1st iLab Europe Workshop

Danilo Garbi Zutin

Carinthia University of Applied Sciences

Batched Laboratories Development

Villach, Austria – November 17th 2009

Online Labs: Implementation Point of View

Different solutions and technologies exist today to implement remote Laboratories as well as different communication standards and data exchange protocols.

Therefore, each Institution/University is likely to adopt its own standards and approaches to perform tasks like handling user‘s accounts and managing experiment data. Because of that, sharing remote Labs becomes more difficult.

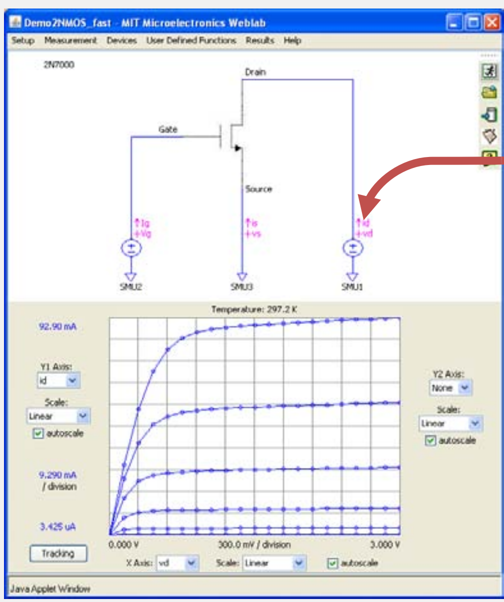
As the number of online labs increases, a highly scalable architecture is desirable in order that labs could be managed in a confortable way and included/disposed easily.

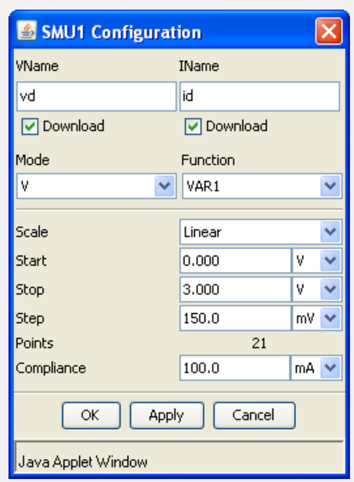
Batched Experiments:

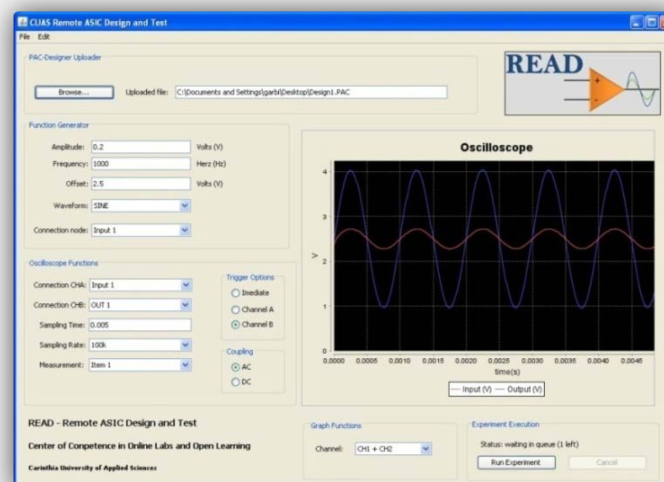
Batched experiments are those in which the entire course of the experiment can be specified before the experiment begins.

Batched experiments should be queued for execution in order to maximize the efficiency of the lab server.

Example: MIT's Microelectronics WebLab for device characterization, CUAS READ System.



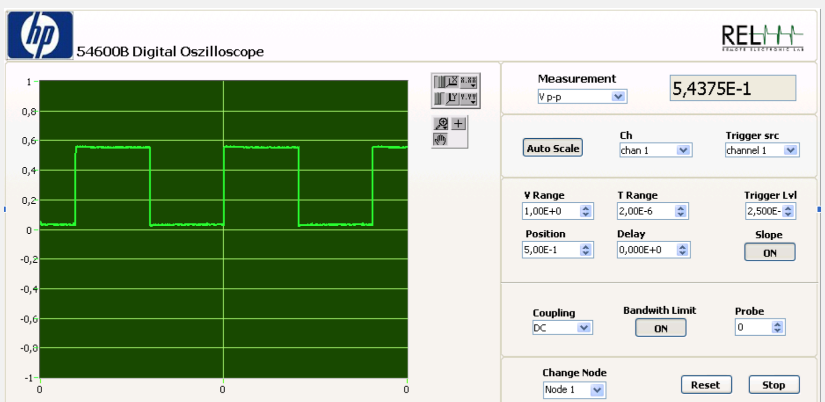


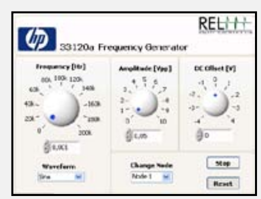


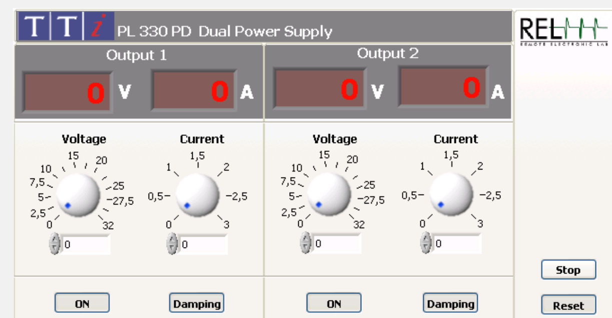
Interactive and Sensor Experiments:

Interactive experiments are those in which the user monitors and can control one or more aspects of the experiment during its execution.

Example: CTI’s REL (Remote Electronic Lab). Students can dynamically change the input to the oscilloscope, function generator, power supply and multimeter and watch live data being displayed on the oscilloscope screen as parameters are changed.







Sensor experiments are those in which users monitor or analyze real-time data streams without influencing the the phenomena being measured. MIT’s online photovoltaic station [11] provides a simple example.

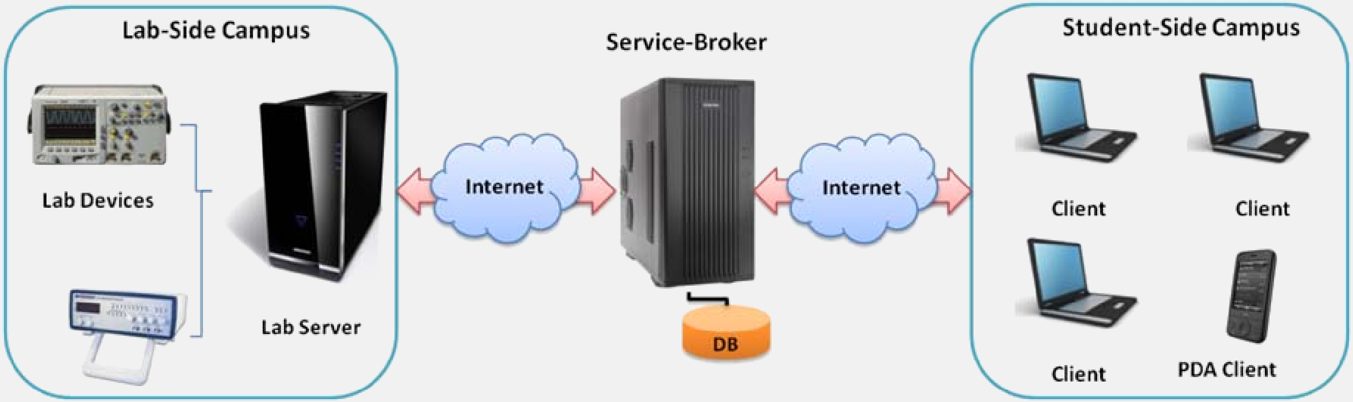
Three-tier Web Application:

Center of Competence in Online Labs and Open Learning

First tier: *Client Application* that either runs as an applet or as a downloaded application on the student's workstation.

Middle tier: The *Service Broker* provides the shared common services. It is backed by a standard relational database (SQL Server™).

Third tier: *The Lab Server* that executes the experiments and notifies the Service Broker when the results are ready.



The Topology of the Batched Experiment Architecture[1]

Lab Server/Client Design

Center of Competence in Online Labs and Open Learning

Batched Clients and Lab Servers Structure:

Messages between a *lab client* and *server* are very specific and should be transmitted through the generic channels of the *Service Broker*.

