

# Web Services Technologies

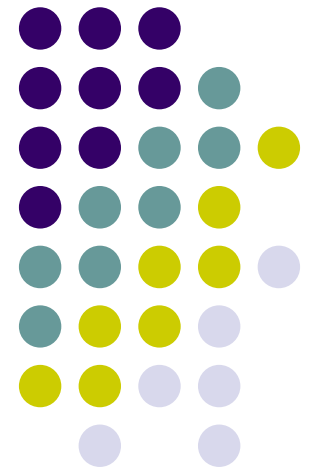
---

## SOAP vs. Jini

Nadir Weibel and Rudi Belotti

June 19th, 2002

ETH Zürich

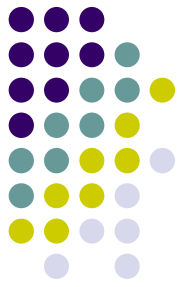


# Goals

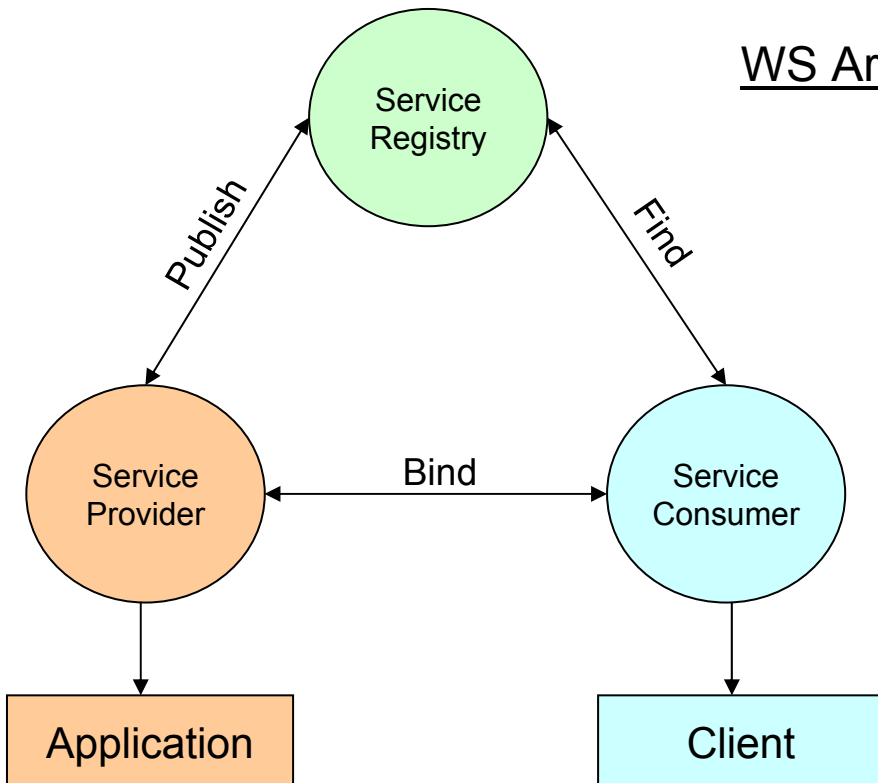


- Design a web service infrastructure with [SOAP/WSDL/UDDI](#) (developed by W3C) and with [Jini](#) (a Sun Java product)
- Implement some web services with both Jini and SOAP
- Compare the 2 technologies:
  - Service Discover Time
  - Service Access Time
  - Scalability
  - Network Congestion
  - Serialization Overhead
  - Code Complexity

# Web Services?

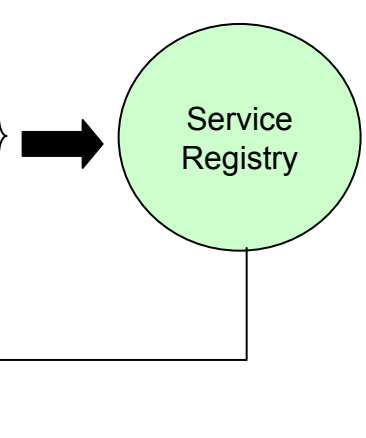


- Software interfaces between a **client** (invokes the service) and an **application** (provides the functionality)



