Tempus DesIRE

Spring School Ilmenau

FACHHOCHSCHULE KÄRNTEN

Center of Competence in Online Labs and Open Learning OL²

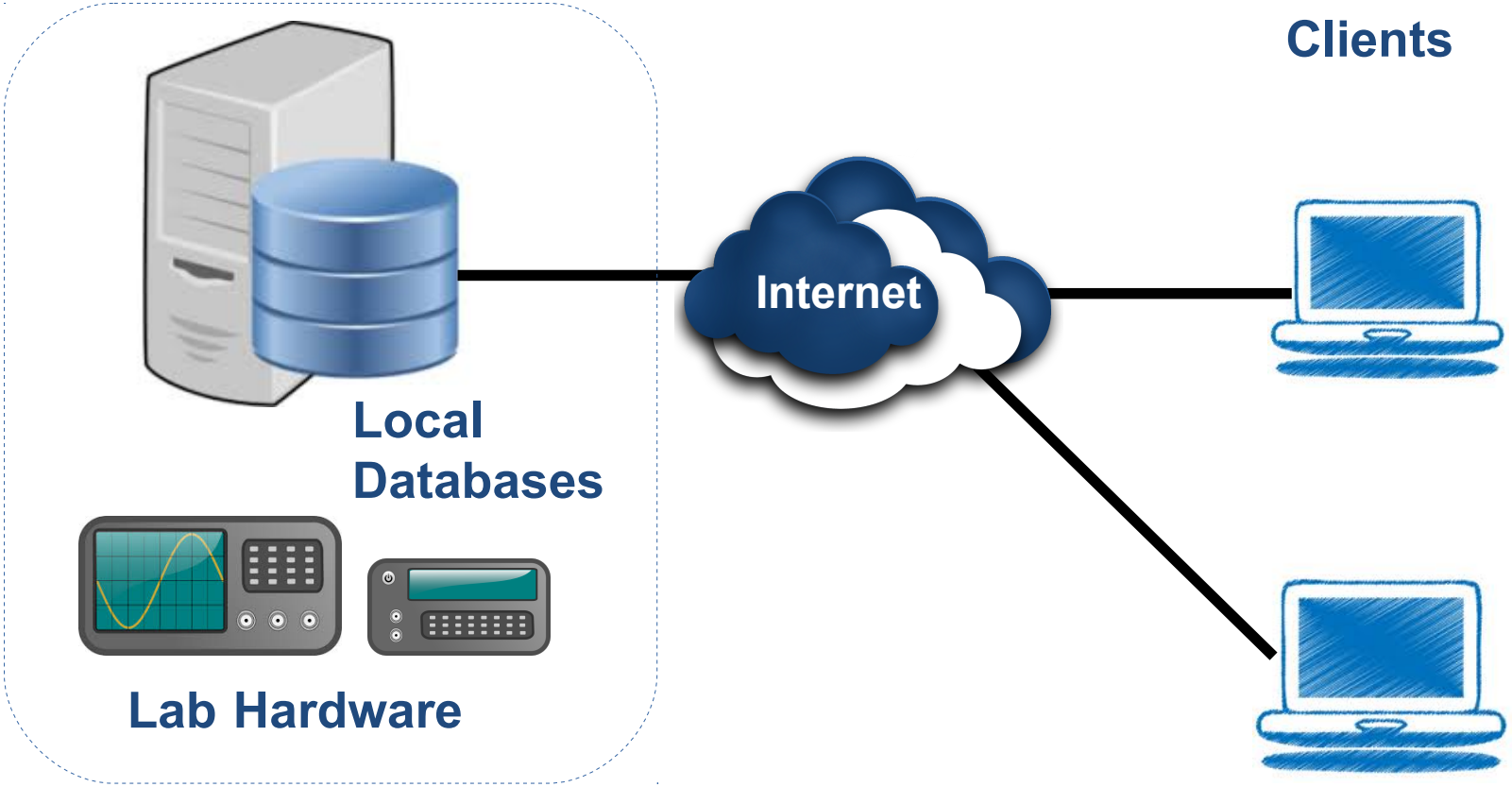
Carinthia University of Applied Sciences
Danilo G. Zutin

Remote Laboratory Management Systems and the iLab Shared Architecture (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.

Typical *Ad hoc* Online Lab Architecture

Lab Server:
manages lab equipment and students




{JSON}


http://


SOAP
Web
services
WSDL
Services
service
REST


LS


AJAX
Asynchronous Javascript And XML


RSS
XML
ATOM


HTML
WebSockets


XML-RPC


JSON-RPC


XML


CORBA


OpenSocial

Common functionalities of an Online Lab system

User Management

Online Labs should be easy to share with colleagues and students of other Universities.

Online Labs require user authentication in order that permissions could be given and data stored and analized.

But, in a Network of online Labs, this discoureged lab owners from sharing their equipment as many times they are obligated to administer accounts of users from other institutions.

If experiment results storage is necessary, the responsible lab server also assumes the responsability of storing or disposing of student experiment results.

Common functionalities of an Online Lab system

Lab Session Management

Calendar based booking

- Lab session typically takes longer
- User can reserve an specific time-slot
- Reliable: lab server will be available at reserved time

Queueing mechanism

- Usually implemented as first in first out (FIFO)
- Experiments usually run fast
- Requests can be prioritized

How much effort is it necessary to implement an Online Lab?

Online Lab: The Challenges

Developing an online lab from scratch is a lot of work!

- Great attention needed to user **scalability**
- Needs to be done by **domain specialist**

Managing a broadly shared Lab is also a lot of work!

- Disincentive for owner to share lab

Key challenge: **Scalability**

Scalability

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 users increases, a highly scalable architecture is desirable in order that labs could be managed in a comfortable way and included/disposed easily.

The importance of sharing a software infrastructure

So that lab developers don't have to start fresh each time but can build upon a **stable foundation**;
So that students can have a consistent interface to multiple laboratories with **single sign-on**;
So that the infrastructure can **separate the task** of *providing the lab* from that of *managing students using the lab*.

Examples of RLMSs

iLABS
Remote Lab Access

weblabdeusto

labshare

RemLabNet
TESTING
BETA
TESTING

Tempus

DesIRE

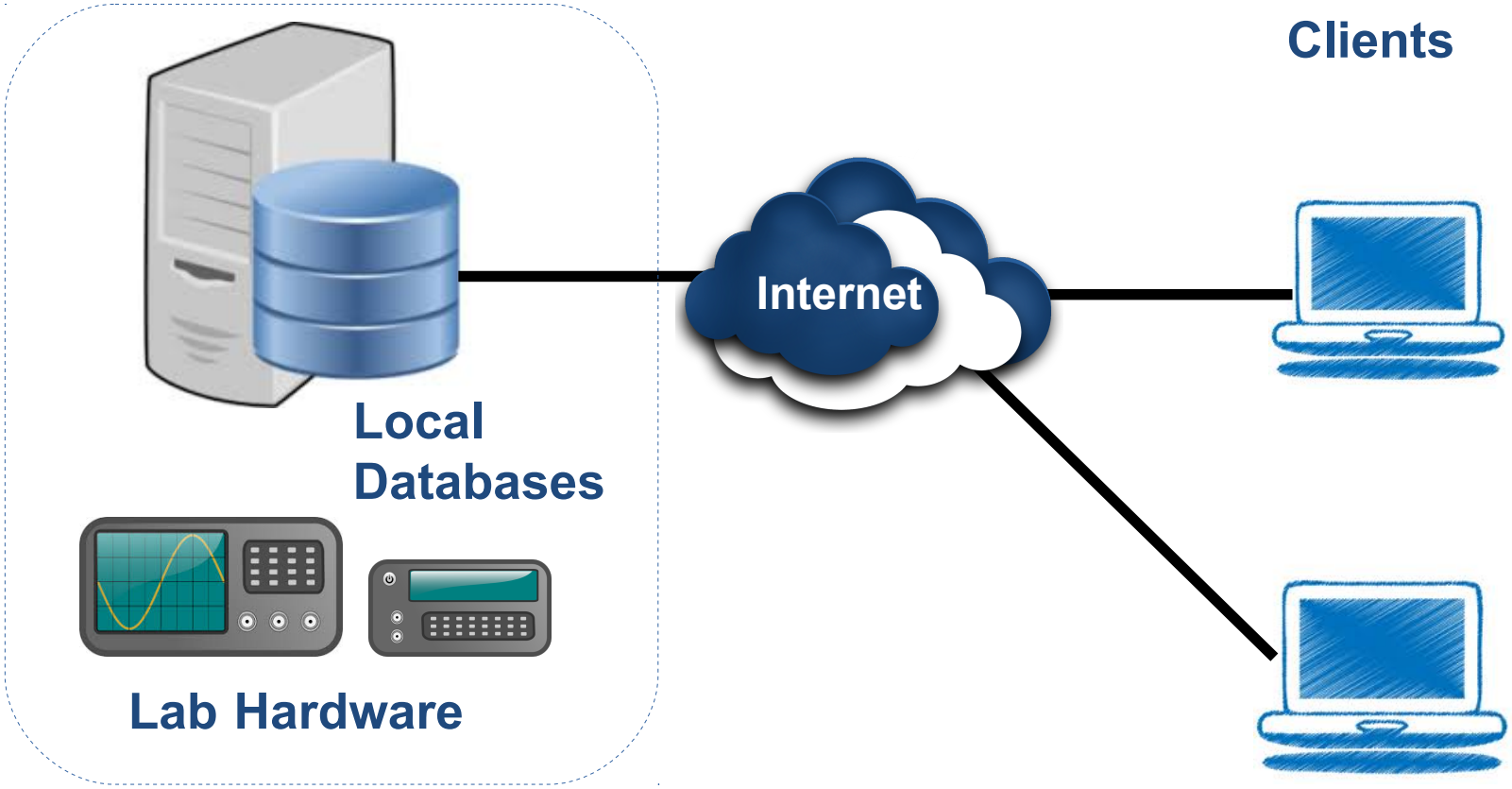
Center of Competence
in Online Labs
and Open Learning
OL²

