Åsmund Sand

Applying the Internet to Instrumentation and Metrology

Thesis for the degree philosophiae doctor

Trondheim, December 2006

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Electronics and Telecommunications

NTNU
Norwegian University of Science and Technology

Thesis for the degree philosophiae doctor

Faculty of Information Technology, Mathematics
and Electrical Engineering
Department of Electronics and Telecommunications

ISBN 978-82-471-0024-0 (printed version)
ISBN 978-82-471-0038-7 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2007:2

Printed by NTNU-trykk

Preface

This thesis is submitted in fulfillment of the requirements for the degree "Philosophiae Doctor" at the Norwegian University of Science and Technology (NTNU). The research was funded in part by and carried out at both Justervesenet and UniK - University Graduate Center. Parts of the work was also done at National Physical Laboratory in England.

Acknowledgments

Professor Tor Fjeldly at UniK and Doctor Harald Slinde at Justervesenet supervised me during these four years. I wish to thank for all support and ideas. I could not have done this without them.

The funding jointly provided by UniK and Justervesenet is gratefully acknowledged.

I would like to thank Graeme Parkin and Mike Stevens at National Physical Laboratory for the cooperation and the ideas for the generic system. I hope the cooperation will continue in the future.

Last I wish to thank my beloved girlfriend, Trine Gabrielsen, for all her patience and encouragement.

Summary

The work describes useful approaches to applying the Internet in electrical metrology and calibration.

The work comprises several approaches to Internet-enabled metrology, including running remote measurements and remotely operating instruments. The author proposes new ways of developing Internet-enabled instrumentation systems, focusing on increasing the availability while preserving the security.

Abbreviations

AC	Alternating Current
ACI	AC Current
ACV	AC Voltage
DC	Direct Current
DCI	DC Current
DCV	DC Voltage
DLL	Dynamic Linking Library
DMM	Digital Multimeter
DUT	Device Under Test
GPIB	General Purpose Interface Bus
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
ICT	Information and Communications Technology
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MAC	Message Authentication Code
NAT	Network Address Translator
NMI	National Metrology Institute
OTDR	Optical Time Domain Reflectometer
PC	Personal Computer
PCI	Peripheral Component Interconnect
PXI	PCI eXtensions for Instrumentation
RS-232	Recommended Standard 232
SCPI	Standard Commands for Programmable Instrumentation
SSL	Secure Sockets Layer
TCP	Transfer Control Protocol

USB	Universal Serial Bus
VME	Versa Module Eurocard
VXI	VME eXtensions for Instrumentation

Definitions

Metrology "the science of measurement - set of operations having the object of determining a value of a quantity"[1]

Calibration "set of operations that establish, under specified conditions, the relationship between values of quantities indicated by a measuring instrument or measuring system, or values represented by a material measure or a reference material, and the corresponding values realized by standards"[1]

Traceability "property of the result of a measurement or the value of a standard whereby it can be related to stated references, usually national or international standards, through an unbroken chain of comparisons all having stated uncertainties"[1]

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Present work

1.1.1 Background

Every day, thousands of electrical instruments are transported between laboratories across the world. Some of these instruments are transported to calibration laboratories to be calibrated, which means checking their measuring capabilities against high-precision standards. This process has several disadvantages. First, the instruments are calibrated in an environment different from their normal conditions, which means it is difficult to tell how they perform when returned to the owner. Second, the instruments are often influenced by transport in a non-predicting way, resulting in increased uncertainty in the calibration result. Third, the instrument is often out of operation for several weeks, which could be very expensive for the owner.

The metrology community has therefore started to look at ways of disseminating calibration values from the calibration laboratories to the instruments, so that the latter could be calibrated directly in the owners' laboratories. Since most instrumentation today is computer-based, it is a natural extension to include the Internet in the instrumentation systems. If developing computer systems that could control and run calibration processes from remote, it would allow accredited operators to perform calibrations from their own office. This work will discuss new ways of utilizing the Internet to achieve this . There are several challenges, concerned with security, accessibility, dissemination of calibration values, user trust and user training.

The main goal of this work has been to look for new ways of utilizing the Internet in the metrology community. The world is getting more connected, and

it is important that the community try and utilize new technologies to increase the quality of their work. As will be seen, there is a potential to allow for higher-precision calibrations than are available today, by including the Internet in the process.

In 2002 the University Graduate Center at Kjeller (UniK) [2] and Justervesenet (JV) [3] agreed to engage in a joint research project regarding applying the Internet to electrical metrology and calibration. Justervesenet was just starting to investigate the possibilities of integrating the Internet in electrical measurements, and they needed partners with knowledge in the area. UniK, experienced in the Internet-enabled metrology field [4],[5],[6],[7],[8],[9],[10],[11],[12],[13], was looking for partners to co-fund Ph.D. scholarships for their students. The working title of the current project was "Applying the Internet to Electrical Metrology and Calibration".

1.1.2 Motivation

As the author of this work it is appropriate to state the reasons for choosing the current subject for my Ph.D. thesis. I have a background in physics and information technology, and I am greatly interested in how the Internet can be utilized to solve everyday tasks in new and practical ways. Through the Ph.D. scholarship above, I was given the opportunity to explore how the Internet may be utilized in the metrology community. This was a great way of combining my background with my interests in the Internet. I also find the interface between the physical and virtual reality very interesting.

Justervesenet wanted to look at effective and secure ways to integrate the Internet in electrical metrology and calibration. Being a National Metrology Institute (NMI), Justervesenet is the leading and highest-level measurement and calibration authority in Norway, and they perform a number of calibrations each year. There is potential to reduce the costs and increase the measurement quality by utilizing the Internet in this process.

Each of Justervesenet's customers need to calibrate different kinds of electrical devices on a regular basis. This process requires the transport of these devices from the customers to Justervesenet, where the calibration process is performed. This way of doing measurements and calibrations introduces uncertainties at several levels. The benefits of integrating the Internet in the process can be seen in the list on page 16.

There is a growing trend to integrate the Internet in the metrology field all over the world. Justervesenet wanted to look at a more general approach to Internet-enabled metrology as well, not limited to Internet-enabled calibration. Different solutions were considered with regard to general remote instrument operation.

1.1.3 Summary

The current work has focused on implementing experimental instrumentation systems, which could solve different metrology tasks. Three instrumentation systems have been developed, comprising different functional parts. The systems have been compared with existing systems. Different means of communicating via the Internet have been investigated. The work also includes several solutions for instrument operation, either directly, or indirectly via measurements.

A lot of the work has been experimental, and much time has been put into implementing or modifying different system requirements. The focus has been on developing practical and smart guidelines that may be used by other NMIs when integrating the Internet into the instrumentation systems.

The output of the work has been a new way of performing Internet-enabled calibrations, which has a lot of potential if a well-suited way of disseminating calibration values is found. Compared to other similar systems, the current work has focused on availability, scalability and security. Nearly all laboratory computers may be used with the system, as long as an Internet-connection is available.

iMet v.1.0

The first system, iMet v.1.0, was developed with Internet-enabled calibration in mind. It allows an operator at Justervesenet to control a calibration process directly at one of its customers, as described in Fig. 1.13.

Traditional Internet-supported measurement systems are not able to solve this task, because they do not allow outside-personnel to communicate with electrical equipment protected by firewalls or proxy servers in a general way. New ways of bi-directional Internet-communication through computer network protection systems have been investigated.

Some firewalls and dedicated proxy servers only allow clients on the inside to communicate over the Internet using the HTTP protocol. The HTTP protocol is by nature asymmetric, and it is traditionally used in a request-response scenario, e.g. for requesting a web page from a server. Using Microsoft .NET, a new communication channel was set up, allowing bi-directional HTTP traffic. This allowed two LAN computers, separated by the Internet, to communicate via a public web server, and solved many of the challenges concerned with controlling a calibration process remotely in a general way.

The iMet system v.1.0 is presented in chapter 2.

Comparison with other systems

The first version of the iMet system, was compared with NPL's system for Internet-enabled calibration. A comparison study was performed at NPL in Teddington, England, in December 2004. The two systems were tested and compared using the list in 1.7, which in large was created during the comparison. The results of the comparison revealed both differences, e.g. in area of use and role of the operator, and similarities, e.g. hardware interface support.

The comparative study is presented in chapter 3.

iMet v.2.0

The second version of the iMet system was developed with the comparison study in mind. Some weaknesses of the system needed attention, and several areas for potential improvement had been identified. The new system was increasingly focusing on a dynamic approach to performing calibrations and measurements. Measurements no longer had to be preinstalled, but could be downloaded from a dedicated database server. This made the system increasingly more scalable, as new measurements could be added to the system without the need for a system recompilation or restart.

This system is presented in chapter 4.

Joint project with NPL

The latest system is the result of a joint effort by Justervesenet and NPL to develop a highly adaptable instrumentation system, which automatically adjusts to the equipment in use. As for iMet v.2.0, new measurement procedures are added to a database. The main difference is that support for new equipment may be added to the database as well. This means that the system may be used to run any measurement or calibration routine using any PC-connected device.

There was a need to develop a system that could communicate with instruments over non-standard hardware interfaces. The software, linking the control application and the hardware interface drivers, has been made available from a dedicated database server. This allows the control application to add support for new and non-standard hardware interfaces at runtime, with no prior knowledge of the hardware, and the system is thus very scalable.

This system is presented in chapter 5.

Presentations and articles

The systems above and the comparative study have been presented at several workshops and conferences:

- Workshop on Internet-Supported Measurements - Future Solutions, Kjeller, Norway, 21 April 2004
- Workshop on Middleware Development, Norwegian Defence Research Establishment (FFI), 4 June 2004
- Conference on Precision Electromagnetic Measurements, London, UK, 27 June - 2 July 2004 [14]
- International Workshop: From Data Acquisition to Data Processing and Retrieval, Ljubljana, Slovenia, 13-15 September 2004 [15]
- NMIJ-BIPM Workshop on the Impact of IT in Metrology, Tsukuba, Japan, 16-20 May 2005, [16]
- VII Advanced Mathematical and Computational Tools in Metrology (AMCTM 2005) Conference, Caparica, Portugal, 27-29 June 2005, [17]
- 2006 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, La Coruña, Spain, 10-12 July 2006, [18]

The following articles have been published of accepted for publication:

- IMet - A Secure and Flexible Approach to Internet-Enabled Calibration at Justervesenet [19]
- A Dynamic Instrumentation Framework for Remote Operation of PC-Connected Devices [20]
- A Secure Approach to Distributed Internet-Enabled Metrology [21]
- Internet-Enabled Calibration: An Analysis of Different Topologies and a Comparison of Two Different Approaches [22]

1.1.4 European survey on ICT tools usage

In 2005, a survey was conducted to find the current status of ICT tools usage in European National Metrology Institutes under the iMERA project [23] funded by the European Commission. The results will be used to decide where to put efforts in order to increase the degree of ICT compatibility between the NMIs. In the future, it will be important for the different NMIs to communicate and collaborate in a seamless way via the Internet. It is also necessary for the NMIs to coordinate their research work.

Parts of the survey are presented in Appendix A.

1.2 Previous work

Several NMIs and calibration laboratories have worked in the area of Internet-enabled metrology. The following is a list of historical and ongoing projects for some of the largest NMIs in the world (the list can also be found in [24]):

- **National Physical Laboratory (NPL[25])**:
  - iPIMMS: Radio frequency (RF) impedance measurement system for calibration vector network analyzers against primary standard artifacts [26]
  - iVR: Voltage and resistance measurement system for calibrating secondary standard voltage sources and resistors against the Fluke 4950 multi-function calibrator, used as a highly characterized transfer standard [27]
  - iCOLOUR: Visible spectrum reflectance/color measurement system for calibrating spectrophotometer color measuring systems against standard tiles of known reflectance [28]
  - iOTDR: Optical time-domain reflectometer (OTDR) measurement system for calibrating OTDR devices against standard optical fiber artifacts [29]
- **National Institute of Standards and Technology (NIST[30])**:
  - SIMnet: the use of Internet-enabled Metrology in inter-laboratory comparisons in the Inter-American Metrology System (SIM) [31]
  - MeasureNet: The same technology used in SIMnet has also been used in MeasureNet since 2000 to support training and collaboration between NIST and State weights and measures laboratories [32]
  - Electrical calibration: Also since 2000, the Internet-enabled calibration capability used in SIMnet has been used to provide a remote calibration service to a few of NIST's customers, specifically for on-site calibration of multi-function calibrators using traveling digital multimeters (DMMs) [33]
  - Dosimetry calibration services: In the e-calibration service, the user logs into the NIST-supplied program resident on their PC. This local program sends a request to the NIST server for authorization. The local program gets approval from the server then provides the user with instructions and stores data as acquired. Upon conclusion of the session, an encrypted log of the calibration data is transferred to the NIST server and a provisional certificate is sent to the user.