## WS Architecture:

- Publishing
- Discovering
- Binding

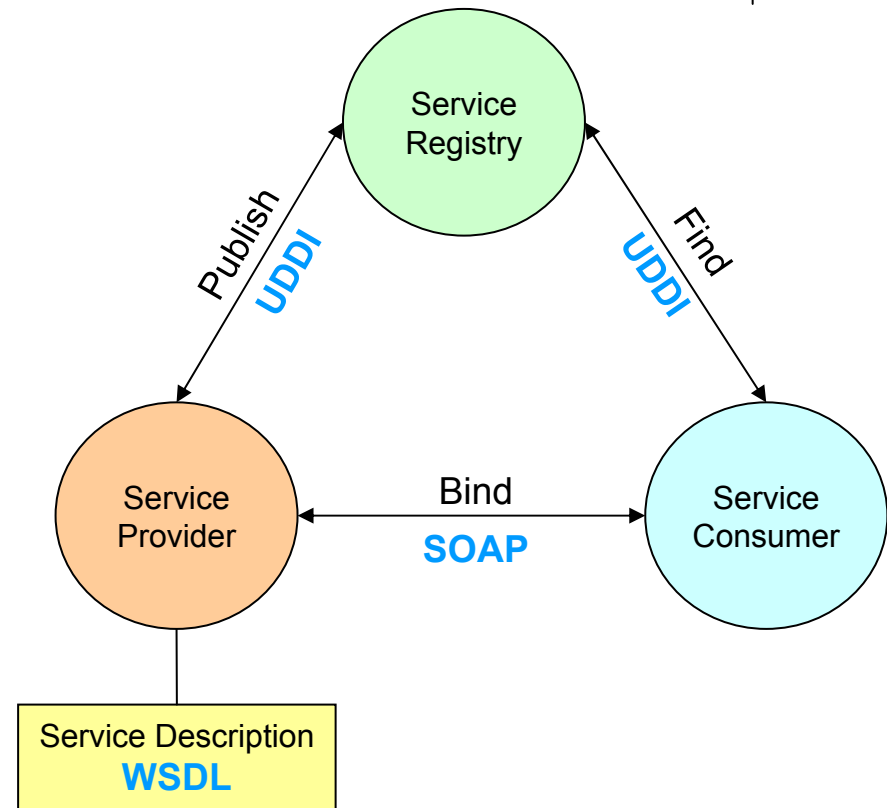


- Maintains a record of the available services
- Allows the client to locate the required services
- Provides a way to communicate between clients and services

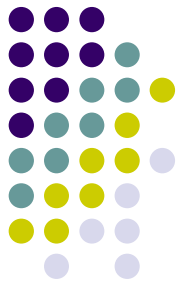
# SOAP and Web Services



- 3 main technologies that build the infrastructure:
  - **WSDL**: Web Service Definition Language
  - **UDDI**: Universal Discovery, Description and Integration
  - **SOAP**: Simple Object Access Protocol
- All 3 technologies are based on **XML**, the eXtensible Markup Language

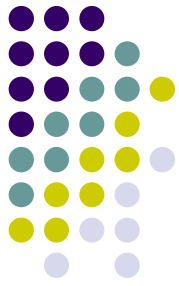


# WSDL: Web Service Definition Language



- The Service Consumer needs to know
  - what kind of message to send
  - what kind of messaging protocol to use
  - what network address to send the message to
- WSDL
  - describes the service interface
  - describes the service implementation
  - allows automatic code generation
  - makes web services self-describing

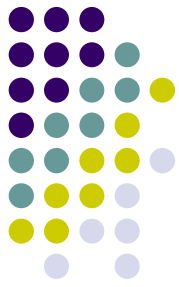
# WSDL example



```
public class CalcService {  
    public double add(double p1, double p2) {  
        return p1 + p2;  
    }  
}
```

```
<?xml version="1.0"?>  
<definitions name="CalcService_Interface" [...]>  
    [...]  
    <message name="addRequest">  
        <part name="p2" type="xsd:double"/>  
        <part name="p1" type="xsd:double"/>  
    </message>  
    <message name="addResponse">  
        <part name="return" type="xsd:double"/>  
    </message>  
    <portType name="SimpleCalculatorService">  
        <operation name="add" parameterOrder="p1 p2">  
            <input message="tns:addRequest"/>  
            <output message="tns:addResponse"/>  
        </operation>  
    </portType>  
    [...]  
</definitions>
```

# UDDI: Universal Discovery, Description and Integration



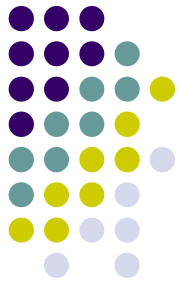
- The UDDI specification
  - defines a way to publish and discover web services
- Concepts:
  - **Business Entity**  
web service provider
  - **Business Service**  
web service within a Business Entity
  - **Binding Template**  
technical description that includes the access point(s) of the web service
  - **TModel**  
description of abstract concepts like the web service's interface

# SOAP: Simple Object Access Protocol



- SOAP
  - protocol that allows communication between applications
  - evolution of XML-RPC
  - based on XML, XML Schema and XML Namespaces
  - supports data-types
  - programming language, platform and hardware neutral
  - simple and flexible