Batched-lab Development:

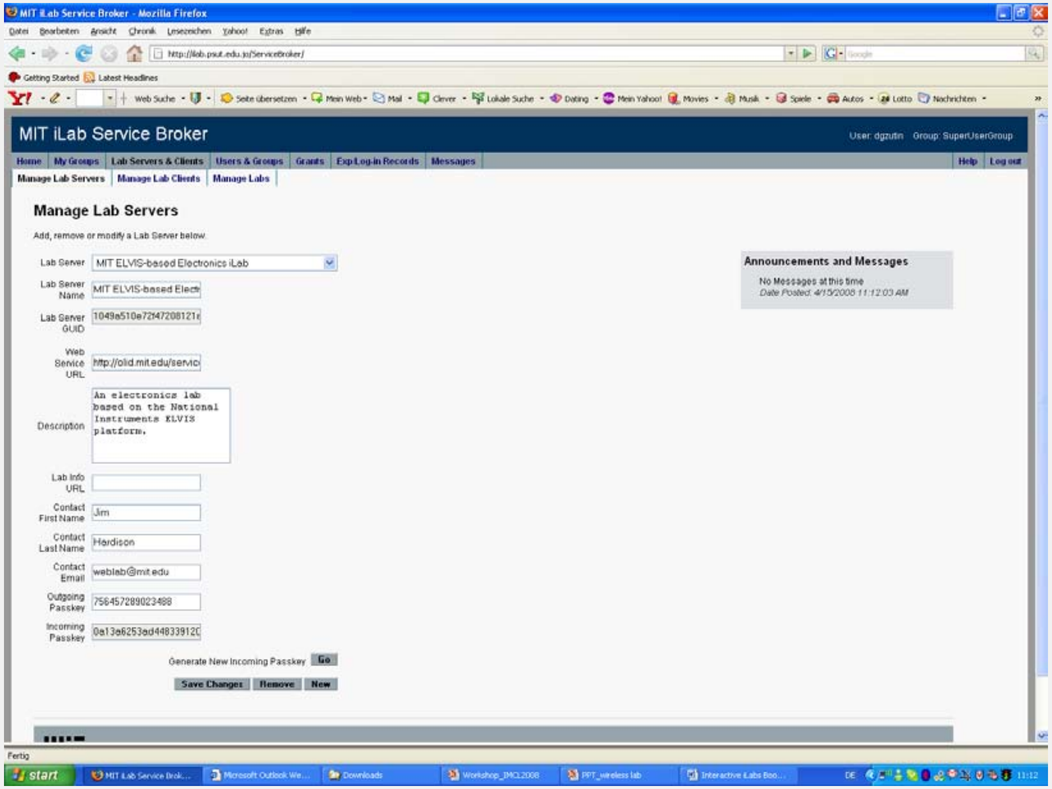
Lab Client development

Lab Server development

Lab Client/Server Communication framework (define inputs and outputs)

The communication framework can be expressed in any text format, but XML documents are an ideal vehicle for that.

Registering a Batched lab on a Service Broker



Install Domain Credentials:

-Contact the process Agent (Lab Server's side) administrator via email and get the web services URL and an initial passkey.

The SB Web Services Interface

Center of Competence in Online Labs and Open Learning

ServiceBrokerService

The following operations are supported. For a formal definition, please review the Service Description.

- GetLabStatus
  Checks on the status of the lab server
- RetrieveAnnotation
  Retrieves a previously saved experiment annotation.
- GetEffectiveQueueLength
  Checks on the effective queue length of the Lab Server
- GetLabConfiguration
  Gets the lab configuration of a lab server
- SaveClientItem
  Sets a client item value in the user's opaque data store
- GetExperimentInformation
  Retrieves experiment metadata for experiments specified by an array of experiment IDs.
- RetrieveLabConfiguration
  Retrieves a previously saved lab configuration for a particular experiment
- Cancel
  Cancels a previously submitted experiment. If the experiment is already running, makes best efforts to abort execution, but there is no guarantee that the experiment will not run to completion
- Submit
  Submits an experiment specification to the lab server for exection
- SaveAnnotation
  Saves or modifies an optional user defined annotation to the experiment record.
- RetrieveResult
  Retrieves the results from (or errors generated by) a previously submitted experiment
- RetrieveSpecification
  Retrieves a previously saved experiment specification
- ListAllClientItems
  Enumerates the names of all client items in the user's opaque data store
- GetLabInfo
  Gets general information about a lab server
- GetExperimentStatus
  Checks on the status of a previously submitted experiment
- Validate
  Submits an experiment specification to the lab server for exection
- Notify
  Notification from the Lab Server that a previously submitted experiment has terminated.
- DeleteClientItem
  Removes an client item from the user's opaque data store
- LoadClientItem
  Returns the value of an client item in the user's opaque data store

Lab Server Web Services Interface

Center of Competence in Online Labs and Open Learning

LabServerAPI

WebLab Lab Server/Service Broker Interface

The following operations are supported. For a formal definition, please review the **Service Description**.

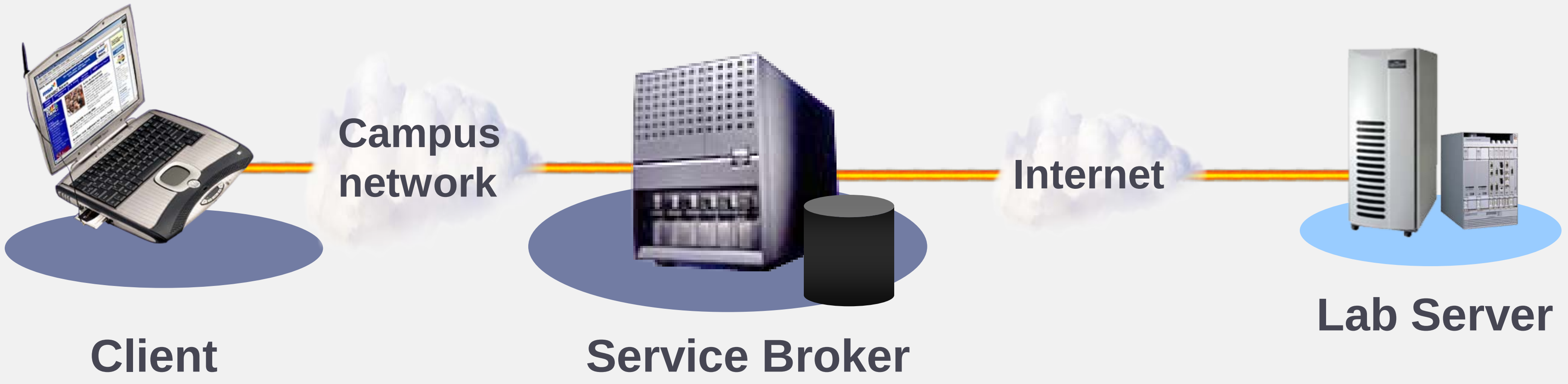
- **Cancel**
  WebLab Lab Server Method: This method cancels the specified spubmitted experiment. A single boolean return value indicates whether or not the cancel operation completed successfully.
- **GetLabConfiguration**
  WebLab Lab Server Method: Returns the valid XML lab configuration document 'labConfiguration'. Used by the lab client to determine experiment setup availability.
- **GetLabInfo**
  WebLab Lab Server Method: Returns a URL where information about that lab may be found.
- **GetEffectiveQueueLength**
  WebLab Lab Server Method: Returns the current length of the experiment queue based on a hypothetical priority, group and broker membership. Members of struct WaitEstimate are 'effectiveQueueLength' (int) and 'estWait' (double).
- **GetExperimentStatus**
  WebLab Lab Server Method: This method returns the status of a previously submitted experiment. Members of the struct LabExperimentStatus are 'statusReport' (struct ExperimentStatus) and 'minTimeToLive' (double). Members of the struct ExperimentStatus are 'statusCode' (int), 'wait' (struct waitEstimate), 'estRuntime' (double) and 'estRemainingRuntime' (double).
- **Submit**
  WebLab Lab Server Method: This method validates an incoming experiment specification, calculates the approriate execution priority and, if no errors are thrown, enters the job into the execution queue. Members of the struct SubmissionReport are 'vReport' (ValidationReport), 'minTimeToLive' (double) and 'wait' (WaitEstimate)
- **GetLabStatus**
  WebLab Lab Server Method: Returns the current status of the lab. Members of struct LabStatus are 'online' (boolean) and 'labStatusMessage' (string).
- **RetrieveResult**
  WebLab Lab Server Method: This method returns the results of the specified experiment. Memebers of the struct ResultReport are 'statusCode' (int), 'experimentResult' (string), 'xmlResultExtension' (string), 'xmlBlobExtension' (string), 'warningMessages' (string()) and 'errorMessage' (string).
- **Validate**
  WebLab Lab Server Method: Checks that the submitted experiment specification is valid and executable by the specified caller. Member of struct ValidationReport are 'accepted' (boolean), 'warningMessages' (string()), 'errorMessage' (string) and 'estRuntime' (double)

Center of Competence in Online Labs and Open Learning

Client and Lab Server Design

(Batched Architecture)

Batched labs in the iLab Shared Architecture



- Service Broker provides generic services, deployment mechanism for the client.
- Lab Server and Client contain lab-specific code.
- All communications pass through Service Broker.