Calibration session spectra and data are evaluated by NIST staff; and a final certificate is prepared, signed and sent electronically to the user [34]

- **Physikalisch-Technische Bundesanstalt (PTB[35])**:
  - Coordinate measuring machine calibration and monitoring [36]
  - Electrical calibration [37]
  - Collaboration with NPL on AC Josephson voltage standard [38]
  - Tele-calibration for the Primary High-Pressure Natural Gas Standard
  - Potential collaboration with NMIJ
  - Audio and video communication
  - Web-based data acquisition, storage and access
  - Internet security
- **Nederlands Meetinstituut (NMi[39])**:
  - Internet-enabled maintenance of a self-calibrating electrical calibrator
- **National Metrology Institute of Japan (NMIJ[40])**:
  - Josephson standard
  - Time and frequency
  - Optical frequency
  - Length
  - AC-DC difference
  - Radioactivity
  - Coordinate measuring machines
  - Temperature
  - Pressure
  - Flow

1.3 Historical background

1.3.1 Personal Computers in Metrology

Early instrumentation systems required the operators to manually operate the instruments, and the measurement process was often tedious and error prone due to human interaction when setting up the equipment, entering data input, reading data output and handling errors. In the mid 1970s, a few companies started developing specialized hardware interfaces and equipment, which enabled scientists to connect electrical laboratory equipment to regular personal computers (PCs) as seen in Fig. 1.1. The instrument operators no longer had to use the physical knobs and displays on the physical instruments, but could utilize the computing power and display capabilities of PCs.

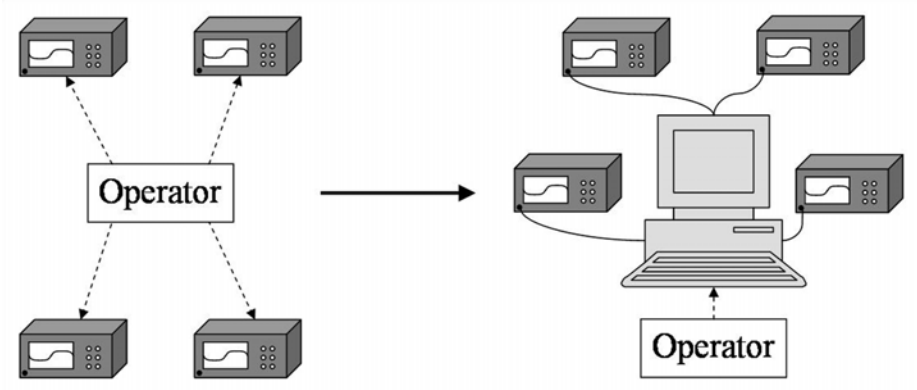

Fig. 1.1: The introduction of personal computers in the metrology field. From operating each instrument manually, the operator now could operate many instruments via one PC.

Hewlett-Packard[41] developed the Hewlett-Packard Instrument Bus (HPIB) card, later named the General Purpose Interface Bus (GPIB[42]) card, which has become the world-leading bus[1] in the instrumentation field. In the following years many PC-based instrumentation solutions were developed, including new instrumentation buses, hardware cards and instruments. The trend today is to use a PC as an instrument controller, and to standardize the interface between the instruments and the PC. New stand-alone instruments quite often also support standard PC buses like Universal Serial Bus (USB[43]) or Ethernet[44], which are common buses on most regular PCs.

[1] An instrumentation bus is a collection of wires, that connects a computer to peripheral instruments or interconnects modular instruments.

Bridge technology also enables e.g. GPIB instruments to be connected via standard USB. An instrumentation bridge acts as a relay between different instrument buses, so that they may be connected to the same, common PC bus. This process is shown in Fig. 1.2.

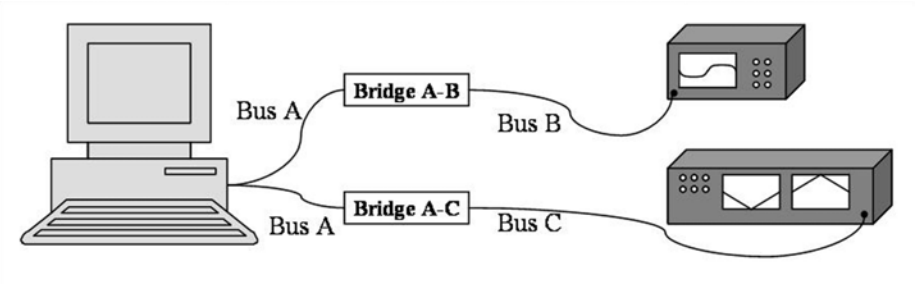

Fig. 1.2: Bridge technology enables instruments with different instrument buses to connect to the same, common PC bus.

1.3.2 Instrument bus technologies

For more than twenty years, the measurement community has regarded the GPIB bus as the leading instrument bus, due to its robustness and simplicity. However, it is not integrated in new PCs, and costly hardware cards are needed to be able to communicate using this interface, if not using some kind of hardware bridge (which can also be costly). Although the number of stand-alone instruments using GPIB is quite large, newer equipment quite often comes with an alternative interface, like USB, IEEE 1394 (Firewire[45]) or Ethernet. Other types of instruments, so-called modular instruments, use integrated buses like VME Extension for Instrumentation (VXI[46]), which is based on the Versa Module Eurocard (VME) bus, and PCI Extension for Instrumentation (PXI[47]), which is based on the Peripheral Component Interconnect (PCI) bus. These buses are integrated into a rack of instruments, which enables fast and reliable inter-instrument communication.

The following is a summary of the most common buses for stand-alone instruments:

- **IEEE 488**
  This bus was initially called the Hewlett-Packard Instrument Bus. It is a digital 8-bit parallel communications interface with data transfer rates up to 1 MB/s. Up to 15 instruments and a controller, usually a PC, may use one GPIB bus. The GPIB bus specification has been extended three times

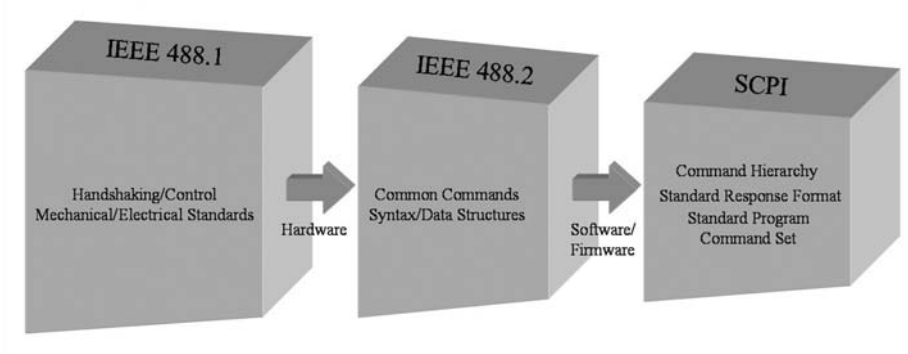

Fig. 1.3: The structure of the GPIB standards. IEEE 488.1 was the first specification, describing technical bus features. IEEE 488.1 addressed problems that had arisen in the original IEEE standard, and standardized the instrument communication protocol. SCPI further standardized the IEEE 488.2 command set.

as seen in Fig. 1.3. GPIB cables and connectors can be fully screened and may be used in most environments.

- **RS232**[48]
  RS-232 is a standard for serial binary data exchange, and was originally developed to interconnect a data terminal with a modem. Though gradually replaced by the more popular USB bus, the standard is still quite common in the instrumentation field.

- **USB**
  This is a fast serial bus integrated in most new computers and laptops, in contrast to IEEE 488 and RS232. USB 1.1 has a transfer rate of up to 1.5 MB/s, while the newer USB 2.0 has a maximum transfer rate of 60 MB/s. USB cables are generally not screened, and are not suited for all environments.

- **IEEE 1394**
  This is a high-performance serial bus, supporting transfer rates of up to 800 Mbps. It is not as often integrated in new PCs as USB, and the cables are not screened. IEEE 1394 differs from USB in that there is an existing protocol defined for controlling instruments over the IEEE 1394 bus.

- **Ethernet**
  Recently, instrument manufacturers have started to include Ethernet connections in their stand-alone equipment. Ethernet connections are found in most PCs, and the maximum transfer rate is 1 Gbps. Due to the existence

of Ethernet networks in most instrumentation companies, this means that Ethernet equipment could easily be integrated, though security measures must be taken.

The most common buses for modular instruments can be summarized as follows:

- **VXI**
  VXI was the first attempt to standardize instrumentation systems for use in the industry. Its goals were among others increased interchangeability and system throughput, and decreased size and costs.

- **PXI**
  PXI is a rugged, high-performance, and inexpensive deployment platform, based on the PCI bus, for measurement and automation systems. It is used for integrating modular instruments from different vendors and integrate them in one PXI system, with integrated timing and synchronization resources.

1.3.3 Virtual instrumentation

When PCs entered the instrumentation field, the measurement processes could be set to run automatically without human interaction. This reduced the error rate concerned with data input and output, and traditionally tedious measurement processes could be run more rapidly. Virtual instrumentation is a joint description for software used to control and communicate with physical devices and to handle the data acquired from them. Data handling includes acquiring, performing calculations on, presenting and storing the data. As the PCs became more common, the so-called virtual instruments were introduced.

A virtual instrument is a piece of software running on a PC, which acts on behalf of a real physical instrument, and the user of the virtual instrument may use it as if it were the real instrument. The virtual instrument handles all communication between the user and the physical device. The concept is shown in Fig. 1.4. Virtual instruments may also emulate the behavior of physical instruments.

Early challenges

From the beginning, virtual instrumentation introduced a few technical challenges:

- Proprietary and non-standardized hardware interfaces required users to install hardware cards in the PC for each instruments to operate

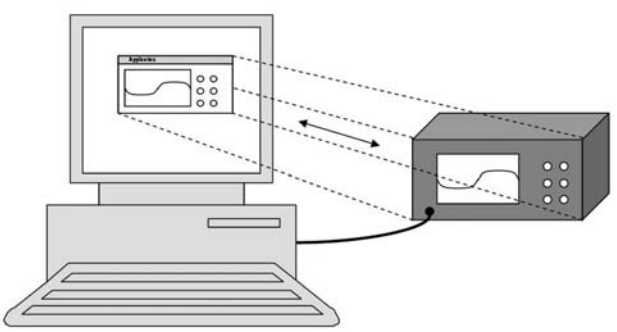

Fig. 1.4: The concept of a virtual instrument. The instrument operator may use the virtual software instrument as if it was the real instrument. The communication with the real instrument is hidden from the operator, and is handled by the underlying software.

- Professional programmers were needed to develop the virtual instruments
- Great variety of instrument command sets prevented code reuse

Today most PC-connected instruments can be set to communicate over standard PC interfaces, either directly or by using bridge technology, though there still are some older instruments using proprietary hardware interfaces. The use of standard PC interfaces simplifies the process of connecting new equipment to a PC without the need to open its cover to insert a hardware card. It also permits laptops to be used, which often have little room for additional hardware cards.

A lot of work has been done in the software development tool area, with special focus on usability and functionality. The programming of virtual instruments may be done using graphical programming tools, like National Instruments'[49] LabVIEW[50], which means that scientists, without special programming skills, may create quite complex virtual instruments relatively quickly. An example is shown in Fig. 1.5. There are also good solutions for people who are more used to textual and object-oriented programming languages. An example using Microsoft C#[51] (pronounced "C Sharp") is shown in Fig. 1.6.

As to the great variety of instruments command sets, there is a trend towards trying to match similar functionality with similar commands. The SCPI Consortium[52] has developed the Standard Commands for Programmable Instrumentation (SCPI[53]), which is a standardization of common instrument commands. An example is the string "*IDN?" for acquiring instrument identification.