The iLab Shared Architecture

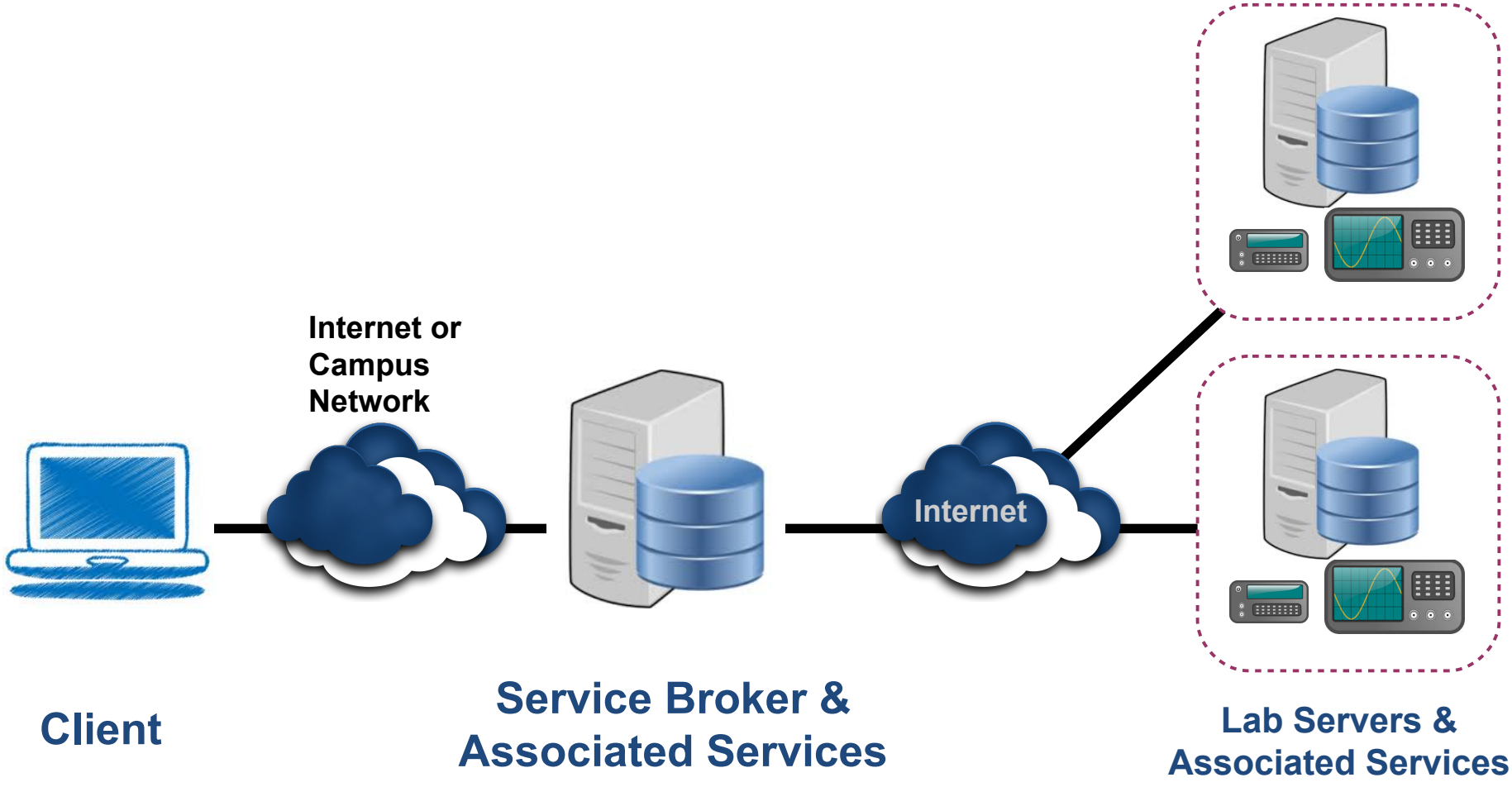
Hal Abelson of MIT and Dave Mitchell of Microsoft suggested building iLabs on top of such a web service infrastructure in 2002.
This initiated the development of the iLab Shared Architecture.

Before...

Lab Server:
manages lab equipment and students



Now… The iLab Shared Architecture

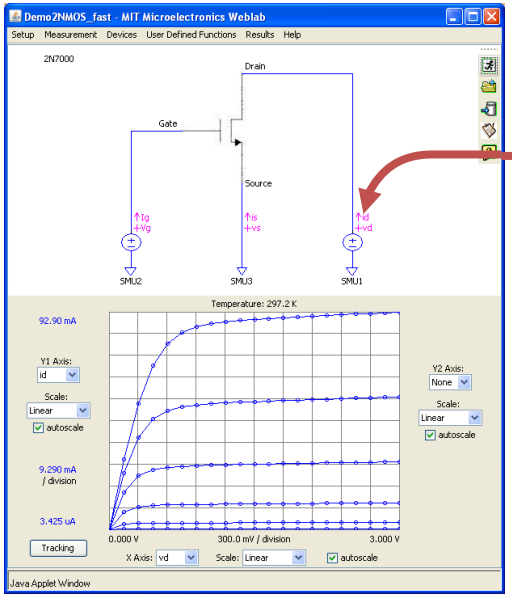


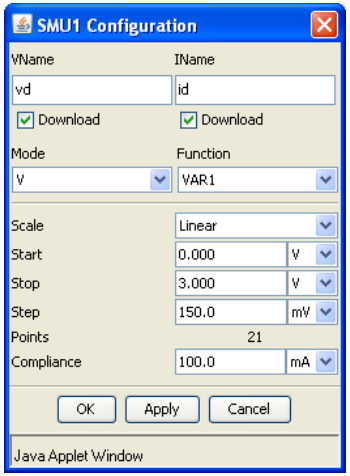
The iLab Shared Architecture

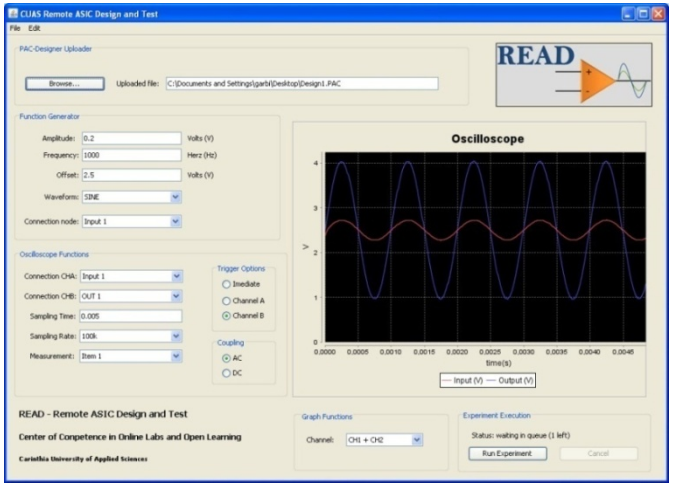
Batched Experiments:

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.







Tempus DesIRE

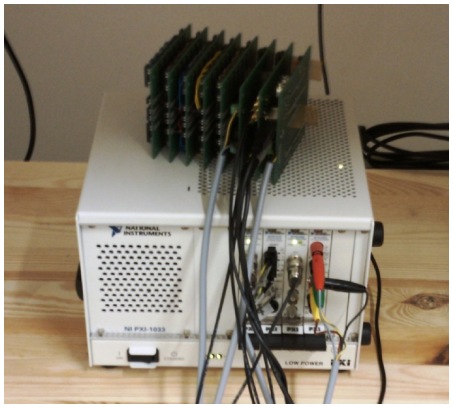
Center of Competence in Online Labs and Open Learning

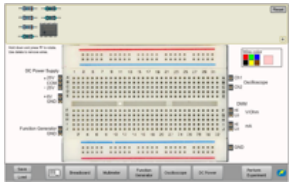
The iLab Shared Architecture

Interactive 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.





