Teletraining for Programmers of Intelligent Networks and Generalisation to a Unified Architecture for Tele-Experimenting

Mueller, Eberhard; Pommer, Hermann; Veichtlbauer, Armin; Hofmann, Ulrich
University of Salzburg

Abstract

This paper describes an environment for remote testing of IN-Services (programs for Intelligent Networks).

This system is realised as a Client/Server Architecture. A programmer of IN-Services is connected to a Test Center via Internet. A Graphical User Interface (GUI) is running on a Windows PC. An FTP connection to a local UNIX machine with program source files is established to download these files.

A Test Requirement can be created and then be sent to the Test Center. After executing the test the results are stored and sent back to the Client.

The final part describes a generalisation of this Remote-Testing system to a Unified Architecture for Tele-Experimenting.

Key words
Tele-Experimenting, Tele-Learning, Intelligent Network (IN), Service Design (SD), IN-Service, Remote Experimenting (REX), Unified Tele-Experimenting Architecture (UTEA)

1. Introduction

With Intelligent Networks value-added services for Telecommunication can be realised. This is a fast growing market with good chances for earning better revenues than with standard network services. But the "time to market" must be short.

The programming of IN-Services (a proprietary system of Siemens for controlling Intelligent Networks [8]) is a complex process and substantial experience is necessary. The complete infrastructure is expensive, especially the Test Executing Environment (Testing of IN-Services).

With a Tele-Experimenting Environment, which connects the remote developer of IN-Services with the Test Center, it is much cheaper to get practice in programming IN-Services without having an expensive locale Test Environment.

After the implementation and successful test of this REX-system and the comparison with an earlier REX project at the Polytecnical University of Salzburg ([1],[2]) the concept of a general REX-architecture was developed. This *Unified Architecture for Tele-Experimenting* ([3],[4]) is presented in the final part of the paper.

2. Teletraining for programmers of Intelligent Networks

2.1. Architecture of Client/Server for Remote Testing of IN-Services

The programmer creates IN-Services with the Service Designer at his own Unix machine. Normally the IN-Services have to be transferred to the local available Test Executing Environment. An operator with experience has to execute the test.

An access via Internet is not available due to security standards. With the Remote Testing System programmers without an own Test Executing Environment and with lack of testing experience can have test runs of their programs executed.

It is realised as a Client/Server system ([5],[6],[7]) as shown by figure 1:

