Spring School Ilmenau

FACHHOCHSCHULE KÄRNTEN

Center of Competence in Online Labs and Open Learning

Carinthia University of Applied Sciences
Danilo G. Zutin

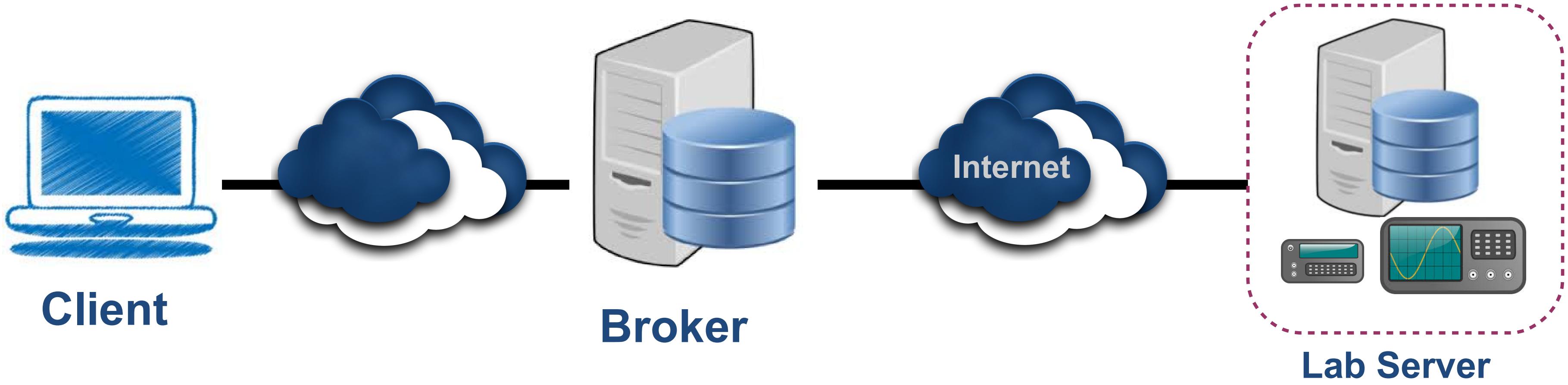
Developing Online Labs Compliant with ISA

Note: Some of the figures shown were taken from presentation slides and/or papers from MIT/CECI. Figures without any notes are own. Also some text excerpts were taken from the references listed in the last slide.

CC BY SA

Client and Lab Server Design

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

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

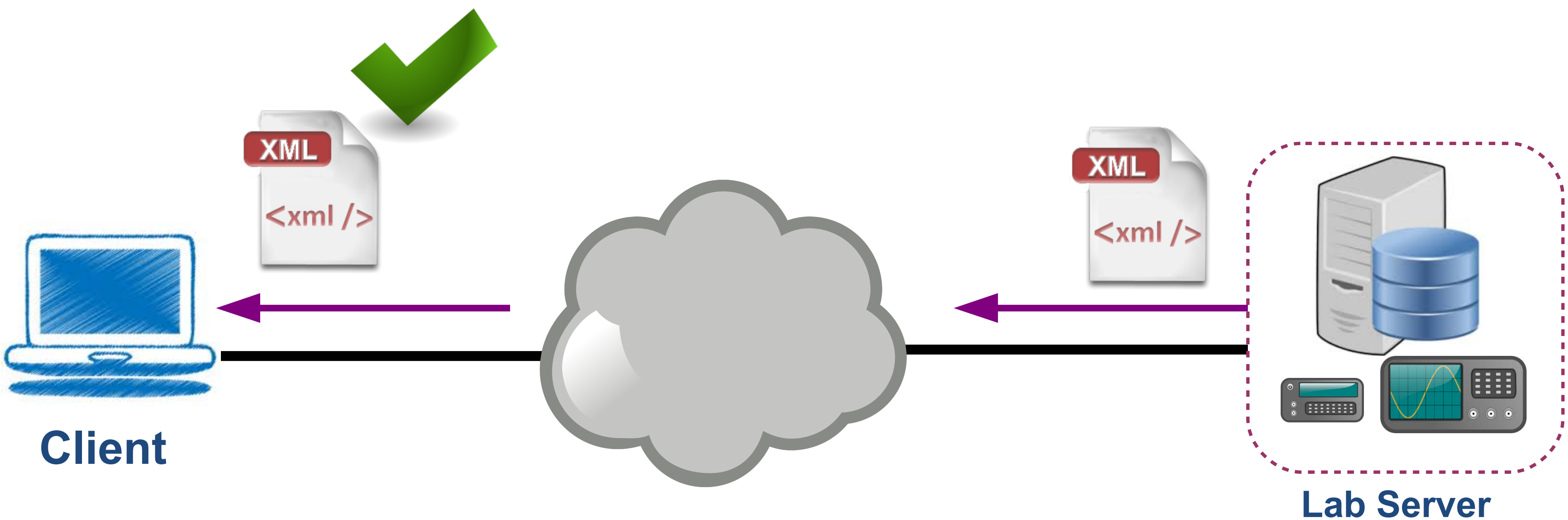
- Specification of batched parameters and results (processed only by Lab server and lab client)
- Definition of messages passed between server and client

Lab Client–Server Communication

Messages passed between Client and Lab Server communicate key lab information.

