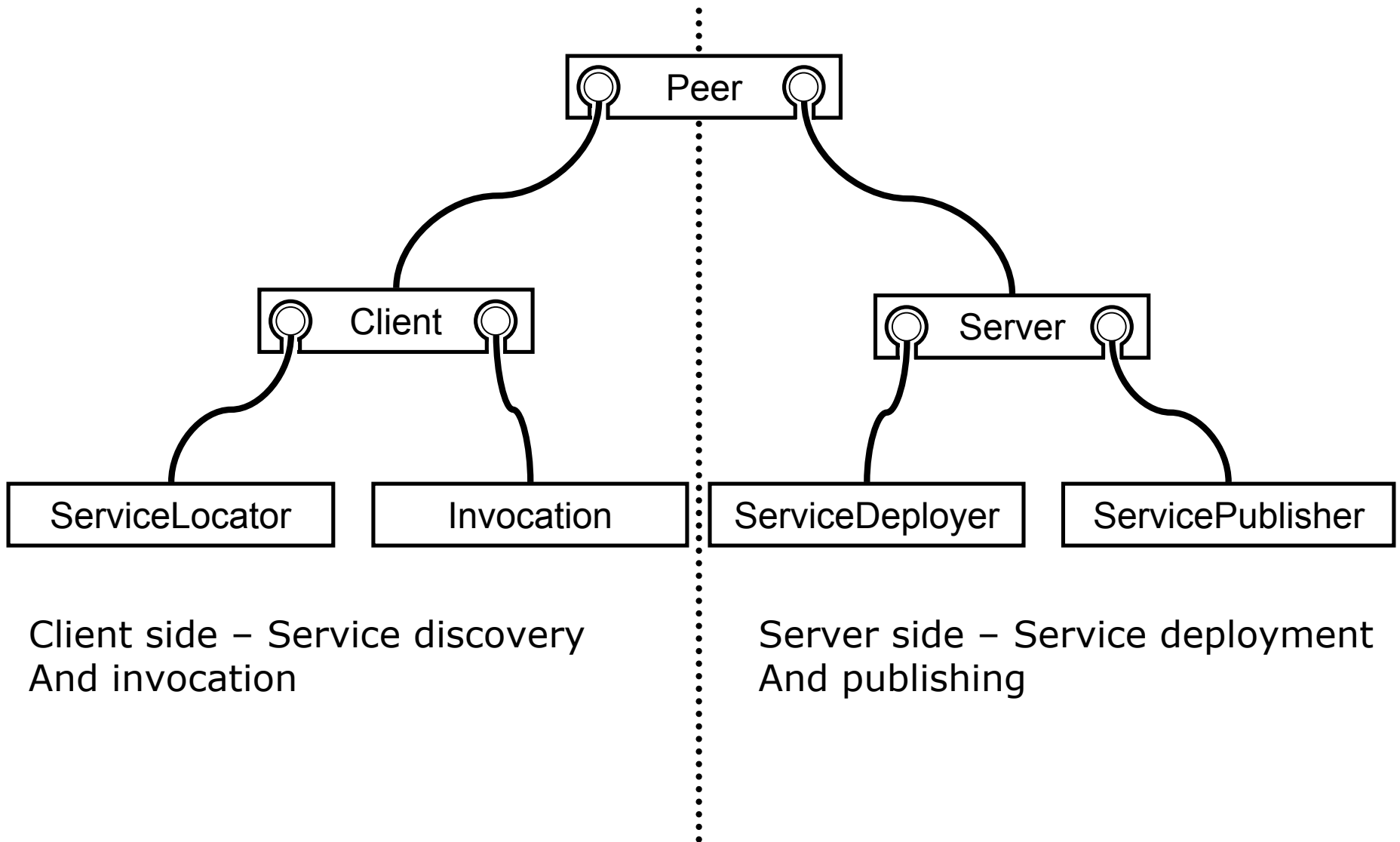
WSPeer

- Evolved out of an interest in combining the strengths of Web services with the strengths of P2P systems.
 - The Web service XML messaging standards are sophisticated, extensible and promote interoperability. They allow service description and data typing.
 - P2P discovery and resource distribution are more scalable and fault tolerant.

WSPeer

- Currently two implementations:
 - A 'standard' implementation using HTTP and UDDI.
 - An experimental P2P implementation using P2PS (Peer-to-Peer Simplified).