# Status of SOAP Technology

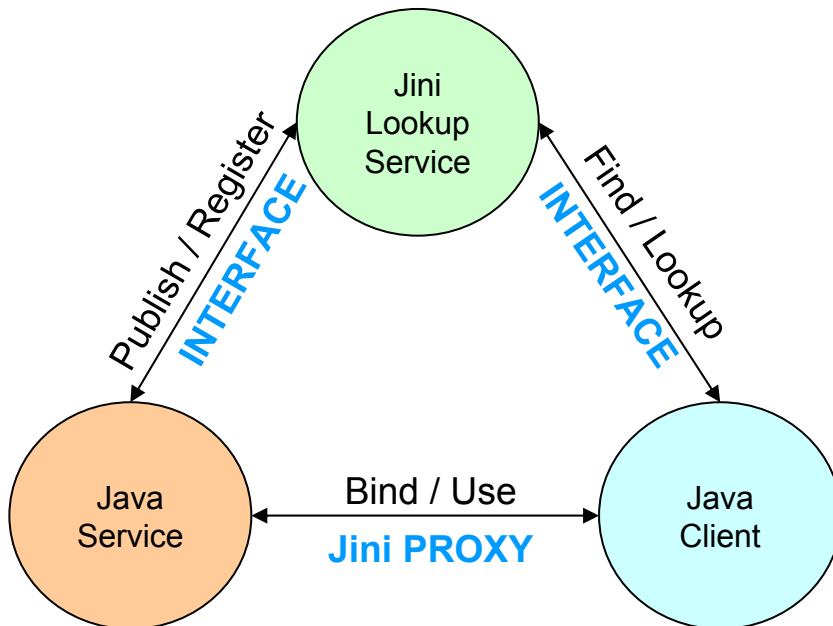


- WSDL
  - Version 1.1
  - W3C Note, March 15, 2001
- UDDI
  - Version 2.0
  - UDDI Open Draft Specification, June 8, 2001
  - Errata 3, February 15, 2002
- SOAP
  - Version 1.2
  - W3C Working Draft December 17, 2001



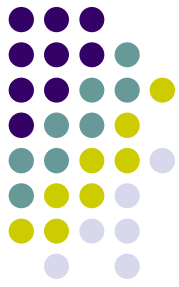
# Jini – An innovative Philosophy

- Jini is a Sun technology built on top of Java **Remote Method Invocation** to provide a distributed computing environment supporting rapid configuration (*plug & play*) and **spontaneous networking**
- Publisher/Subscriber Paradigm implemented using **LEASES**, the **Lookup Service** and Service **Proxies**



Service: • Discover the Lookup Service  
• Register into the Lookup Service

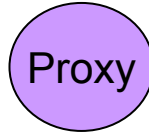
Client: • Discover the Lookup Service  
• Look for the required Service



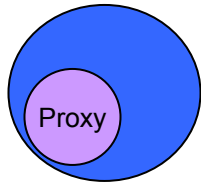
# Jini Service Publishing

1.

Create the **Proxy** to be used by the client implementing the **Service Interface**



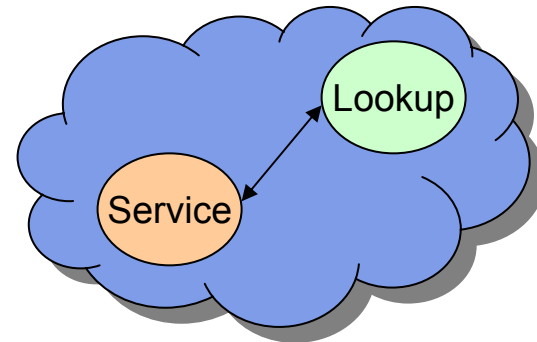
Create a **Service Item** containing information about service (name, ID, Lease, ...) and the proxy



Service Item

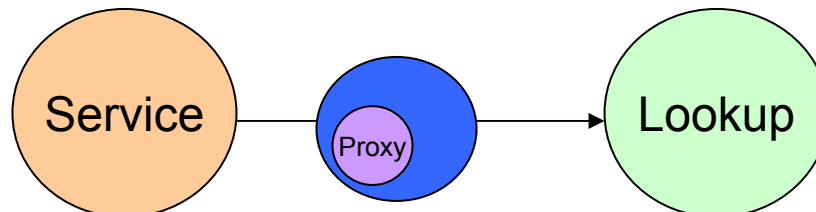
2.

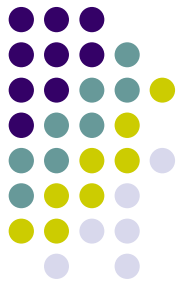
Look in the Subnet to **discover** the Lookup service



3.

**Register** the service Item (with the proxy) within the Lookup Service



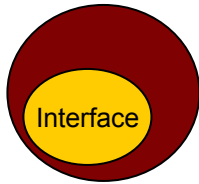


# Jini Service Lookup (Client)

1.

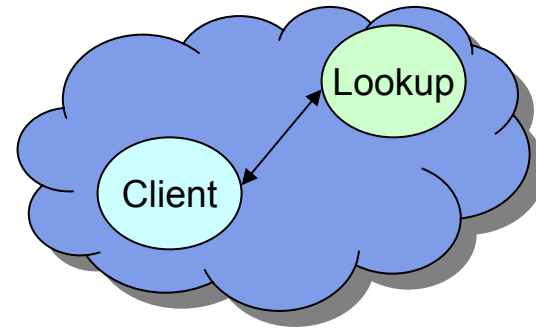
Create a **Service Template** for the required service to be transmitted to the lookup service using the **Service Interface**

Service Template



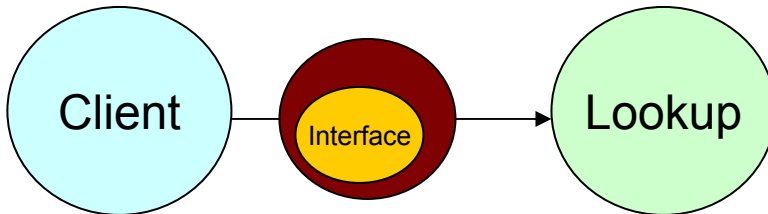
2.