- Lab Hardware Configuration/Status
- Experiment Parameters & Results

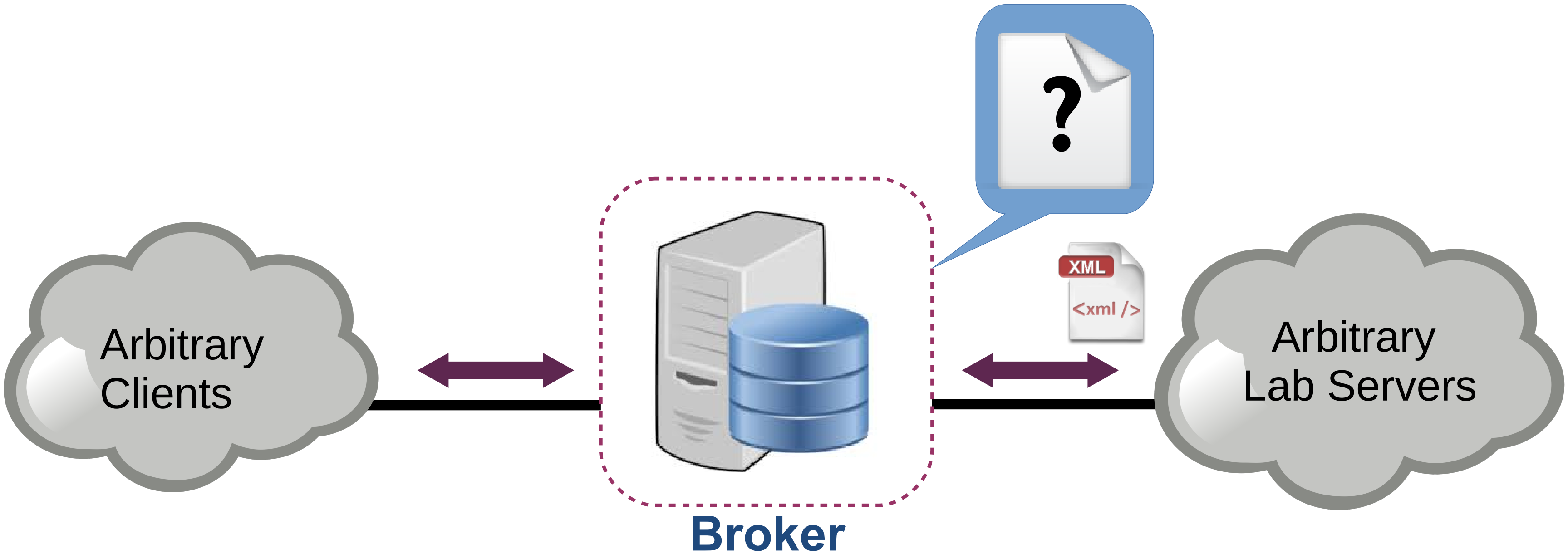
This information is necessarily lab-specific.



Server–Client communication framework

- All Lab Client-Server Messages must be passed through Service Broker.
  Generic mechanism.

XML and JSON are ideal technologies for this application.



Lab Server Design

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.

Lab Client Design: Basic requirements

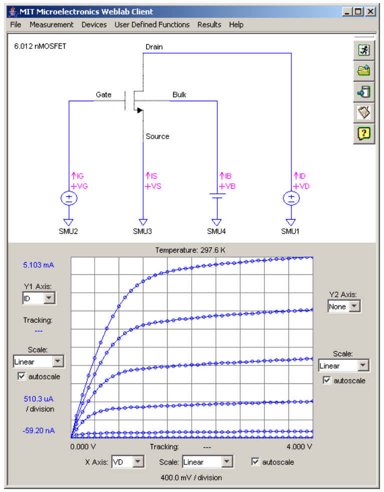
Basic requirements:

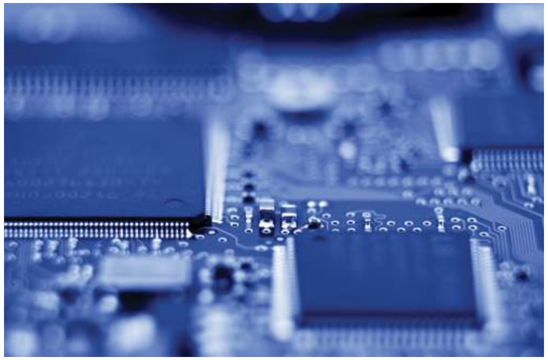
- Provide an educationally valuable user interface to the lab, embody pedagogical aspects
- Implement the iLab Client-Service Broker API
- Create & Interpret lab-specific communication messages with Lab Server

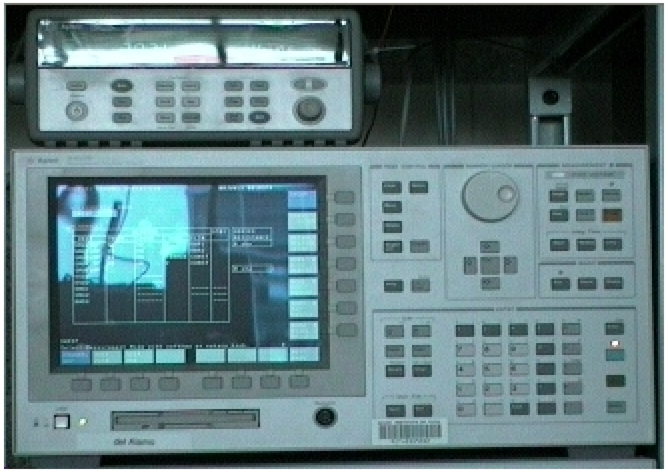
Again… iLab Architecture APIs are platform-neutral. Lab developer can select the best technology for their Client.

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.







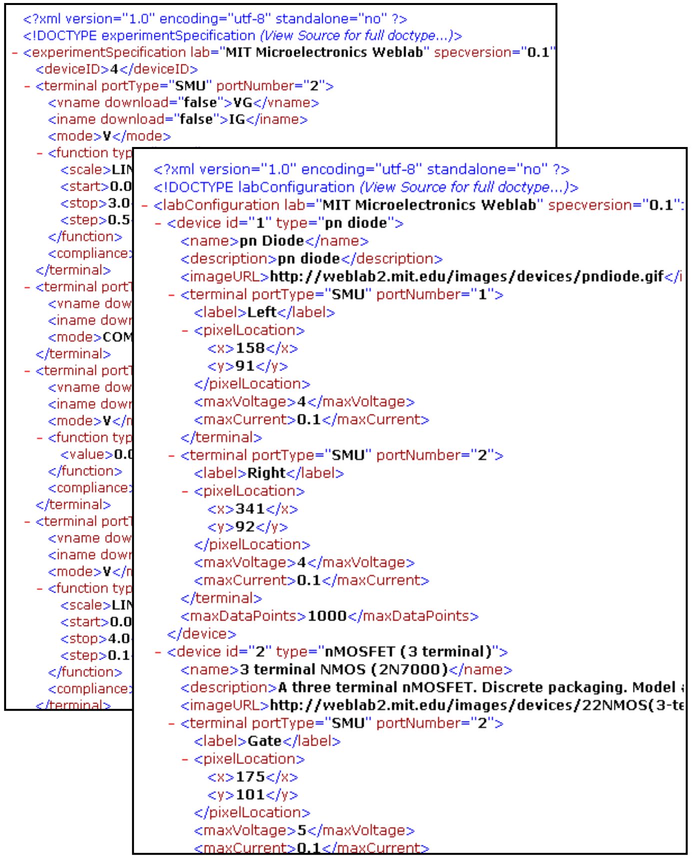
Semiconductor Parameter Analyzer

Tempus DesIRE

Center of Competence in Online Labs and Open Learning

Lab Server Development Examples

- 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.