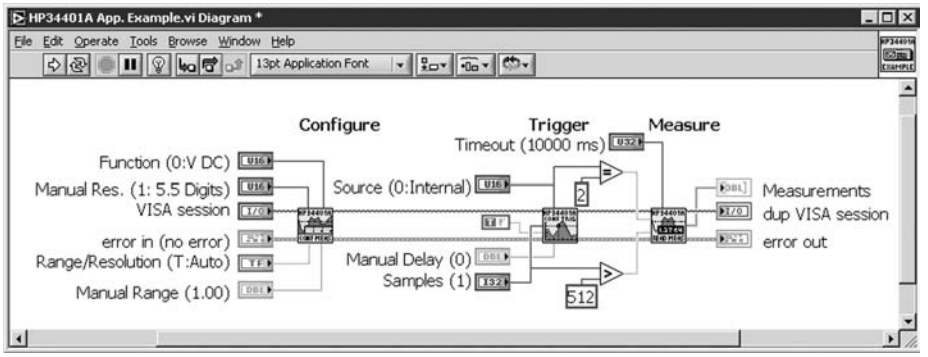

Fig. 1.5: The LabVIEW programming language. Programming flow is represented by lines (left to right).

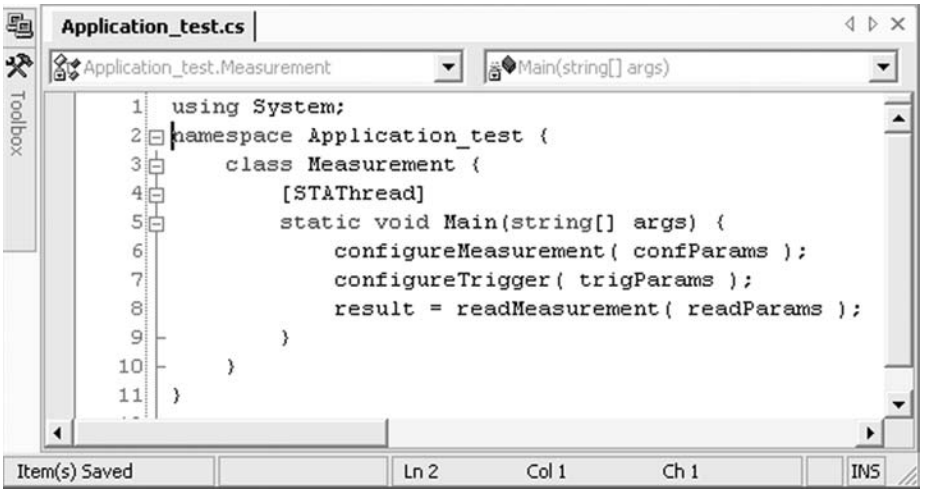

Fig. 1.6: The C# programming language. Programming flow is represented by text lines (top to bottom).

Virtual Instrumentation Software Architecture

A common problem with traditional control applications is that low-level driver communication happens directly in the control application. This becomes a problem when adding new hardware interfaces to the system, which requires the control application to be rewritten. The Virtual Instrumentation Software Architecture (VISA[54]) solves this problem, as it takes care of

the bus communication. That is, it provides a programming interface between the hardware and several development environments like LabVIEW[50], LabWindows/CVI[55], Measurement Studio[56], Microsoft Visual Studio[57], and Sun's Java[58]. This process is shown in Fig. 1.7. VISA comprises GPIB, VXI[46], PXI[47], RS232, Ethernet, and USB interfaces. Using VISA, a programmer may choose between several programming languages, develop the control application, and then operate instruments without ever knowing how the bus driver works.

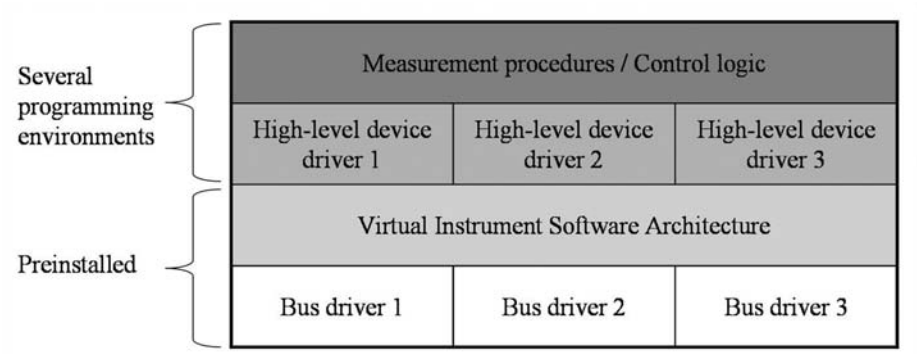

Fig. 1.7: The Virtual Instrumentation Software Architecture enables a control application to communicate over several instrument buses. The user develops the control application and high-level device drivers.

1.3.4 The introduction of the Internet

The introduction of the Internet during the 1990s enabled people to communicate digitally over long physical distances. This communication can be done in a fast and secure way, and today a lot of services are available to the public. These include text chat, video conferencing, IP telephony, document collaboration and online gaming. When including the Internet in metrology, several benefits are obtained compared to traditional metrology:

- fast transfer of measurement results and control signals over long physical distances
- physical separation of expertise and equipment
- online availability of measurement results
- online availability of measurement procedures

- sharing and reuse of resources (equipment, expertise)
- consistent measurement procedures

1.4 Metrology and calibration

Calibration within the metrology field deals with determining the correctness of measured values. To be used in real measurements, an electrical instrument periodically needs to be calibrated against known standards. The values measured or generated by the instrument must be traceable to values with known uncertainties, which means determining how much the instrument reading is in error by checking it against a measurement standard of known error. The calibration gives information about the error of the equipment with respect to the accepted reference value. Usually the instrument is sent to a National Measurement Institute (NMI) or a calibration laboratory, where the instrument is calibrated.

A calibration performed at an NMI is of the highest level of accuracy. Usually, calibration laboratories calibrate their equipment at an NMI.

The calibration uncertainties make up a hierarchical tree, as seen in Fig. 1.8, with the uncertainty increasing downwards.

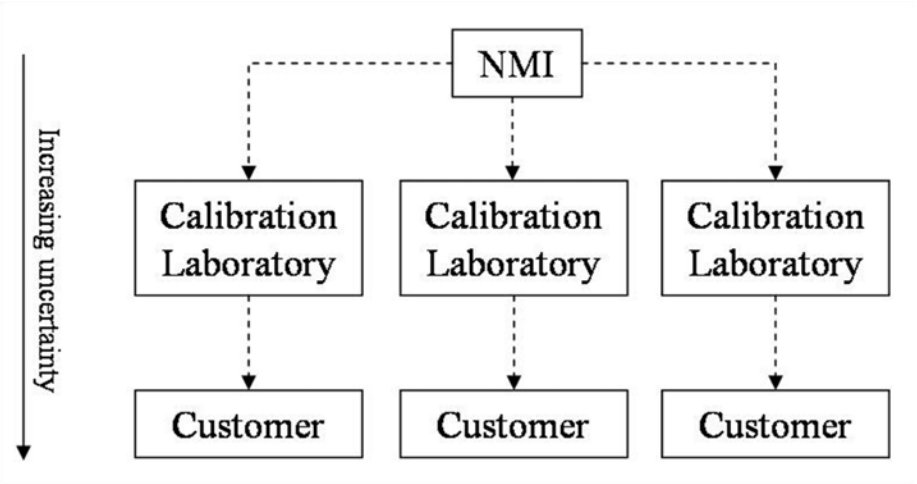

Fig. 1.8: Traceability. The values measured or generated at the customer are traceable to the calibration laboratory's values, which again are traceable to the NMI's standards. The uncertainties increase toward the customer.

1.5 Internet-enabled metrology and calibration

The traditional way of performing calibrations is to send the Device Under Test (DUT) to a calibration laboratory where the calibration takes place. This is shown in Fig. 1.9. In recent years the international metrology community has started to explore the possibilities of remotely monitoring and controlling measurements and calibrations via the Internet [59, 60, 61, 62].

When performing remote calibrations, a calibrated transfer standard is first transported to the DUT as shown in Fig. 1.10. Both the standard and the DUT are then connected to a PC, which again connects to the Internet. The calibration process may be controlled or monitored in many ways. The operator may either be co-localized with the instruments, or separated from them by a network.

The advantages of running calibrations in such a way are many:

- The DUT is calibrated in its working environment, thus lowering the total calibration uncertainty
- The DUT is out of operation for a much shorter period than for traditional calibrations, which is often critical for the owner of the DUT
- The effects of transporting the transfer standard are often much better understood than the effects of transporting the DUT
- If a suitable transfer standard is found, a much better cost to accuracy ratio is obtained
- The DUT owners usually get more involved in the calibration process, which may be a benefit for their normal laboratory work

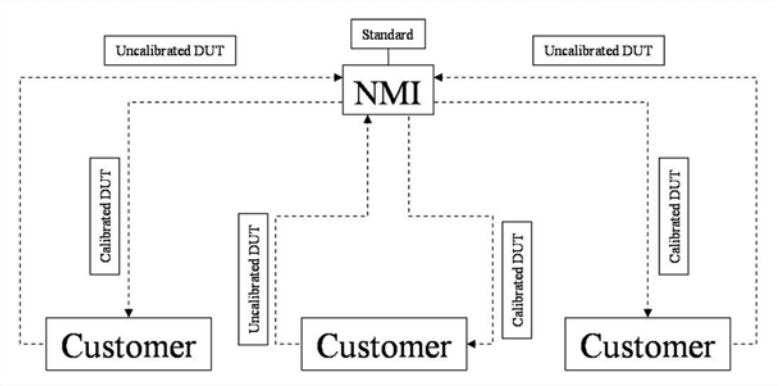

Fig. 1.9: Traditional Calibration. A DUT is sent from each customer to the NMI (or calibration laboratory). The calibration process is then controlled by NMI personnel. When the process is finished, the DUT is returned to the customer.

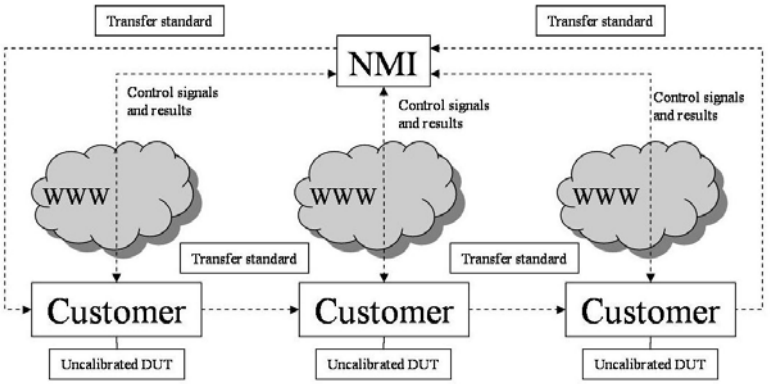

Fig. 1.10: Internet-Enabled Calibration. A transfer standard is sent from the NMI (or calibration laboratory) to a customer. The calibration process is then controlled by NMI personnel via the Internet or by skilled personnel at the customer. When the process is finished, the transfer standard may be sent to another customer or returned to the NMI for recalibration.

1.6 System architectures and network topologies

There are normally three different system architectures that can be used for instrument operation and remote calibration:

1. The operator and instruments are located at the same local area network (LAN) computer connected to the Internet. The measurement control application may be downloaded from a public server or it may be preinstalled on the instrument computer. Measurements may sometimes be downloaded in the form of software routines from the same public server. See Fig. 1.11.

2. The operator uses a LAN computer while the instruments are connected to a public web server or a computer with easy access to a public web server. This architecture is quite useful when setting up online laboratories. See Fig. 1.12.

3. The operator and the instruments are located at different LAN computers separated by the Internet. Instrument control commands and measurements may be sent from the operator to the instruments via a dedicated relay server. Architecture 2 and 3 are similar in that the operator and instruments are separated by the Internet. The latter architecture is more complex due to the instrument-side firewall or proxy server, which only support outbound connections. See Fig. 1.13.

The architectures mentioned have different advantages and disadvantages.

Most of todays Internet-enabled measurement or calibration systems belong to the first two architectures. The third architecture, which is the focus of this work, is not so common, due to the challenges associated with accessibility and security.

Some systems exist for the third architecture, but often some network configuration is needed, like adding new firewall rules, adjusting proxy server settings or changing Network Address Translation (NAT) configurations (effectively changing it to a architecture 2 system). For more information about network security components, see Appendix B. Few companies are interested in opening up firewalls, or configuring proxy servers or NATs, to enable the use of Internet in metrology.

The reason why some network configuration is necessary is that most firewalls, proxy servers and NATs only support outbound network connections. This means that two computers behind different firewalls are unable to communicate directly. Often proxy servers limit which communication protocols are allowed to be used, thus making the inter-communication even more challenging.

1.6.1 Architecture 1 - LAN A to LAN A

Type 1 architecture is quite robust to unstable network connections since the instrument communication is done locally. The instrument communication is also very fast. However, this architecture requires the operator to travel to the instruments to control them.