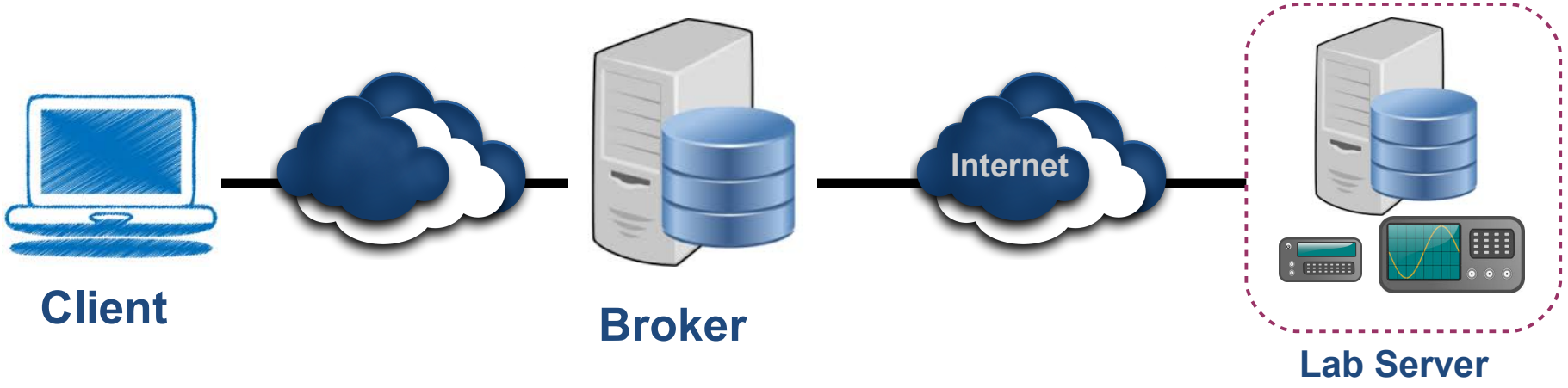
The iLab Shared Architecture

Three-tier Web Application:

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 ServerTM). The Service Broker should reside on a server at the student's institution.

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



The iLab Shared Architecture

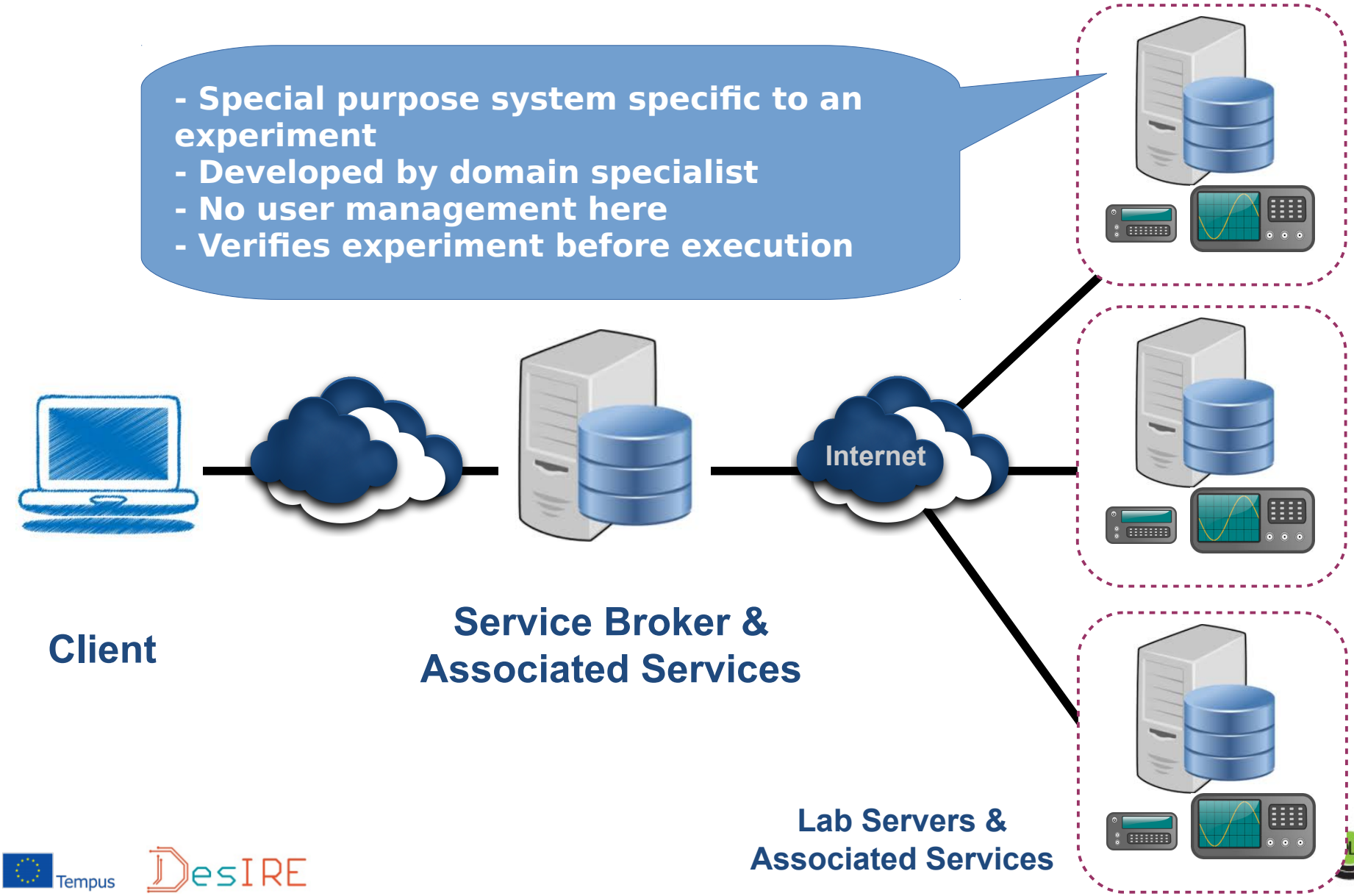
Student's Client: Lab-dependent piece of code that must understand the protocol for specifying a particular experiment's parameters.

The Service Broker authenticates students, checks on their authorization to contact a particular Lab server, accepts an experiment specification from the client and waits to retrieve the results. The SB also stores experiment specification and results.
The SB is a completely generic code. It can interoperate with any combination of client and lab server that implement the appropriate interfaces expressed in terms of web service SOAP calls defined in WSDL

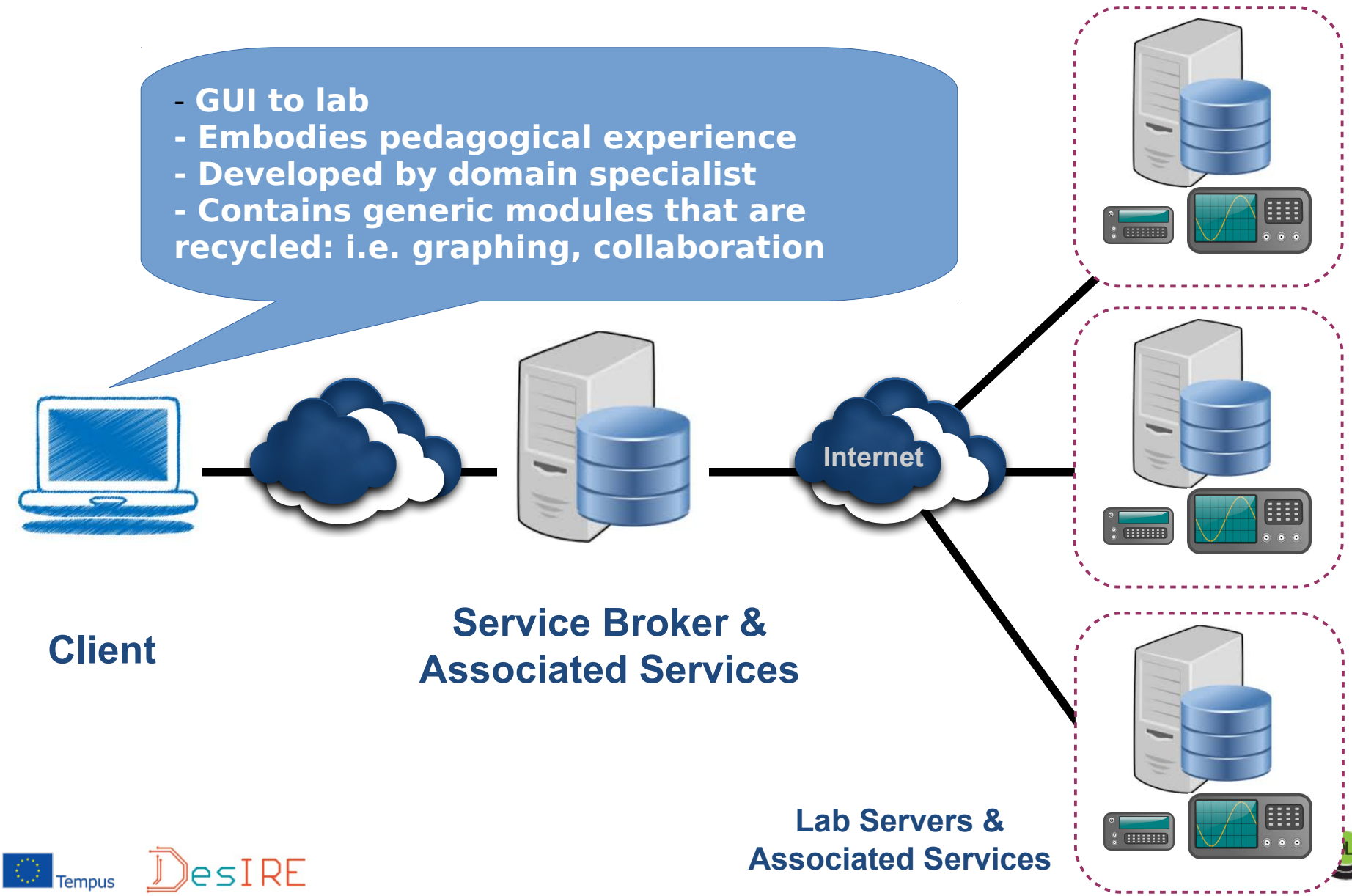
The Lab Server knows nothing about the students using the system and stores results and experiments specification temporarily .