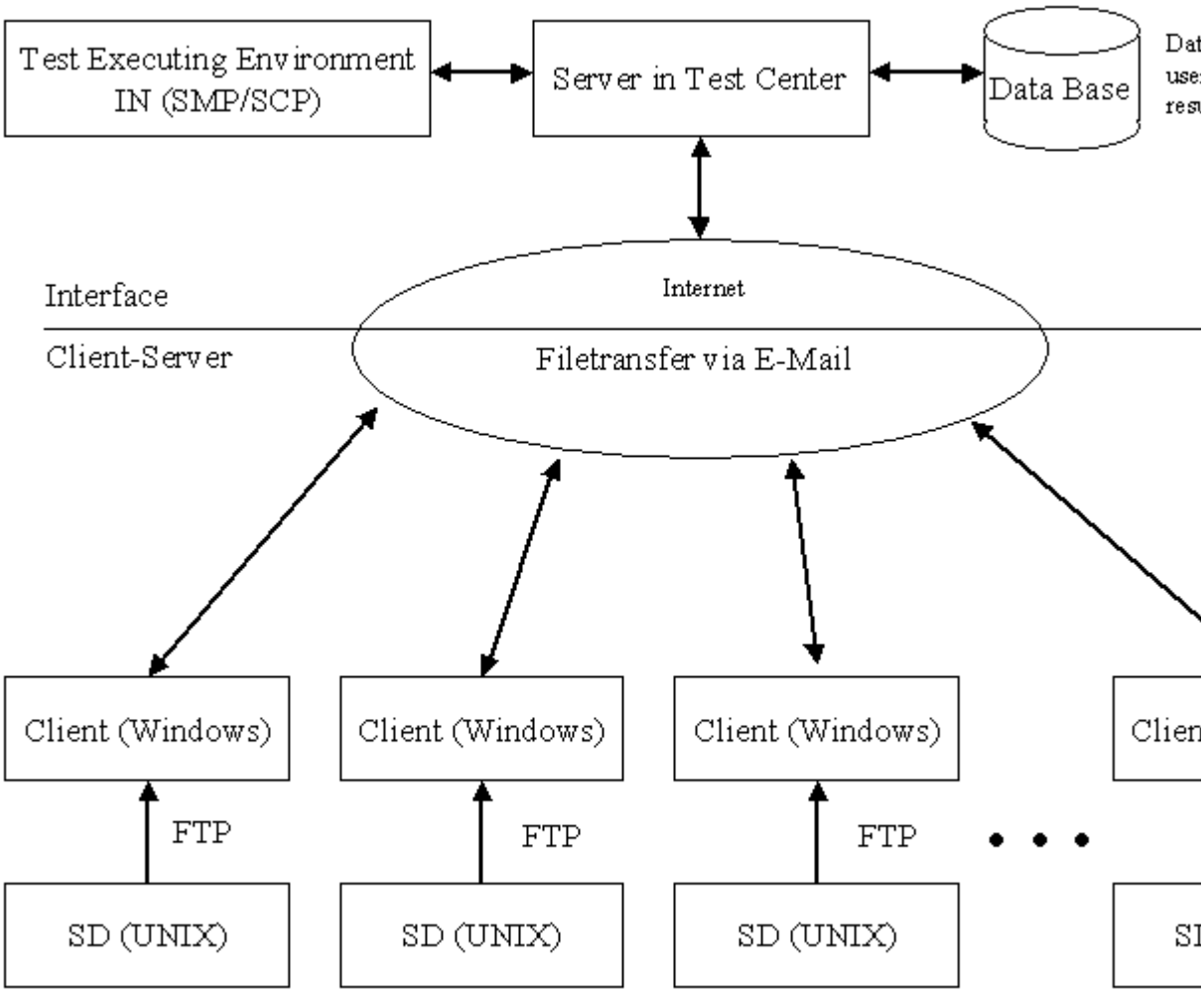

Figure 1: Client/Server-Architecture of Remote Test-Executing

Every client can send Test Requests to a Test Center. After Test Execution the client gets back the commented Test Results.

In the Test Center the tests are stored in a stack and executed offline (batch mode).

Priorities in the execution order can be allocated for different customers. Billing is also available.

2.2. Implementation of the Client/Server System

From a client machine with a GUI (running under Windows) a connection to the SD machine (running under UNIX) can be established. A list of available IN-Services on this SD machine is derived and displayed in the Client-GUI. Every selected IN-Service can be downloaded to the client machine via FTP. Then the user can create one ore more Test Requests for every IN-Service. Every Test Request contains a description of the test.

Before sending a Test Request a registration call to the Test Center is necessary. The server in the Test Center sends back a unique User-ID for further communication.

Then the client can send the selected test requirement to the Test Center via email. The server in the Test Center controls the incoming mail, extracts all information and stores it in a data base (User-ID, IN-Service, Test Requirement, ...). Then the tests are executed manually by an operator in the Test Executing Environment. The operator can add a comment for every result. Then the results are stored in the data base and are sent back automatically by the server to the sending client.

The Test Execution is done in batch mode. It is possible to consider different priorities for premium customers or teaching courses. Also a billing system can optionally be integrated.

A larger number of customers can be made motivated to use the "Tele-Experimenting of IN-Services".