Lab Server Developer Tasks

Center of Competence in Online Labs and Open Learning

Design Lab Server

- Bound by lab instrumentation, desired functionality, iLab API

Design Lab Client

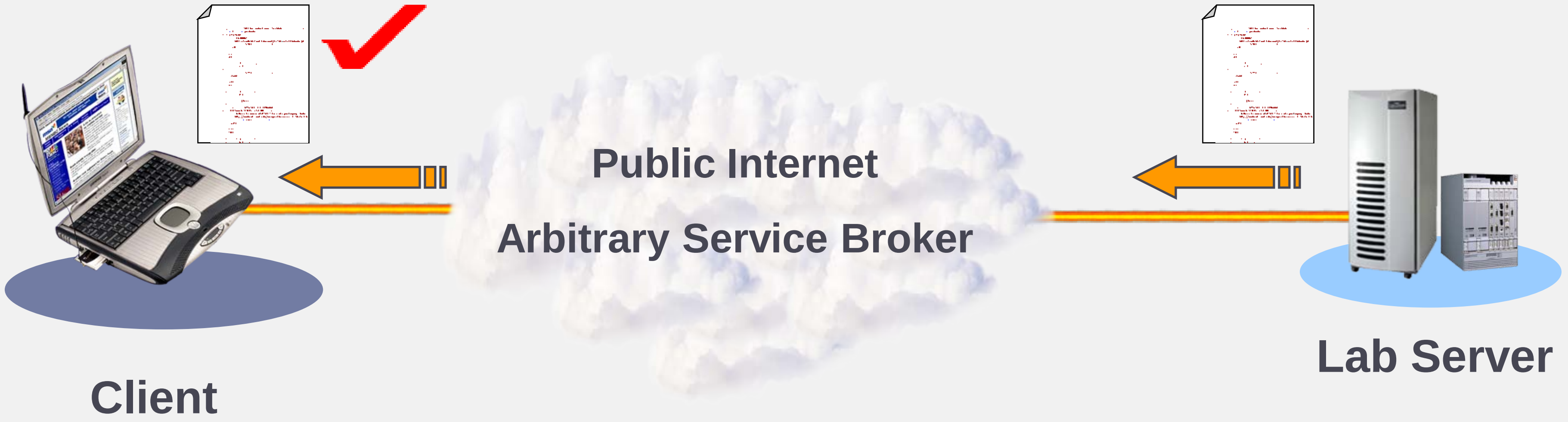
- Bound by Lab-Specific UI requirements, iLab API

Design Server–Client communication framework

- Specification of batched parameters and results (processed only by Lab server and lab client)
- Definition of messages passed between server and client

Lab Client–Server Communication

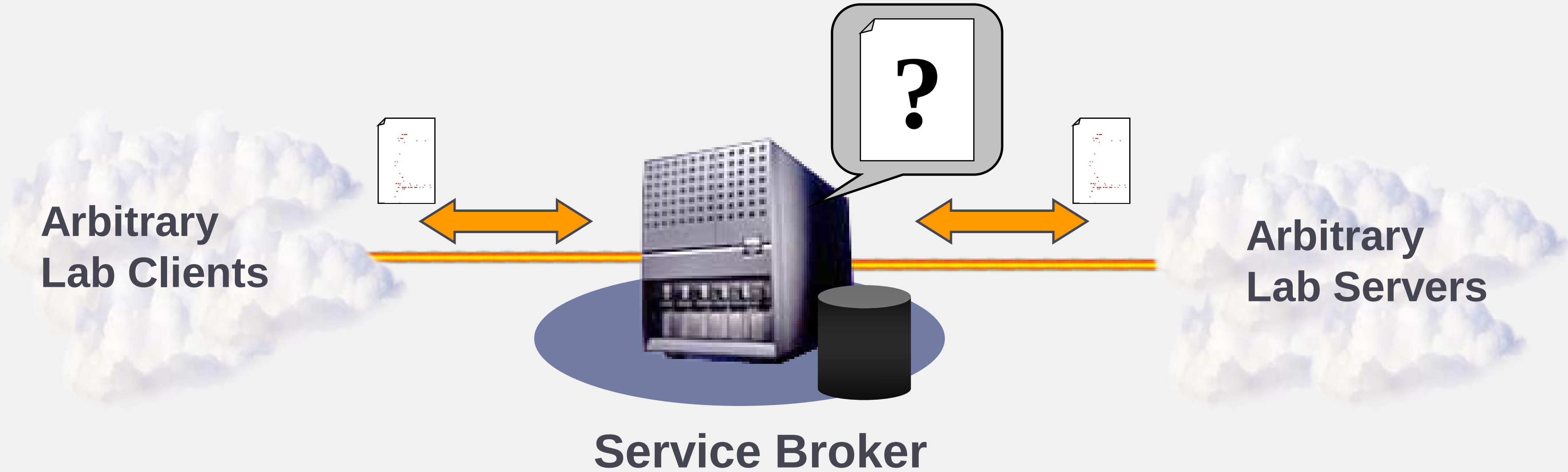
- Messages passed between Client and Lab Server communicate key lab information.
  - Lab Hardware Configuration/Status
  - Experiment Parameters & Results
- This information is necessarily lab-specific.



Server–Client communication framework

Center of Competence in Online Labs and Open Learning

- All Lab Client-Server Messages must be passed through Service Broker.
  - Generic mechanism.
- XML an ideal technology for this application.

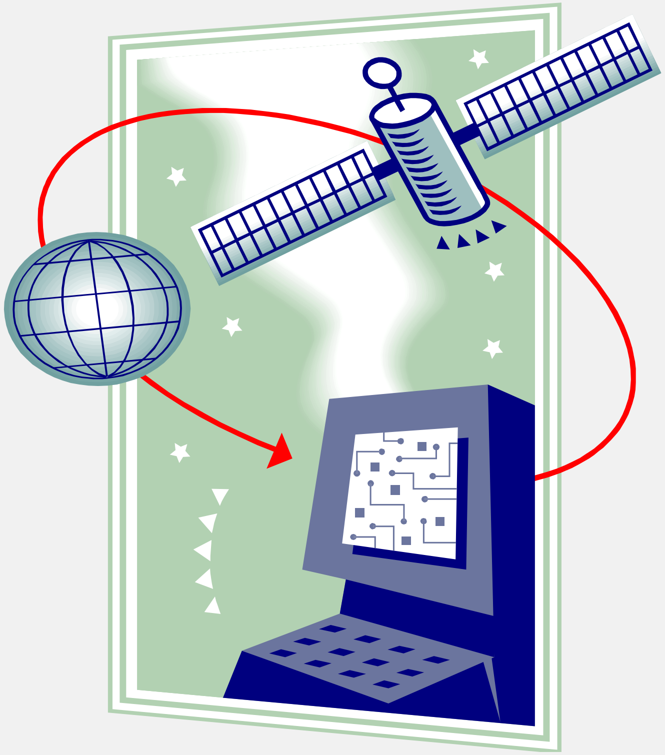


Lab Server Design

Center of Competence in Online Labs and Open Learning

Basic Requirements:

- Provide access to lab hardware.
- Implement the iLab Lab Server API
- Define & utilize format for lab-specific communication with the Client.
- Provide any other functionality necessary for lab operation



Note: **iLab Architecture APIs are platform-neutral. Lab Server technology driven by lab resources, hardware requirements.**

Center of Competence in Online Labs and Open Learning

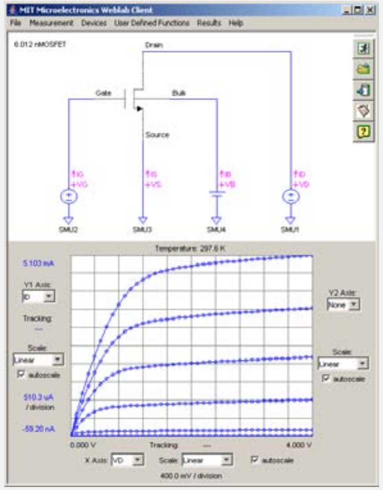
Basic requirements:

- Provide an educationally valuable user interface to the lab, embody pedagogical aspects
- Implement the iLab Client-Service Broker API
- Create & Interpret lab-specific communication messages with Lab Server

Again… iLab Architecture APIs are platform-neutral. Lab developer can select the best technology for their Client.

Example: MIT Microelectronics Device Characterization iLab

- Online microelectronic device characterization lab.
- First lab deployed using the iLab Architecture.
- Used by students, guests & OCW users worldwide.



Semiconductor Parameter Analyzer

Lab Server Development Examples

Center of Competence in Online Labs and Open Learning

- Three distinct message types used for lab-specific communication between Client and Lab Server.
  - Lab Configuration
  - Experiment Specifications
  - Experiment Results
- XML is used to encode information.
- Passed through the Service Broker as generic text.


```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentSpecification (View Source for full doctype...)>
- <experimentSpecification lab="MIT Microelectronics Weblab" specversion="0.1"
  <deviceID>4</deviceID>
 - <terminal portType="SMU" portNumber="2">
    <vname download="false">VG</vname>
    <iname download="false">IG</iname>
    <mode>V</mode>
  - <function typ
      <scale>LIN
      <start>0.0
      <stop>3.0
      <step>0.5
    </function>
    <compliance>
  </terminal>
 - <terminal port
    <vname dow
    <iname dow
    <mode>COM
  </terminal>
 - <terminal port
    <vname dow
    <iname dow
    <mode>V</
  - <function typ
      <value>0.0
    </function>
    <compliance>
  </terminal>
 - <terminal port
    <vname dow
    <iname dow
    <mode>V</
  - <function typ
      <scale>LIN
      <start>0.0
      <stop>4.0
      <step>0.1
    </function>
    <compliance>
  </terminal>
```