WSPeer Structure



PeerMessageListener Interface

Each interface in the WSPeer tree fires events. These events are propagated up to the **Peer** interface. An application implementing the **PeerMessageListener** interface is notified of these events.

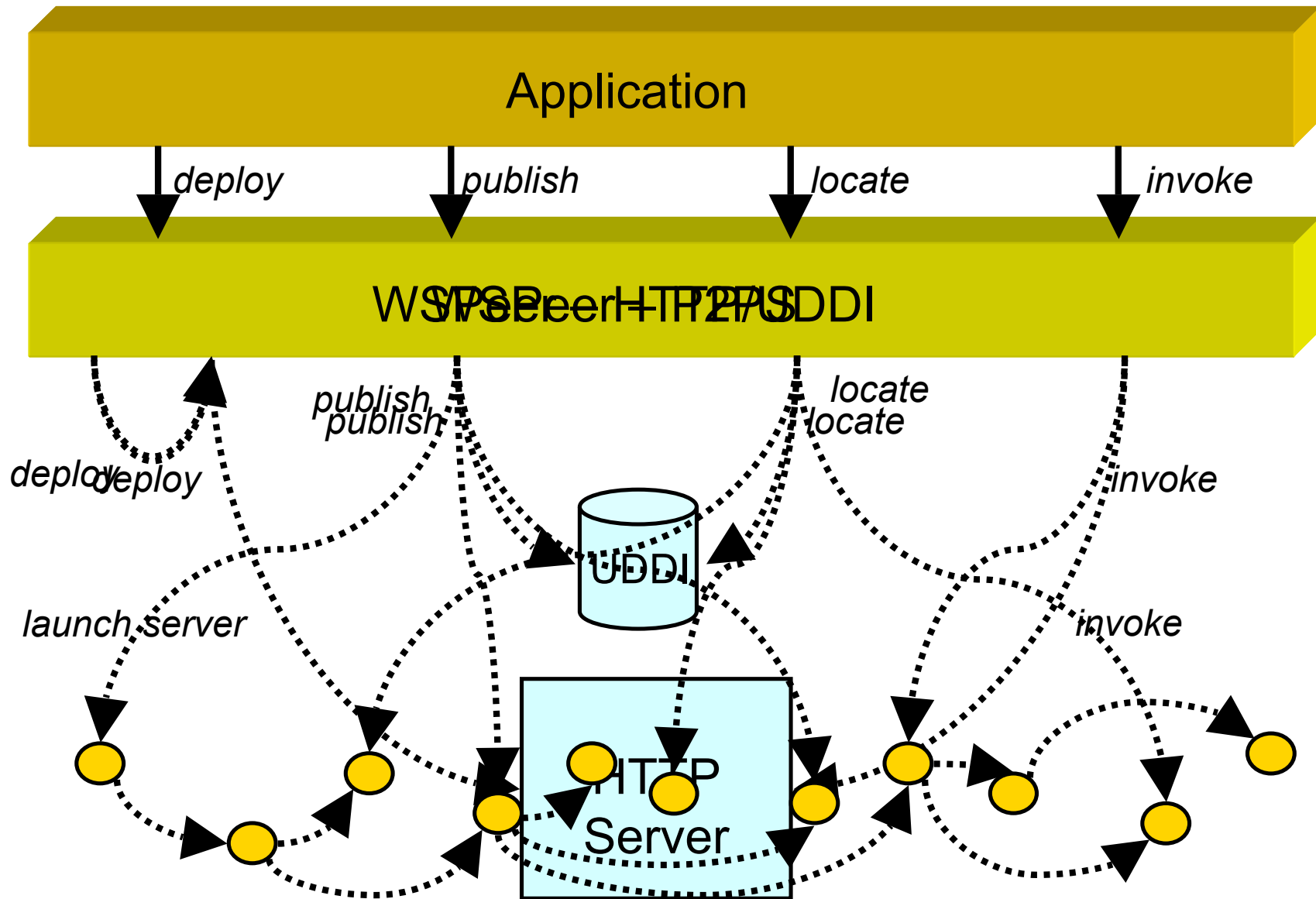
```
public interface PeerMessageListener {  
    public void peerMessageReceived(DeploymentMessageEvent evt);  
    public void peerMessageReceived(PublishMessageEvent evt);  
    public void peerMessageReceived(DiscoveryMessageEvent evt);  
    public void peerMessageReceived(ClientMessageEvent evt);  
    public void peerMessageReceived(ServerMessageEvent evt);  
}
```

This listener is attached to the **Peer** interface at the root of the WSPeer interface tree.