In the batched experiment architecture the student's workstation never contacts the Lab Server directly. We can maintain this strict discipline because the batched experiment requires so little communication between the client and the Lab Server

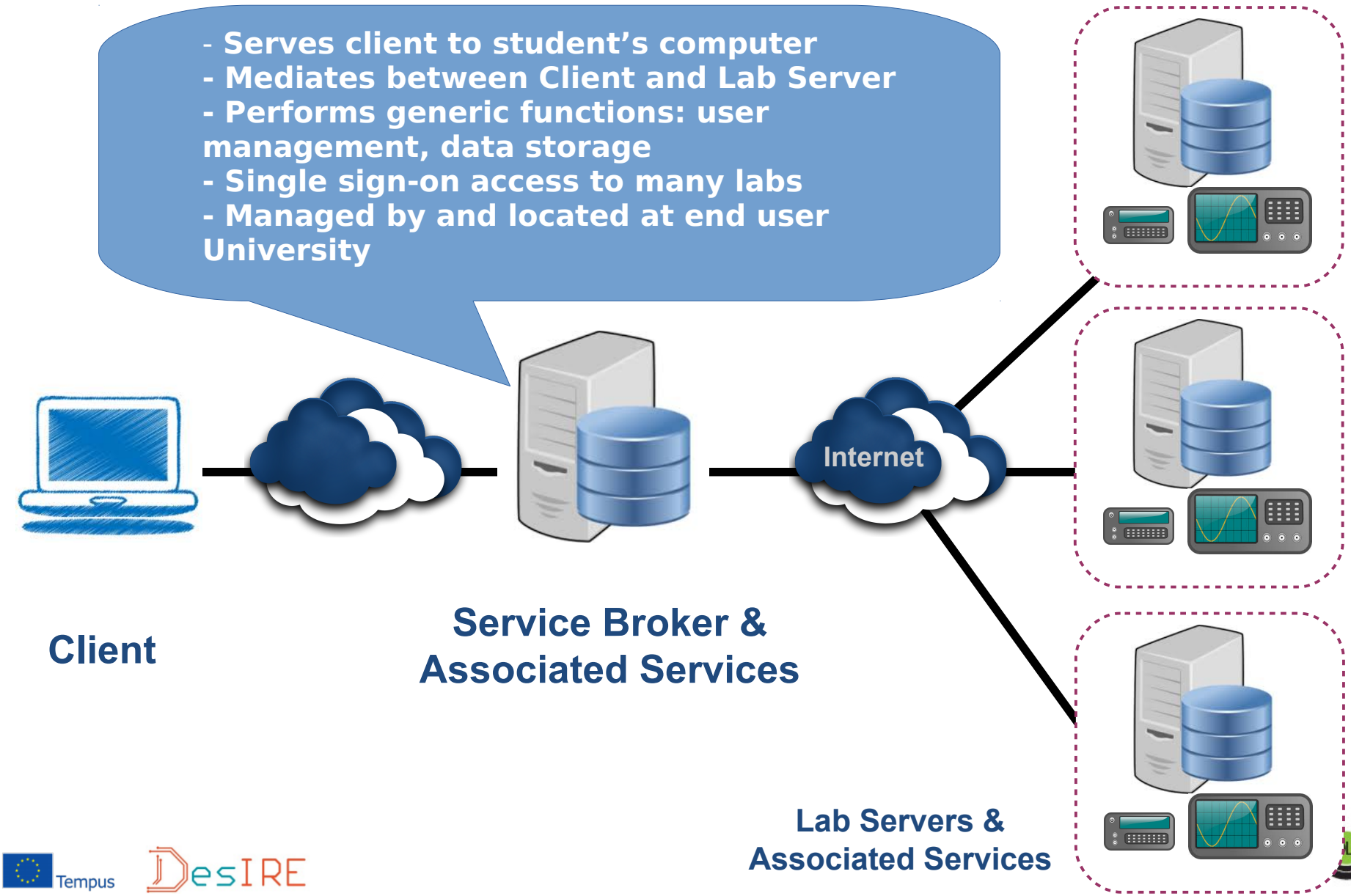
- Special purpose system specific to an experiment
- Developed by domain specialist
- No user management here
- Verifies experiment before execution



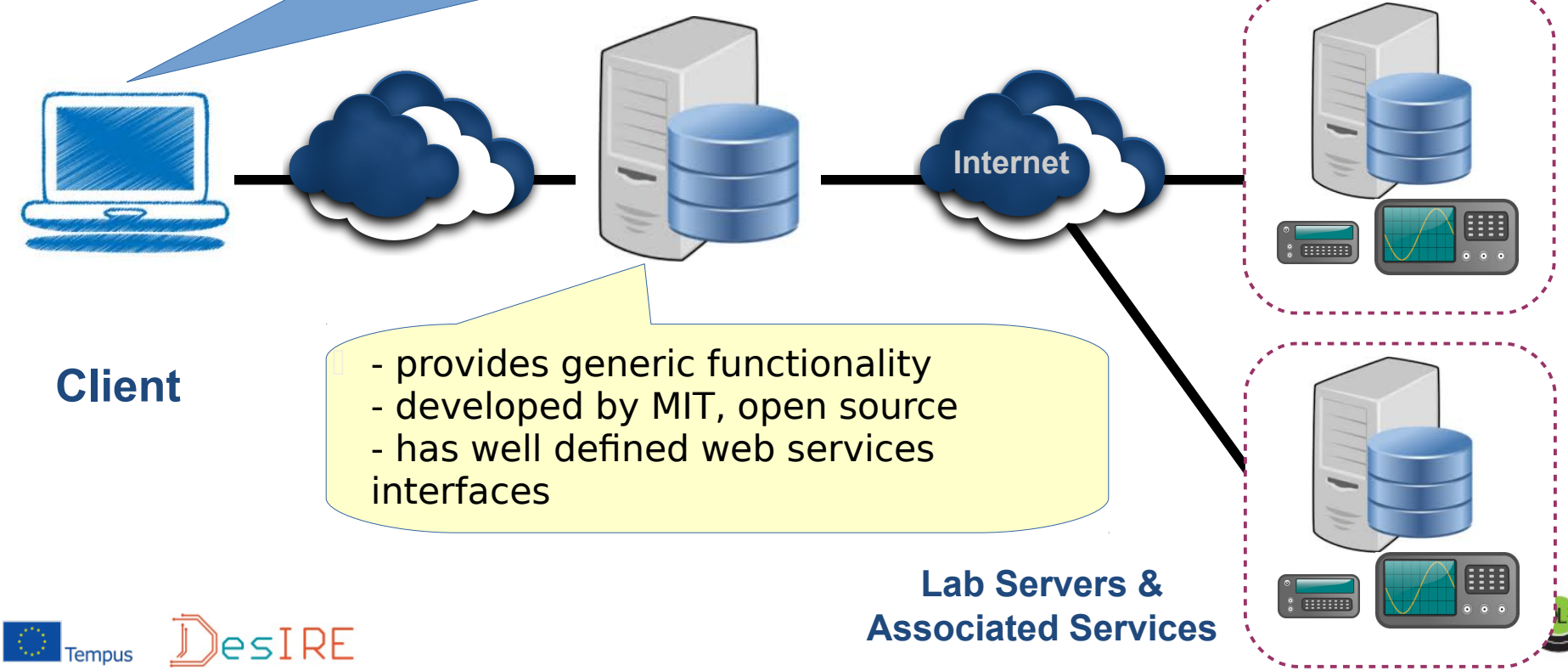
- GUI to lab
- Embodies pedagogical experience
- Developed by domain specialist
- Contains generic modules that are recycled: i.e. graphing, collaboration



- Serves client to student's computer
- Mediates between Client and Lab Server
- Performs generic functions: user management, data storage
- Single sign-on access to many labs
- Managed by and located at end user University



Lab provider:
- develops Lab Server and Client
- can customize modules developed
- registers them with Service Brokers