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE labConfiguration (View Source for full doctype...)>
- <labConfiguration lab="MIT Microelectronics Weblab" specversion="0.1"
 - <device id="1" type="pn diode">
    <name>pn Diode</name>
    <description>pn diode</description>
    <imageURL>http://weblab2.mit.edu/images/devices/pndiode.gif</i
  - <terminal portType="SMU" portNumber="1">
      <label>Left</label>
    - <pixelLocation>
        <x>158</x>
        <y>91</y>
      </pixelLocation>
      <maxVoltage>4</maxVoltage>
      <maxCurrent>0.1</maxCurrent>
    </terminal>
  - <terminal portType="SMU" portNumber="2">
      <label>Right</label>
    - <pixelLocation>
        <x>341</x>
        <y>92</y>
      </pixelLocation>
      <maxVoltage>4</maxVoltage>
      <maxCurrent>0.1</maxCurrent>
    </terminal>
    <maxDataPoints>1000</maxDataPoints>
  </device>
 - <device id="2" type="nMOSFET (3 terminal)">
    <name>3 terminal NMOS (2N7000)</name>
    <description>A three terminal nMOSFET. Discrete packaging. Model
    <imageURL>http://weblab2.mit.edu/images/devices/22NMOS(3-te
  - <terminal portType="SMU" portNumber="2">
      <label>Gate</label>
    - <pixelLocation>
        <x>175</x>
        <y>101</y>
      </pixelLocation>
      <maxVoltage>5</maxVoltage>
      <maxCurrent>0.1</maxCurrent>
```

MIT Microelectronics iLab Lab Server

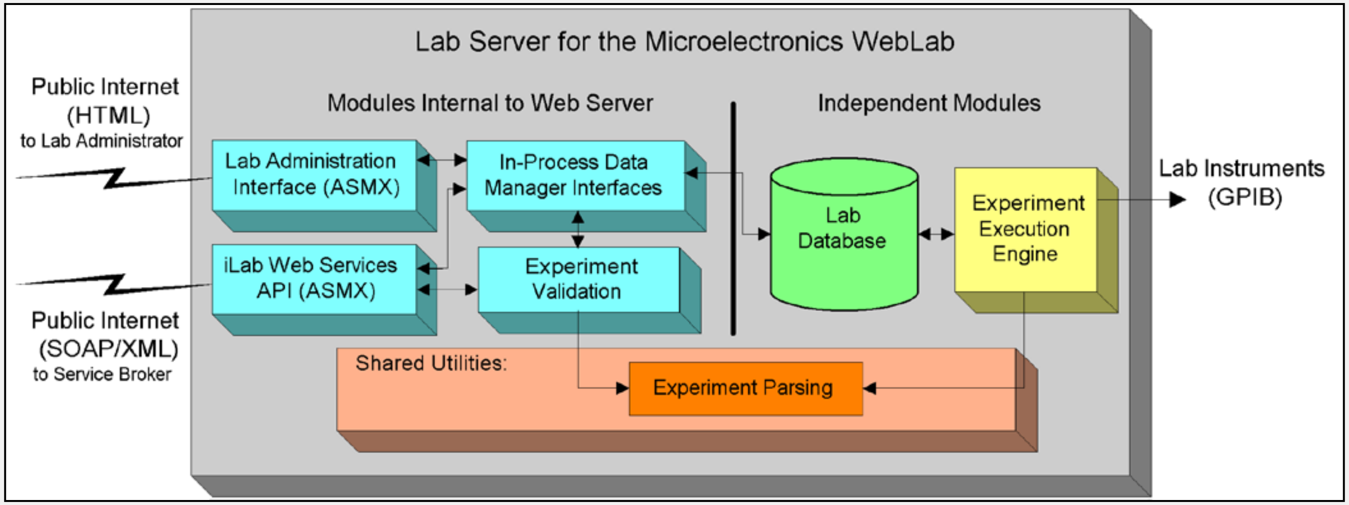
Center of Competence in Online Labs and Open Learning

Lab Server Requirements:

- Scalable performance and reliability.
  - Asynchronous experiment submission and execution
- Built-in lab management utilities.
- Highly modular, extensible.

The Lab Server

Center of Competence in Online Labs and Open Learning



Picture from MIT/CECI

Built on Windows using .NET Framework and MS SQL Server.

The Lab Server: Experiment Validation

Center of Competence in Online Labs and Open Learning

All experiments are validated on the server before they are queued:

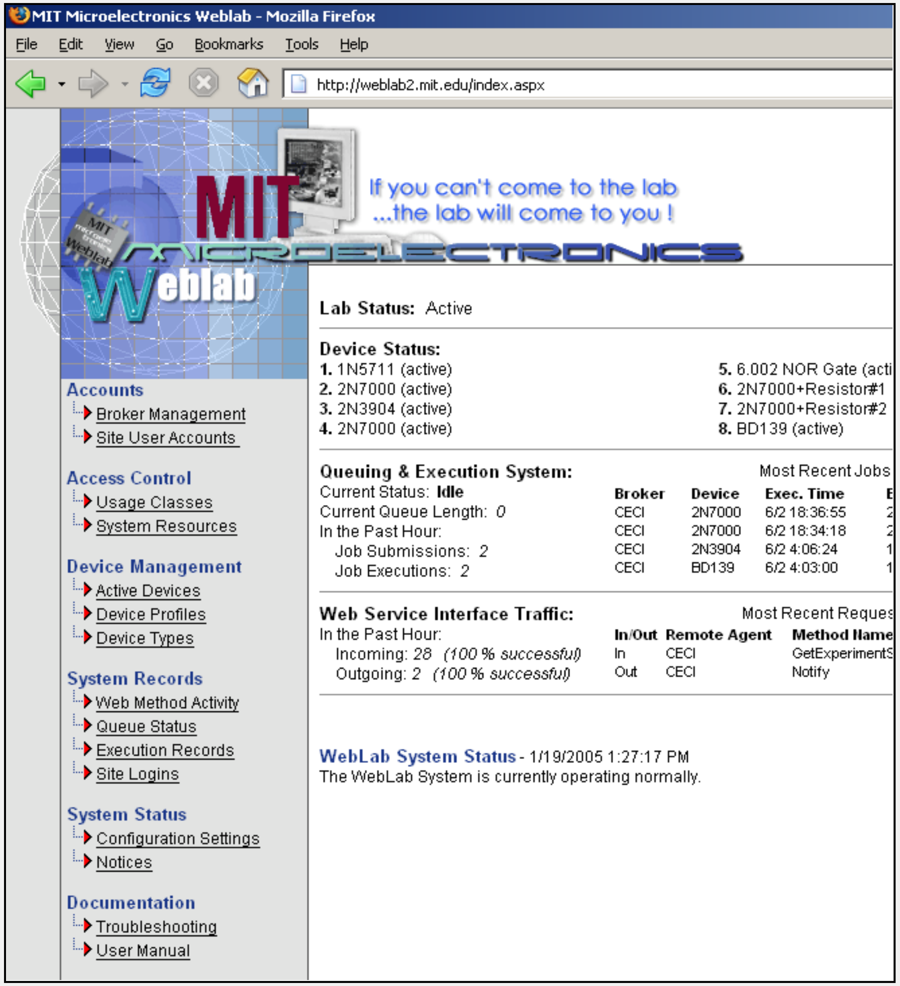
- Jobs are checked for:
  - Basic Correctness
  - Compliance with Hardware capabilities
  - Compliance with Server-imposed rules
- Reduces resources spent on incorrectly specified jobs.
- Server-based validation ensures uniformity, rapid application of changes

Lab Server Highlights: Lab Management

Center of Competence in Online Labs and Open Learning

Most Lab Management functions available online:

- Used to view system status/logs, edit system configuration
- Interface geared towards common functions
- Allows rapid response to events

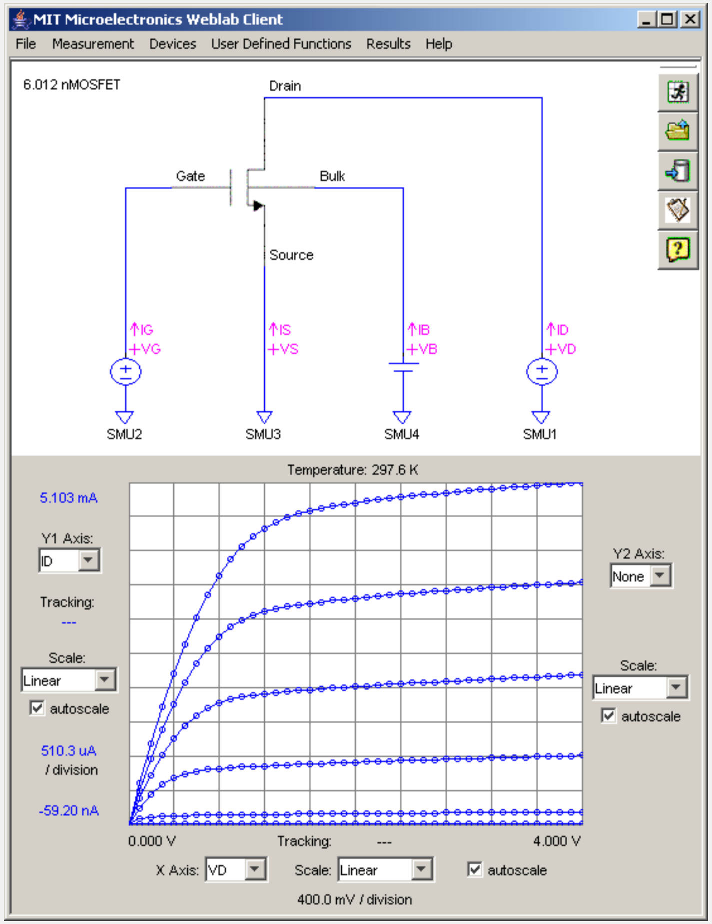


The µElectronic WebLab Client