Protecting Applications

- The events and all data structures received by the application are WSPeer abstractions.
- This protects the application from changes in the network topology as well as developments in/changes to standards, for example, Service definition.

...Protecting Applications

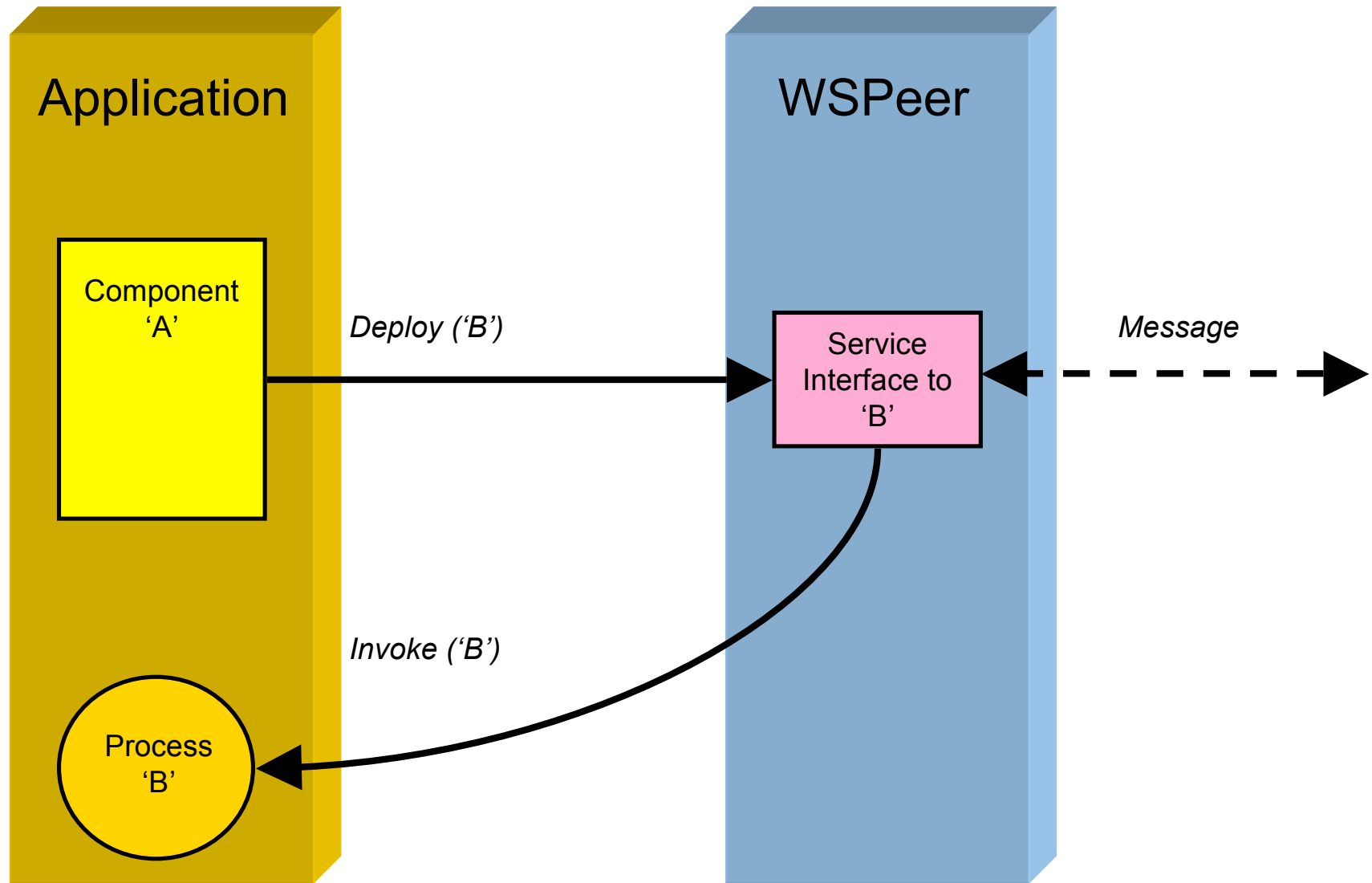


WSPeer Services