Many Internet-enabled calibration systems use this architecture type [26], [29], [34]. An application is either downloaded from a dedicated server, or individual measurement procedures are downloaded to a preinstalled control application.

The security of the system is dependent on the ability to securely authenticate the database server and the customer, secure the data traffic between them, and to validate the data downloaded from and uploaded to the database server.

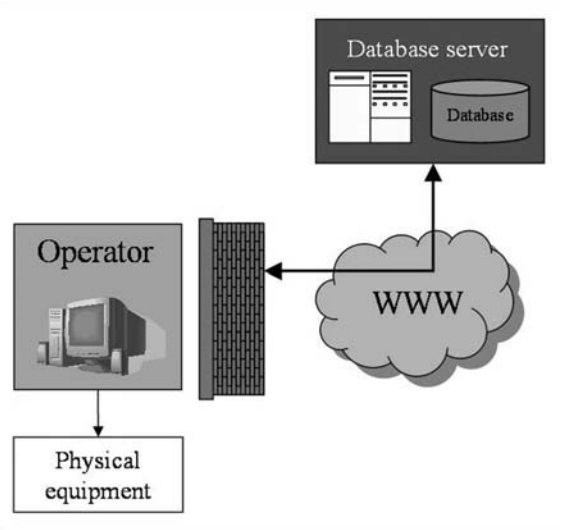

Fig. 1.11: Architecture 1. The operator sits at the LAN computer where the instruments are connected. A database server can be accessed in order to download measurement procedures and historical data or to upload measurement results.

1.6.2 Architecture 2 - LAN A to server

Type 2 architecture promotes the use of very thin clients (often a regular web browser is enough on the client side), thus little needs to be installed on the client computer before using the system. Since the instruments need to be connected to a dedicated web server, this architecture is not suitable for general instrument operation (most instruments are connected to regular LAN computers behind strict firewalls). For direct instrument control the communication is dependent on the available bandwidth.

The security challenges for this system is quite similar to architecture 1. In addition, the instrument server needs to be authenticated. The instrument server also needs extra security routines to limit access to the connected instruments.

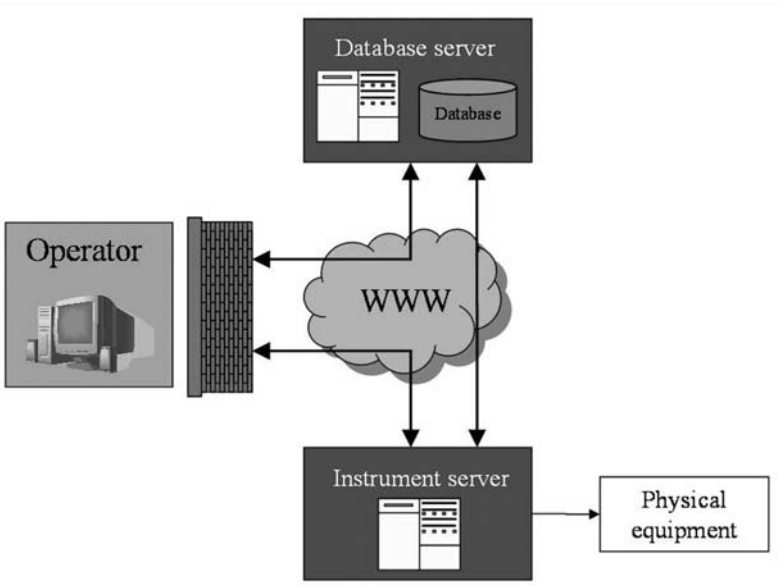

Fig. 1.12: Architecture 2. The operator sits at a LAN computer while the instruments are connected to a dedicated instrument server. A database server can be accessed in order to download measurement procedures and historical data or to upload measurement results.

1.6.3 Architecture 3 - LAN A to LAN B

Type 3 architecture enables users to remotely operate instruments anywhere, as long as the instruments are connected to a computer connected to the Internet. There is also no need for the operator to travel to the instruments to control them, thus third-party experts could easily take part in the control process. However, this architecture can be sensitive to unstable network traffic and network congestions, since persistent connections are often needed when doing direct instrument control. Direct instrument control is dependent on the bandwidth available, and the occurrence of long delay times may distract the operator.

The security challenges of the system is quite similar to architecture 2. In addition, the instrument computer must be able to authenticate the operator and vice versa.

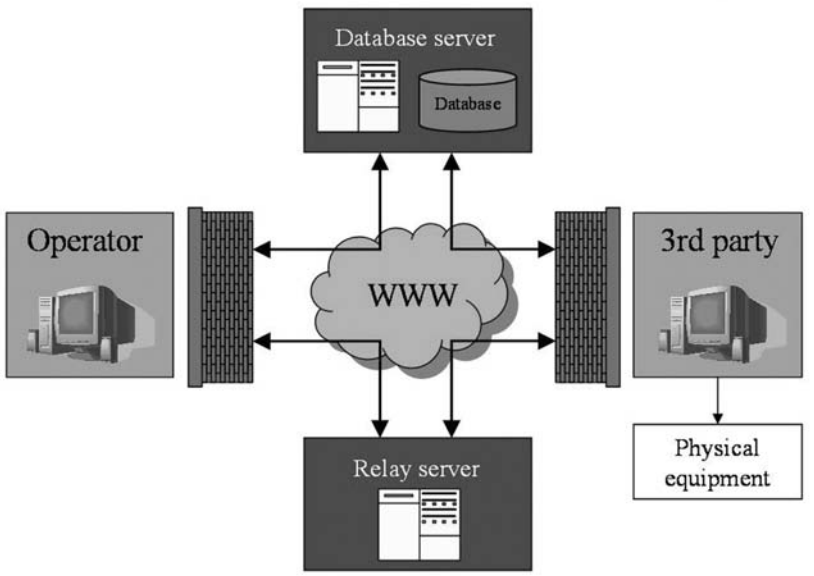

Fig. 1.13: Architecture 3. The operator sits at one LAN computer while the instruments are connected to another LAN computer. The LAN computers are separated by the Internet, and generally located behind several network security entities like firewalls and proxy servers with network address translators (NATs). A database server can be used to store measurement procedures, measurement results and historical data. The relay server is used to enable the instrument computer and the operator's computer to communicate.

Of the three architecture types presented, only types 1 and 3 seem suitable for Internet-enabled calibration, since type 2 requires the setting up and configuration of a dedicated server for each instrument to be calibrated.

The database server, accessible to all users of the systems, enables the operators to work from anywhere. It is also responsible for the consistency of the system.

1.7 Planning a system

Before developing systems used for Internet-enabled calibration, several areas need to be addressed by a national metrology institute (NMI) or a calibration laboratory.

Important parts of an Internet-enabled calibration system include the involved companies, the system users, the system maintainers, the electronic data transferred across the Internet, and the physical system components.

Regarding the users, its important to identify who will use the system and where they will be located. This will affect several aspects of the whole system, including usability, configuration capabilities, communication complexity, and security.

If the system will be further developed and improved in the future, e.g. adding support for new measurement procedures, it should be designed for maintenance. The people responsible for maintaining the system should be equipped with easy-to-use development tools.

The sensitivity of the data transferred across the Internet, such as personal information, calibration data or measurement procedures, will affect the level of security required, like choice of communication protocols, cryptographic techniques, key lengths and security policy.

It is also necessary to decide upon what kind of hardware resources, e.g. computers and instruments, will be involved, so that sufficient security and protection procedures are built into the system. When expensive equipment is used, potential hackers could be more attracted to compromising the security of the system.

The following is a list based on experience from work done by JV and NPL when comparing their systems, iGen and iMet. This comparison is presented in chapter 3. More work could be put into systematizing the list in the future, to enable more general comparisons of Internet-enabled instrumentation systems, perhaps using Common Criteria to specify security requirements. As for the JV-NPL comparison, the list seemed good enough to compare the two systems in question.

1. **The operator**
   A decision needs to be made about who should be the operator: in-house experts, third-party experts, or maybe the customer?

2. **Operator localization**
   This means choosing between one of the three architectures previously discussed. For Internet-enabled calibration this usually means type 1 or 3, as explained in 1.6.3.

3. **Usability**
   The usability of the system is dependent on who should be the operator. Should anyone be able to use the system, or will user training be needed? Should it be a system for experts? What is the level of complexity?

4. **Operator interaction**
   What degree of user interaction is required by the system? Should the operator always initiate and be in control of a process, or could the system be self-driven?

5. **Configuration**
   Often, addresses on the Internet change, and sometimes new communication protocols are needed. Will the system be developed with some kind of configuration possibilities? Can the system be configured using text files, or will the configuration be hard-coded in the assembly (need to recompile for every change made)?

6. **Data handling**
   Should data produced in the measurements be analyzed right away, or should it just be stored for future analysis? Should the analysis be done manually or automatically?

7. **Security**
   The system owner must have a clear understanding of who should be authorized to use the system. If the data generated by the system is sensitive, it should be encrypted.

8. **Availability**
   Should the system always be up-and-running, or will there be downtimes? The system functionality often depends on the stability of the network conditions. Does the system work from all LANs, or is involvement from network administrators necessary? On which type of software platform should the system be able to run? (E.g. Windows, Linux or Mac)

9. **Development**
   Will it be easy to develop and test new calibration routines, or are software experts needed? Should the system be static, or will it be necessary to add extra functionality to the system?

10. **Scalability**
    Will new calibration routines be added to the system? Will this be done while the system is running, or must the system be shut down? A decision needs to be made about which hardware interfaces should be supported and what equipment will be connected. Will it be necessary to add new hardware interfaces or equipment? Must the system support several concurrent users?

The answers to these questions may be used to describe the system requirements. Where available, standardized solutions should be used. It is important to involve different people in the process, like managers, users, IT administrators and the people responsible for maintenance. This way, different perspectives of the same issues can be used to ensure and optimize the qualities of services offered to the customers.

Chapter 2

The iMet system v.1.0

This chapter presents the first version of the iMet system. A secure and firewall-friendly communication channel has been utilized to enable bidirectional communication between two hosts separated by the Internet. While the instruments are connected to only one of them, measurement procedures may be implemented and run on both hosts.

2.1 Background

Many solutions for remote operation of instruments exist today. Common to most of them is that the instruments are connected to a server [59], [60], [61], [62], [63], [64], [9], [10]. As will be seen in chapter 3, some systems co-locate the operator and the instruments on a LAN computer, while instrument information, procedures and measurement data are exchanged with a dedicated server [26], [65]. Few systems exist that allow the operator and instruments be to be separated, without one of them being located at a dedicated web server. The exceptions found, e.g. [66], use symmetrical protocols, like TCP, thus enabling full-duplex channels between operator and instruments. Direct TCP communication is often not allowed through some firewalls or dedicated proxy servers.

2.2 Introduction

The first version of the iMet system is illustrated in Fig. 2.1. It was designed to allow a person to remotely operate and monitor instruments at remote locations, e.g. another LAN. To the operator, the instruments seemed locally connected.

A viable application is Internet-enabled calibration, as described in 1.5 using the architecture given in 1.6.3, where skilled personnel at an NMI control and monitor the calibration process directly at a customer's laboratory.

The calibration procedures were implemented to run locally at the customer's computer, but the NMI personnel were also allowed to send individual instrument commands to operate the instruments directly.

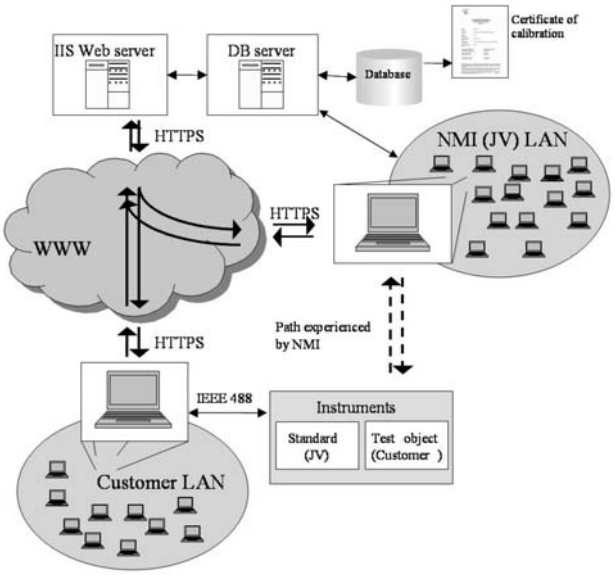

Fig. 2.1: The architecture of the iMet system v.1.0. Personnel at an NMI (JV) may operate instruments and control calibration processes remotely at the customer's site. The Internet Information Services (IIS) Web Server is responsible for relaying the inter-client communication. The database is used to store historical data, measurement configuration data, customer information, and, potentially, certificates of calibration.