Center of Competence in Online Labs and Open Learning

Client Requirements:

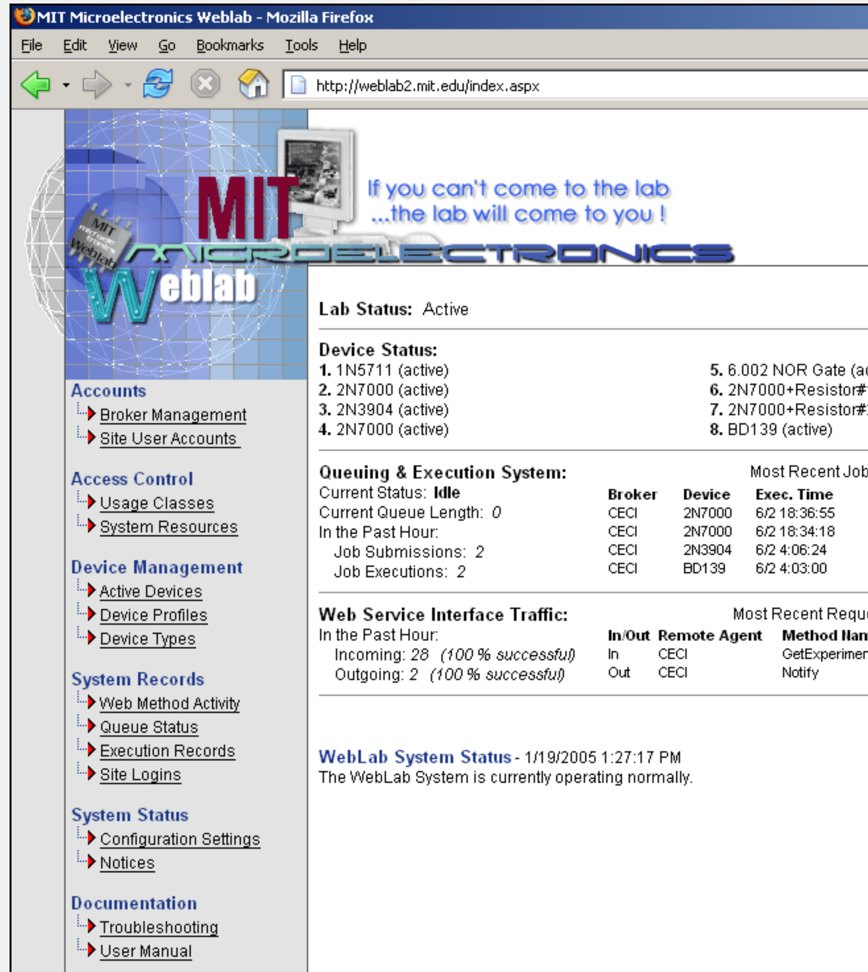
- Intuitive interface
- Easily deployed on many platforms
- Minimal user requirements
- Highly modular design
- Easily extensible



Picture from MIT/CECI

Most Lab Management functions available online:

- Used to view system status/logs, edit system configuration
- Interface geared towards common functions
- Allows rapid response to events



Meeting Client Design Goals: Portability

Java used to develop client.

- Often present as client execution environment
  - Good cross-platform compatibility
  - Places few special requirements on end-user
- Packages/toolkits provide necessary functionality
  - Graphical UI, Web Services, XML all within reach
- Versatility
  - Few constraints imposed by technology

Other Client Technology Options

Center of Competence in Online Labs and Open Learning

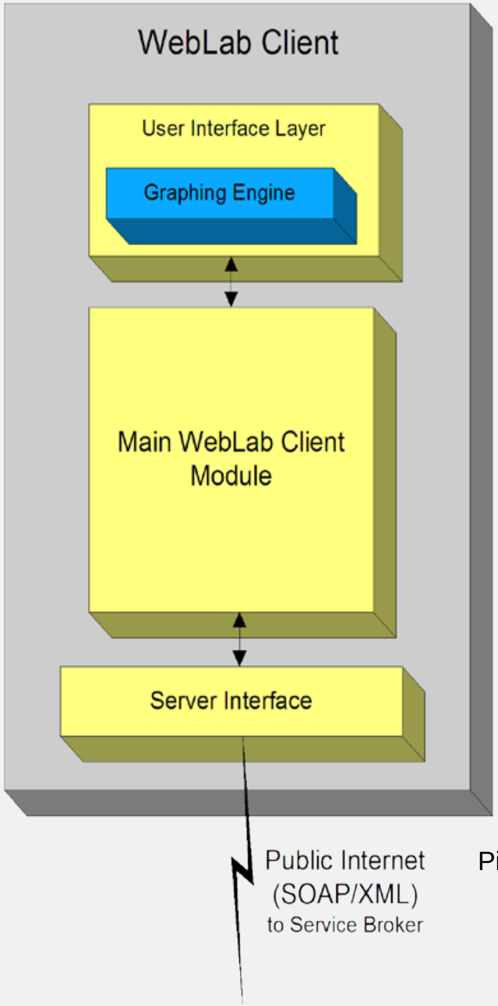
- Stand-alone application (.NET, Java, C/C++, etc.)
  - Versatile
  - Typically more platform dependent
  - User must download/install client
- HTML/Web Script based client (.NET, Java/JSP, PHP, etc.)
  - Typically more portable, easy to deploy
  - .NET WebForms are an attractive option
- Client development packages (LabView)
  - Rapid deployment, flexible interfaces
  - Traditionally hard to integrate with Batched-Lab Architecture
  - Potential to integrate LabView UI layer with .NET Server Interface

Client Design Goals: Modularity/Extensibility

Client built from three modules:

- User Interface Layer
  - Only presentation code
- Main Client Module
  - Contains core functionality
- Server Interface
  - Translates Core commands to Web Service Calls

Many changes can be isolated.



Picture from MIT/CECI

Reusability of Lab Code

Center of Competence in Online Labs and Open Learning

- Lab Client/Server code is lab-specific
  - Exception is Client graphing module
- However, some parts can be reused with modification
  - Client/Server – Broker Interfaces, some management tools, Execution queuing, Client/Server infrastructure…
- Deployed labs always valuable as working examples

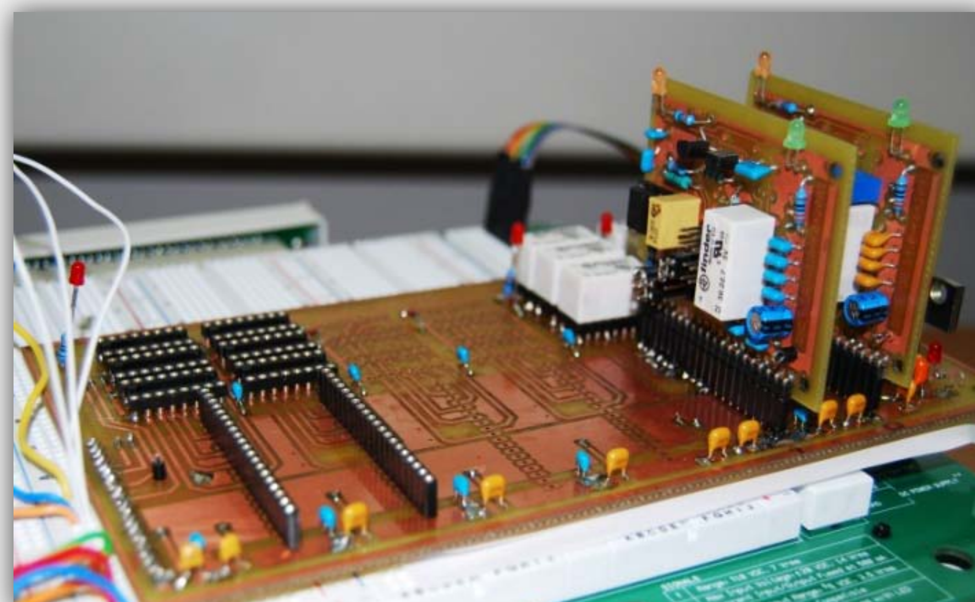
Reusability of Lab Code: Building new iLabs

New ilabs needed to expand into other electronics courses.

- …reuse as much lab code as possible
  - Build upon success of Microelectronics iLab
  - Deploy quick
- …take advantage of platforms like NI ELVIS and lower level LabVIEW functions (DAQ)

NI ELVIS

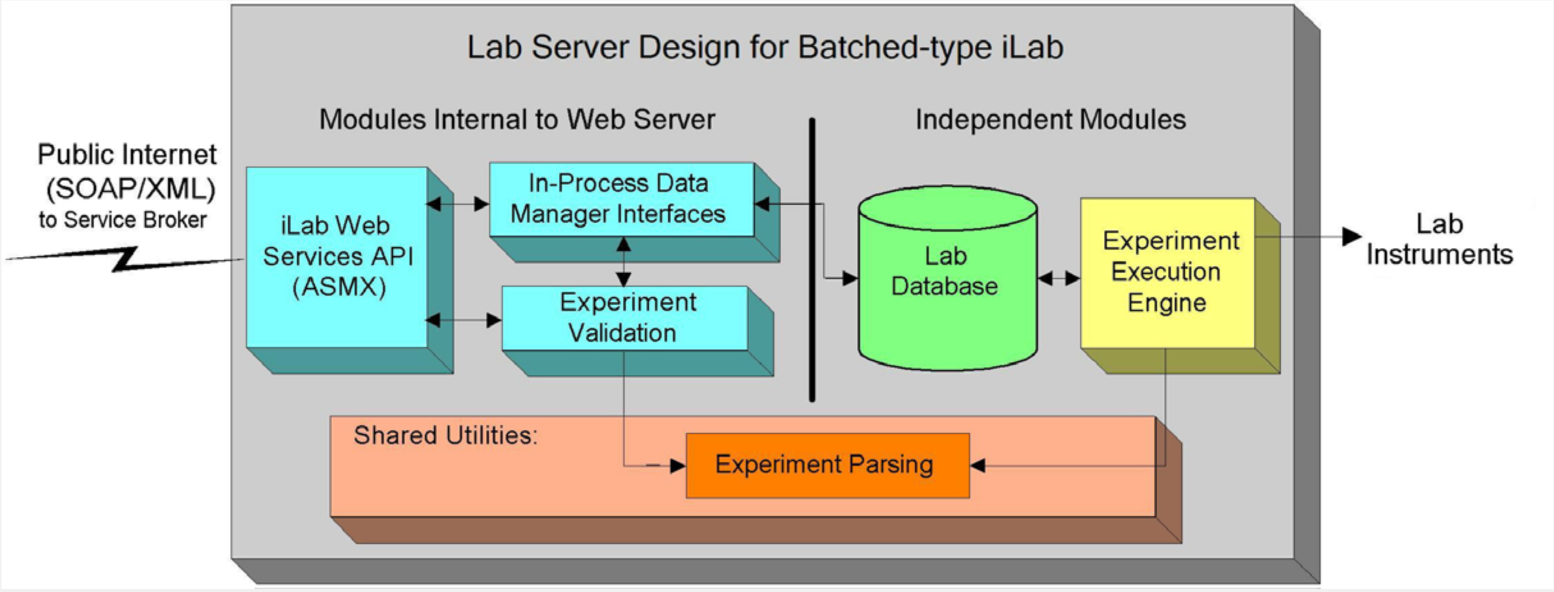
- All-in-one electronics workbench
- Performs variety of basic functions