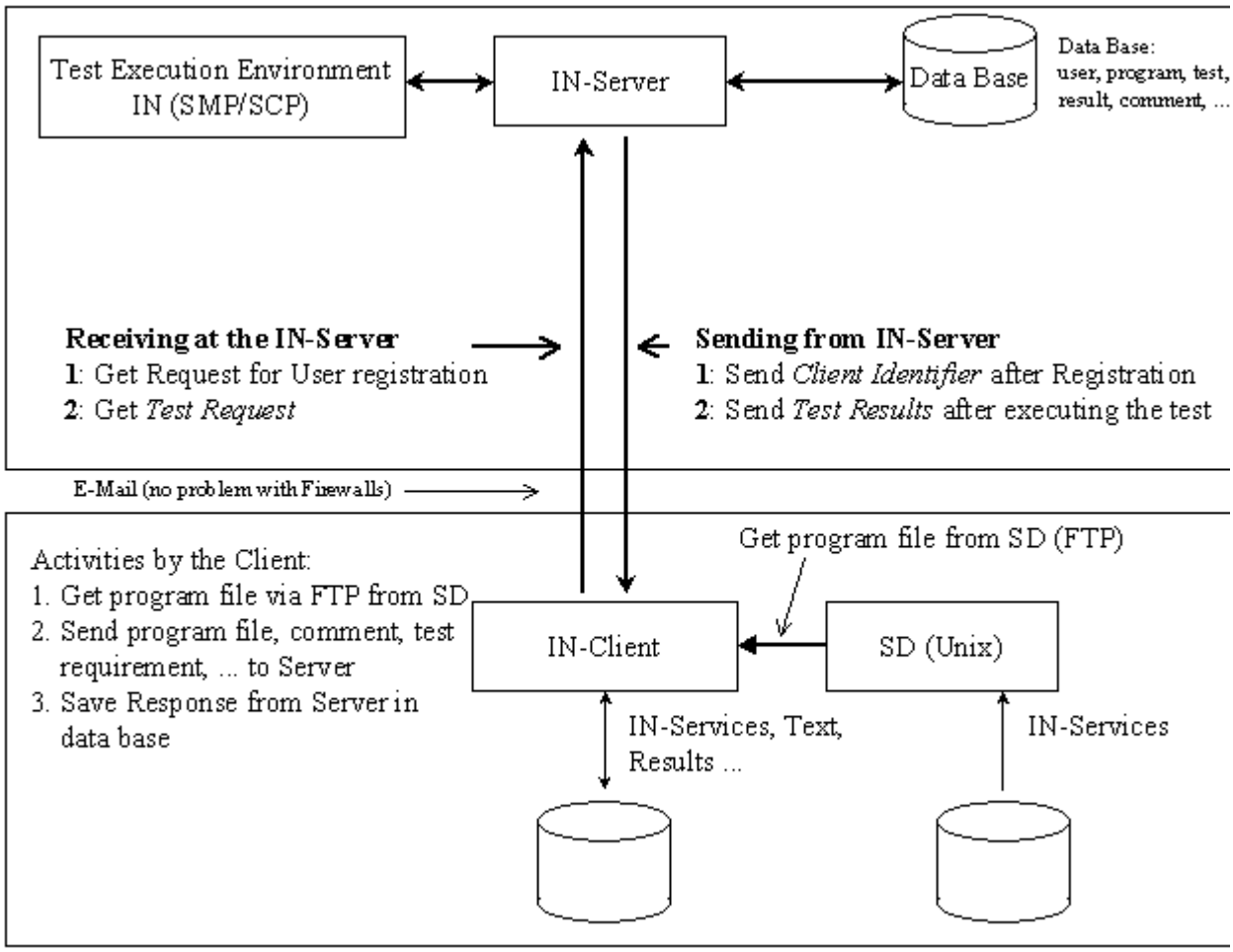

Figure 2: Implementation of the Client/Server-System

2.2.1. Configuration of Client

Before opening the connections, some configuration entries have to be made by the user.