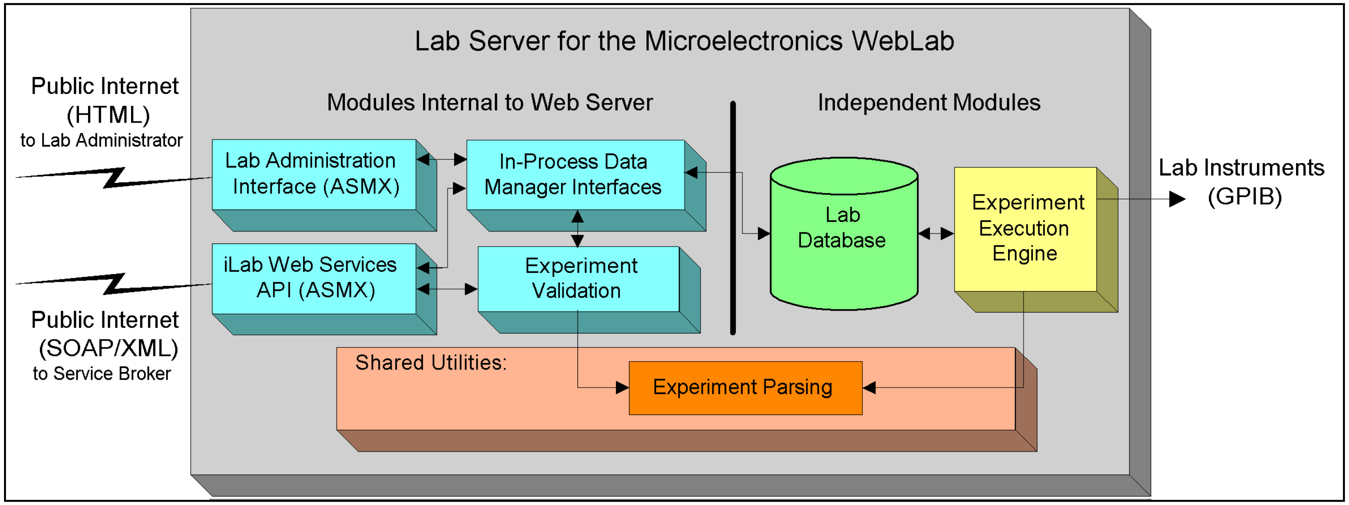


MIT Microelectronics iLab Lab Server

Lab Server Requirements:

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

The Lab Server



Picture from MIT/CECI

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

The Lab Server: Experiment Validation

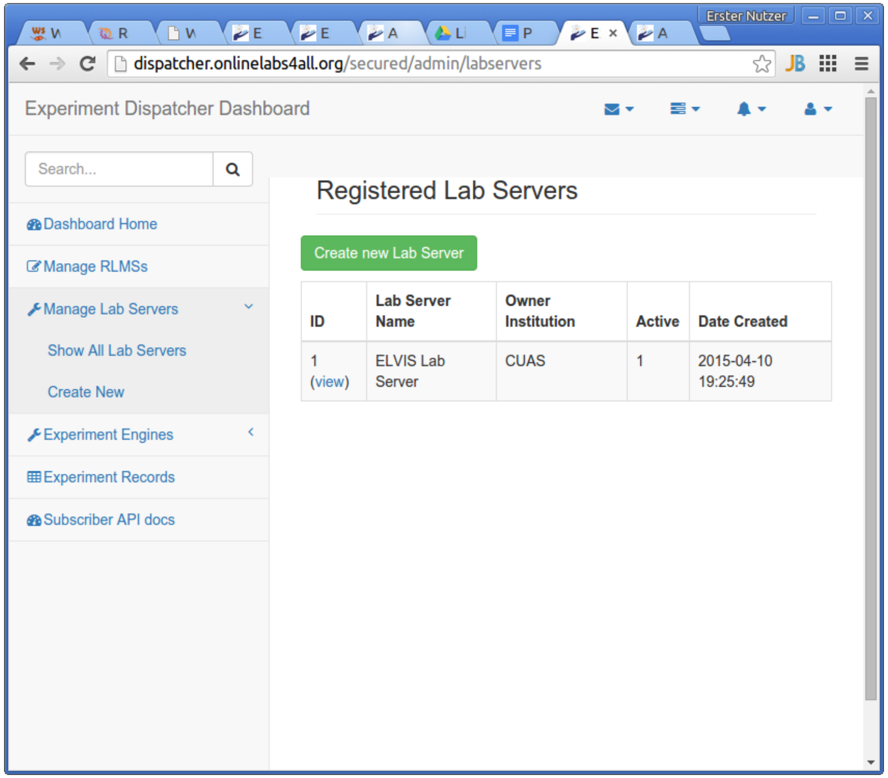
All experiments are validated on the server before they are queued:

- 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

Most Lab Management functions available online:

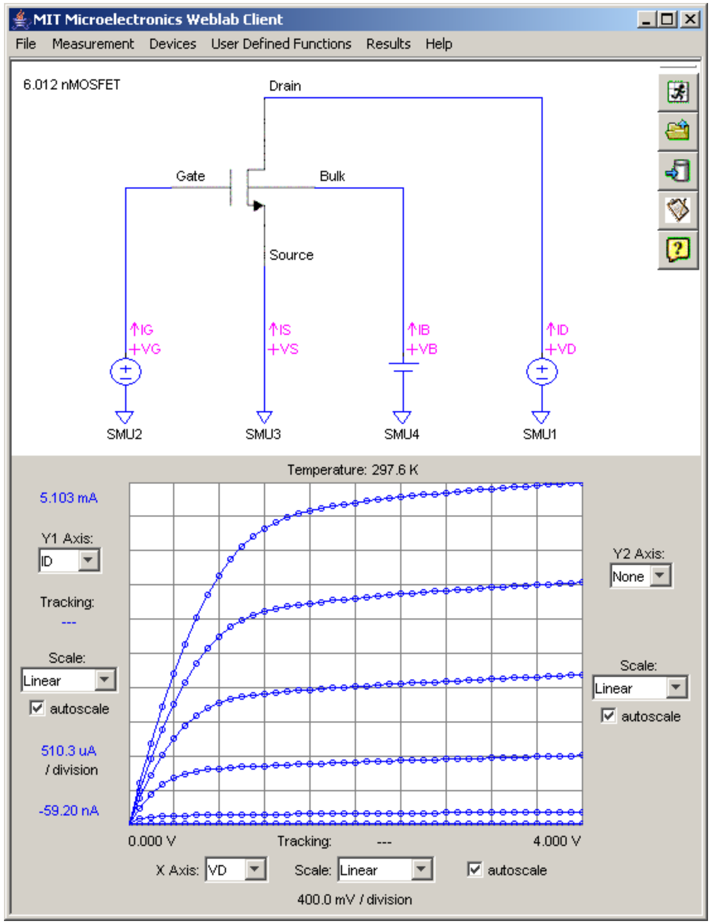
- Used to view system status/logs, edit system configuration
- Interface geared towards common functions
- Allows rapid response to events



The µElectronic WebLab Client

Client Requirements:

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



Picture from MIT/CECI

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

- 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, node.js 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.

Reusability of Lab Code

- 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

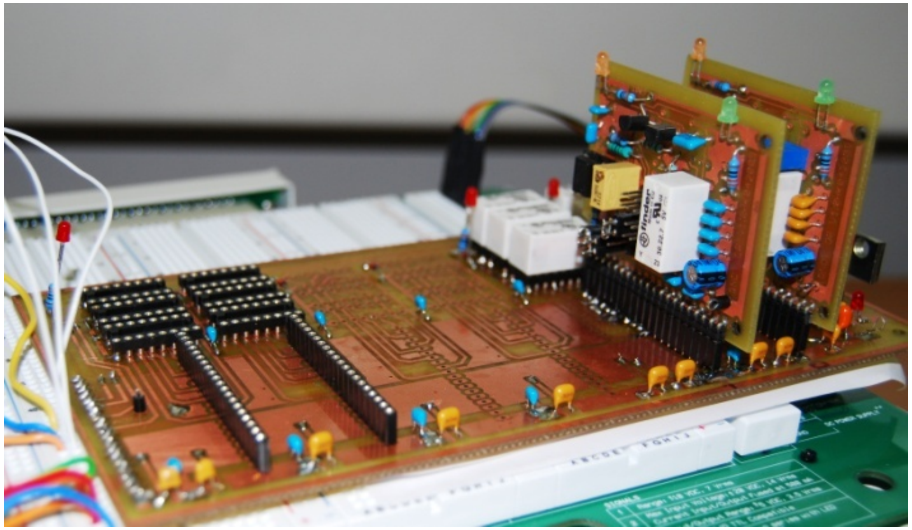
Reusability of Lab Code: Building new iLabs

New ilabs needed to expand into other electronics courses.