Look in the Subnet to **discover** the Lookup service



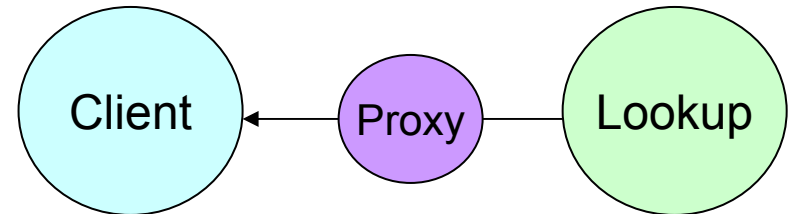
3.

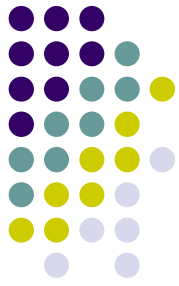
**Lookup** for the required service Using the Service Template



4.

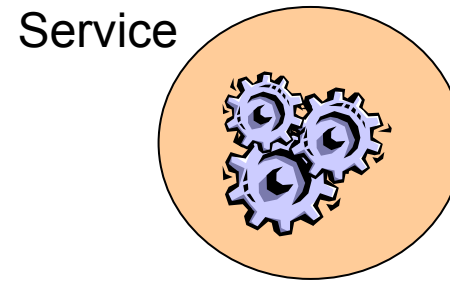
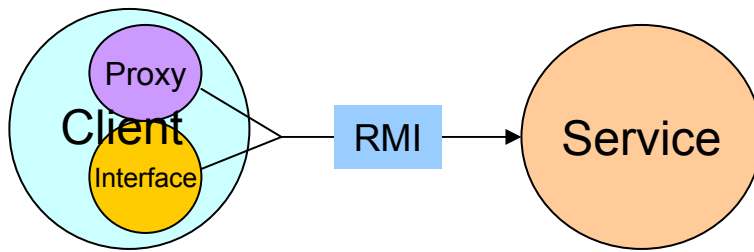
Get the **Proxy** of the required Service from the Lookup Service



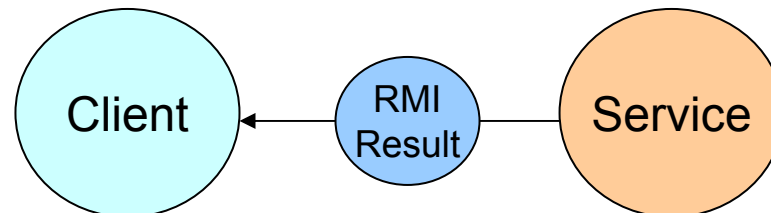


# Jini Binding – Using the service

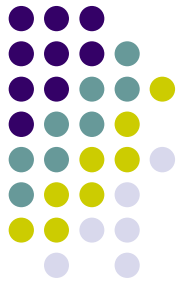
1. Use the proxy to communicate with the Service and invoking it via **RMI** using the local Service Interface
2. The Service executes the Remote Method Invocation and find some result



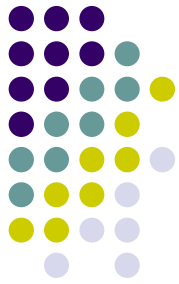
3. Get the result of the Service invocation from the service