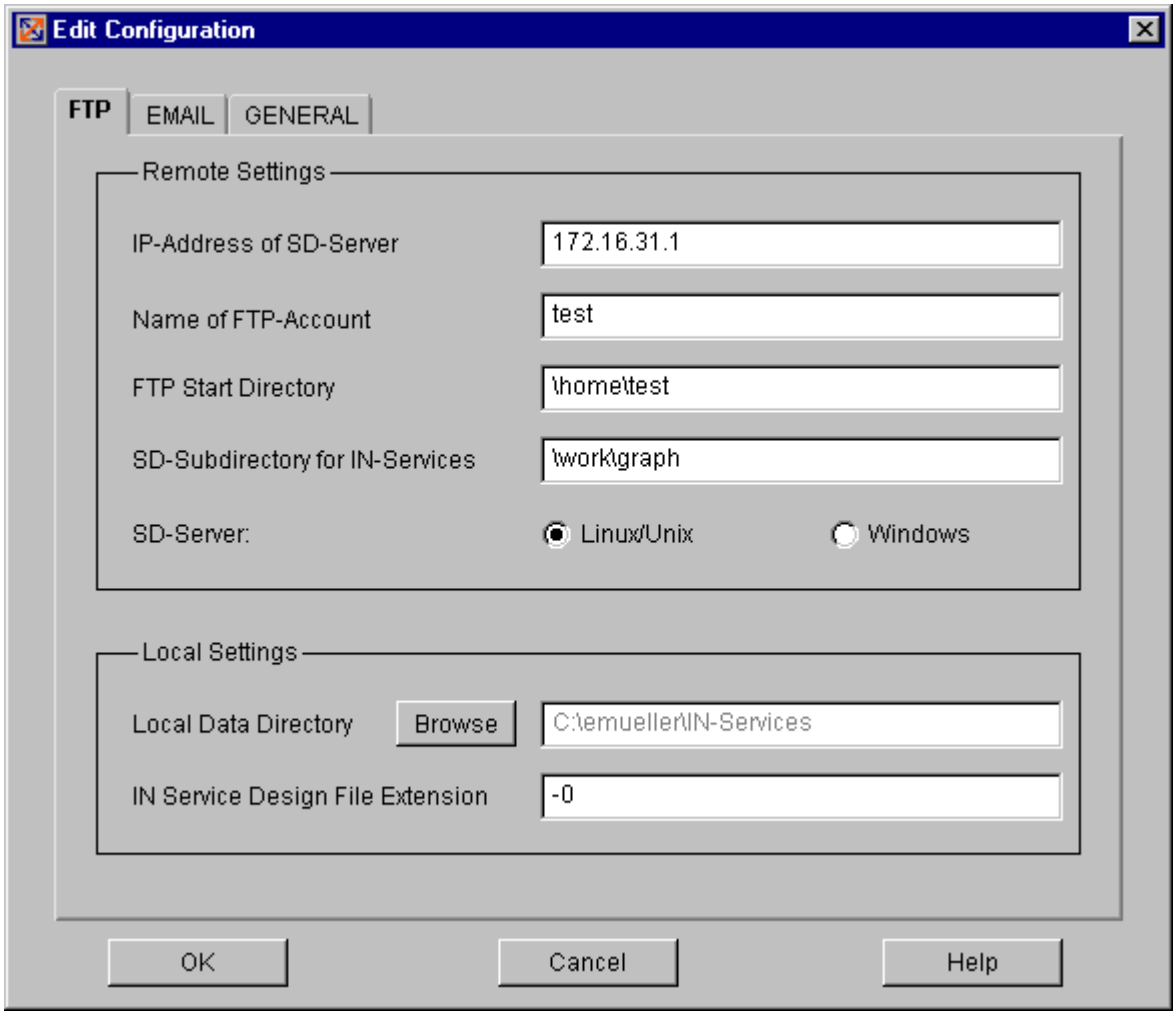

Figure 3: Configuration data of IN Tele-Experimenting: FTP-Section

In the FTP-section this data has to be edited:

- IP address of SD machine
- Name of FTP user account on SD machine
- FTP Start Directory on SD machine
- SD-Subdirectory for IN-Services on SD machine (default: \work\graph)
- Local Data Directory on Client machine
- IN-Service file extension (default: -0)
- SD operating system: Linux/Unix or Windows (default: Linux/Unix)