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

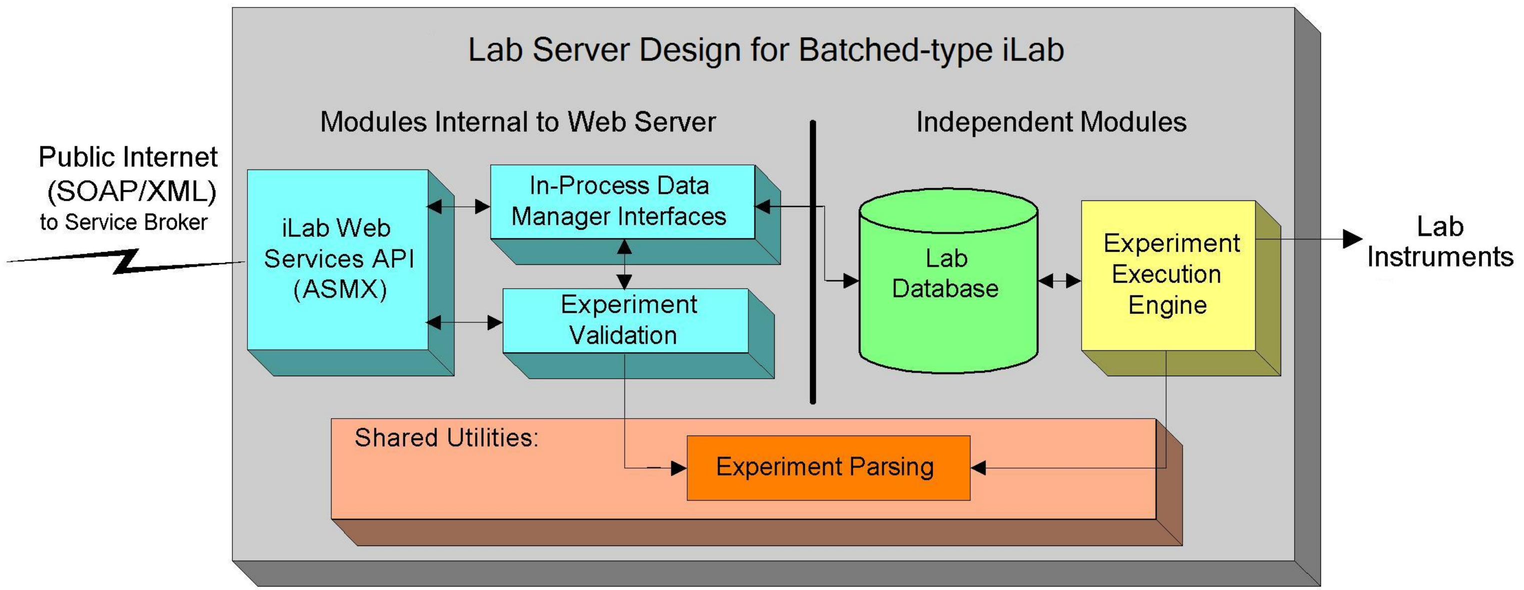
NI ELVIS

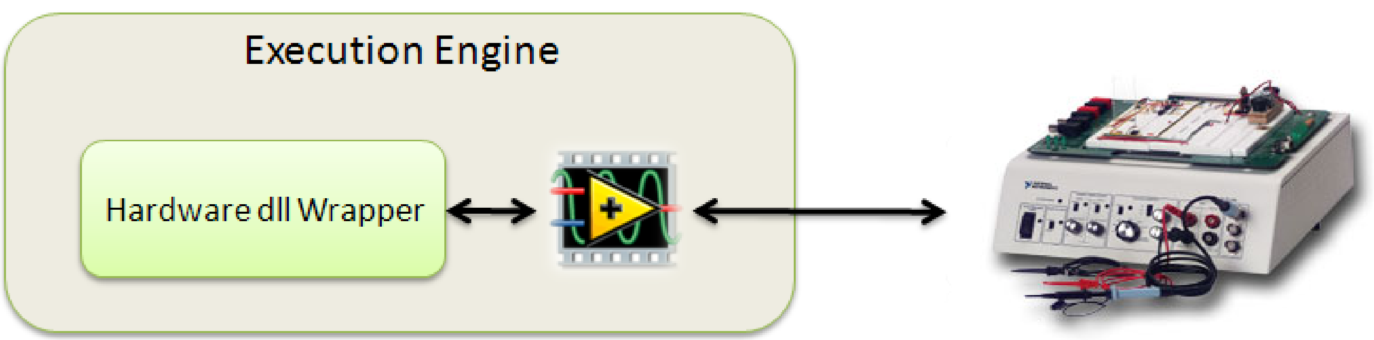
- All-in-one electronics workbench
- Performs variety of basic functions
- Readily software controllable (LabView)
- Compact
- Cost-effective



ELVIS-based iLabs: Version 1

- ELVIS integrated into batched-lab architecture

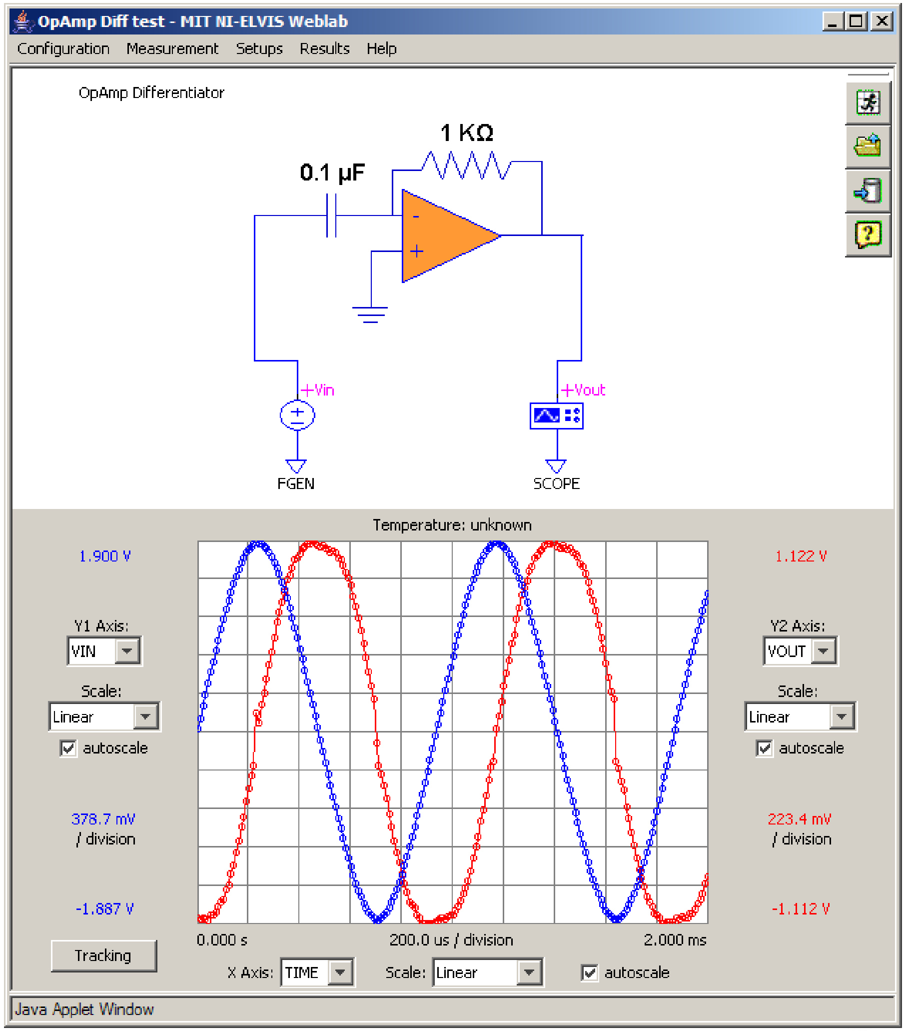




ELVIS-based iLabs: Version 1 (cont.)

