Setting Up and Hosting Your Solver as Web Services via Optimization Services (OS)

Robert Fourer
Jun Ma
Northwestern University
Kipp Martin
University of Chicago

Jun Ma

maj@northwestern.edu
Industrial Engineering and Management Sciences, Northwestern University

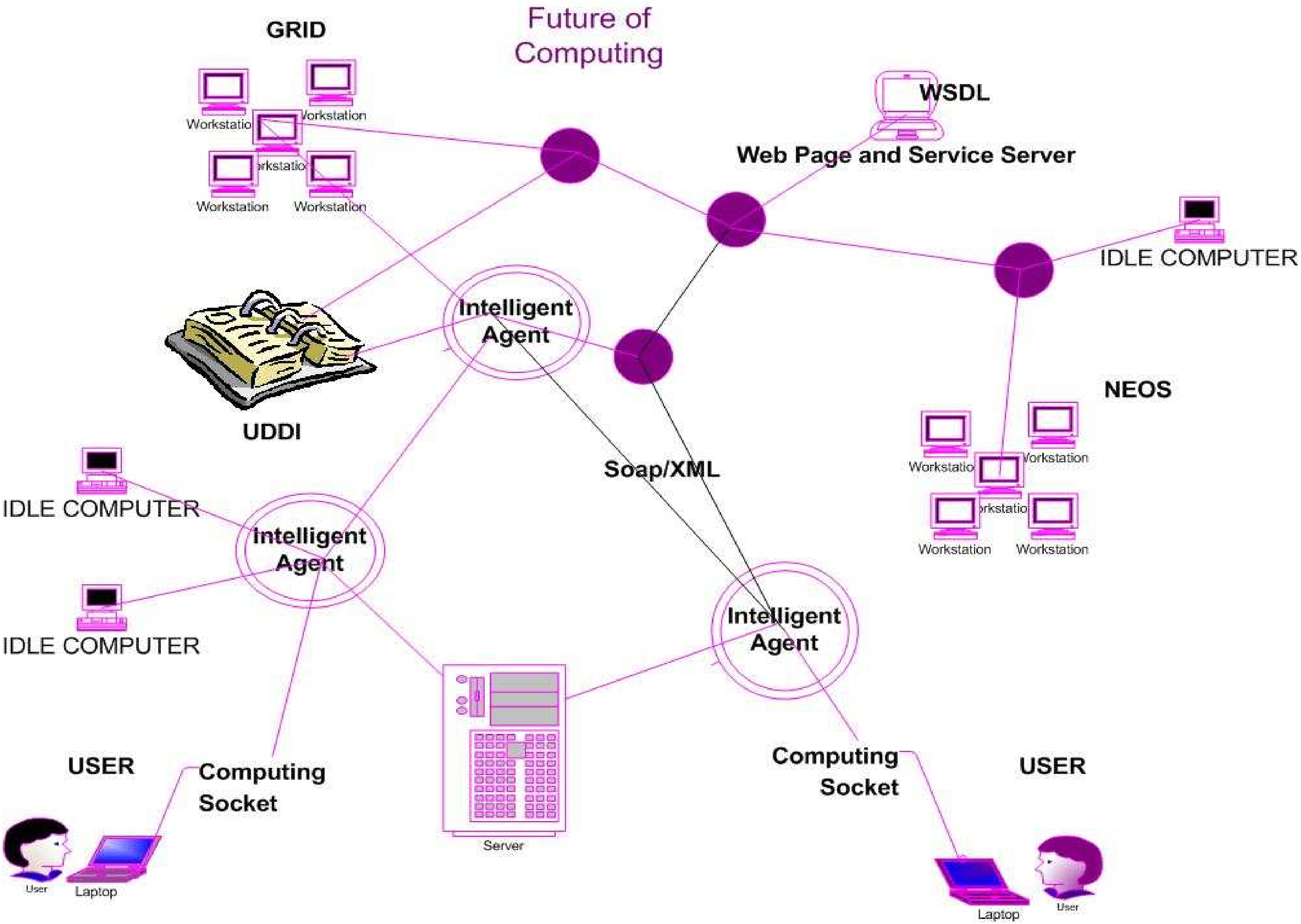
11/04/2007

Outline

- Motivation
- OS Framework
- OS Library
- OS Server
- Conclusion/User Experience