2.2.1 Security issues

Responsive, bidirectional and secure communication between two PCs, located on different LANs separated by the Internet, is challenging. The Internet is an open network, which allows general insight into all the traffic flowing between the connected computers. When dealing with Internet communication, it is therefore important to have focus on security.

A lot of work has been done on Internet security in recent years, and

several security recommendations have been proposed for dealing with Internet-connected systems. Common Criteria [67], Misuse Cases [68] and Attack Trees [69] are different approaches to testing the security and specifying the security requirements of IT systems. The iMet system is currently being tested using the latter.

Traditionally there have been three basic security services: *confidentiality, integrity, and availability.* X.800 [70] is a security recommendation, which adds: *authentication, access control, data confidentiality, data integrity, and non-repudiation.*

An explanation of each service is given below.

- **Confidentiality**
  Make information non-interpretable to unauthorized users
- **Integrity**
  Prevent unauthorized altering of information, and provide detection mechanisms.
- **Availability**
  Prevent unauthorized blocking of information, e.g. Denial-of-Service (DoS) attacks
- **Authentication**
  The receiver and the sender of data can both prove their identity.
- **Access control**
  Protect against unauthorized use of resources (communication resources, information resources, processing resources) or all accesses to a resource.
- **Non-repudiation**
  The recipient of data is provided with proof of the origin of data, while the sender of data is provided with proof of delivery of data.

The security services mentioned only define what needs to be addressed to increase the security of the system. The following list describes concrete tools which can be used to obtain the security services.

- **Encipherment**
  Provides confidentiality of traffic flow. There are two types of reversible (encryption/decryption) encipherment, also shown in Fig. 2.2:
  - *Symmetric or secret key encipherment*
    Sender and receiver share a secret key used for encipherment and decipherment

  - *Asymmetric or public key encipherment*
    Two different keys (private and public) are used for encipherment and decipherment

- **Digital signature mechanisms**
  - Signing a data unit
  - Verifying a signed data unit
- **Access control mechanisms**
  - Use of authenticated identity of an entity to determine access rights
  - Use of access control databases to maintain entity access rights
  - Access rights based on time of access, route of access and duration of access
- **Data integrity mechanisms**
  - Sender generates a cryptographic check-value associated with the data, and the receiver regenerates the check-value and compare them when receiving the data
  - Use of sequence numbering, time stamping and secret keys to prevent misordering, losing, replaying, inserting or modifying data
- **Authentication exchange mechanisms**
  - Use of username and password
  - Use of cryptographic techniques
  - Use of digital signatures
  - Use of characteristics of the entity
- **Traffic padding mechanisms**
  - Provide protection against traffic analysis
- **Routing control mechanisms**
  - Route data based on content, labels, sender or receiver
- **Notarization mechanisms**
  - Use of third-party notary, trusted by all parties, to ensure integrity, origin, time and destination of data

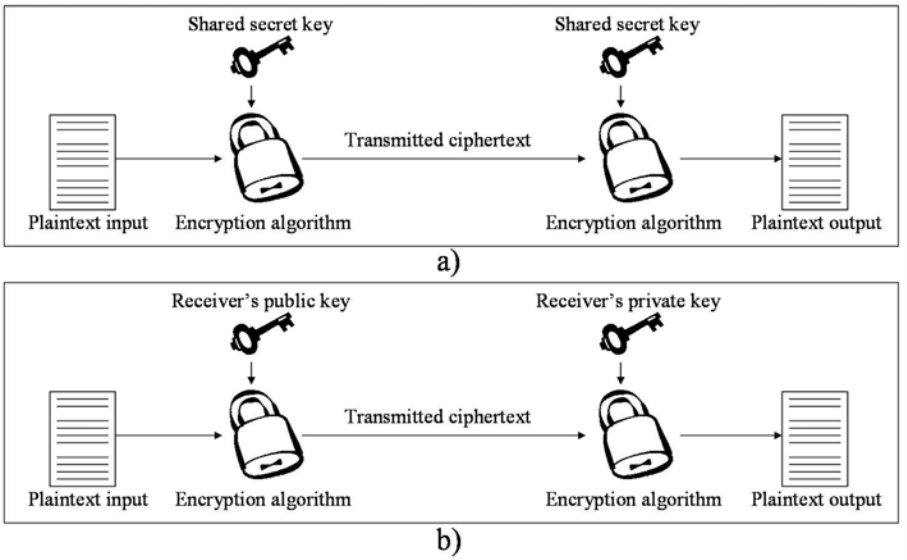

Fig. 2.2: a) Symmetric and b) asymmetric encryption. Symmetric encryption is used to gain confidentiality and integrity. Asymmetric encryption can be used to obtain confidentiality and integrity, when encrypting with the public key, and authentication and non-repudiation, when encrypting with the private key.

Confidentiality and integrity are obtained by encrypting the data exchanged. Authentication and non-repudiation are obtained by using digital signatures or credentials like username and password. As we shall see later, authentication of persons is somewhat standardized, as opposed to equipment authentication, which can be quite difficult. Access control is applied after an authentication process, to establish the authorization level of the authenticated person. By using standard, firewall-friendly communication protocols and standard ports, thus allowing authorized users to access the system, even behind strict firewalls, the availability of the system may be increased. Preventing DoS attacks is beyond the scope of this work and will not be treated here.

Probably the most crucial aspect of the system security, is providing information and education to all system users about using cryptographic services and especially handling cryptographic keys. It is important that all users are aware of which procedures to follow when using the system.

2.3 Middleware

2.3.1 Object-oriented middleware

The current work has focused on finding efficient and secure software solutions to the third architecture described in 1.6.3. This means allowing two Internet-separated LAN computers to inter-communicate in a fast and secure way. To provide direct method invocation between network-separated application domains, the use of so-called middleware simplifies the process of transferring information about the method call over the network. Object-Oriented Middleware (OOM) is a communication technology, which enables data objects to communicate across networks.

There are several OOM technologies available, including among others CORBA [71], Java Remote Method Invocation (RMI) [72], and Microsoft .NET Remoting [73]. These technologies use interface-based method invocation over Transmission Control Protocol/Internet Protocol (TCP/IP) networks. They act as black boxes to the users, and handle all network-related issues involved in the method invocation process. Fig. 2.3 shows a general OOM architecture.

For all three OOM technologies mentioned, the following applies. After setting up a connection to a remote server, the client has access to a local proxy representation of the remote server object. This object proxy has the same interface as the real server object. When calling a method on the object proxy, the method call is automatically transferred across the Internet by the underlying middleware layer to the real server object. This way the client may use the remote server object as if it was local, as shown in Fig. 2.3. When two clients are connected to the same remote server object, inter-client communication is possible.

2.3.2 Two-way method invocation

Direct communication between two Internet-separated LAN computers is often prohibited due to intermediate firewalls and proxy servers. Firewalls often only allow out-bound connections from internal LAN clients, and they often restrict the external addresses and ports to connect to. Application-level firewalls may also restrict the protocol in use, e.g. only HTTP connections on external port 80 (HTTP) or 443 (HTTPS). Proxy servers act on behalf of the internal clients, so that no direct connection can be made from an internal client to an external server. Instead the proxy server connects to the server on behalf of the client. This process is shown in Fig. 2.4.

A client's choice for connecting to the Internet may sometimes be restricted to using HTTP on external port 80 or 443 via a proxy server. This means, the client is invisible to external parties (they can only see the proxy server or firewall).

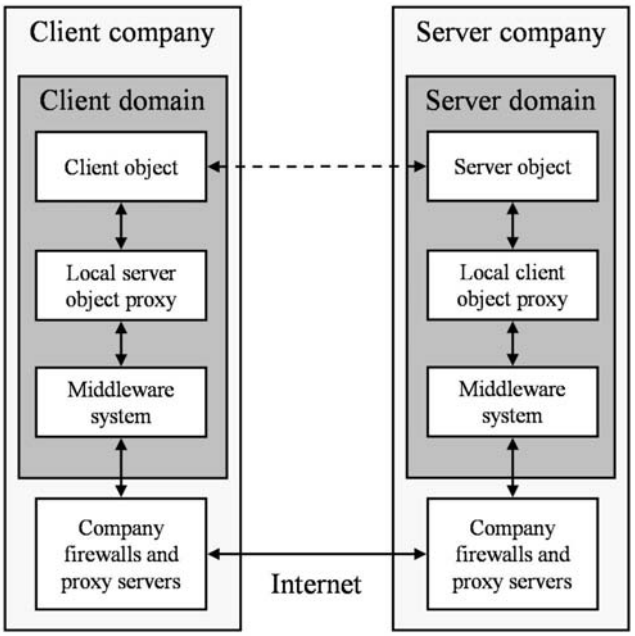

Fig. 2.3: General object-oriented middleware architecture. The communication process is made transparent to the interacting objects by the underlying middleware. The server object and the server object proxy implement the same software interface. The same applies to the client object and the client object proxy.

It also means that the client only can communicate with remote web servers, or applications that can act as web servers (handle HTTP traffic). For two such clients to inter-communicate, without reconfiguring intermediate firewalls or proxy servers, it would require them to set up connections to an external web server, which would relay the communication signals between them. In the following discussion, clients which can only connect to the Internet using HTTP on external port 80 or 443 via a proxy server will be referred to as *restricted clients*.

The three OOM technologies all support using HTTP on external port 80 or 443. This is called HTTP tunneling, which means wrapping the real communication protocol inside HTTP packets. Looking at Fig. 2.3, the middleware layer is responsible for this HTTP tunneling, which is transparent to the client object and the server object. Though both TCP and HTTP communication is supported for CORBA, Java RMI and .NET Remoting, the communication is only one-way (or simplex). When used outside the middleware

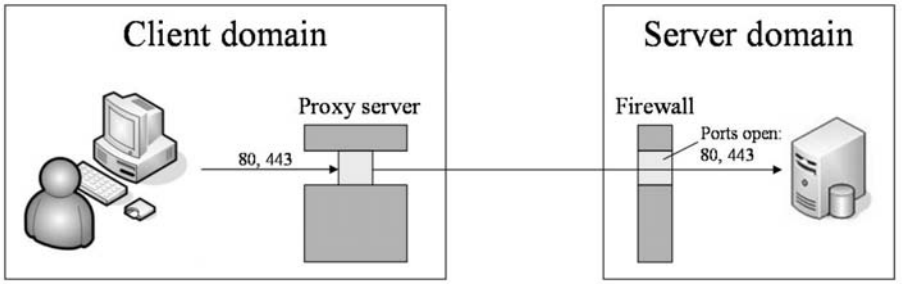

Fig. 2.4: The figure shows the function of a proxy server. Instead of setting up a TCP connection to an external server directly, an internal client sets up a TCP connection to a proxy server, which then sets up a separate TCP connection to the remote server on behalf of the requesting client.

frameworks, TCP is full-duplex. This simplex-restriction means that a client may call methods on a remote server object and receive method replies using TCP or HTTP, but it is difficult for the server object to call methods on the client object. The latter would be possible, if the server opened a new TCP or HTTP connection to the client (which is efficiently stopped by the client-side firewall).

When implementing architecture 3, described in 1.6.3, the relay server object needs to be able to call methods directly at the connected client objects to avoid inefficient client polling. As seen before, the server can only send data to the clients as method replies or by setting up new connections. It is possible to configure CORBA, Java RMI and .NET Remoting, so that the server object can send a method call reusing an existent connection. This only applies to direct TCP connections, and would prevent *restricted clients* from using the system.

To include *restricted clients*, one solution would be to develop a full-duplex HTTP channel, which would support transferring method calls in both directions using HTTP. This has been done for .NET Remoting by GenuineChannels [74]. Instead of forcing the server object to open a new HTTP connection to the client object, the server object can reuse an existing HTTP connection (set up by the client object). This process will be described in 2.3.4. This functionality is difficult to implement in CORBA or Java RMI in an efficient way, and was one of the reasons why .NET Remoting was chosen in the current work.

To provide this functionality, some restrictions will dictate the choice of platform. Software developed for .NET may only be run on computers with the .NET Framework installed. Traditionally, this means that the computers involved must run the Windows platform. Platform-independent versions of the .NET Framework are under development, e.g. the Mono project [75] and the

DotGNU project [76], but these have yet to be tested for the present system.