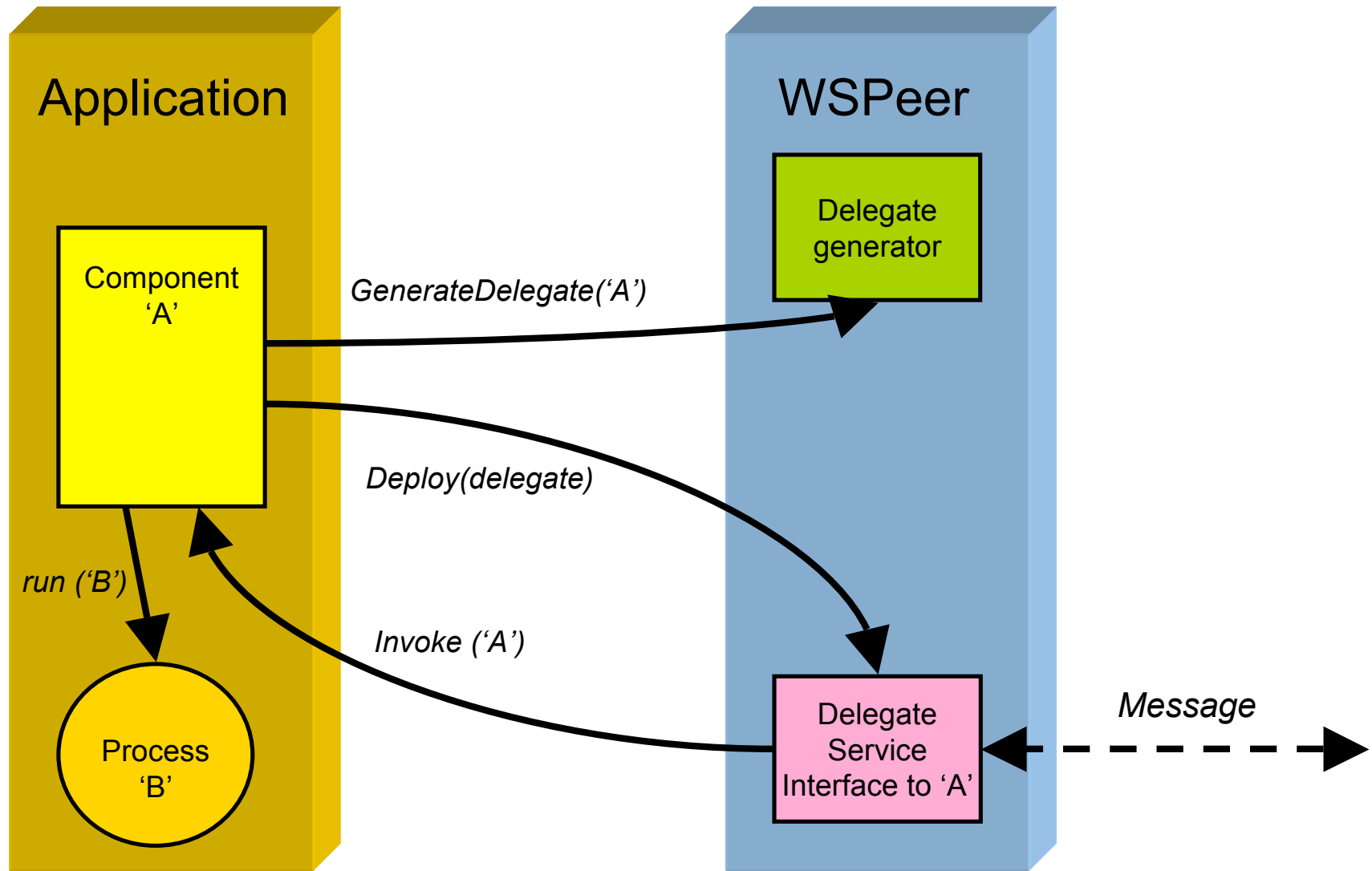
Two types of service deployment:

- '*Direct*' – expose an aspect of an application directly as a Web service. Here the state of the service lasts for the duration of the invocation.
- '*Delegated*' – Generate a proxy that calls back to an object in memory in the application. Here, the service can be perceived to have the state of the object.