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



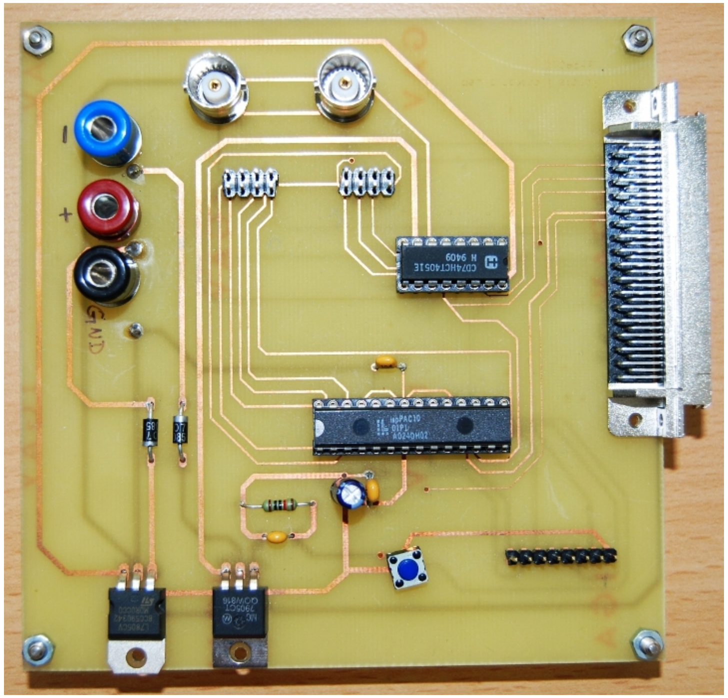
Example of Lab Batched Lab

Another Batched Lab Example

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

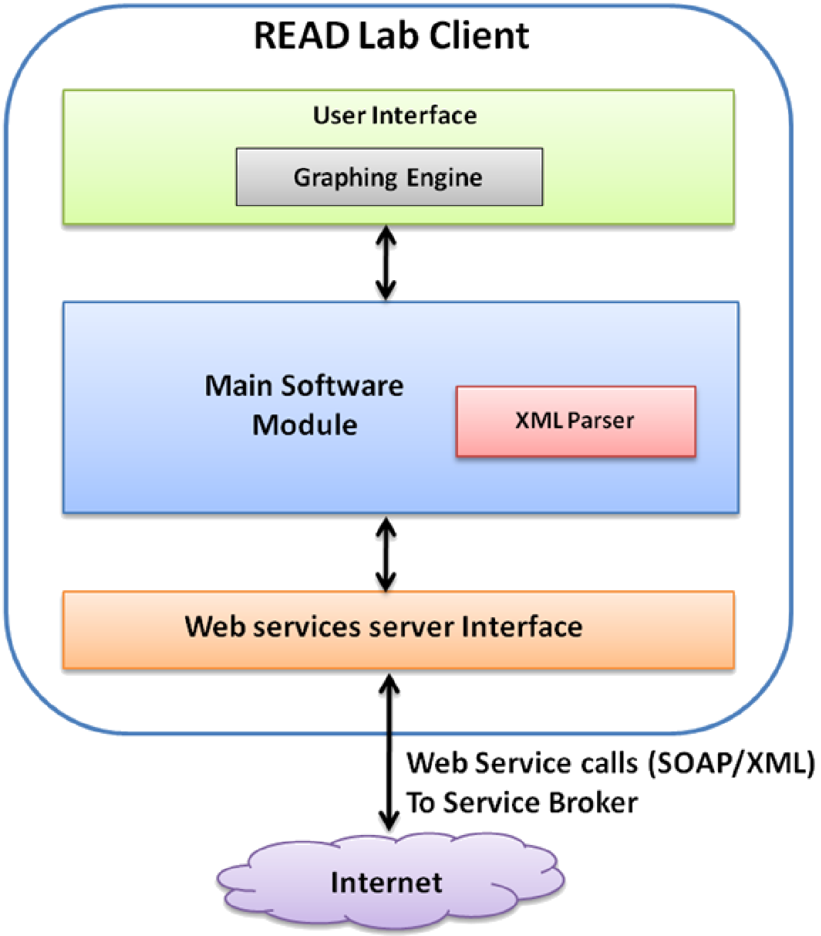
The Decision for the Batched Architecture

- 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)

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)

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

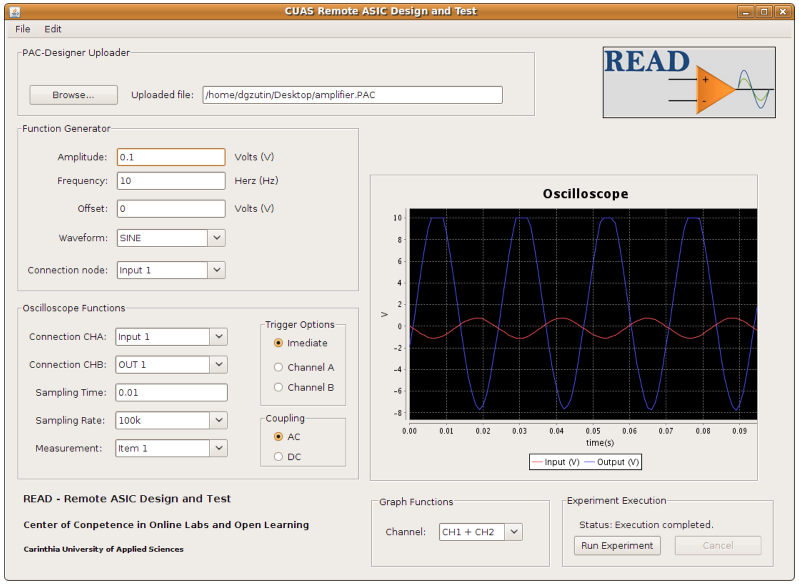
Parse experiment results

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)

The XML Experiment Specification and Results


```xml
<?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>
```