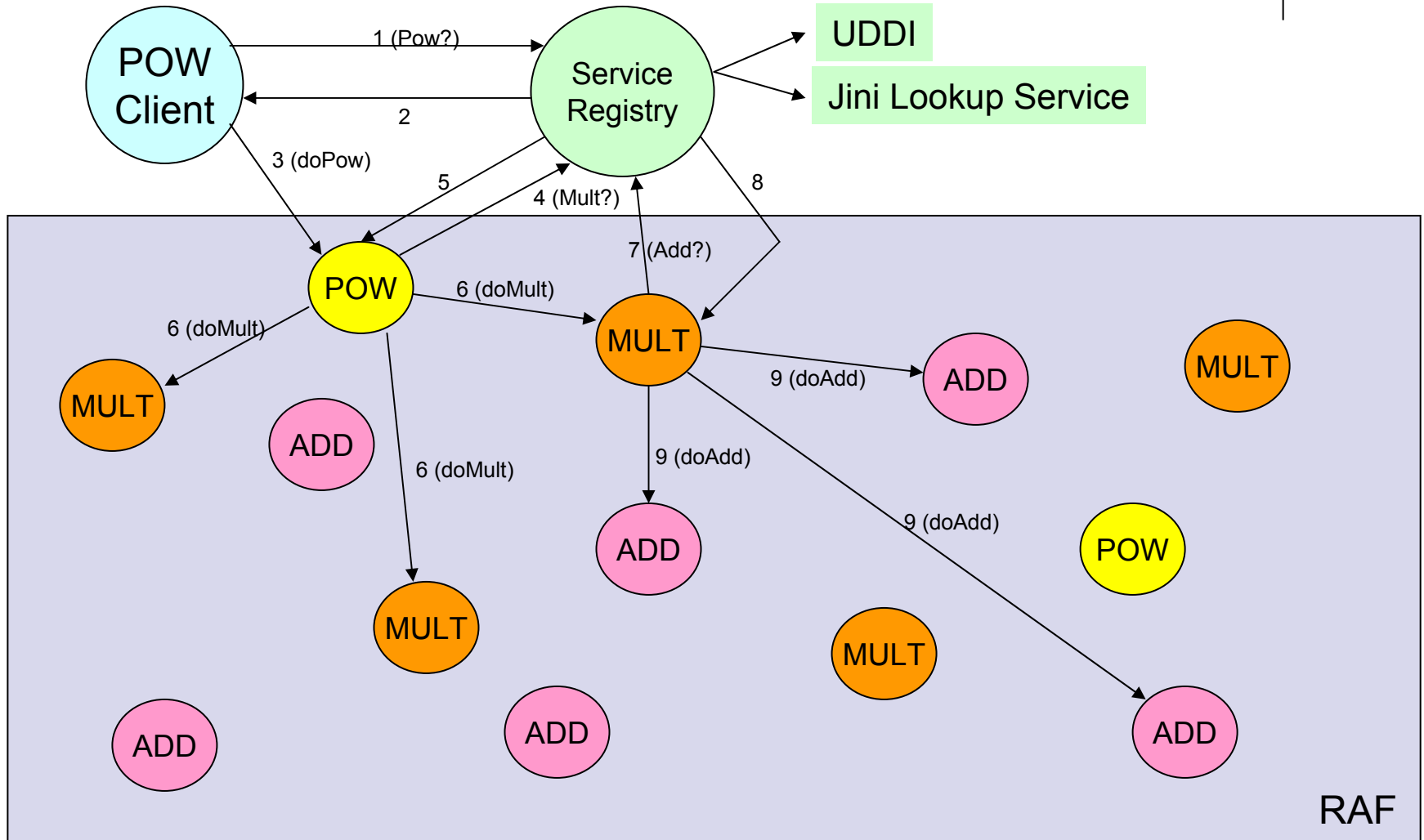
# Status of Jini Technology



- Jini Architecture Specification
- Jini Technology Core Platform Specification
  - Version 1.2
  - December 2001
- Java SDK 1.4