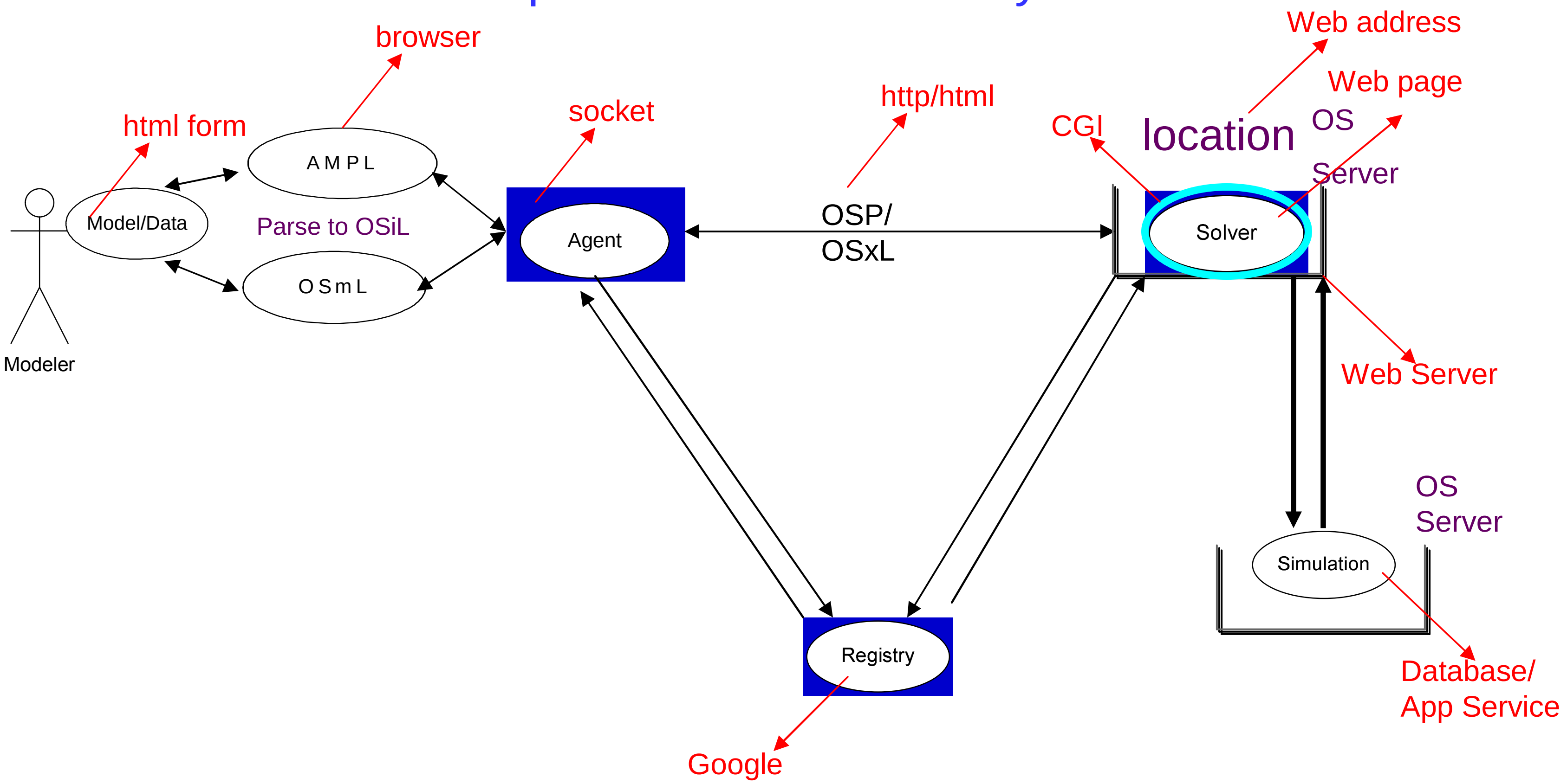
Motivation

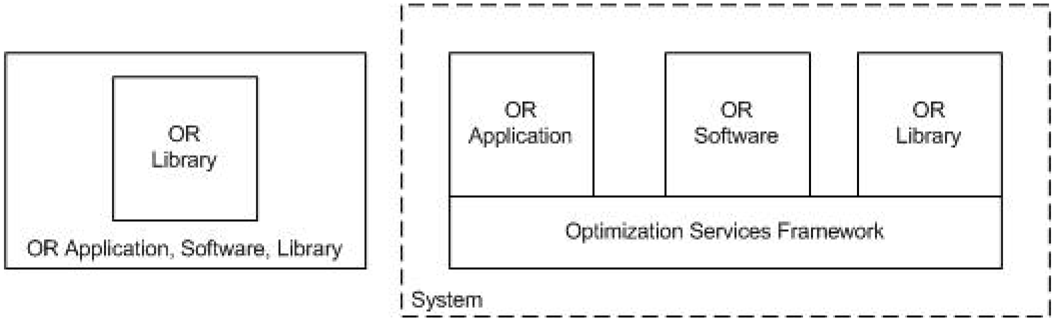
Future of Computing



OS Framework

Optimization Services System




OR
Library
OR Application, Software, Library
OR
Application
OR
Software
OR
Library
Optimization Services Framework
System

OS Library

- OSCommon
  - representationParser
    - OSiL Reader/Writer
    - OSrL result
    - OSoL option
    - Etc.
  - util
    - data structure
    - io
    - xml
    - etc
  - communicationInterface
    - OShL (hook up to solvers/analyzers: solve, send, retrieve)