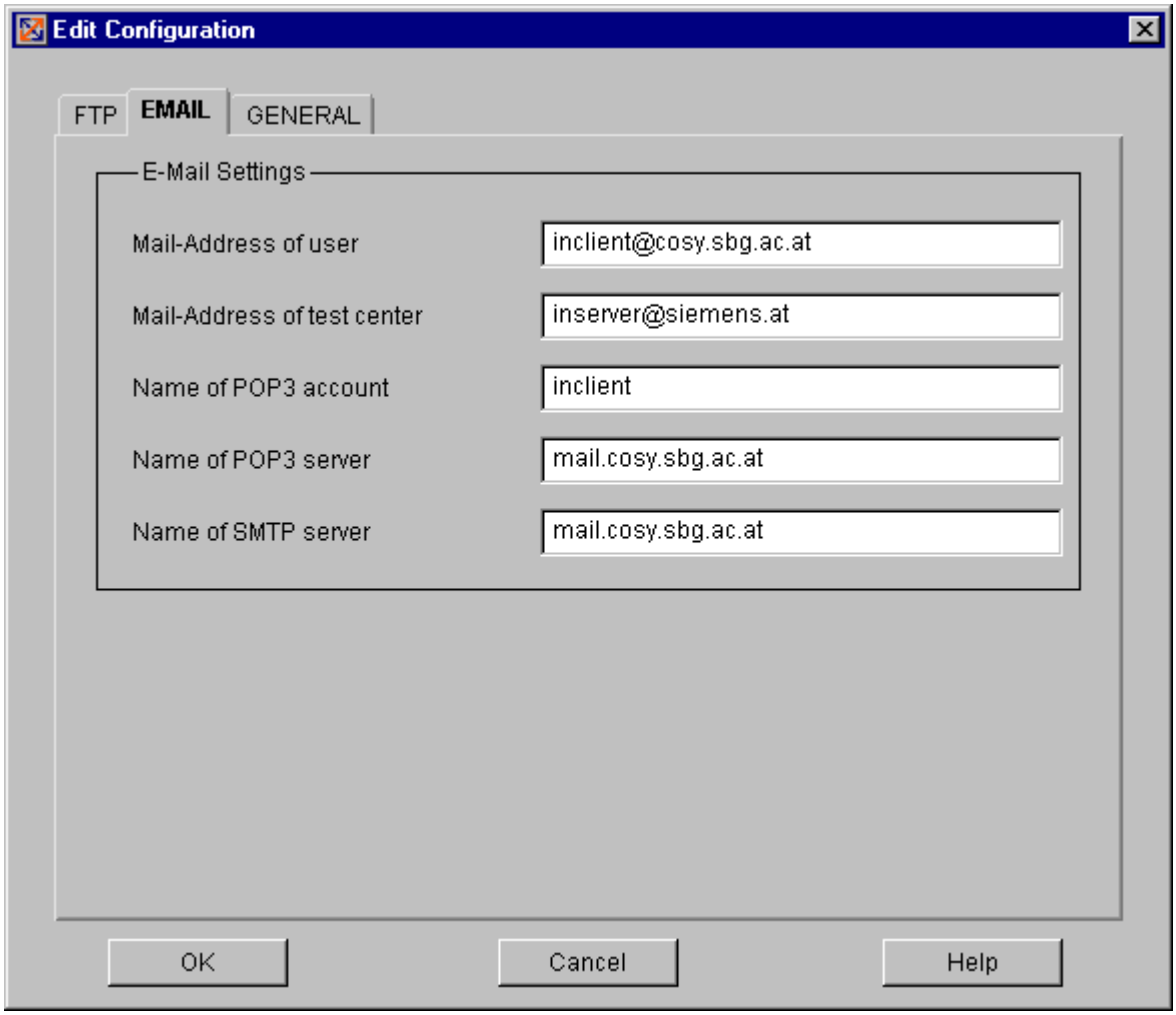

Figure 4: Configuration data of IN Tele-Experimenting: EMAIL-Section

In the EMAIL-section this data has to be edited:

- mail address of user on client
- mail address of Test Center
- Name of POP3 account
- Name of SMTP server

After configuration all connections can be established.

The User-ID is automatically extracted from the response of the server to the Registration Request and stored in the config-file on the client.

2.2.2 Description of connections

1.: FTP-Connection to SD machine by client:

Before first using the FTP connection to the SD machine the configuration data has to be entered in the form of figure 3.