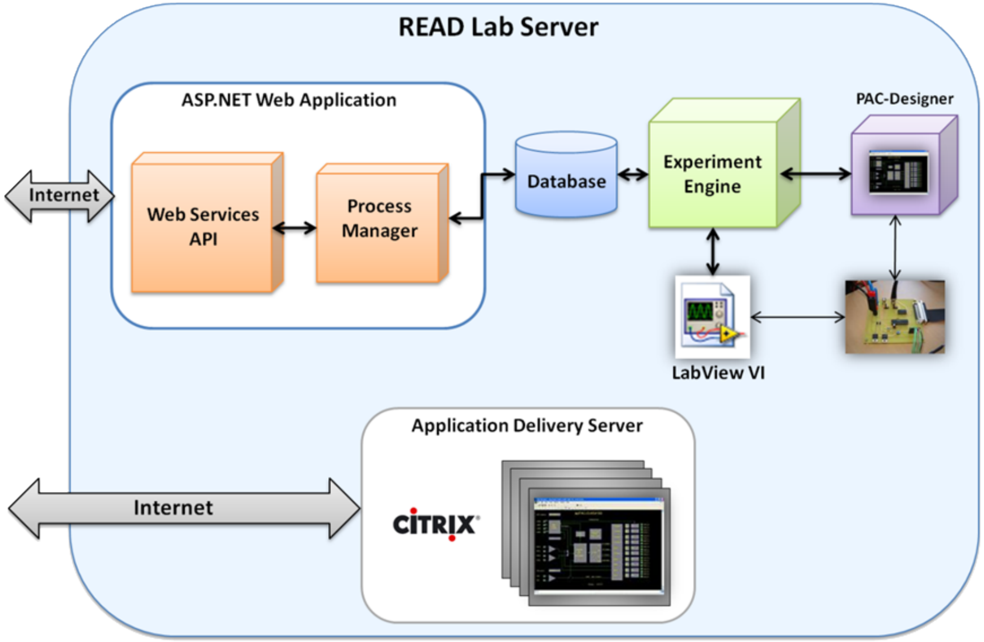


```xml
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE experimentResult SYSTEM 'http://exp04.cti.ac.at/elvis/xml/experimentResult.dtd'>
  <experimentResult lab="CUAS READ Lab" specversion="0.1">
  <datavector name="TIME" units="s">0 1E-05 2E-05 3E-05 4E-05</datavector>
  <datavector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

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



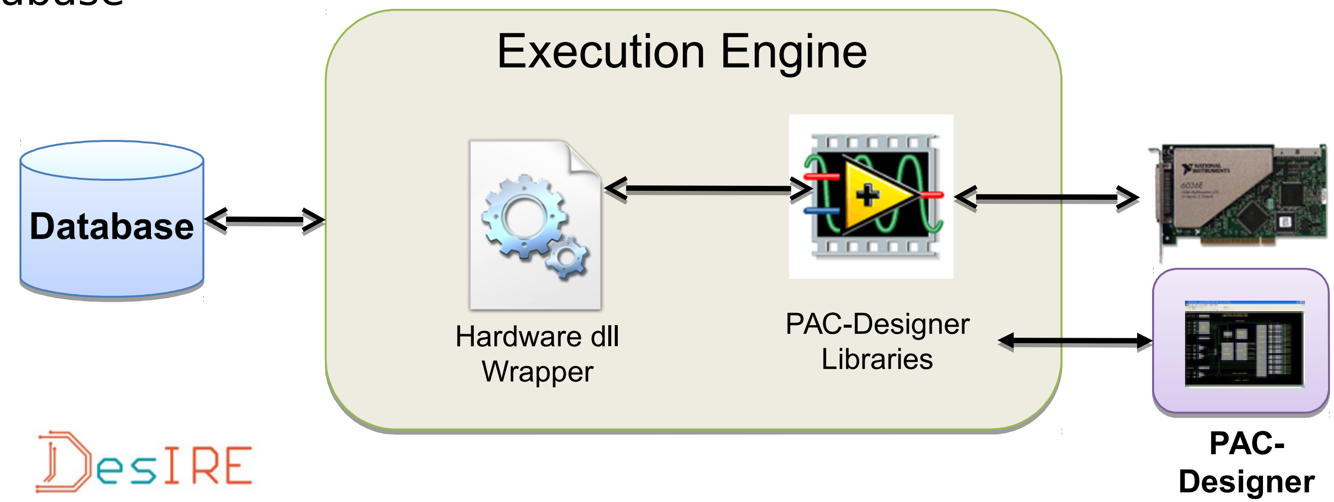
The READ Lab Server (3)

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



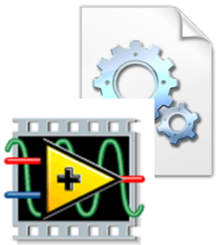
Tempus
DesIRE
Center of Competence in Online Labs and Open Learning

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

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



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*

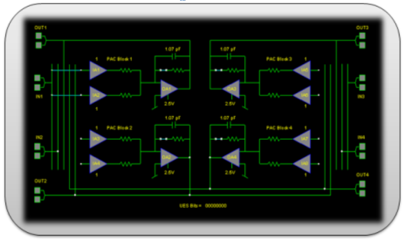
- .PAC files are XML based describing simulation parameters and Information for the JTAG interface.

  The client:

 - Reads the .PAC file