- Readily software controllable (LabView)
- Compact
- Cost-effective

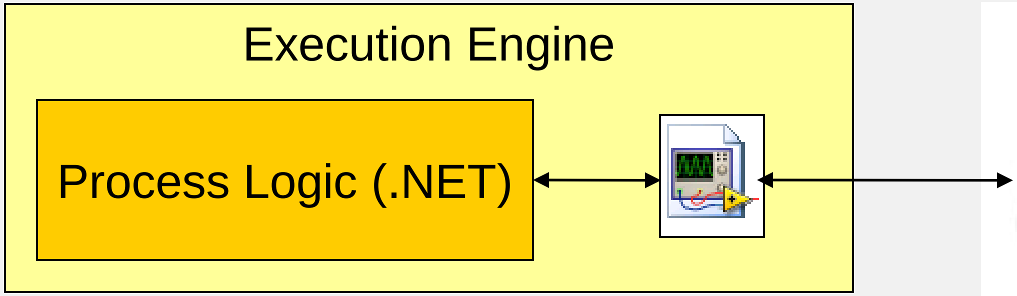


ELVIS-based iLabs: Version 1

Center of Competence in Online Labs and Open Learning

- ELVIS integrated into batched-lab architecture



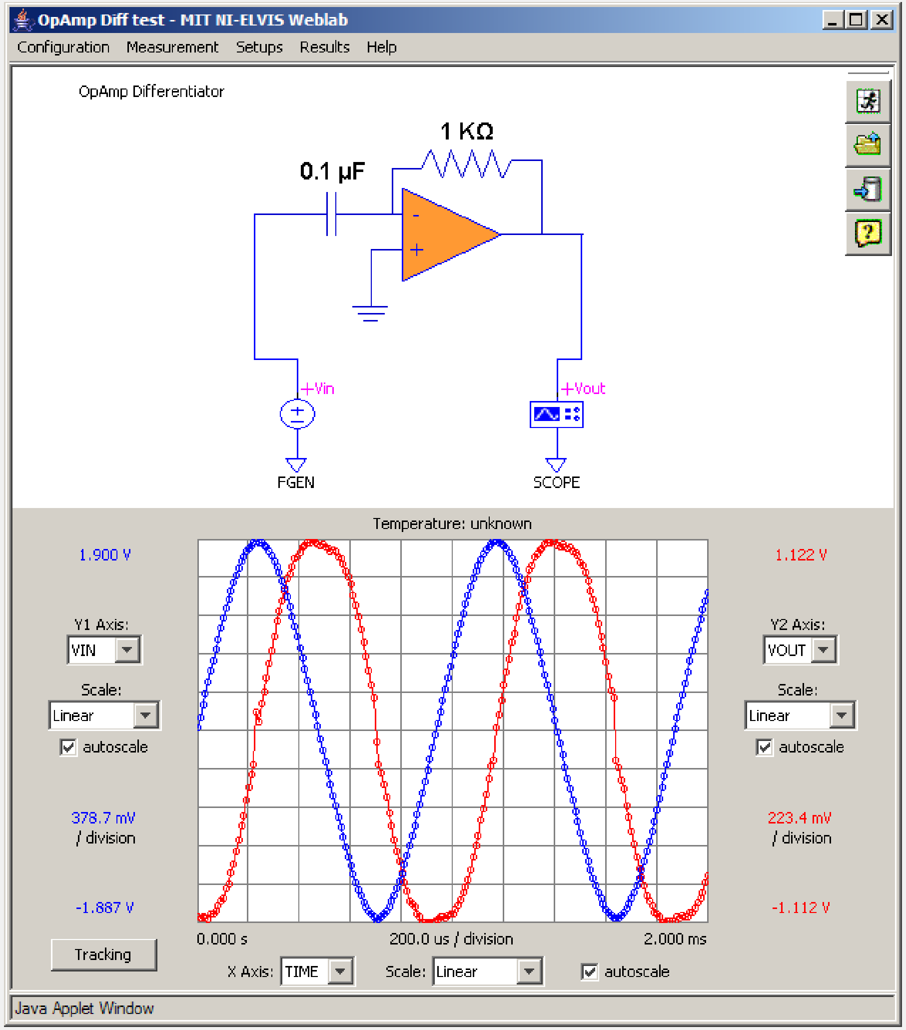


ELVIS-based iLabs: Version 1 (cont.)

Center of Competence in Online Labs and Open Learning

Lab Client very similar to that of the Microelectronics iLab

- UI elements are similar
  - Graphing engine, layout templates reused
  - Changes in parameter input controls
- Web Service Interface reused
- Main changes in Client Core
  - Interpreting new experiment parameters
  - Using a new Lab Client to Lab Server Communication format



Example of Lab Batched Lab

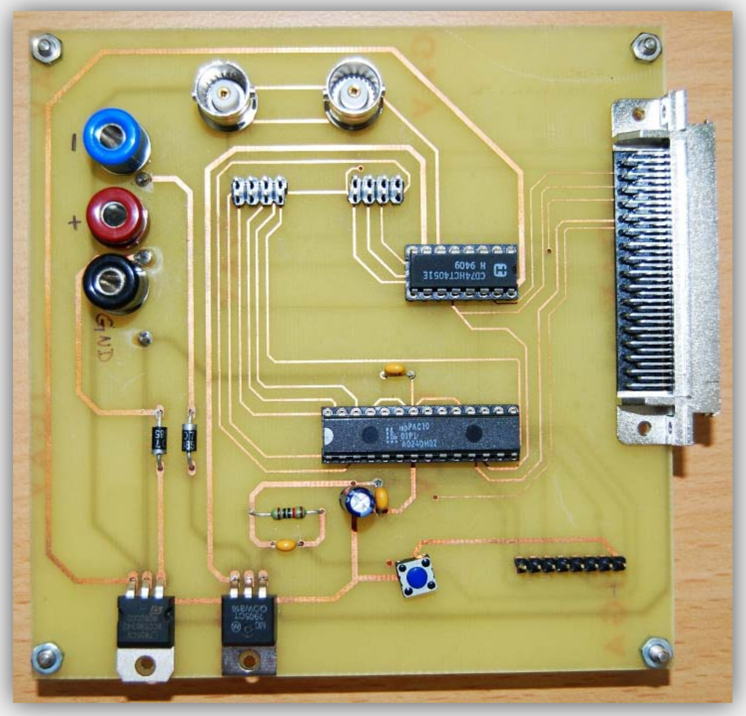
A Batched Lab Example at CUAS

The READ Hybrid Laboratory

READ – Remote ASIC Design and Test

- Allows for the realization of Electronics Experiments with an analogue programmable device (ispPAC10).
- A hybrid laboratory, allowing the design, simulation and test of real devices.

- Design and Simulations: PAC-Designer 5.0
- Test and Measurements: READ Lab Server via a Java Applet Client.
- Runs within the iLabs Shared Architecture (batched experiment)



READ Redesign – Batched versus Interactive

Center of Competence in Online Labs and Open Learning

The Decision for the Batched Architecture

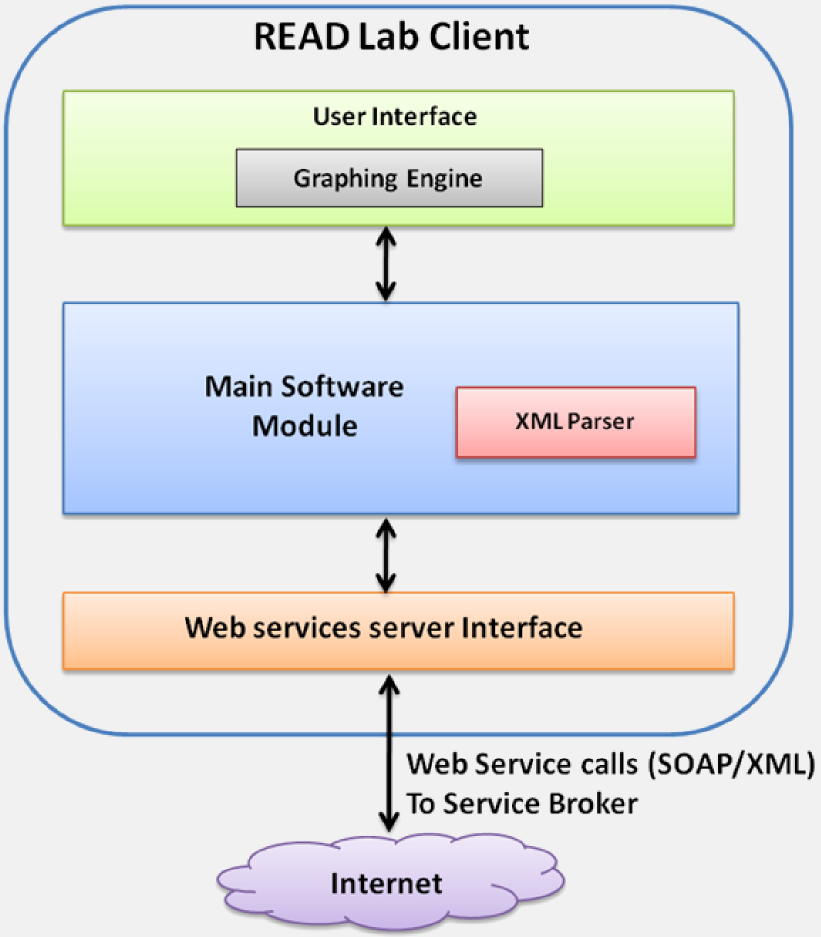
- Circuit under test is kept in an idle state during great part of the execution cycle.
- Take advantage of the queuing mechanism of previous lab servers
- Low amount of data is exchanged during each experiment execution

The READ Lab Client (1)

Center of Competence in Online Labs and Open Learning

Client Functionalities:

- Provide the lab Graphical User Interfaces
- Include pedagogical aspects
- Implement the Web Services interface to communicate with the Service Broker
- Create experiment specification protocols
- Parse experiment results received from the server



The READ Lab Client (2)

Center of Competence in Online Labs and Open Learning

The Web Services Interface

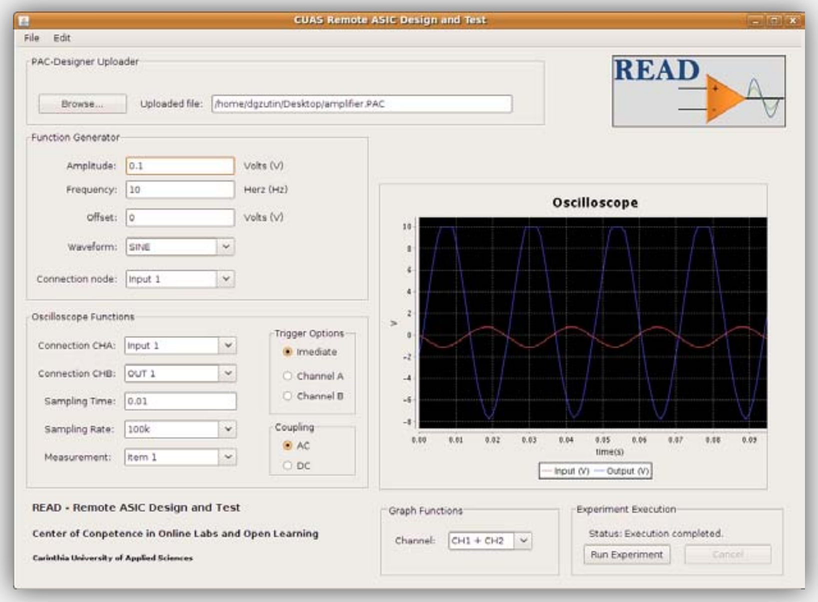
- Translate internal method calls do Web service calls
- Manages full cycle of an experiment execution

Main Client Module

- Create experiment specification
- Parse experiment results
- Process the data (if necessary)

The User Interface

- Provide the lab Graphical User Interfaces
- Display the results with graphing functions



The READ Lab Client (3)

Center of Competence in Online Labs and Open Learning

The XML Experiment Specification and Results