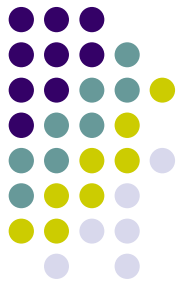


# Web Service Infrastructure

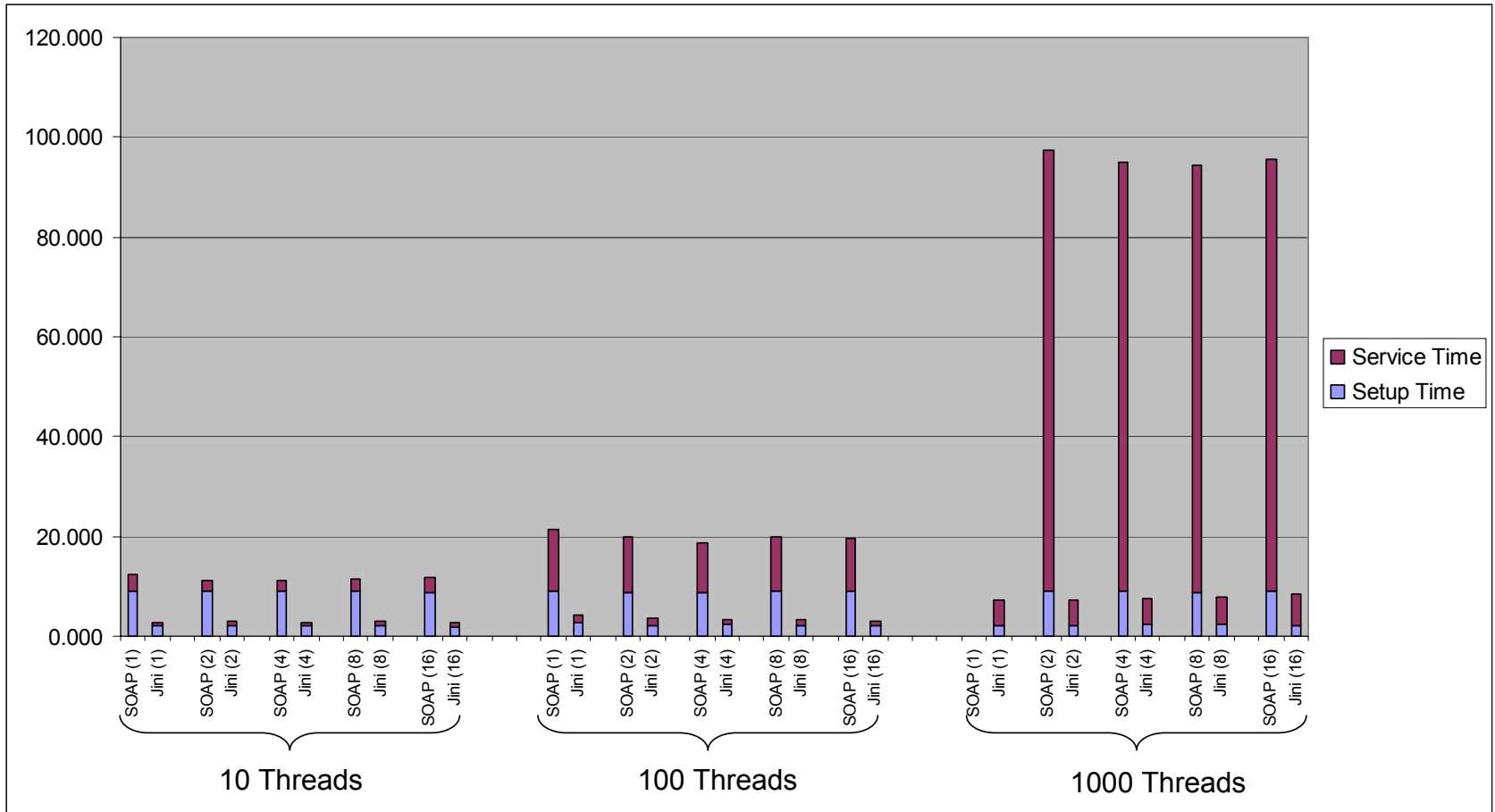


# Preliminary Results

## (Multiply Service with variable Thread number)



### 1. Jini vs. SOAP

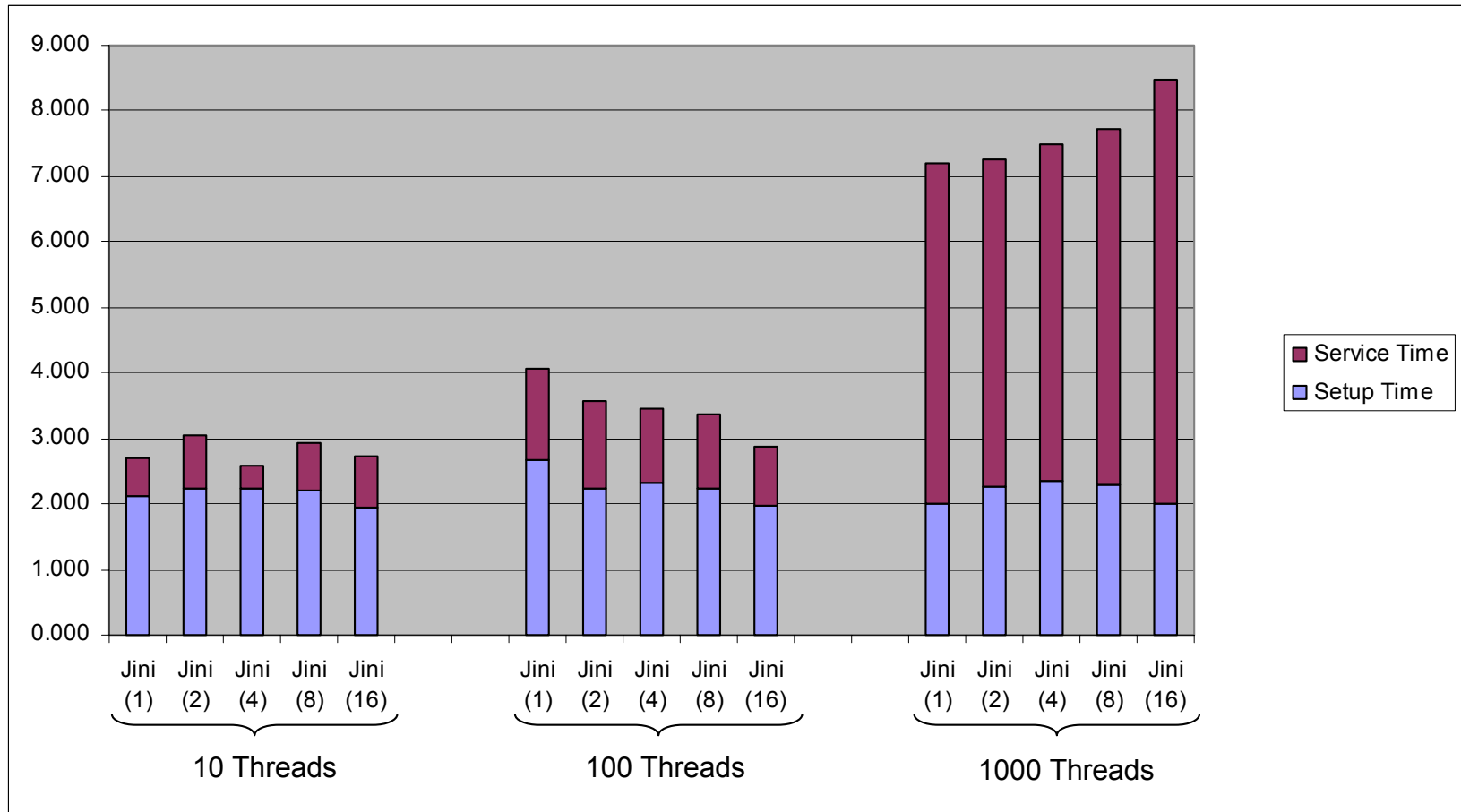