 - Wraps it inside the Experiment Specification XML string

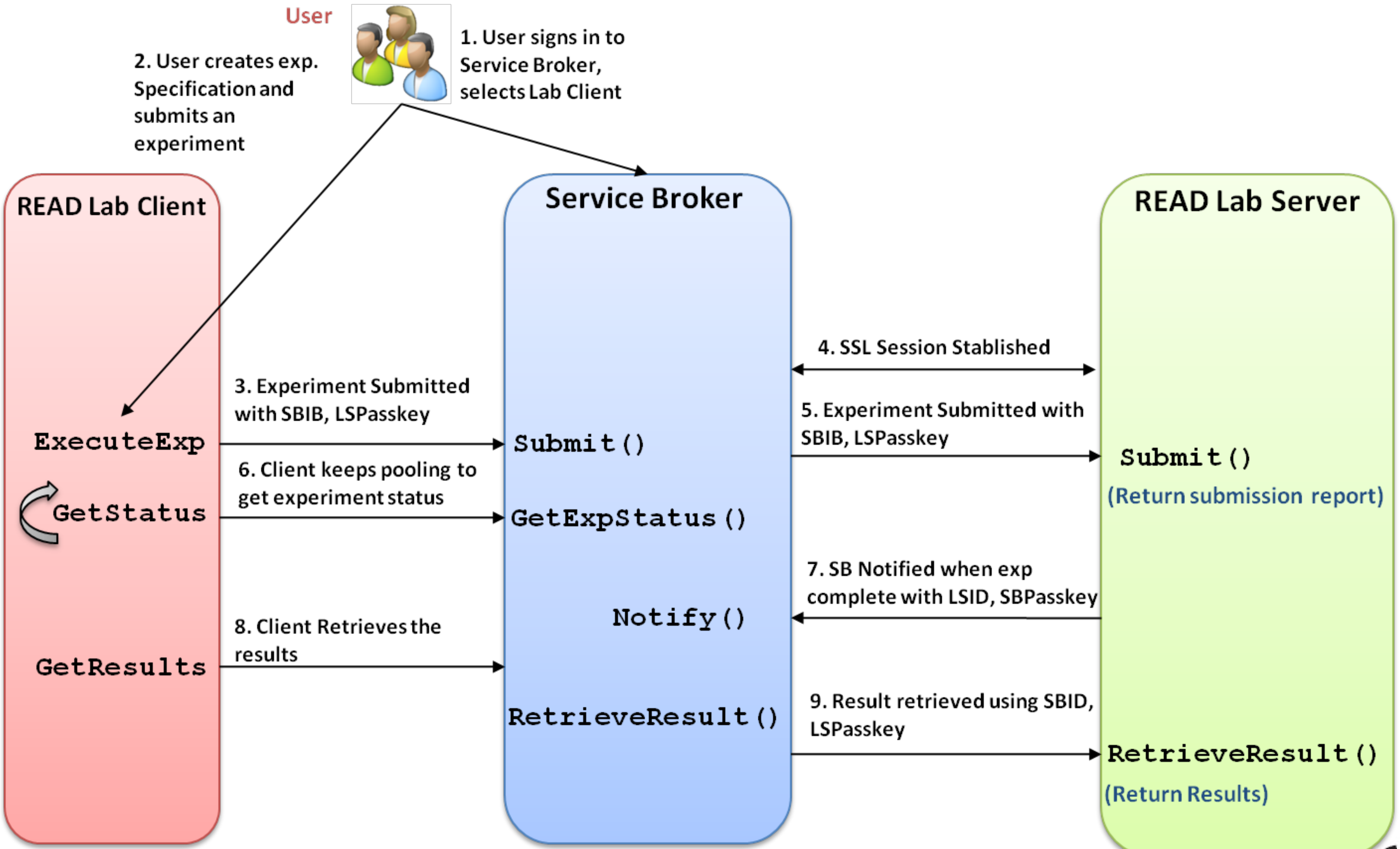
- Sends it to the Lab Server



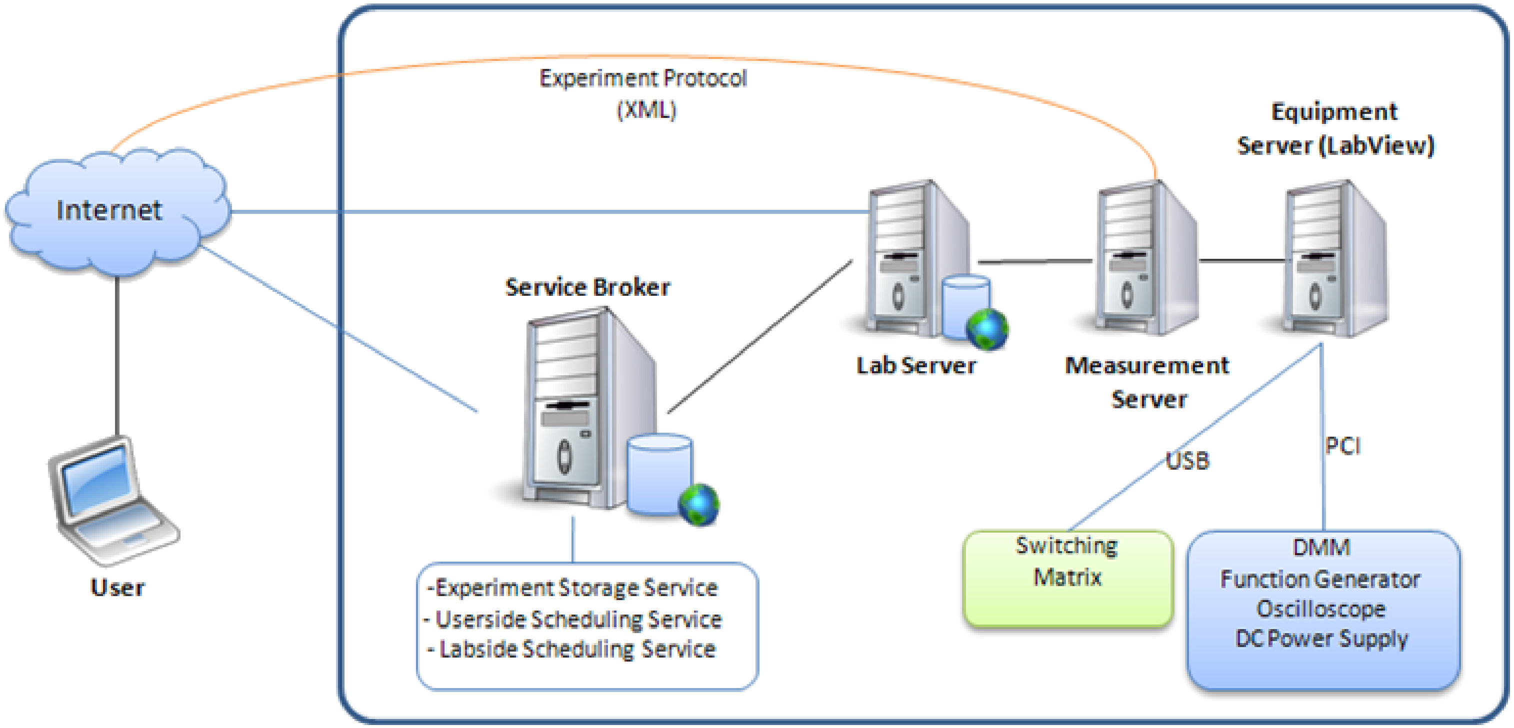

```
<?xml version="1.0" ?>
- <PacDesignData>
  <DocFmtVersion>1</DocFmtVersion>
  <DeviceType>ispPAC10</DeviceType>

    <FuseMap>001110000000001000001110100000000000000000000001
  - <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



Interactive Lab Server/Client Design



Interactive Lab Server/Client Design

- Implement the ISA Interactive Lab Server Services
- User launches the lab and is redirected to the lab client
- Service broker forwards the credentials to the Lab Server
- Lab Server uses the credentials to validate the ticket and check if user is authorized to carry out experiments
- If user is Authorized, Lab Server launches the client
- Lab Server and client should also implement mechanism to close the connection if session expires

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

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.

Thank you for your attention!