If, or when, CORBA or Java RMI add support for bidirectional, full-duplex HTTP channels, the system could be ported to use these technologies as well. Here we will just demonstrate the principle of the channel, which is only dependent on the HTTP functionality, and could, in principle, be implemented on any platform, e.g. Linux. A short presentation of CORBA and Java RMI is given in the two following sections, while a more thorough description of the .NET Framework and .NET Remoting is given in 2.3.3 and 2.3.4.

CORBA

CORBA is an acronym for Common Object Request Broker Architecture. It enables heterogeneous applications, written in a number of different programming languages, to inter-operate by defining all aspects of the inter-communication process. These include the interfaces to be used when communicating with remote objects, the protocols to be used in the communication process and a standardized way to describe objects and services. The result is platform and location transparency for sharing well-defined objects among distributed applications. The CORBA architecture follows the architecture described in Fig. 2.3. The server first binds the remote object to a naming service accessible to the clients. The clients may then use the naming service to locate and get a reference to the remote object.

To let heterogeneous applications inter-operate, CORBA first wraps a specialized interface around each application. The application with the interface is called a CORBA object. The interface describes the application inside and how other CORBA objects may communicate with it. CORBA uses the Object Management Group (OMG) [77] Interface Definition Language (OMG IDL) [78] to describe the interfaces of all CORBA objects.

CORBA is able to map IDL to several programming languages, including C, C++, Java, COBOL, Ada, Lisp, Python and Smalltalk, which means that two functionally equal applications, written in different programming languages, look the same when looking at the IDL interface descriptions. The Remoting.Corba project [79] tries to add C# and Visual Basic to this list.

Java Remote Method Invocation

Java Remote Method Invocation (Java RMI) is a middleware framework, where Java objects may communicate seamlessly across networks, e.g. the Internet. The RMI architecture also follows the architecture described in Fig. 2.3.

RMI applications often comprises three separate components, a server, a client and a public registry. The server application usually creates one or more remote objects, binds them to a public registry and then lets clients invoke methods

on these objects. The client application gets a reference to one or more remote objects from the RMI registry, and then invokes methods on them directly at the server. It is also possible to pass object references in method calls or replies.

The underlying Java RMI middleware takes care of the communication between the client, server and the RMI registry. To the client it seems as if the remote objects are parts of the client application.

2.3.3 Microsoft .NET Framework

.NET [80] is Microsoft's strategy for connecting people, systems, computers, and devices. All software written for .NET needs to be run within the .NET Framework [81].

This framework is a common platform on which to run software written in programming languages [82] supporting the Common Language Infrastructure (CLI) [83], approved by the international standardization organization ECMA [84]. The software is executed in the Common Language Runtime [85]. The compilation of the code is a two-stepped process. First the source code is transformed into Common Intermediate Language (CIL), which is something between source code and CPU-specific code. When this CIL code is executed, the code is compiled Just-In-Time (JIT) into CPU-specific code.

This is quite similar to Java. Java applications are developed using one language, Java, and may be run on any platform supporting the Java Runtime Environment (JRE). .NET applications can be written in any language, following the CLI standard, and the applications can be run on any platform supporting the CLR. The JRE is much more widespread than the CLR, though there are projects working to spread the latter, as can be seen in 2.3.5. A structural comparison of CLR and JRE can be seen in Fig. 2.5.

2.3.4 Microsoft .NET Remoting

Microsoft .NET Remoting is a relatively new OOM middleware technology, which enables applications in different application domains to communicate as described in 2.3.1.

Full-duplex HTTP channel

As seen in 2.3.2, a full-duplex HTTP channel is available for .NET Remoting. Such a channel allows *restricted clients* to use systems of architecture 3. As will be seen, this HTTP channel follows the HTTP 1.1 standard [86], utilizing the *keep-alive* property which keeps an HTTP connection open during a session. Each client sets up two keep-alive HTTP connections to the server, *sender* and *listener*, where the former is used by the client and the latter by the server. This

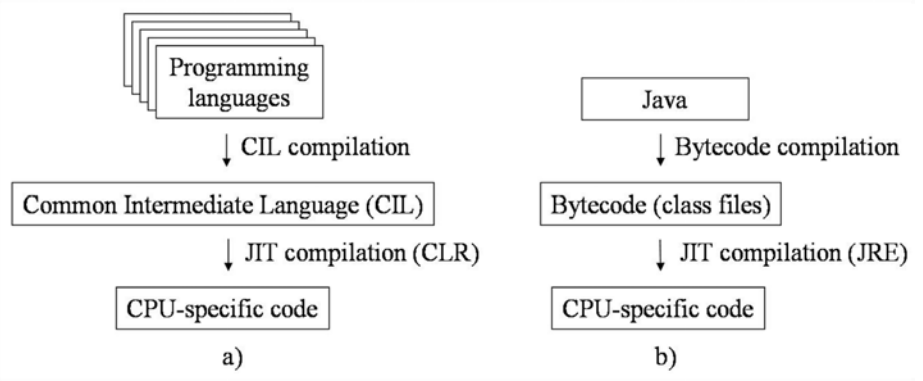

Fig. 2.5: Similarities and differences between a) the Common Language Runtime and b) the Java Runtime Environment. There are many programming languages conforming to the CIL standard, but not so many platforms supporting the CLR. Java is the only programming language for the JRE, but the JRE is supported by many platforms.

way, full-duplex communication over HTTP is achieved. The reason why two connections are needed, is the asymmetrical request-respond design of the HTTP protocol (the server can only send data in an HTTP response packet after the client has sent an HTTP request). Since the solution is based on reusing HTTP connections set up by the clients, the solution will work with any firewall or proxy server (as long as HTTP connections on external port 80 or 443 to the remote server are allowed, which is usually true). The client always has an "unanswered" or pending HTTP request on the *listener* connection, which the server may use to send data or method calls in an HTTP response. The pending HTTP request has a well-defined timeout value, typically set to about 1-2 minutes, after which the request is canceled and a new request is initiated. This will happen every time the server does not send data within 1-2 minutes. The underlying middleware is responsible for this utilization of the HTTP protocol, which is an efficient form of client polling. Since the communication is handled by the middleware, and thus hidden from the overlying application, the instrumentation system as such is not affected.

Fig 2.6 and 2.7 explains the client-to-server and server-to-client method invocations. When combining the two, we may obtain client-to-client method invocation, as shown in Fig. 2.6. The latter describes a situation where a client needs to call a remote method on another client connected to the same server object. The following three sections also describe the client-to-server, server-to-client, and client-to-client method invocation.

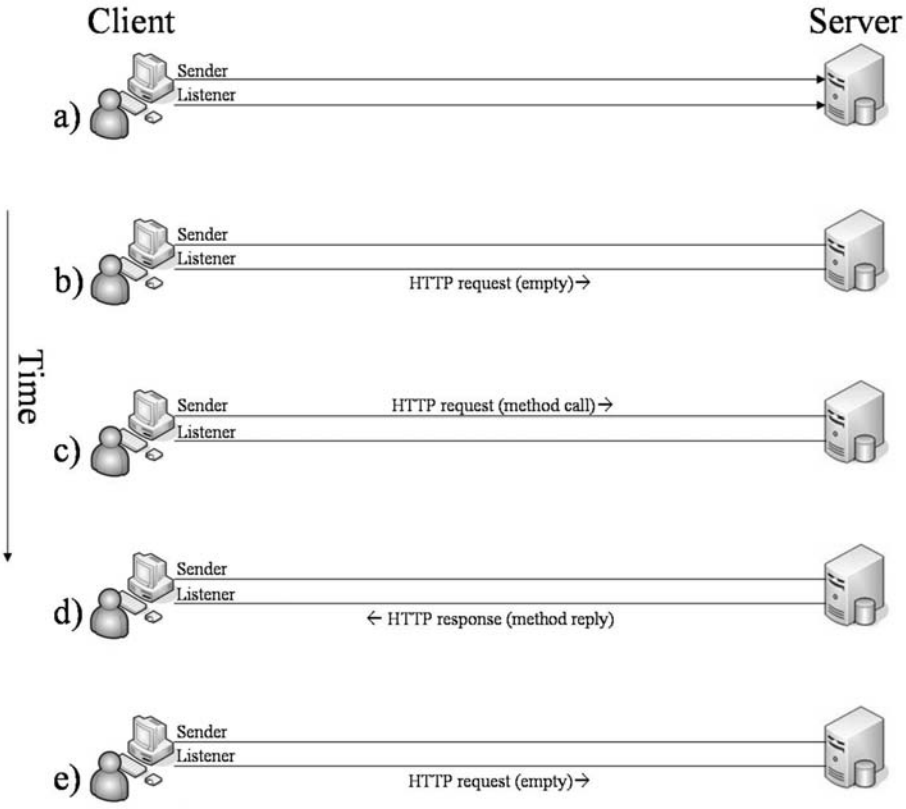

Fig. 2.6: Client-to-server method invocation. a) The client sets up two HTTP connections, *sender* and *listener*, to port 443 at the server. b) The client sends an empty HTTP request on the *listener* connection. This can then be used by the server to send method calls. c) The client sends a method call in an HTTP request on the *sender* connection. d) The method is handled by the server, and a method reply is sent back in an HTTP response on the *listener* connection. e) The client receives the method reply, and instantly sends a new empty HTTP request on the *listener* connection.

Client-to-server method invocation

When the client calls a method on the local proxy server object, the underlying .NET Remoting system wraps the method call in a message and sends it in an HTTP request to the server using the *sender* connection. On the server, the .NET Remoting system unwraps the method call and calls the correct method on the server object. It then sends the method reply back on the *listener* connection in

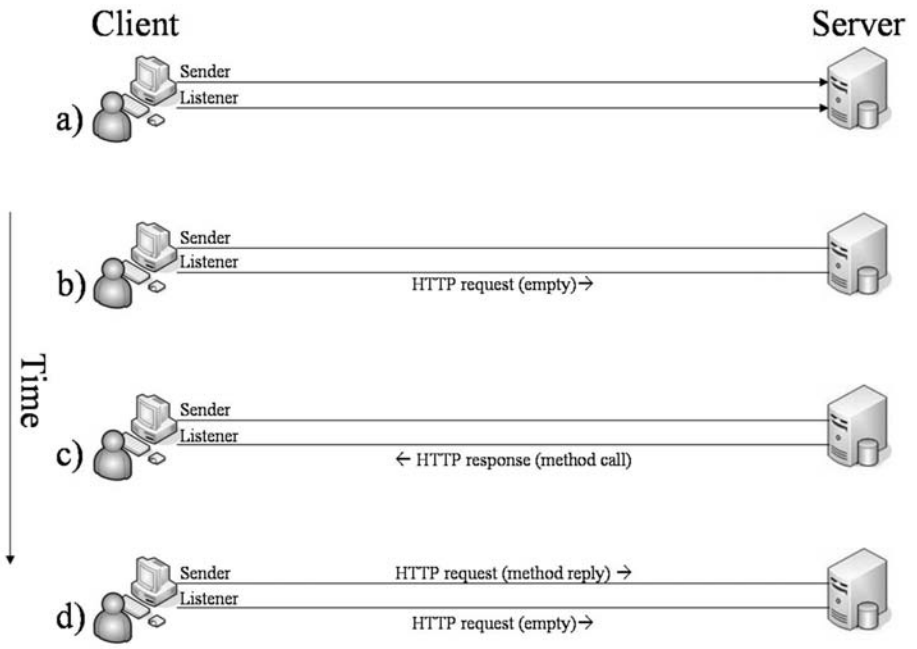

Fig. 2.7: Server-to-client method invocation. a) The client sets up two HTTP connections, *sender* and *listener*, to port 443 at the server. b) The client sends an empty HTTP request on the *listener* connection. This can then be used by the server to send method calls. c) The server sends a method call in an HTTP response on the *listener* connection. d) The client instantly sends an empty HTTP request on the *listener* connection. The method call is then handled by the client, and a method reply is sent back in an HTTP request on the *sender* connection.

an HTTP response.