```
<?xml version="1.0" encoding="utf-8" standalone="no" ?><!DOCTYPE experimentSpecification SYSTEM
"http://ilabs.cti.ac.at/xml/experimentSpecification.dtd">
    <experimentSpecification lab="CUAS READ Lab" specversion="0.1">"
    <PACString>PAC-Designer PAC String</PACString>
    <terminal instrumentType="FGEN" instrumentNumber="1">
    <vname download="true">Vin</vname>
    <function type="WAVEFORM">
    <waveformType>SINE</waveformType>
    <frequency>1000</frequency>
    <amplitude>0.5</amplitude>
    <offset>2.5</offset>
    <connInput>1</connInput>
    </function>
    </terminal>
    <terminal instrumentType="SCOPE" instrumentNumber="2">
    <vname download="true">Vout</vname>
    <function type="SAMPLING">
    <samplingRate>500000</samplingRate>
    <samplingTime>0.02</samplingTime>
    <connProbe_CHA>1</connProbe_CHA>
    <connProbe_CHB>1</connProbe_CHB>
    <coupling>0</coupling>
    <triggerSource>1</triggerSource>
    </function></terminal>
    </experimentSpecification>
```

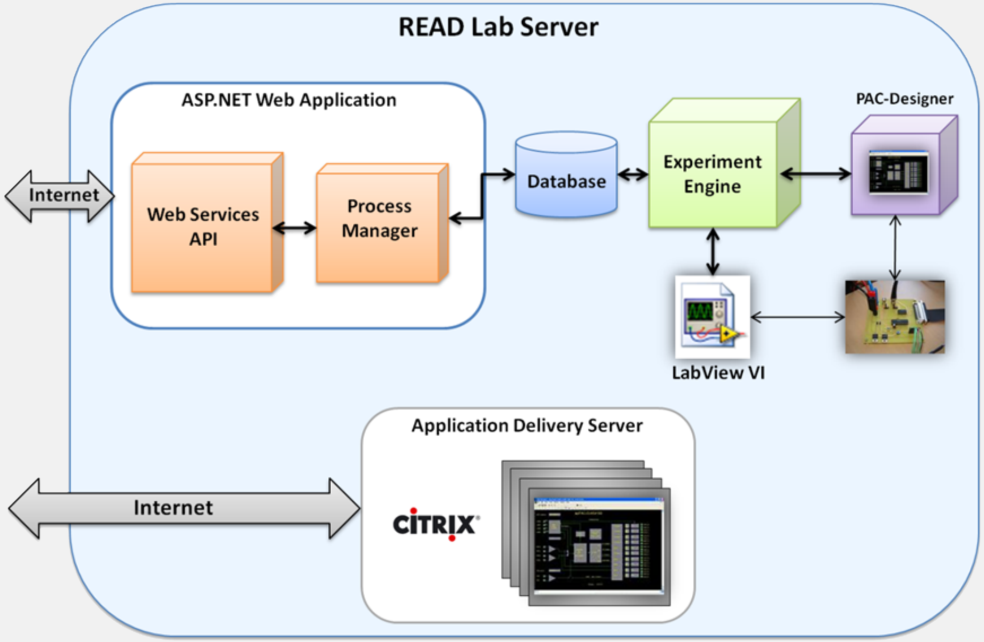
```
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE experimentResult SYSTEM 'http://exp04.cti.ac.at/elvis/xml/experimentResult.dtd'>
  <experimentResult lab="CUAS READ Lab" specversion="0.1">
  <datavector name="TIME" units="s">0 1E-05 2E-05 3E-05 4E-05</datavector>
  <datavector name="VIN" units="V">0.402830402191198 0.569602385435954 0.722355996130949
0.850456447078121</datavector>
  <datavector name="VOUT" units="V">2.50433404263296 2.5049785833044 2.50401177229907 2.5049785833044
2.50481744813608</datavector>
</experimentResult>
```

The READ Lab Server (1)

Center of Competence in Online Labs and Open Learning

Server Functionalities:

- Implement the Web Services interface to communicate with the Service Broker
- Queue experiments for execution
- Parse experiment specification protocols and perform validation
- Create experiment results received from the server
- Provide interface to lab hardware
- Assure the correct circuit is measured



The READ Lab Server (3)

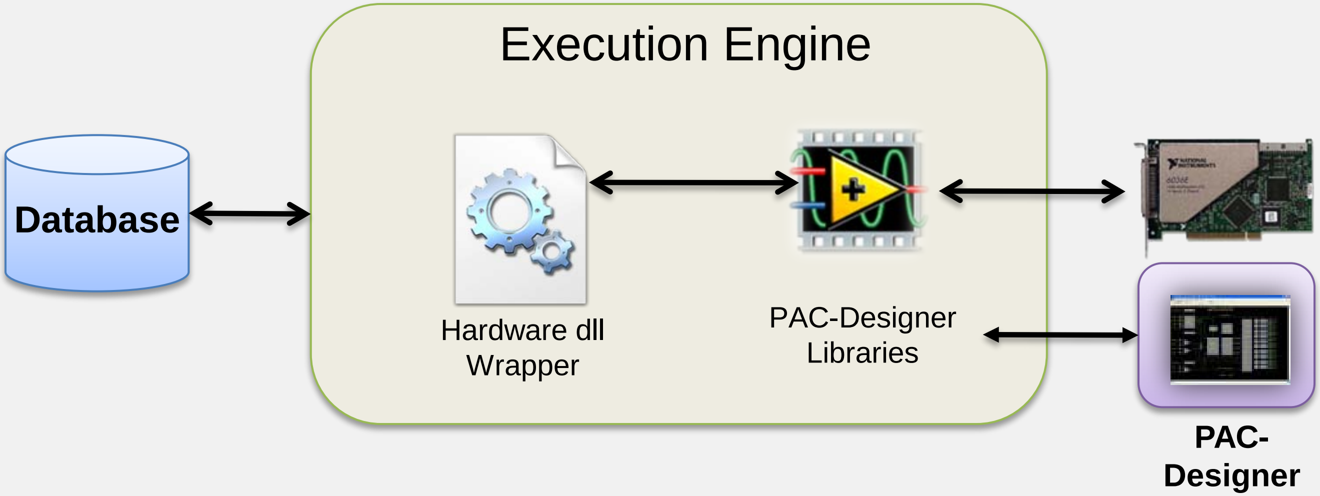
Center of Competence in Online Labs and Open Learning

The Web Server and Services Interface

- Exposes Web methods to be called by the Service Broker.
- Validate experiments
- Queues experiment requests to be executed by writing them into the database

The Experiment Execution Engine (1)

- Communicates with the low level libraries that control the Laboratory Hardware
- Parses the experiment specification
- Dequeues experiments, executes them and writes the results back into the database



The READ Lab Server (4)

Lab Hardware Control with LabVIEW

- DAQ NI PCI-6251 (DAQmx Library of Vis)
- Original virtual instruments could be kept with minor changes (function generator and oscilloscope)
- Virtual instruments run in a straightforward fashion

Receive parameters from wrapper → **Start signal generation** → **Acquire finite number of samples** → **Stop signal generation** → **Return measurements to wrapper**

- Compiled as a *DLL* to be called from the experiment engine

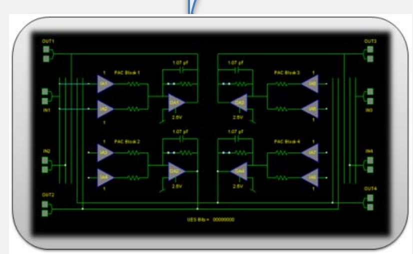
Center of Competence in Online Labs and Open Learning

The ispPAC Uploader Module

- Ensures that the desired circuit is being tested.
- Extra module added to the experiment engine
- Developed with the *PAC-Designer Software Development Kit*

- .PAC files are XML based describing simulation parameters and Information for the JTAG interface.
- The client:

 - Reads the .PAC file
 - Wraps it inside the Experiment Specification XML string
 - Sends it to the Lab Server

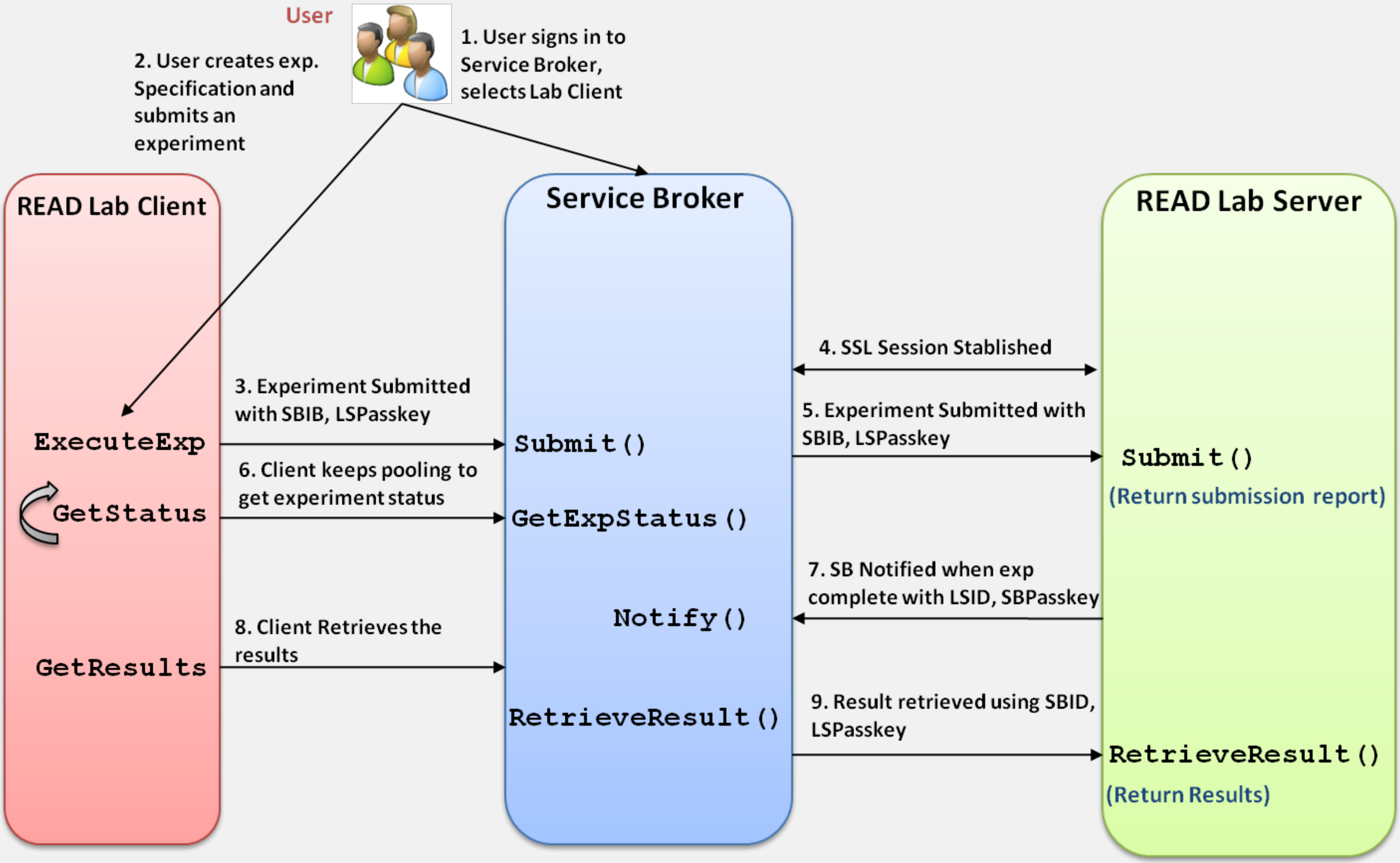