After activating the corresponding menu entry the user has to enter the password of the FTP account. Then a list of all available IN-Services on the SD machine is shown.

The selected IN-Service can be downloaded and after a successful download it is visualised as an entry in the list of local IN-Services on the client.

2.: The structure of the automatically generated mails:

Registration Request:

- Titel: "Request for Registration"
- Body:empty
- Attachment:no

Response of Registration Request:

- Titel: "Re: Request for Registration"
- Body:User-ID
- Attachment:no

Test Request:

- Titel: User-ID, IN-Service, Test Request
- Body:Mail with text created by the user
- Attachment:Files: IN-Service, Test Request

Response on Test Request:

- Titel: Re: User-ID, IN-Service, Test Request
- Body:Comment created by the operator
- Attachment:Files: Report

3. Unified Architecture for Tele-Experimenting

3.1 Client-Server Architecture of UTEA

Remote Testing is a reasonable way to make a better use of expensive experimenting equipment for a large number of users.

For the realisation a large quantity of programming effort is necessary. To avoid a "multiplication" of this effort for different REX applications it was analysed, which components and interfaces of a REX system can be a part of a general REX architecture and which are application specific [3],[4].

Two types of experimenting are possible: online and offline (batch-mode, e. g. the IN Tele-Experimenting). The architecture covers both types. The next figure shows the general system for online-experimenting.