    - OScL (call to simulations)
    - OSdL (discover in registries)
  - localInterface
    - OSInstance
    - etc.
  - nonlinear: defines all the nonlinear operator/operands/functions

```
OSiLReader reader = new OSiLReader();

reader.read(example.osil);

reader.getLinearConstraintCoefficients();

reader.calculateNonlinearFunction(5, x); //x is double[]
```

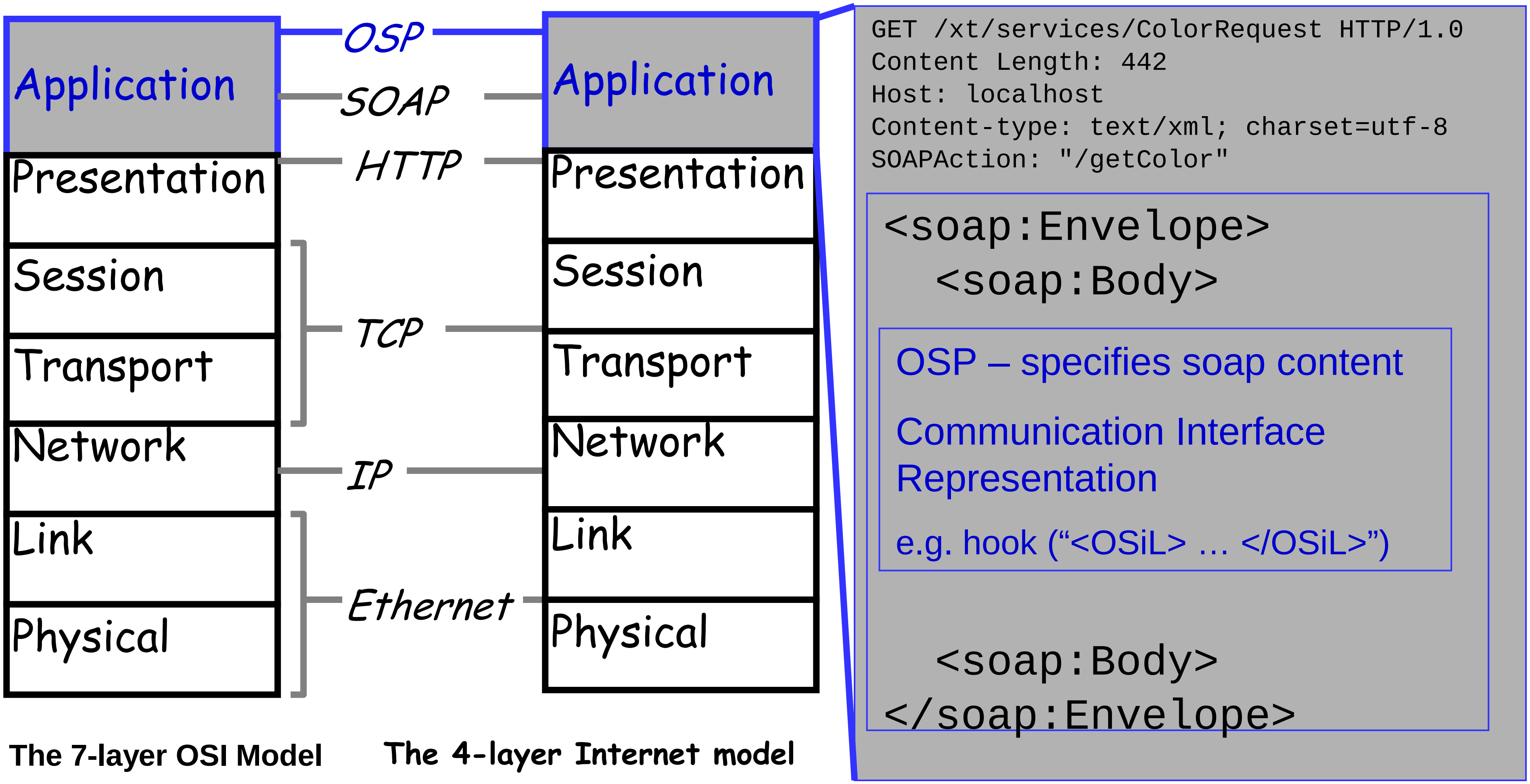
OS Library

```
OSSolverAgent agent = new OSSolverAgent();

agent.solverAddress = “http://1.2.3.6/CbcSolverService”;

String osrlResult = agent.solve(osilInstance, osolOption);
```