# Preliminary Results

## (Multiply Service with variable Thread number)

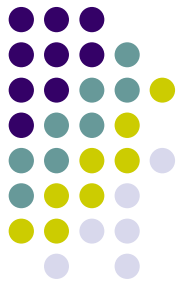


### 2. Jini Scalability

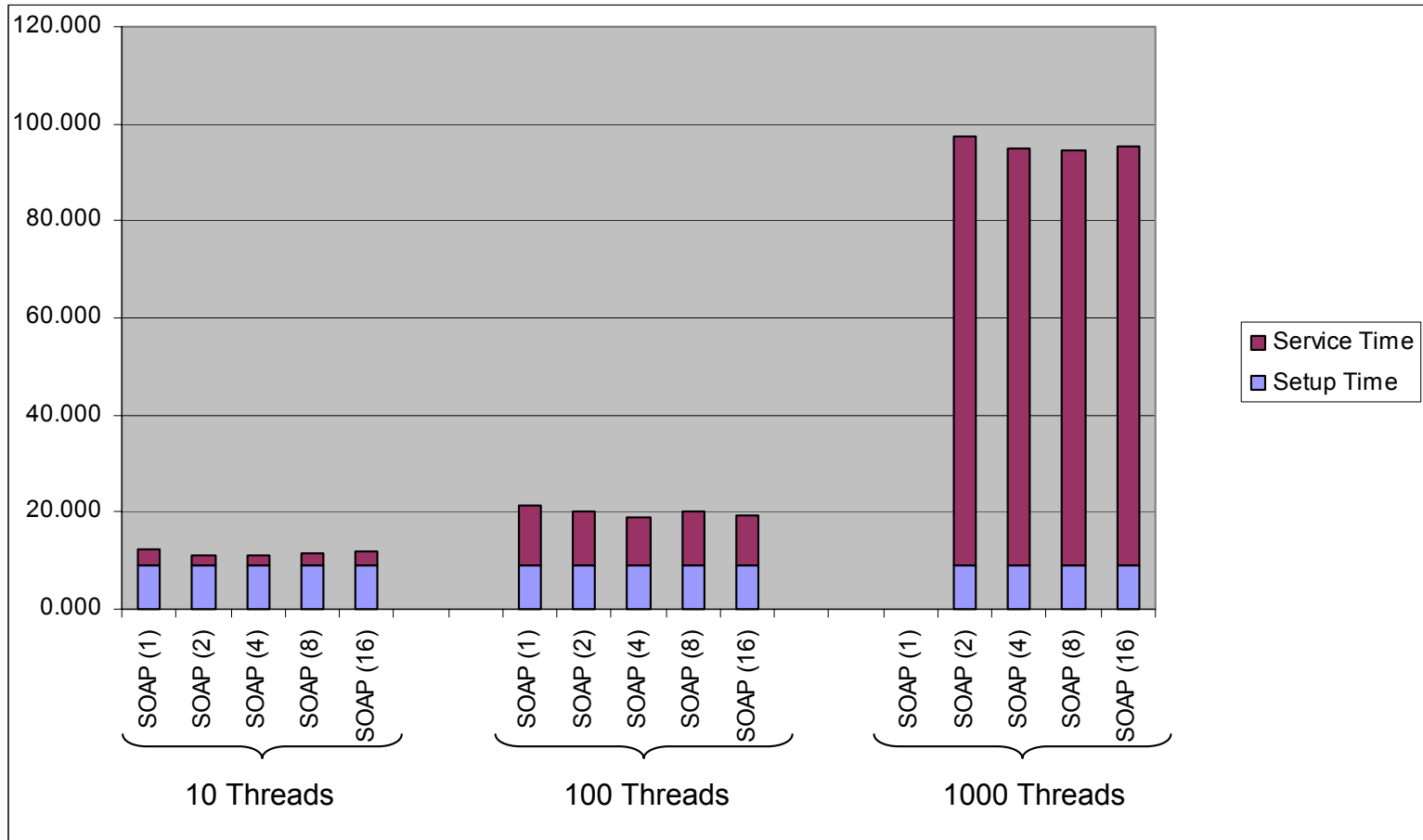


# Preliminary Results

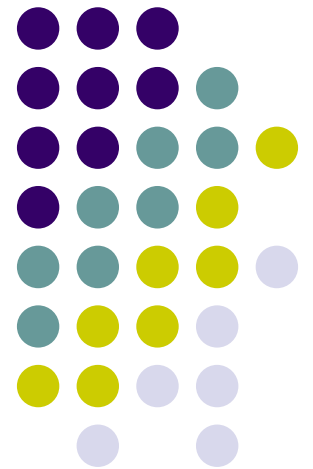
## (Multiply Service with variable Thread number)