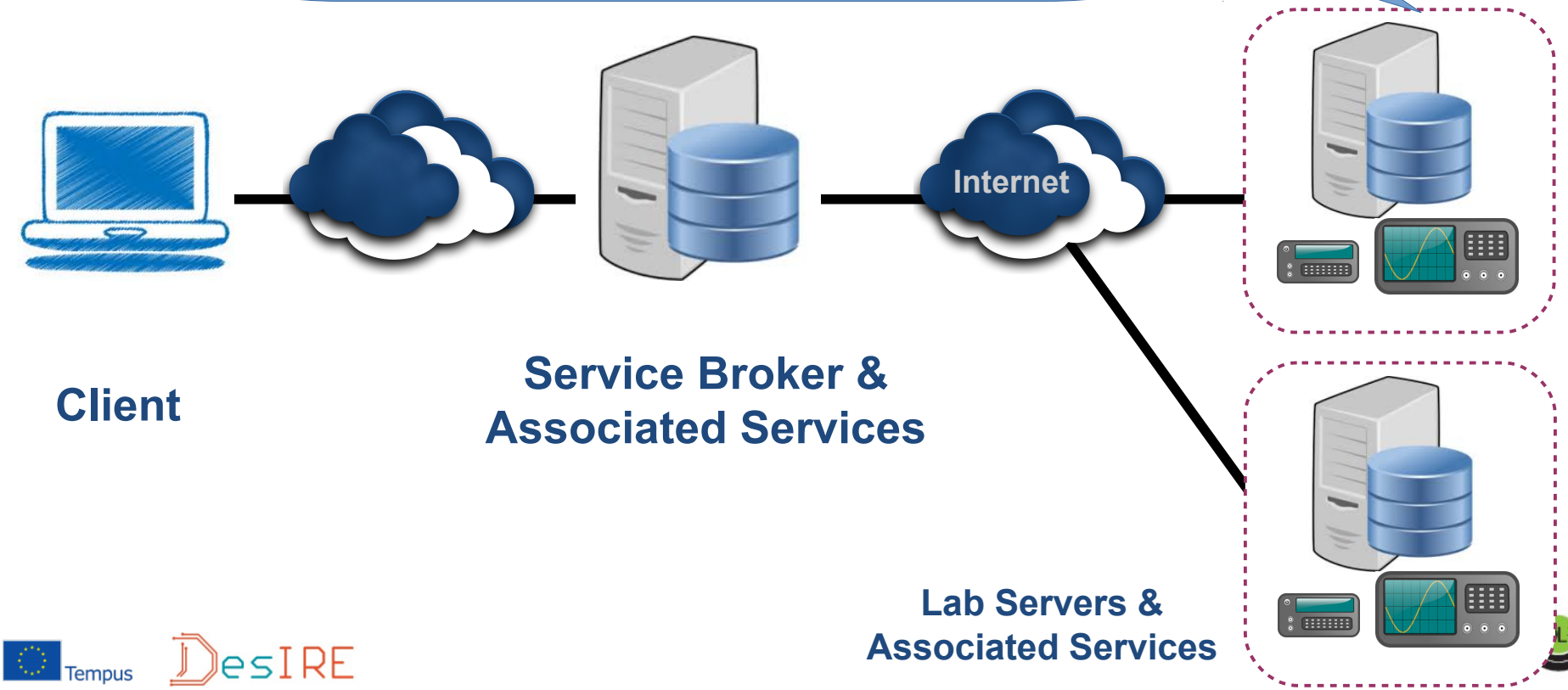


Client

- provides generic functionality
- developed by MIT, open source
- has well defined web services interfaces

Lab Servers & Associated Services

Lab provider:
- manages Lab Server
- sets lab policy
- does not know the user identity



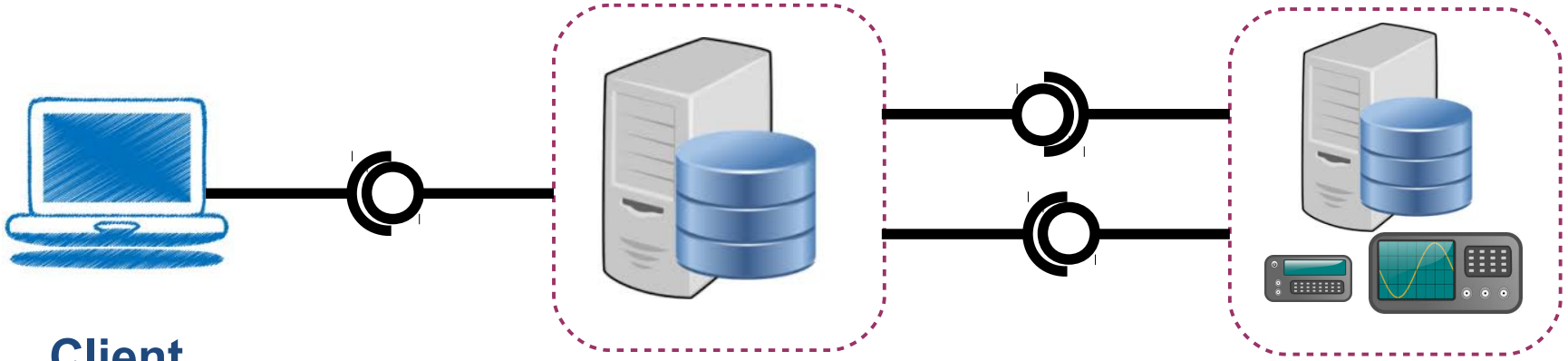
ISA Web Services APIs (Batched)

Service Consumers:

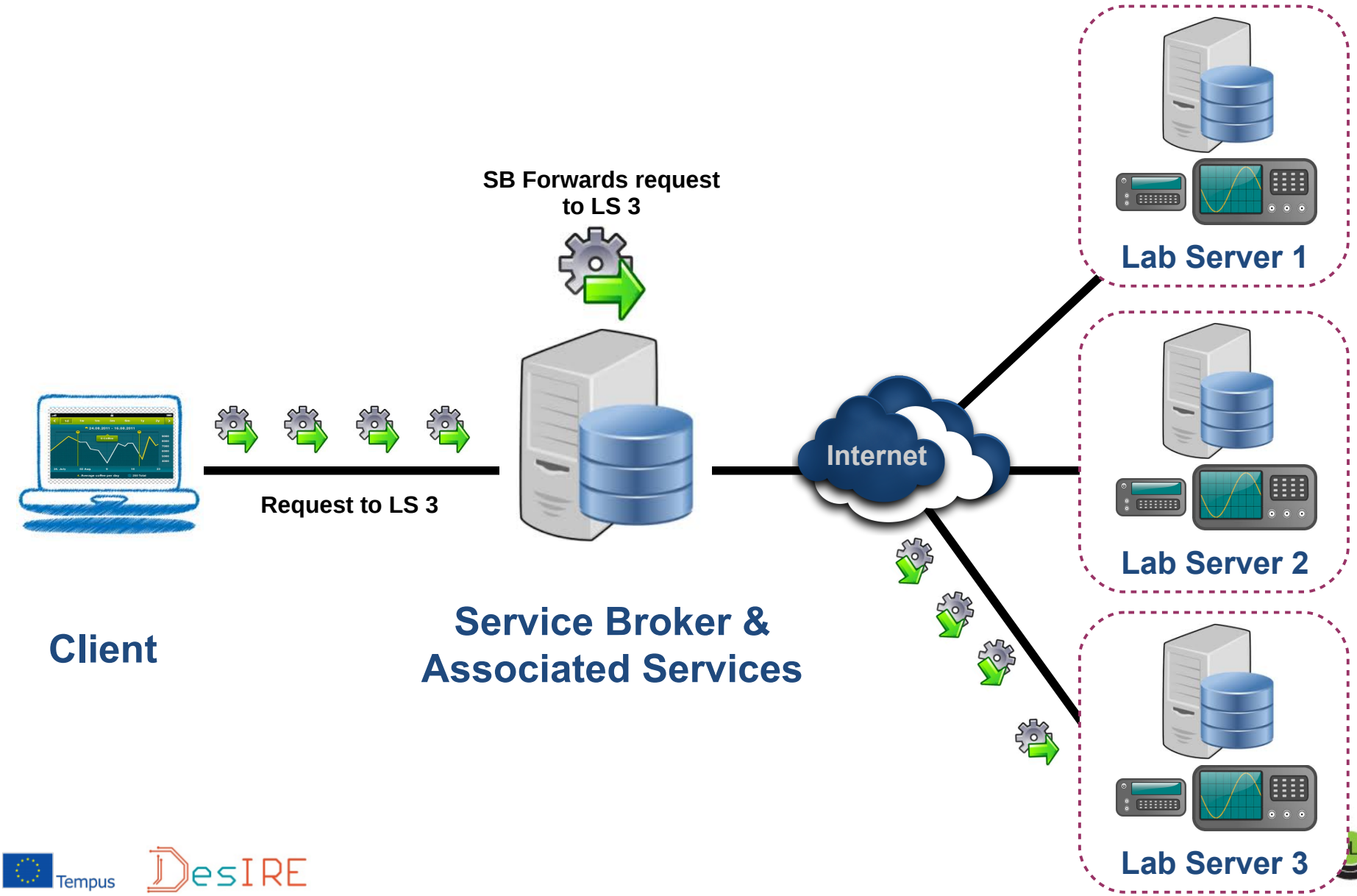
- **Client App**
- **Service Broker**

Service Providers:

- **Service Broker**
- **Lab Server**



ISA Web Services APIs (Batched)



The ISA API

Web Services: Are a method of making information available in a standardized way that could be accessed by any developer‘s application over the Web.

Web Services have clear W3C standards , all standards based on XML

Source: Beginning ASP.NET 2.0 with C#
Chris Hart, John Kauffman, David Sussman, Chris Ullman

The iLab Shared Architecture

Outline of a Student Batched Experiment Session

User

1. User signs in to Service Broker, selects Lab Client

2. User creates exp. Specification and submits an experiment

READ Lab Client

ExecuteExp

GetStatus

GetResults

Service Broker

Submit()

GetExpStatus()

Notify()

RetrieveResult()

READ Lab Server

Submit()

(Return submission report)

RetrieveResult()

(Return Results)

3. Experiment Submitted with SBIB, LSPasskey

4. SSL Session Stablished

5. Experiment Submitted with SBIB, LSPasskey

6. Client keeps pooling to get experiment status

7. SB Notified when exp complete with LSID, SBPasskey

8. Client Retrieves the results

9. Result retrieved using SBID, LSPasskey

ISA Web Services API (Request/Response Example)

SOAP Request

```
POST /ServiceBroker/Services/ServiceBrokerService.asmx HTTP/1.1
Host: ilabs.cti.ac.at
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://ilab.mit.edu/Submit"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
  <soap:Header>
    <sbAuthHeader xmlns="http://ilab.mit.edu">
      <couponID>long</couponID>
      <couponPassKey>string</couponPassKey>
    </sbAuthHeader>
  </soap:Header>
  <soap:Body>
    <Submit xmlns="http://ilab.mit.edu">
      <labServerID>string</labServerID>
      <experimentSpecification>string</experimentSpecification>
      <priorityHint>int</priorityHint>
      <emailNotification>boolean</emailNotification>
    </Submit>
  </soap:Body>
</soap:Envelope>
```

SOAP (XML) documents are sent as part of the HTTP data.