Server-to-client method invocation

When the server calls a method on the local proxy client object, it wraps the method call in a message and uses the client's pending HTTP request to send an HTTP response, containing the message object, on the client's *listener* connection. After the client receives this HTTP response, the underlying .NET Remoting system unwraps the message and calls the correct method at the client. It then sends back the method reply in an HTTP request on the *sender* connection. Thus, bi-directional method invocation is obtained, without inefficient client polling, and standard security protocols, like HTTPS, can still

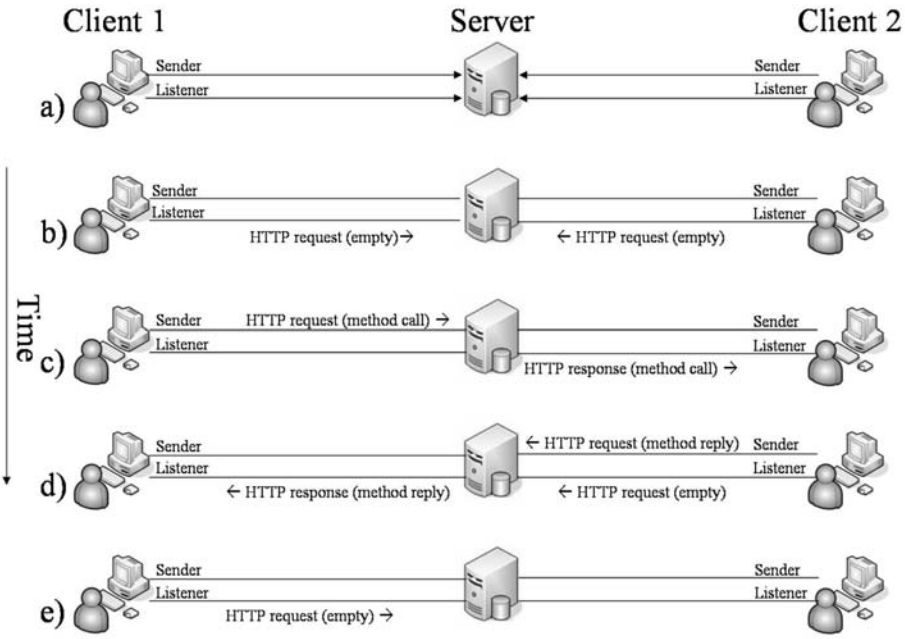

Fig. 2.8: Client-to-client method invocation. a) Two clients set up two HTTP connections each, *sender* and *listener*, to port 443 at the server. b) The clients then send an empty HTTP request on the *listener* connections. This can then be used by the server to send method calls. c) One of the clients then send a method call in a HTTP request on the *sender* connection. When received, the server forwards the method call in an HTTP response on the other client's *listener* connection. d) The second client instantly sends an empty HTTP request on the *listener* connection. The method call is then handled, and a method reply is sent back in an HTTP request on the *sender* connection. On retrieval, the server forwards the method reply in an HTTP response on the first client's *listener* connection. e) When receiving the method reply, the first client sends an empty HTTP request on the *listener* connection.

be used.

Client-to-client method invocation

When combining the client-to-server and the server-to-client method calls, it is possible to let two clients call methods directly at each other. First the call is transferred on the sending client's *sender* connection to the server. The server then forwards the call onto the receiving client's *listener* connection. On receiving

the method call, the receiving client calls the local method. The method reply is returned on the receiving client's *sender* connection. On receiving the method reply, the server then forwards the method reply onto the sending client's *listener* connection. Thus, a client-to-client method call is obtained even when both clients are behind different firewalls or proxy servers.

There is apparently no need to limit the usage of the system to device operation. In addition to communicating with external devices, the system might perform many other operations as well. If e.g. the customer's computer is powerful, the operator may utilize this extra power to perform heavy calculations. This is, however, beyond the scope of this work, and the following discussion will focus on instrument control and performing remote measurements.

Channel security

Microsoft's web server, Internet Information Services (IIS), hosts the remote server object, and all security features of the IIS architecture are available. The full-duplex HTTP channel, described in this section, supports the default security services of IIS, and encryption and compression for faster transfer of data. To set up the two connections, the clients must be able to connect to the remote server on port 443 SSL/TLS, which most firewalls and proxy servers support. The security of the whole system will be discussed in 2.4.

2.3.5 Porting the .NET Framework to other platforms

There are projects concerned with porting the .NET Framework to platforms like Linux, Mac OS X, Solaris and BSD. Below is a description of two of them, the Mono Project and the DotGNU Project.

The Mono project

The Mono Project is a software development project sponsored by Novell [87]. The project's main task is to port the Microsoft .Net Framework to several platforms like Linux, BSD, Solaris, Microsoft, and Mac OS X. The result will be that .NET client and server applications may run on most platforms, just like Java applications may run on any platform supported by the Java Runtime Environment.

The project includes:

- a compiler
- a class library
- a Common Language Infrastructure (CLI) runtime engine

The DotGNU project

The DotGNU Project is, like the Mono Project, trying to port the .NET Framework to other platforms, like GNU/Linux, Cygwin/Mingw32, Mac OS X, Solaris, and AIX. Currently there are three subprojects:

- DotGNU Portable.NET, an implementation of the Common Language Infrastructure (CLI)
- phpGroupWare, a multi-user web-based GroupWare suite, which also serves to provide a collection of web service components
- The DGEE web service server

2.4 Security

When using the Internet for general instrument control, and especially when offering callback methods from the server object directly to the connected clients, several security concerns arise. If unwanted intruders could gain access to the remote object and fake the identity of a reference laboratory authority or replay old method calls, serious damage could be done at the customer-side. The customers must therefore be absolutely confident that they are communicating with a real authority. The authority must also be confident about the customer's identity, e.g. when Internet-enabled calibration is performed. The information sent across the network should be impossible to alter or replay, at least not without detection. It would also be critical if a hacker could emulate the behavior of the remote object, and make customers or calibration authorities connect to it.

The .NET Remoting architecture has no default security features, but it is prepared for custom security implementations. By hosting the remote object in Microsoft IIS, the system can leverage all security features provided by the IIS architecture.

The following discussion of the implemented system's security model is based on the services described in 2.2.1.

2.4.1 Confidentiality

To obtain confidentiality the iMet system v.1.0 uses Hypertext Transfer Text Protocol (HTTP) over Secure Sockets Layer (SSL), or HTTPS. HTTPS uses Public Key Infrastructure (PKI) to exchange a secret key when initializing a session between a client and a server. A secret key is used in symmetric encryption, while a private and public key pair is used in PKI (asymmetric

encryption). During the client-server session all communication signals are encrypted according to the SSL standard. Any hackers listening to the communication, would not be able to decrypt the encrypted data in an easy way. The security is not so much dependent on the security protocol as on the handling of the private and secret keys. If private or secret keys are lost, the security of the system would be compromised.

Initializing an SSL session, or performing the SSL Handshake, includes several steps:

1. Client and server negotiate which encryption algorithm to use
2. Authentication is done using certificates (when server and client must be authenticated)
3. A shared key is agreed upon
4. Symmetric encryption is used with the shared key

The iMet system v.1.0 operates over HTTPS with both server and client authentication, using x.509 certificates. Each client certificate is signed by the server, and the server certificate will be signed by a trusted certificate authority (CA). When using certificates in the authentication process, it is extremely difficult to fake the identity of a customer computer, authority computer or the server.

Before engaging in an Internet-enabled calibration process, each client needs to request JV for a server-signed X.509 certificate. This certificate is used to authenticate the client's computer when connecting to the server, as described in Fig. 2.9. The number of authorized users are thereby held at a minimum, thus lowering the risk of compromising the system.

2.4.2 Integrity

SSL also ensures the integrity of the data exchanged between the client and the server. As can be seen in Fig. 2.10, the application data is split into fragments, which are then concatenated with a Message Authentication Code (MAC), before being encrypted. The fragments may also be compressed before being concatenated, which results in faster transfer of data.

The MAC is simply a small *hash* value, created from the original data fragment and some shared, secret MAC key. The *hash* value is a unique fingerprint of the fragment, generated by a one-way *hash* function. It is virtually impossible to find two fragments which give the same *hash* value, and it is impossible to find the fragment from the *hash* value.

After encrypting the fragment and the MAC, the client sends the packet to the server. On receiving the packet, the server decrypts it, creates a new MAC

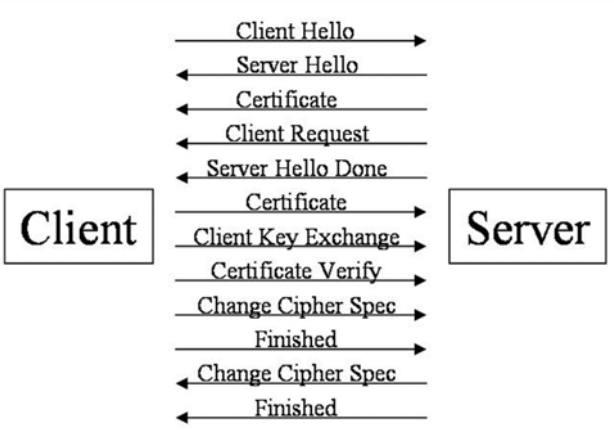

Fig. 2.9: The SSL Handshake with server and client authentication. During the initializations, the client and server agree upon encryption algorithms, perform certificate-based authentication, and create a shared symmetric key.

from the contained fragment, and compares it with the contained MAC. If these match, the server can be sure that the packet has not been tampered with during transportation. After encrypting the data and MAC, it is virtually impossible to change the data without corrupting the MAC.

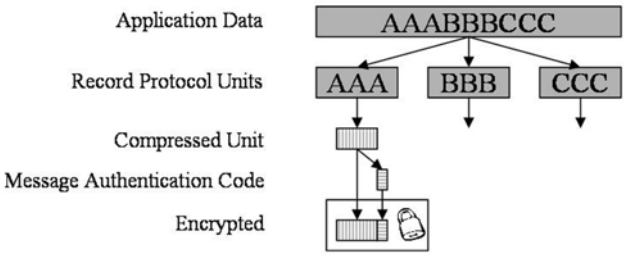

Fig. 2.10: The SSL Record Protocol. Application data is fragmented, and each fragment is then compressed. A MAC is created from each compressed fragment, and the compressed fragment and the MAC is concatenated and encrypted.

2.4.3 Authentication

In SSL, digital certificates are used to authenticate specific entities, such as companies, specific computers and specific user accounts on a computer. The private key, associated with a certificate, or client certificate, is stored on the computer where the certificate is installed. It is important that this private key is stored in a secure way. On Windows systems, the private key is encrypted using a key based on the logged-in user's Windows password. If an unwanted intruder could gain access to the computer and the correct user account's password, the security of the system would be compromised.

In the iMet system v.1.0 additional steps are performed in order to authenticate the specific person authorized to connect to the server. After authenticating the computer and user account to the server, the user needs to supply the server with a username and password. In failing to do so, the user can not use the remote object's methods. As long as the private key, iMet username and iMet password are not lost or given away, the authentication process is quite secure. This is how the authentication process is performed in e.g. some online banking services. The client certificates must be distributed to the customers and authority in a secure fashion. If laptops are sent out to the customers in advance, the certificates should be installed before sending. The customers must be thoroughly educated in how certificates should be handled and stored.

In addition to authenticating persons, each connected instrument also needs to be authenticated. When performing remote calibrations, it is important to be sure of the identity of the instruments involved. As we shall see, this identity is quite complicated to obtain securely, and will be discussed in 2.5.

2.4.4 Non-repudiation

SSL enables secure communication between a client and a server. The data exchanged is kept secret (confidentiality) and possible tampering is detected (integrity). The server and the client may be authenticated using digital certificates and usernames/passwords (authentication).

SSL only provides a secure channel, and it does not cover the area of non-repudiation. As data is sent through the channel, and looked at on the other side, there is no linkage to the original sender. Non-repudiation would be obtained with digital signing of the data. If the data received is stored with the sender's digital signature of the same data, the receiver could at a later time prove that the data was actually sent by the sender at a specific time.

This digital signing of data would need to be implemented on top of the SSL architecture, usually in the application. For the iMet system v.1.0 the focus has been on securing the channel between the NMI and the customer. Non-repudiation could be implemented at a later stage using digital signatures. This

could require each user to use two digital certificates; one for authentication and one for signing data. Not all data would need to be signed. When performing remote calibrations, potential data to be signed might be measurement results and measurement configuration data.

2.4.5 Access control

The users of the system are believed to be located behind their own company firewall, proxy server or NAT. The system server is also behind a firewall, with port 443 open for incoming HTTPS connections. This firewall will be configured only to allow certain IP address ranges to connect, so that only computers belonging to known customer may connect to the server.

In the SSL handshake process, the server requests a client certificate from each connecting client. If a client fails to provide a certificate, the connection is closed. If a certificate is provided, the secure channel is established. The connecting client is then required to log in with a username and password. The username and password are checked against a database, containing authorization information. There are two types of user roles: *customer* and *authority*. Each logged in client is associated with one of these roles, based on the credentials provided, and the roles are associated with different authority levels. E.g. only *authority* clients are allowed to operate other clients' instruments remotely or run remote measurements.