- OSAgent
  - Solver agent
  - Simulation agent
  - Solver agent
- OSSolver
  - Utility and implementation of os-compatible solvers
- OSSimulation
  - Utility and implementation of os-compatible simulation.
- OSRegistry
  - Allows os developers to register their services
  - Lets os users discover os services
  - Let os users/developers validate instances
- OSAnalyzer
  - Utility and implementation of os-compatible analyzers.
- OSScheduler
  - Schedules optimization jobs over the distributed system
  - Takes care of all the non-optimization related chores.

OS Framework

Optimization Services Protocol (OSP)



```
GET /xt/services/ColorRequest HTTP/1.0
Content Length: 442
Host: localhost
Content-type: text/xml; charset=utf-8
SOAPAction: "/getColor"
```

```
<soap:Envelope>
  <soap:Body>
```

OSP – specifies soap content

Communication Interface Representation

e.g. hook ("<OSiL> … </OSiL>")

```
  <soap:Body>
</soap:Envelope>
```

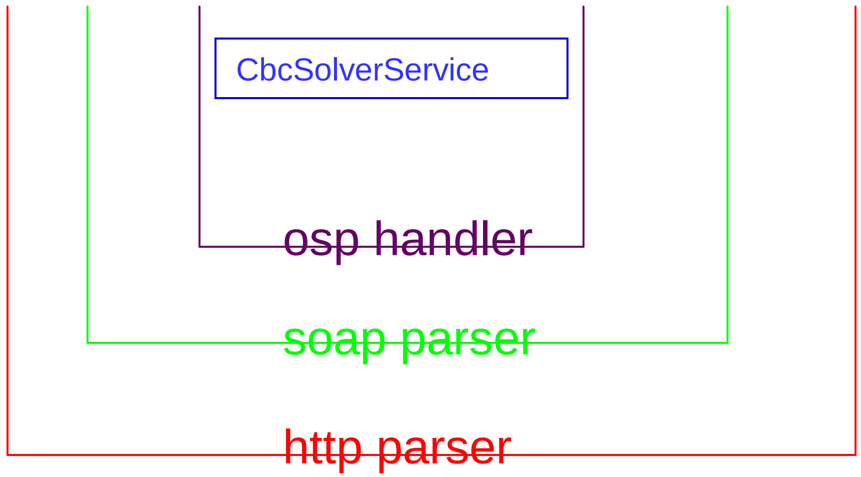
The 7-layer OSI Model

The 4-layer Internet model

OS Server

- **Networking Protocols: HTTP, SOAP, OSP**
  (OS server: Tomcat, Axis, OS library)