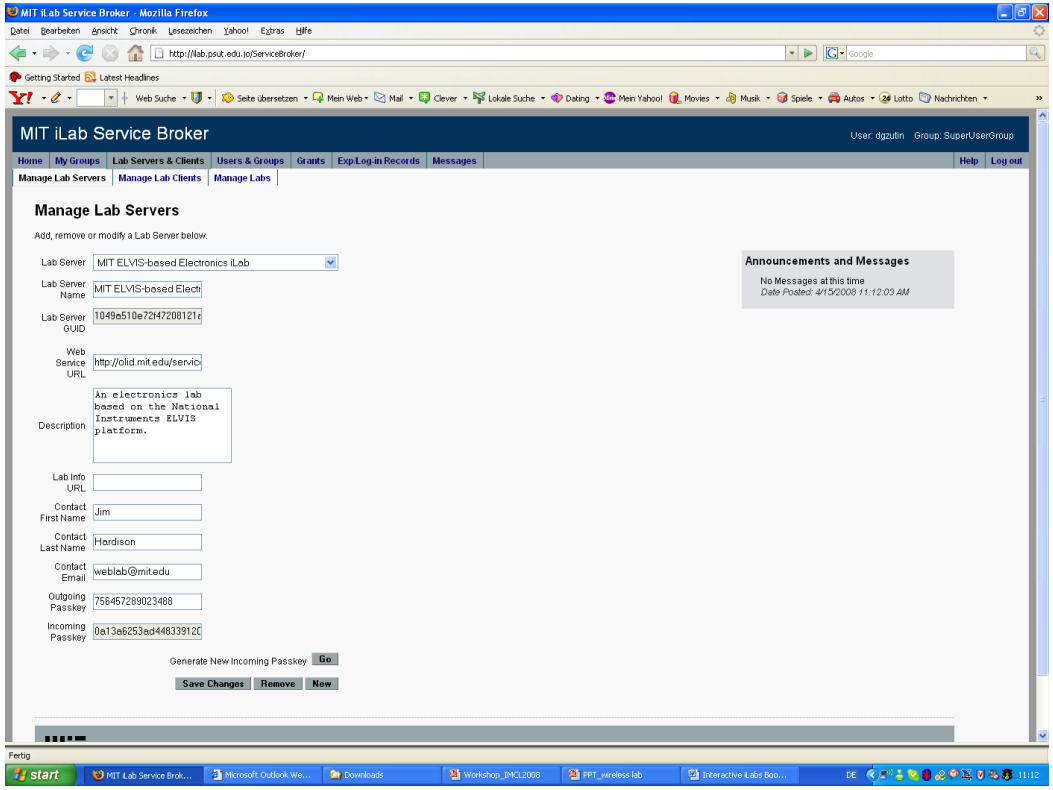
SOAP Response

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

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
  <soap:Body>
    <SubmitResponse xmlns="http://ilab.mit.edu">
      <SubmitResult>
        <vReport>
          <accepted>boolean</accepted>
          <warningMessages>
            <string>string</string>
            <string>string</string>
          </warningMessages>
          <errorMessage>string</errorMessage>
          <estRuntime>double</estRuntime>
        </vReport>
        <experimentID>int</experimentID>
        <minTimeToLive>double</minTimeToLive>
        <wait>
          <effectiveQueueLength>int</effectiveQueueLength>
          <estWait>double</estWait>
        </wait>
      </SubmitResult>
    </SubmitResponse>
  </soap:Body>
</soap:Envelope>
```

The iLab Shared Architecture

Including a lab in the Batched Architecture:



Install Domain Credentials:

- Contact the process Agent (Lab Server's side) administrator via email and get the web services URL and an initial passkey.
- Send the Service Broker's GUI (Global Unique Identifier).

Supporting Interactive Experiments

Interactive experiments, by their nature…

- …require ***real-time*** control.

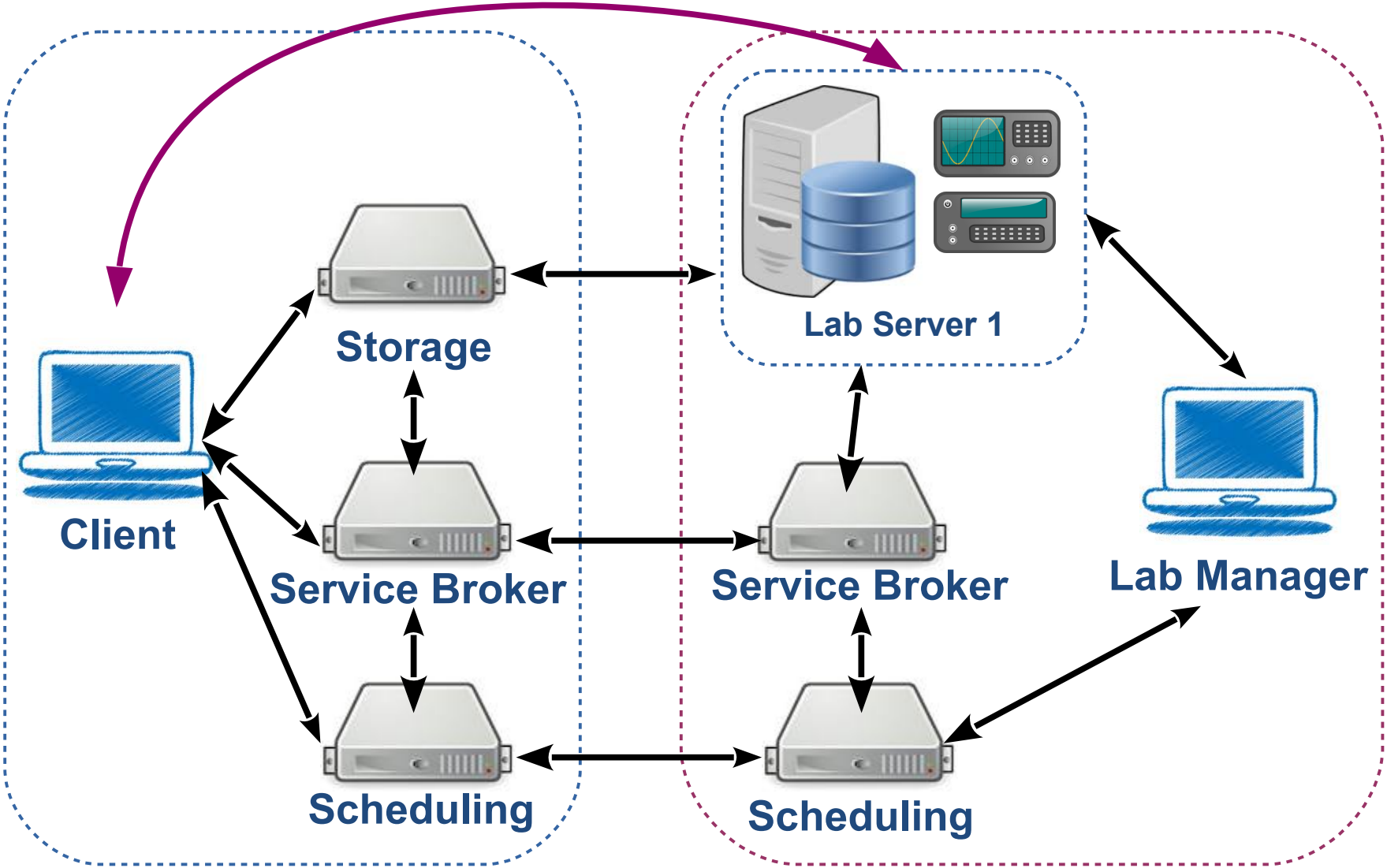
Direct client to lab server connection needed (possibly high-bandwidth, possibly proprietary protocols)

- …are performed in ***human-time***.

Longer periods of single user control

A revision to the iLab Shared Architecture (and the Service Broker) was necessary to provide services for interactive experiments.