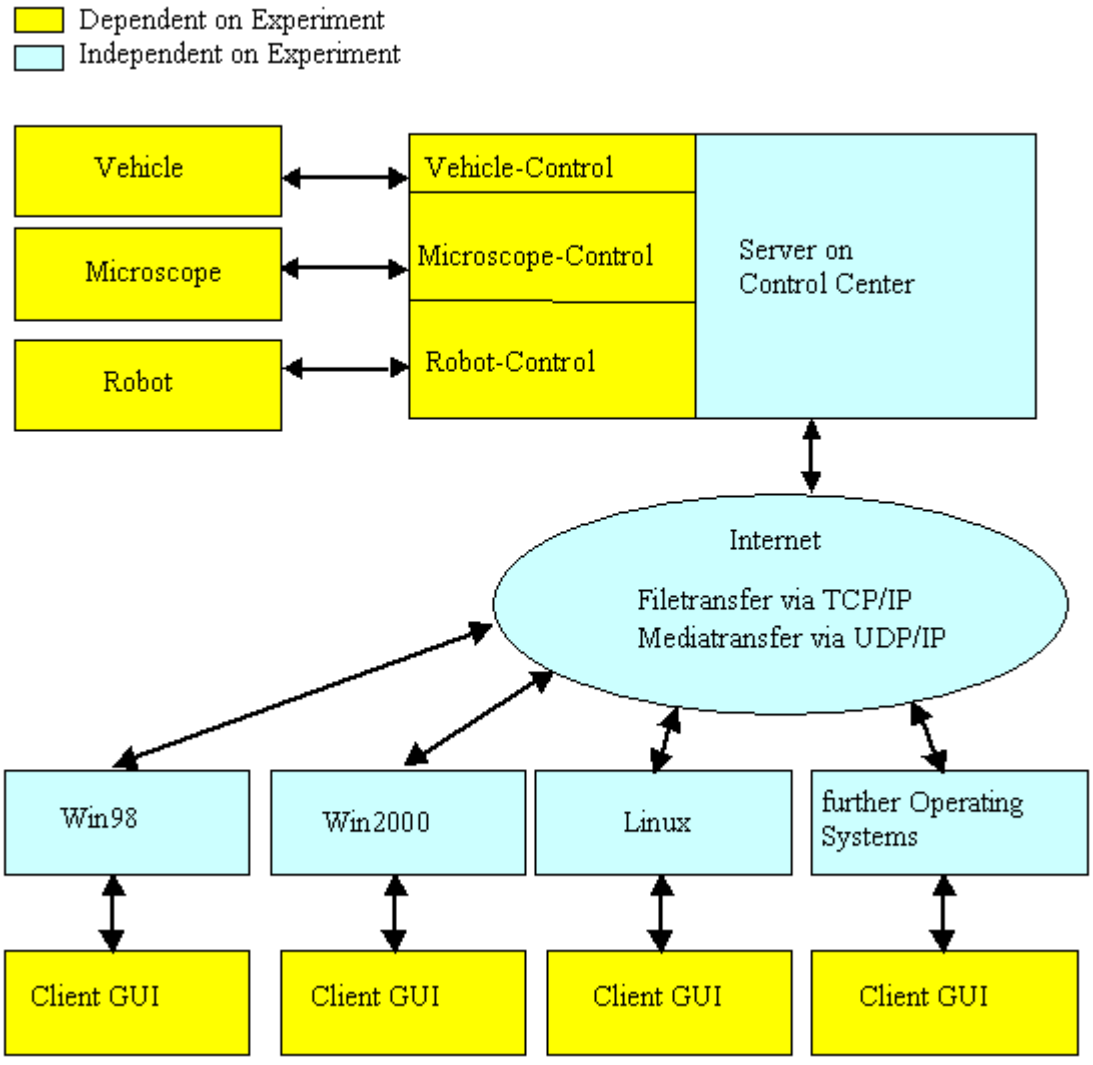

Figure 5: Client-Server-Architecture of UTEA

The Equipment for executing experiments is located at a Control Center. The Server in the Control Center is responsible for the connections to the Test Execution Equipment and to the clients.

At first a client has to get a registration from the Control Center via Internet. After this procedure the client is authorised to make one ore more types of experiments. Before executing any experiment the client has to book a free time slot at the scheduler. A short time before the beginning of the reserved time the client receives a remembering and has to connect to the server to get the control for executing the experiment. The actual state of the experiment (data, video and voice streams ...) is sent back to the Client.

References

[1] Lugger, Benjamin: *Remote Experimenting für QOS-Management in Diffserv-Netzen - Diplomarbeit*, Fachhochschule Salzburg, 2001

[2] Lugger, Benjamin: FH Salzburg (Austria): REX-DiffServ for QoS-Management in IP-Networks, contributed at DETECH 2001 Workshop, Maribor, 2001

[3] Pommer, Hermann: *Software-Design für Tele-Experimenting* - Diplomarbeit, University of Salzburg, 2001

[4] Veichtlbauer, Armin; Müller, Eberhard; Pommer, Hermann: Unified Architecture for Tele-Experimenting, University of Salzburg, contributed at DETECH 2001 Workshop, Maribor, 2001

[5] Schimpf, Andreas: *Client/Server-Konzepte: der Einstieg für Datenbank-Entwickler und -Entscheider*, Haar bei München, Markt-&-Technik-Verl.. 1995

[6] Orfali, Robert: *Client-server programming with Java and CORBA*, New York, NY [u.a.], Wiley, 1998

[7] Orfali, Robert; Harkey, Dan; Edwards, Jeri: *Client/Server Survival Guide*, 3rd Edition, Wiley, 1999

[8] The Solution Provider Lexikon Intelligentes Netz, Siemens AG, http://w3.siemens.de/solutionprovider /_online_lexikon/4/f000684.htm

Authors

Müller, Eberhard, Dipl. Phys.
University of Salzburg, Software Quality Lab
Jakob-Haringer-Str. 1
A-5020 Salzburg
emueller@cosy.sbg.ac.at

Pommer, Hermann, Dipl. Ing.
University of Salzburg, Software Quality Lab
Jakob-Haringer-Str. 1
A-5020 Salzburg
hpommer@cosy.sbg.ac.at

Veichtlbauer, Armin, Dipl. Ing.
University of Salzburg, Software Quality Lab
Jakob-Haringer-Str. 1
A-5020 Salzburg
aveichtl@cosy.sbg.ac.at

Hofmann, Ulrich, Prof. Dr. Ing. habil.
Polytechnical University of Salzburg, TKS
Schillerstr. 30
A-5020 Salzburg
ulrich.hofmann@fh-sbg.ac.at