OSServer =



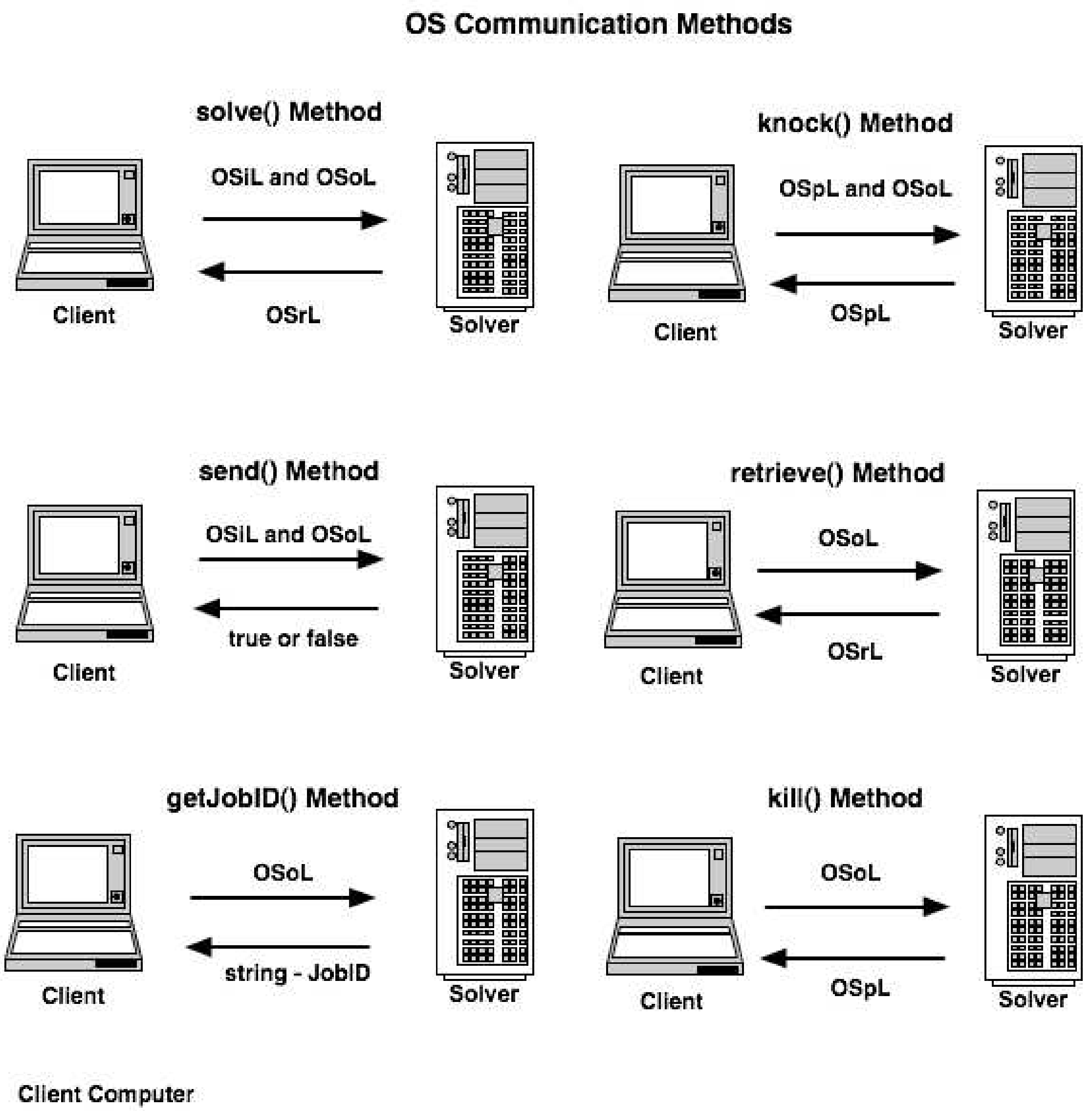
CbcSolverService

osp handler

soap parser

http parser

OS Server


OS Communication Methods
solve() Method
OSiL and OSoL
OSrL
Client
Solver
knock() Method
OSpL and OSoL
OSpL
Client
Solver
send() Method
OSiL and OSoL
true or false
Client
Solver
retrieve() Method
OSoL
OSrL
Client
Solver
getJobID() Method
OSoL
string - JobID
Client
Solver
kill() Method
OSoL
OSpL
Client
Solver
Client Computer

Download the OSServer

- Download the binary distribution:

  os-distribution-release_number.zip.