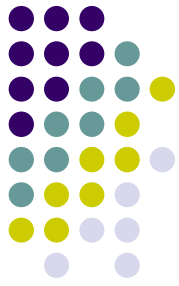
### 3. SOAP Scalability



# Any Question?



# SOAP Example



- SOAP Request (Subtract Service)

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: 465
SOAPAction: ""
```

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
    instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:subtract xmlns:ns1="urn:SimpleCalc" SOAP-
      ENV:encodingStyle="http://schemas.xmlsoap.org/s
      oap/encoding/">
      <p1 xsi:type="xsd:int">3</p1>
      <p2 xsi:type="xsd:int">1</p2>
    </ns1:subtract>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- SOAP Response (Subtract Service)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 460
```

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
    instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:subtractResponse
      xmlns:ns1="urn:SimpleCalc" SOAP-
      ENV:encodingStyle="http://schemas.xmlsoap.org/s
      oap/encoding/">
      <return xsi:type="xsd:int">2</return>
    </ns1:subtractResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

# UDDI Example



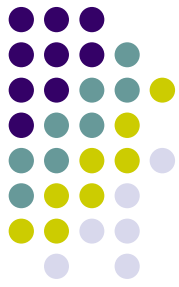
- Publish a binding for a service

```
<?xml version="1.0"?>
<uddi:save_binding generic="2.0" xmlns="urn:uddi-org:api_v2">
  <authInfo>[AUTHINFO]</authInfo>
  <bindingTemplate serviceKey="[SERVICE KEY]" bindingKey="">
    <accessPoint URLType="http">
      http://munro.inf.ethz.ch/soap/servlet/rpcrouter
    </accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo tModelKey="[TMODEL KEY]">
        <instanceDetails>
          <overviewDoc>
            <overviewURL>
              http://munro.inf.ethz.ch/~rbelotti/wsd/service.wsdl
            </overviewURL>
          </overviewDoc>
        </instanceDetails>
      </tModelInstanceInfo>
    </tModelInstanceDetails>
  </bindingTemplate>
</uddi:save_binding>
```

- Find the service using the interface description

```
<?xml version="1.0"?>
<uddi:find_service businessKey="[BUSINESS KEY]"
  generic="2.0" xmlns="urn:uddi-org:api_v2">
  <tModelBag>
    <tModelKey>[TMODEL KEY]</tModelKey>
  </tModelBag>
</uddi:find_service>
```

# Jini Service Publishing



- Define the implementation of the service Interface
- Set Leases, Service Informations and construct the proxy Object which will be sent to the client
- Create a ServiceItem to be registered within the lookup service
- DiscoveryListener listen for a DiscoveryEvent and if a new Lookup Service is discovered it register the service to it.
- Service Items are registered within the Lookup Service

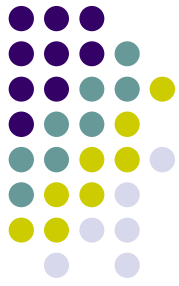
```
class ServiceImpl implements ServiceInterface, Serializable{  
    .....  
}
```

```
LEASE_TIME = 5 * 60 * 1000 // 5 minutes Lease  
ServiceInterface proxjObject = new ServiceImpl();  
ServiceInfo slnfo = new ServiceInfo(manufacturer, model, name,  
    serial, vendor, version);  
ServiceItem servItem = new ServiceItem(ID, proxyObject, slnfo);
```

```
class Listener implements DiscoveryListener{  
    public void discovered(DiscoveryEvent ev){  
        ServiceRegistrar [] newregs= ev.getRegistrars();  
        registerWithLookup(newregs);  
    }  
    public void discarded(DiscoveryEvent ev){  
        ServiceRegistrar [] deadregs= ev.getRegistrars();  
        remove(deadregs);  
    }  
}
```

```
public void registerWithLookup(ServiceRegistrar registrar){  
    registrar.register(servItem, LEASE_TIME);  
}
```

# Software (SOAP)



- WSDL
  - Borland Web Services Kit for JBuilder
- SOAP
  - Apache SOAP 2.2
- UDDI
  - Systinet's WASP UDDI 3.1 Standard
  - Systinet's WASP UDDI 4.0 Beta 1
    - PostgreSQL 7.2
- Servlet Engine
  - Apache Tomcat 4.0.3