ISA Interactive Lab Support

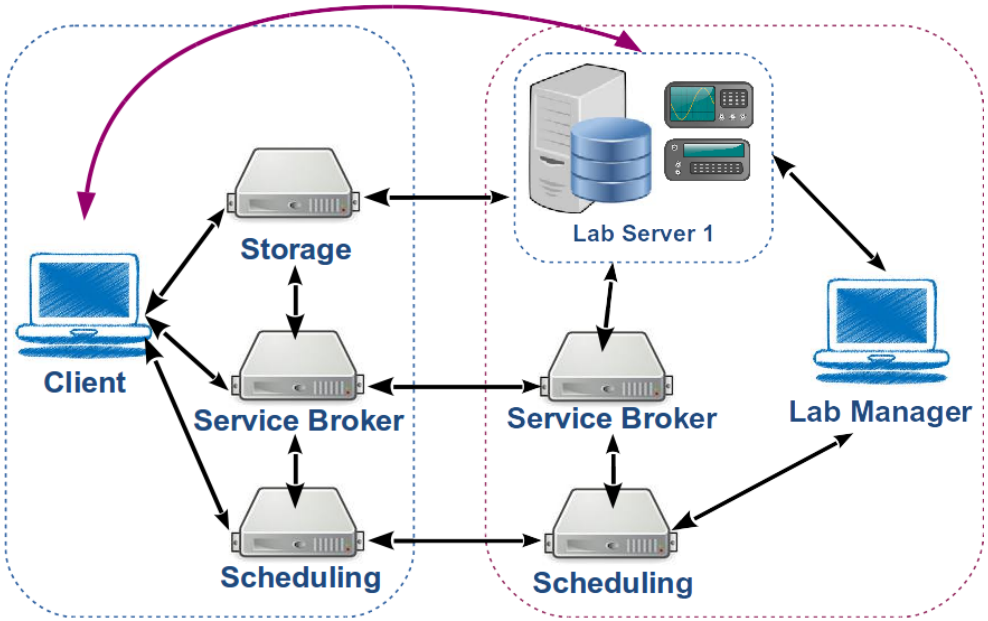


Interactive ISA

iLab Service Broker orchestrates access to a set of distributed, stand-alone services for…

- Experiment Storage
- Scheduling
- Cross-service

Authorization Mechanism:
(Ticketing)



Service Broker sets up the experiment session and then steps out of the way.

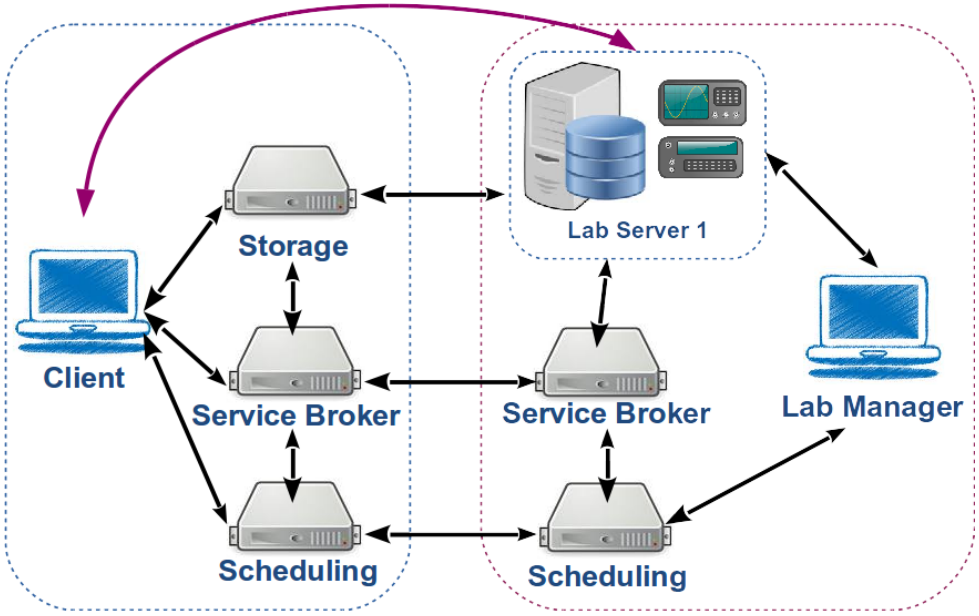
The Experiment Storage Service (ESS)

A generic stand-alone service responsible storing student experiment information

- Lab configurations, input parameters, results

Service Brokers, Lab Servers and Clients all interface to an ESS

- Service Broker responsible for record management
- Clients & Lab Servers contribute to the record



A single ESS may service many iLab installations

Scheduling Services

Used to manage student access to the lab. Composed of two complementary services:

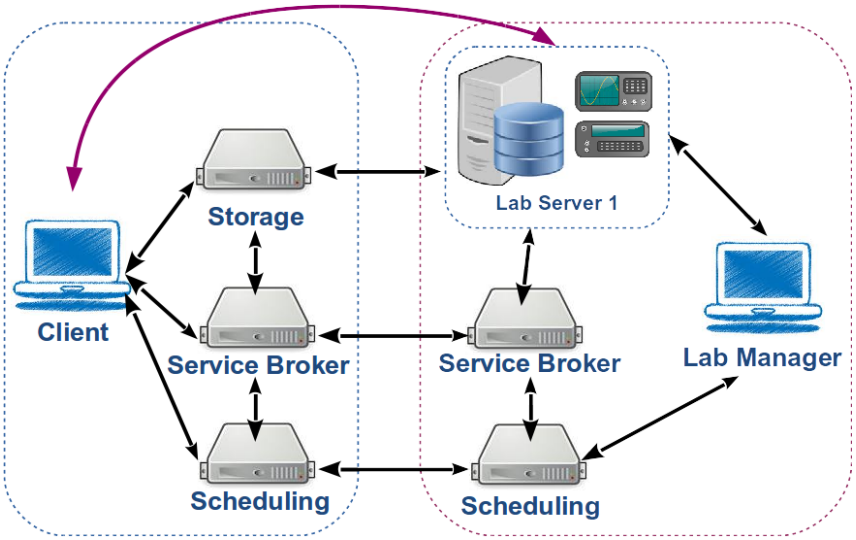
Lab-side Scheduling Service (LSS)

Allocates available lab time to sets of students.

User-side Scheduling Service (USS)

Reserves blocks of available time for individual student use.

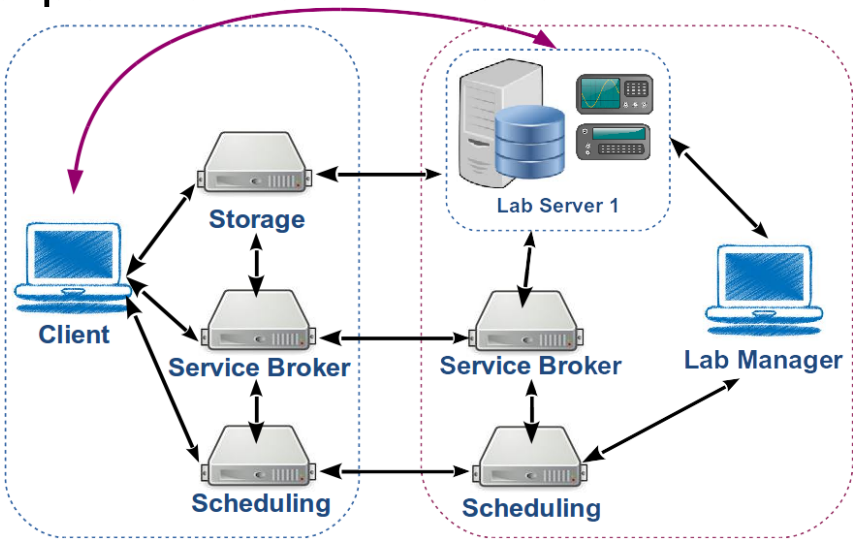
Service Brokers manage connections between USS And LSS pairs



Ticketing

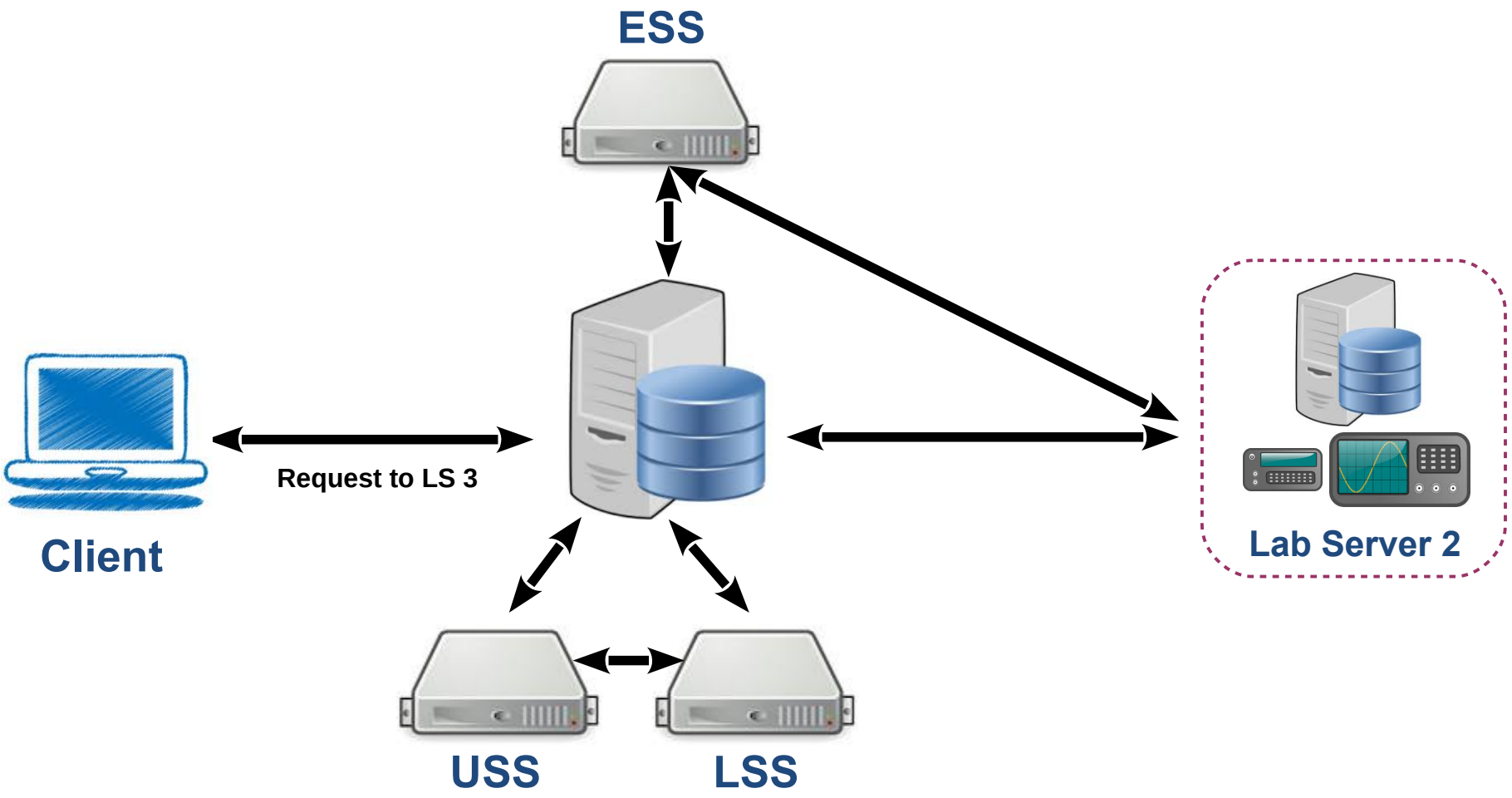
Provides a common authorization system for systems in the iLab Shared Architecture, based on the following ideas:

Service Brokers authenticate users and can permit access to ISA resources by issuing Tickets.
Agents seeking access receive only a pointer to the Ticket (Coupon).
Only issuing and redeeming agents are able to access Ticket information.