  The server side of the Java distribution is based on the Tomcat 5.5 implementation.
- After unpacking os-distribution-release_number.zip there is a directory os-server-1.0 and a single file os.war.
- For users that have not installed the Tomcat server, os-server-1.0 contains all of the necessary files for a OS Solver Service. If you do not have a Tomcat server running do the following to setup a Tomcat server with the OS Solver Service :

Setting up the OSServer

- Step 1. Put the folder os-server-1.0 in the desired location for the OS Solver Service on the server machine.
- Step 2. Connect to the Tomcat bin directory in the os-server-1.0 root and execute ./startup.sh (Linux) or ./start.bat (Windows)
- Step 3. Test to see if the server is running the OSSolverService. Open a browser on the server and enter the URL

http://localhost:8080/os/OSSolverService.jws

or

http://127.0.0.1:8080/os/OSSolverService.jws

You should see a message Click to see the WSDL. Click on the link and you should see an XML description of the various methods available from the OSSolverService.

- Step 4. On a client machine, create the file testremote.config with the following lines of text

```
-serviceLocation http://***.***.***.***:8080/os/OSSolverService.jws
-osil parincLinear.osil
```

where ***.***.***.*** is the IP address of the Tomcat server machine. Then, assuming the files testremote.config and parincLinear.osil are in the same directory on the client machine as the OSSolverService execute: ./OSSolverService -config testremote.config

- You should get back an OSrL message saying the problem was optimized.

Connect to the OSServer with OSSolverService

- At present, the OSSolverService takes the following parameters. The order of the parameters is irrelevant. Not all the parameters are required. However, if the solve or send service methods are invoked a problem instance location must be specified.
- -osil xxx.osil this is the name of the file that contains the optimization instance in OSiL format. It is assumed that this file is available in a directory on the machine that is running OSSolverService. If this option is not specified then the instance location must be specified in the OSoL solver options file.
- -osol xxx.osol this is the name of the file that contains the solver options. It is assumed that this file is available in a directory on the machine that is running OSSolverService. It is not necessary to specify this option.
- -osrl xxx.osrl this is the name of the file that contains the solver solution. A valid file path must be given on the machine that is running OSSolverService. It is not necessary to specify this option.

Connect to the OSServer with OSSolverService

- -serviceLocation url is the URL of the solver service. This is not required, and if not specified it is assumed that the problem is solved locally.
- -serviceMethod methodName this is the method on the solver service to be invoked. The options are **solve**, **send**, **kill**, **knock**, **getJobID**, and **retrieve**. The use of these options is illustrated in the examples below. This option is not required, and the default value is solve.
- -solver solverName Possible values for default OS installation are clp (COIN-OR Clp), cbc (COIN-OR Cbc), dylp (COIN-OR DyLP), and symphony (COIN-OR SYMPHONY). Other solvers supported (if the necessary libraries are present) are cplex (Cplex through COIN-OR Osi), glpk (glpk through COIN-OR Osi), ipopt (COIN-OR Ipopt), knitro (Knitro), and lindo LINDO. If no value is specified for this parameter, then cbc is the default value of this parameter if the the solve or send service methods are used.

Connect to the OSServer with OSSolverService

- -mps xxx.mps this is the name of the mps file if the problem instance is in mps format. It is assumed that this file is available in a directory on the machine that is running OSSolverService. The default file format is OSiL so this option is not required.
- -nl xxx.nl this is the name of the AMPL nl file if the problem instance is in AMPL nl format. It is assumed that this file is available in a directory on the machine that is
- -browser browserName this paramater is a path to the browser on the local machine. If this optional parameter is specified then the solver result in OSrL format is transformed using XSLT into HTML and displayed in the browser.
- -config pathToConfigureFile this parameter specifies a path on the local machine to a text file containing values for the input parameters. This is convenient for the user not wishing to constantly retype parameter values.