Direct Deployment



Delegated Deployment



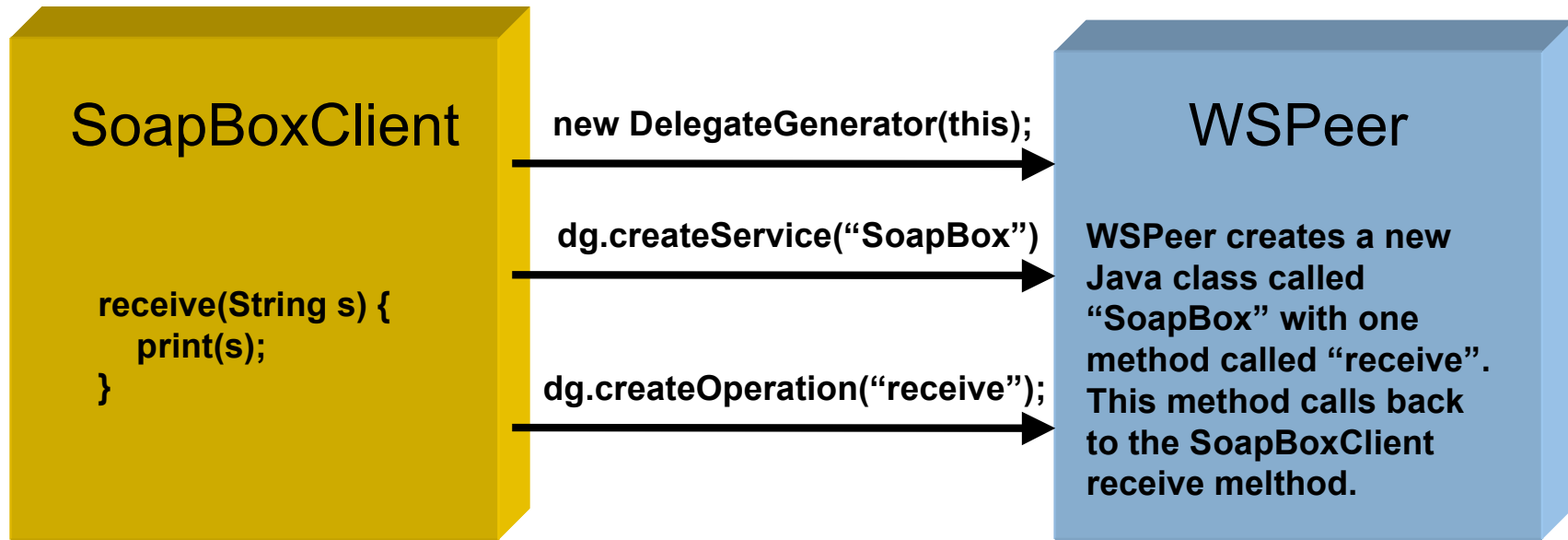
An Example - The "SoapBox" Service

- A Simple Chat service using Web services.
- Employs delegated deployment to maintain state across invocations.
- The `SoapBoxClient` class implements the `PeerMessageListener` interface, allowing it to receive messages from the `WSPeer` system.

SoapBox Using HTTP/UDDI

- The `SoapBoxClient` class has a method:
`void receive (String message) ;`
that prints out the input string.
- It creates a delegate service called "SoapBox" with one operation:
`void receive (String message) ;`

...SoapBoxClient



The Delegate Service

The delegate service method simply echoes that of the SoapBoxClient:

```
public void receive(java.lang.String in0) throws RemoteException {
```

Get the instance of the class that has been stored in a table:

```
    SoapBoxClient inst = null;
    try {
        inst = (SoapBoxClient)InstanceTable.
            getInstanceTable().get("1103048238871-36807");
    } catch(ClassCastException e) {
        throw new RemoteException("Object is of wrong type.");
    }
    if(inst == null) {
        throw new RemoteException("Object cannot be found.");
    }
}
```