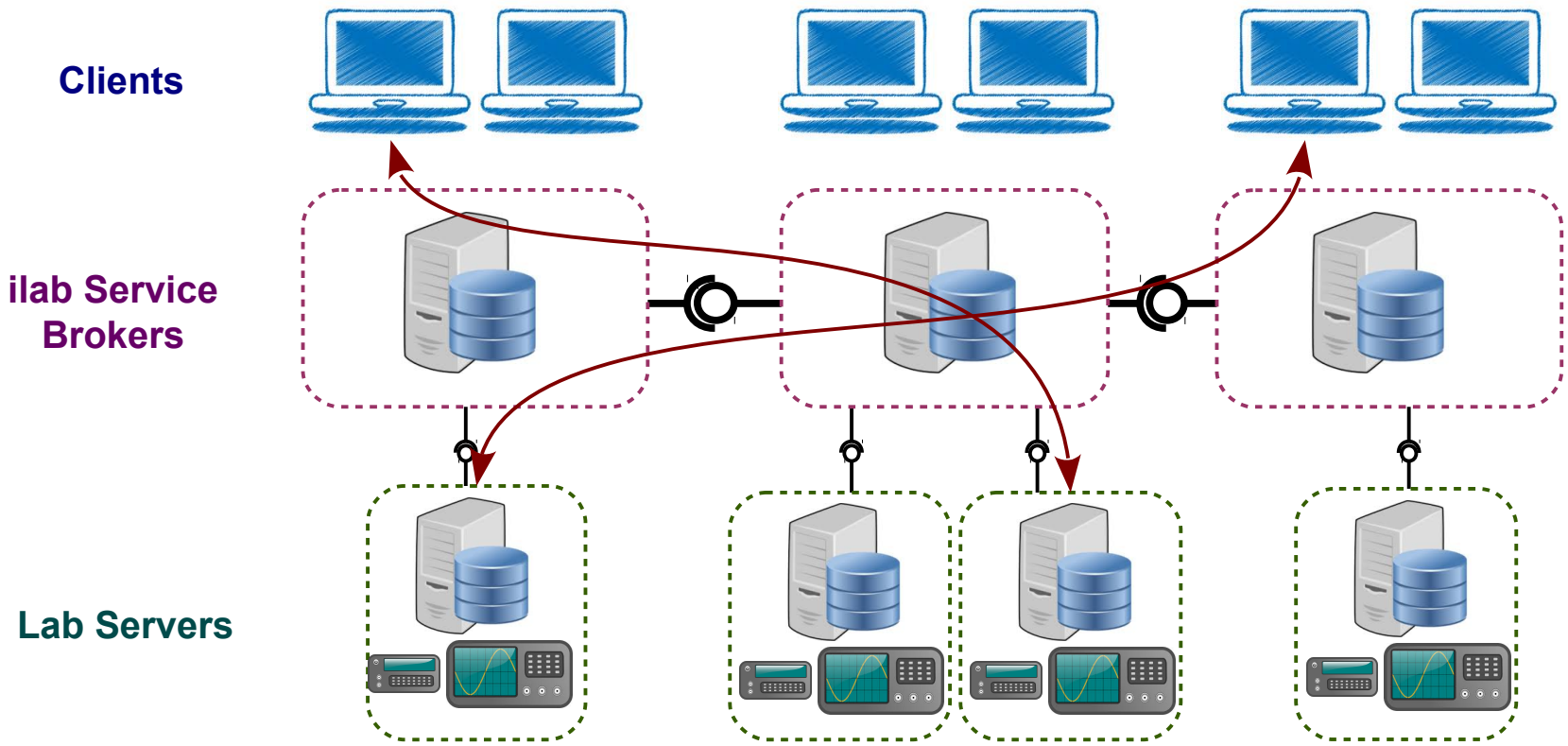
Ticketing is used to manage access to all ISA systems.

Interactive Single Campus Model



Multi-campus case

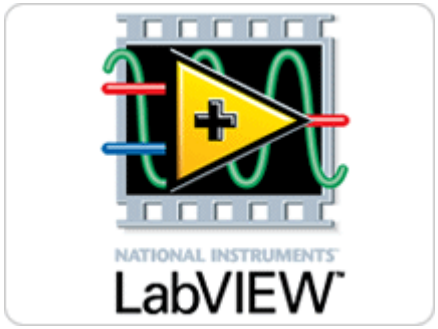
Trust relationships are setup between Service Brokers. Brokers act as an iLab communication backbone



The iLab Shared Architecture

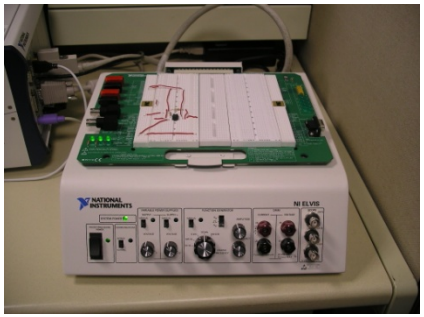
Due to the wide acceptance of LabVIEW by lab developers, a LabVIEW Integrated Interactive Lab Server (LVILS) was released. This distribution was built upon the generic ILS and provides interfaces between the .NET Web Services and LabVIEW processes.

It supports:
- Writing data to the ESS provided via DataSocket using a specific XML record format.
- Management of the LabVIEW process (terminating session)



iLabs Around the World

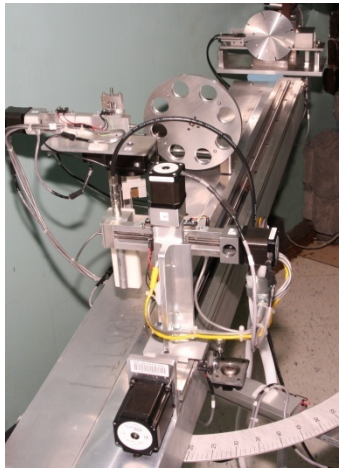
Microelectronics Device Characterization
(MIT-EECS, deployed 1998)



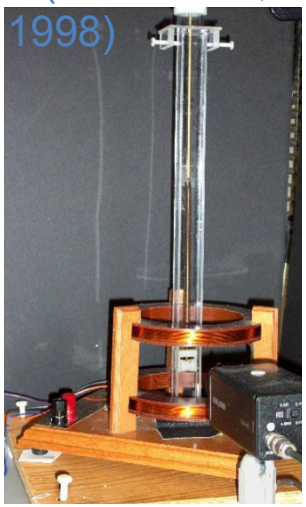
ELVIS

(MIT-EECS, deployed 2006)

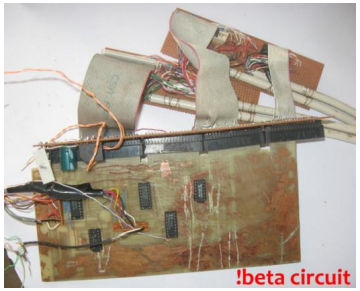
Dynamic Signal Analyzer
(MIT-EECS, deployed 2004)



Neutron Spectrometer
(MIT-Nuclear Eng., deployed 2008)

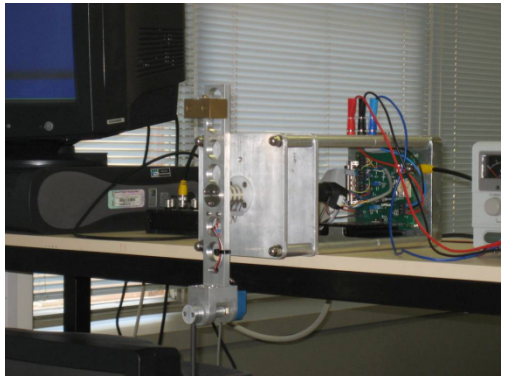


Force on a Dipole
(MIT-Physics, deployed 2008)



Logic Lab
(OAU, Nigeria, deployed 2007)

Radioactivity
(University of Queensland, Australia, deployed 2007)



Inverted Pendulum
(University of Queensland, Australia, deployed 2004)

Tempus

DesIRE

Center of Competence Online Labs and Open Learning

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.
- Hardison/MIT , Slides from „The iLab Project: Introduction and Overview”