```
<?xml version="1.0" ?>
- <PacDesignData>
  <DocFmtVersion>1</DocFmtVersion>
  <DeviceType>ispPAC10</DeviceType>

    <FuseMap>001110000000001000000111010000000000000000000000001
- <SummaryInformation>
    <Title>Voltage Gain = 1/4 for PAC10</Title>
    <Subject>Circuits Library</Subject>
    <Author>Lattice PAC Applications Engineering</Author>
    <Keywords>ispPAC10, gain, attenuation</Keywords>
  - <Comments>
      <![CDATA[ Attenuation mode with gain of 0.25. To prevent
      oscillation, additional feedback capacitance must be
      added for stability (-3dB=600kHz). Input is IN1 and
      output is OUT1. ]]>
    </Comments>
  </SummaryInformation>
- <SimulationSetup>
  - <Curve>
      <CurveNum>0</CurveNum>
      <InputNode>1</InputNode>
      <OutputNode>1</OutputNode>
      <ConfigAB>0</ConfigAB>
      <StartFrequency>10</StartFrequency>
      <StopFrequency>10000000</StopFrequency>
      <PointsPerDecade>500</PointsPerDecade>
    </Curve>
  - <Curve>
```

An Experiment Execution Scenario

Center of Competence in Online Labs and Open Learning



Conclusion and Considerations

- With iLabs it was achieved a fully multiple user system
- iLabs facilitate sharing this labs and managing its users
- Works behind proxies servers and firewalls
- ISA-compliant laboratories
- Considerations on the migration of existing labs to ISA

Future work:

- More interactive graph display, supporting zoom functions
- Extract measurements out of the data set
- Include possibility to perform measurements on frequency domain

Thank you

Thank you for your attention!

References

- MIT, iLab: A Scalable Architecture for Sharing Online Experiments, ICEE2004.
- MIT, The Challenge of Building Internet Accessible Labs.
- MIT, Client to Service Broker API.
- Hardison/MIT, iLab Batched Experiment Architecture: Client and Lab Server Design, ppt slides.