Examples (1)

- ./OSSolverService -solver clp -osil ./parincLinear.osil
- ./OSSolverService –config ./testlocalclp.config

  where testlocalclp.config looks like:

  -osil ./parincLinear.osil

  -solver clp
- ./OSSolverService –config ./testlocal.config

  where testlocalclp.config looks like:

  -osil ../data/osilFiles/parincQuadratic.osil

  -solver ipopt

  -serviceMethod solve

  -browser /Applications/

  -osrl ./test.osrl
- ./OSSolverService –c

  where testlocalclp.conf

  -osol ./demo.osol

  -solver clp

```
<?xml version="1.0" encoding="UTF-8"?>
<osol xmlns="os.optimizationservices.org">
<general>
<instanceLocation locationType="local">
../data/osilFiles/parincLinear.osil
</instanceLocation>
</general>
</osol>
```

Examples (2)

- ./OSSolverService –config ./testremote.config

  where testlocalclp.config looks like:

  -osil ./parincLinear.osil

  -serviceLocation http://gsbkip.chicagogsb.edu/os/OSSolverService.jws

  -serviceMethod send

- ./OSSolverService -config .testremote.config -solver clp
- or by adding the line -solver clp to the testremote.config file.
- ./OSSolverService -osol ./remoteSolve1.osol -serviceLocation http://gsbkip.chicagogsb.edu/os/OSSolverService.jws

  where remoteSolve1.osol looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<osol xmlns="os.optimizationservices.org">
<general>
<instanceLocation locationType="local">c:\parincLinear.osil</instanceLocation>
<contact transportType="smtp">maj@northwestern.edu</contact>
</general>
<optimization>
<other name="os_solver">ipopt</other>
</optimization>
</osol>
```

Usage Summary

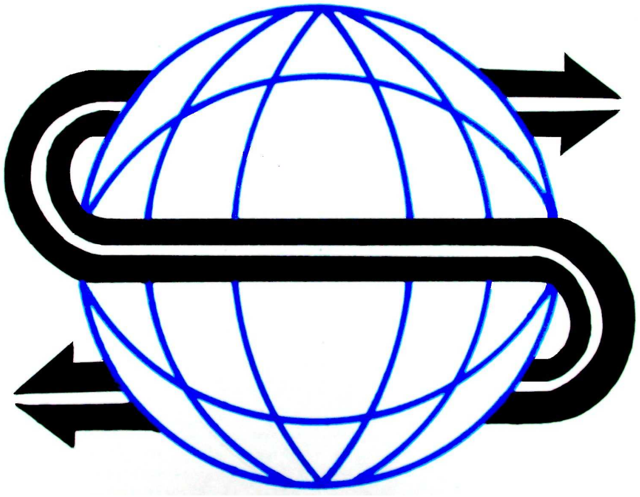
- solve(osil, osol):
  - Inputs: a string with the instance in OSiL format and an optional string with the solver options in OSoL format
  - Returns: a string with the solver solution in OSrL format
  - Synchronous call, blocking request/response
- send(osil, osol):
  - Inputs: a string with the instance in OSiL format and a string with the solver options in OSoL format (same as in solve)
  - Returns: a boolean, true if the problem was successfully submitted, false otherwise
  - Has the same signature as solve
  - Asynchronous (server side), non-blocking call
  - The osol string should have a JobID in the <jobID> element

Usage Summary

- getJobID(osol)
  - Inputs: a string with the solver options in OSoL format (in this case, the string may be empty because no options are required to get the JobID)
  - Returns: a string which is the unique job id generated by the solver service
  - Used to maintain session and state on a distributed system
- knock(ospl, osol)
  - Inputs: a string in OSpL format and an optional string with the solver options in OSoL format
  - Returns: process and job status information from the remote server in OSpL format
- retrieve(osol)
  - Inputs: a string with the solver options in OSoL format
  - Returns: a string with the solver solution in OSrL format
  - The osol string should have a JobID in the <jobID> element
- kill(osol)
  - Inputs: a string with the solver options in OSoL format
  - Returns: process and job status information from the remote server in OSpL format
  - Critical in long running optimization jobs

Conclusion/User Experience

- Open Environment
- Convenience just like Using Utility Services
- No High Computing Power Needed
- No Knowledge in Optimization Algorithms and Software (solvers, options, etc.)
- Better and More Choices of Modeling Languages
- More Solver Choices
- Solve More Types of Problems
- Automatic Optimization Services Discovery
- Decentralized Optimization Services Development and Registration
- More Types of Optimization Services Components Integrated (Analyzers/Preprocessors, Problem Providers, Bench Markers)
- Smooth Flow and Coordination of Various Optimization Services Components.
- A Universal, Scalable and Standard Infrastructure that promotes Collaboration and Other Related Researches
- Concentration on Good Modeling



- www.optimizationservices.org
- www.coin-or.org/OS