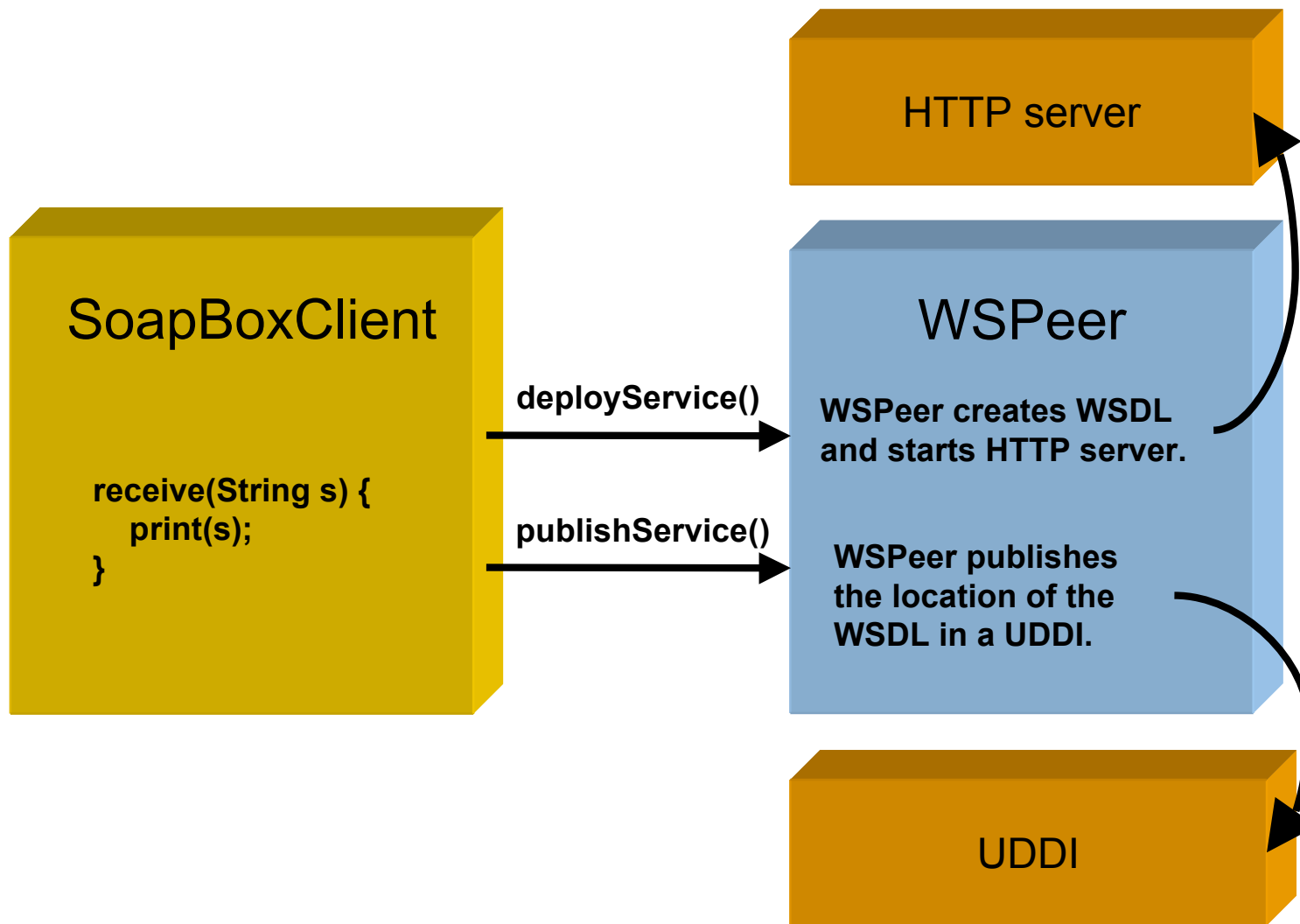
Call the "receive" method of the SoapBoxClient instance:

```
    inst.receive(in0);
}
```

...SoapBox Using HTTP/UDDI

- The `SoapBoxClient` deploys the service. `WSPeer` creates the WSDL from the delegate service class and kicks off an HTTP server.
- The `SoapBoxClient` publishes the WSDL in a UDDI registry.

...SoapBox Using HTTP/UDDI



...SoapBox Using HTTP/UDDI

- Then it tries to discover other services by the same name – “SoapBox” – in the UDDI registry.
- If a remote chat service is found, the application invokes its
`receive(String message);`
passing it a known introductory message.
- This message prompts the remote service to undertake a search for this service.

...SoapBox Using HTTP/UDDI

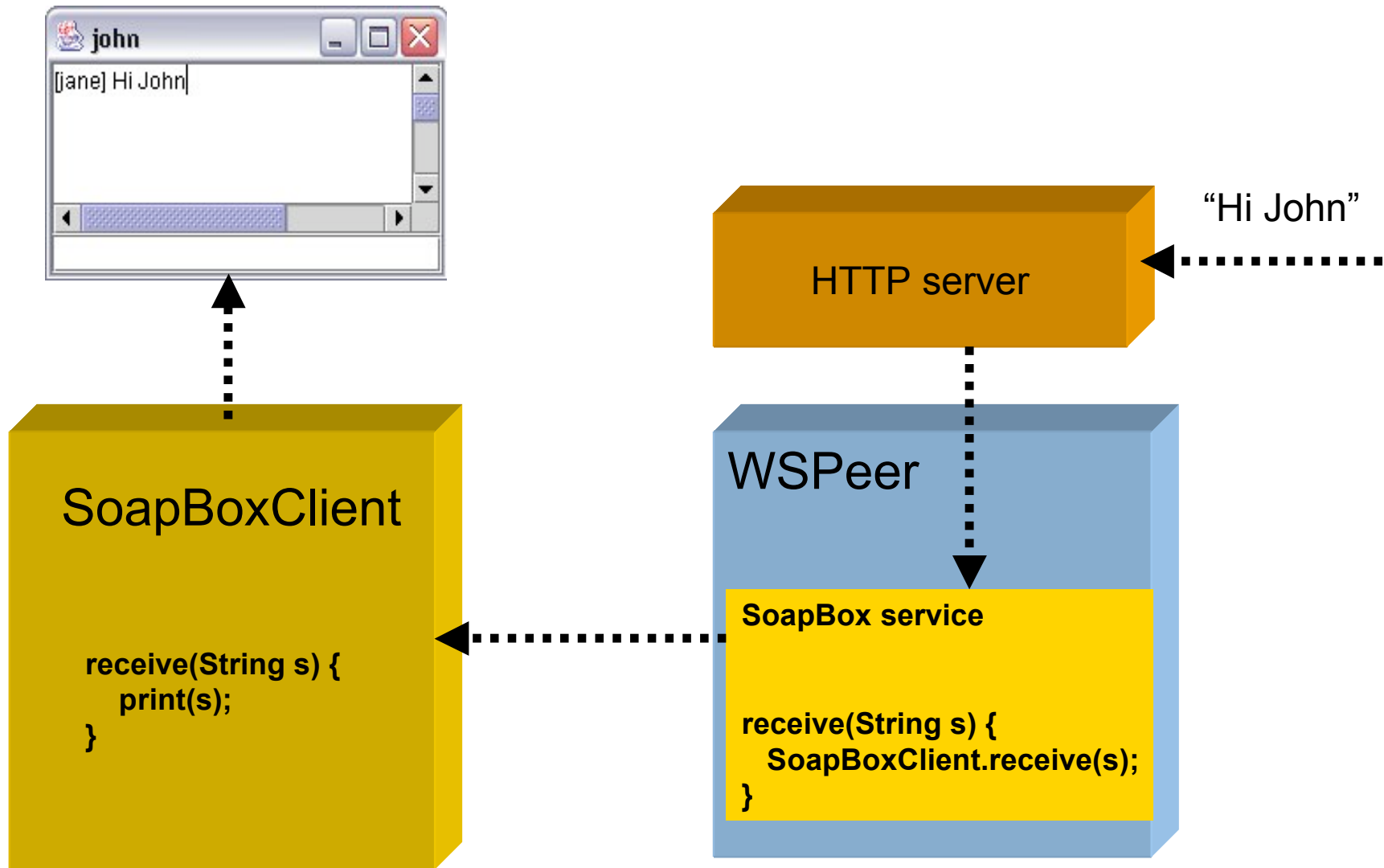
- Once they have both discovered each other they can invoke one-another's

`receive (String message) ;`

operations.

- The actual services that are interacting live only for the duration of a single invocation to this method. The `SoapBoxClient` class maintains the state, i.e. the conversation.

...SoapBoxClient



Summery

- WSPeer does not use a container.
- Instead control of environment, life-cycle, state and invocation is left with the application.
- This allows services to be deployed that can respond to application needs in a more dynamic way.

Thanks!

More information from:

<http://www.wspeer.org>