The use of digital certificates requires that procedures must be established concerning the handling of certificates. Should each person have their own certificate, or should each customer company have a certificate? For how long time should each certificate be valid? For the iMet system v.1.0 a certificate was issued to a specific logged-in user on a specific computer for one year (might be changed in the future).

A hacker could attack the system in several ways. The hacker could request JV for a valid certificate and credentials. This would allow him to connect to the server object. Procedures need to be developed in order to prevent this. He could also try to steal a valid certificate from a customer of JV. He would then need to connect from a customer's LAN (due to firewall rules) and provide a valid username and password. The username and password of a regular customer, would only give *customer* access rights. All customers need to have clear procedures on how to store and use digital certificates.

JV is responsible for creating passwords for each user. Strong and long passwords should always be used, containing capital letter, numbers and special characters. This would make it difficult to guess the correct password, even when having a valid username.

2.4.6 Availability

The Internet is a highly insecure network, and firewalls and proxy servers are required to separate a LAN from exposure to unwanted intruders. Firewalls and proxy servers usually prohibit external hosts to initialize connections to internal hosts, and often put restrictions on which network protocols are allowed. There is also a shortage of public Internet Protocol (IP) addresses, which for some businesses require the installation of Source Network Address Translators (SNATs). A SNAT maps internal LAN addresses to external Internet addresses, such that data packets originating from different internal hosts appear to come from the same public host. This make internal hosts unreachable and invisible to the general public, and all connections between a LAN PC and the outside of the firewall/proxy server/SNAT therefore need to be initialized from the LAN PC.

Several hardware and software components are used to secure a LAN from the Internet. A condensed presentation of commonly used components is given in Appendix B.

To establish reliable communication between two PCs in different LANs, the iMet system sets up persistent connections from both PCs to a public server, which binds the two PCs together in a so-called Meet-in-Middle (MiM) configuration. All communication signals between the two LAN PCs pass through this server. This type of communication is used by many Instant Messaging (IM) services, of which a general description can be found in [88].

To improve the availability of the system, one approach could be to let dedicated servers, e.g. Kerberos, take care of the authentication process, to avoid overloading the application server. This will also make the system more resistant to Denial of Service (DoS) attacks. In an Internet-enabled calibration system, there would be very few users at any given time and the application server would then have free resources to handle unwanted connection attempts.

2.4.7 Organizational and human security policies

So far the security discussion of the iMet system has focused on the technical aspects. Whenever human interaction is required, it is also important to look at how the security could be breached by intentional or unintentional misuse, and weak security policies. Although unintentional misuse may be a usability issue and a result of poor design, it will also affect the security of the system, and needs to be addressed in a security context as well.

The customer

Three issues need to be addressed at the customer side

1. Weak or non-existent security policy

2. Unintentional misuse of the system

3. Intentional misuse of the system

Up to this point in the security discussion, the customer, once authenticated, has been trusted. To further secure the system, it should be required by the customer to have an active security policy, with up-to-date firewalls and virus protection. In addition, the security policy should be communicated to all customer personnel.

Before using the iMet system, a customer should receive adequate user training and system documentation describing how to use the system. Usability issues should be addressed when designing the system, such that the risk of unintentional misuse would be minimized.

It is difficult to prevent intentional misuse, but measures has been added to reduce the effects. Because the customer logs on with the *customer* role, the possible actions to misuse the system are kept at a minimum. E.g. as a *customer* it is only possible to communicate with your own instruments. Additional measures could include the certification of specific persons and extensive logging of every action made by users.

The NMI

It is of utmost importance that the NMI has an adequate security policy, since the customers depend on the software and the dedicated servers provided by the NMI.

JV is currently working on an extensive security analysis of the iMet system, based on the attack tree methodology [69]. This will reveal places in the system where potential attacks might occur. The analysis could also be used as a guide as to where extra resources may be added.

The operators, responsible for performing Internet-enabled calibrations at customers, should receive extensive user education, thus reducing the rate of user mistakes. User mistakes done by operators, often linked to the usability of the system, may strongly compromise the security of the system. Contracts should be made beforehand so that everyone has a clear understanding of who is responsible if discrepancies occur.

2.5 Equipment authentication

When integrating the Internet in electrical metrology and calibration, it becomes increasingly important to be able to securely authenticate the equipment in use. This differs from operating the equipment locally, where the operator can inspect the equipment directly. When calibrating an instrument via the Internet, it is

of utmost importance to authenticate the instrument beforehand, so that the operator can be sure that the correct instrument is being calibrated.

2.5.1 Traditional authentication

There are numerous ways to authenticate electrical PC-connected equipment via the Internet:

Obtaining information from trusted local personnel

The remote operator can be told which equipment is connected by a local person, sitting by the equipment. This requires the operator to trust that local person. Instead of personally authenticating the equipment, the operator bases the authenticity of the equipment on the information provided by the trusted local person.

Inspecting external identification tag visually

By looking at the equipment via a web camera and reading some external identification tag, shown in Fig. 2.11, it is possible to tell the identity of the equipment. This identification tag can comprise manufacturer name, model name and a serial number.

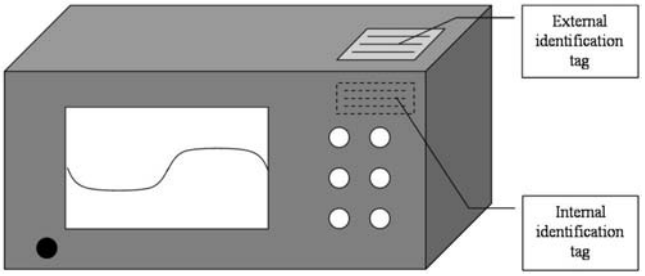

Fig. 2.11: Traditional ways of authenticating electrical PC-connected equipment. The external or internal identification tags are obtained either via trusted local personnel or via a web camera and software.

Using software to obtain internal identification tag

Some instruments have an internal software identification tag, also shown in Fig. 2.11, which may be read using the connected computer. This tag can be used to establish the identity of the equipment. Often the internal identification tag

comprises a manufacturer name, a model name, a serial number and sometimes the version number of the installed software.

There are weaknesses related to each of the methods above. Local personnel may be falsely trusted, external tags of equipment may be tampered with, and internal identification tags may be overwritten.

2.5.2 Authentication based on historical data

Another approach for authentication of PC-connected devices is to make use of historical measurement data, an example of which can be seen in Fig. 2.12. After calibrating an instrument a number of times, the accumulated data, or the historical data, may be used to predict the future behavior of the instrument. All instruments drift, which means that they change behavior over time, and a model of the instrument may be developed based on the drift or the historical data e.g. using Bayesian methods. JV is currently working to test this approach.

This methodology can be used to build a dynamic *finger print* of the equipment. Looking at several relative measurement values (the measured value divided by the nominal value), potential fingerprints of the physical properties of the device can be constructed. E.g. some electrical multifunction calibrators contains fixed resistors, with unique drift behaviors, which could be used to construct unique fingerprints. By using the historical data, and choosing a few key measurement points, the new *finger print* is predicted each time the instrument is to be calibrated. The selected key measurement points are then read from the instrument, and the predicted *finger print* is compared to the measured *finger print*. The *finger print* would be associated with an uncertainty, and the comparison process leads to a calculation of the degree of authenticity. E.g. it may be possible to calculate a 99% certainty of authenticity for an equipment, meaning that we are 99% sure that we are using the correct equipment.

This method may have potential, but needs to be further investigated. There are a few problems or challenges that must be addressed

- Instruments of the same type or model tend to drift in a similar way
- Instruments of the same type or model often have similar calibration characteristics
- Instruments often change characteristics due to repair or accidents. This will introduce problems in the described process, in that the instruments no longer can be authenticated.

All of the above methods could be used to obtain the best possible degree of authenticity.

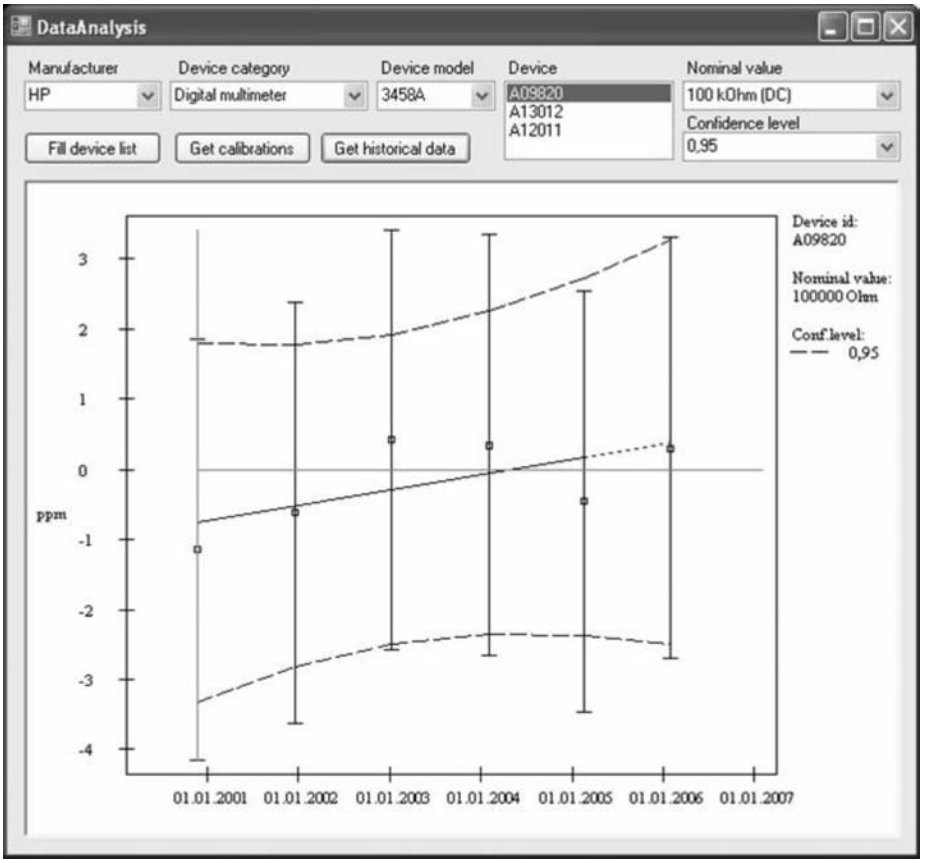

Fig. 2.12: The use of historical key measurement data to predict the behavior of electrical PC-connected equipment. The five first points are historical values, and they are used to predict the future value (the sixth point). In the figure, linear regression is used to model the drift, which is often a good approximation on shorter time intervals.

Protecting the reference values

To use historical measurement data to authenticate electrical PC-connected equipment, the reference values need to be protected. To obtain a reliable measurement value from the equipment to be authenticated, it is required that the reference value is traceable to a known standard value.

Physikalisch-Technische Bundesanstalt (PTB) in Germany is currently involved in a project concerning secure transfer and authentication of measurement data from household electricity meters. The project, named SELMA [89], has

looked at installing small hardware chips in the meter to be used in the data authentication process. Before transfer the measurement data is signed by the meter, which ensures integrity while linking the data to a specific meter. This could be a potential solution for future transfer standards.

2.6 Instrument operation

2.6.1 Customer computer

The implemented system uses the Virtual Instrumentation Software Architecture (VISA [54]) for instrument operation, through a preinstalled dynamic linking library (DLL) file, visa32.dll. VISA is a standardized interface operating between the control application and the instruments. It makes the appropriate driver calls depending on the type of instrument used. The visa32.dll library contains several methods for communicating with hardware over the hardware interfaces mentioned in 1.3.3. In the iMet system v.1.0 only support for GPIB connections have been implemented.

A small .NET component, called the GPIB Communicator, was developed to load this VISA library at startup. The component contains four methods:

- **FindResources**
  Scans the client-side instrument bus for connected instruments
- **Read**
  Receives data from a connected instrument
- **Write**
  Sends data to a connected instrument
- **Query**
  Combines the Read and Write operations into one bulk operation

The installed .NET customer application loads this component at startup, and may then call the contained methods as needed. This way, all implementation concerning the measurement setup and measurement logic is done in the .NET environment. The customer software architecture is shown in Fig. 2.13.

All exported methods at the instrument computer are available to external authorities, and the methods communicate with the physical instrument via the GPIB Communicator object. This object has access to the Virtual Instrument Repository, where software representations of all connected instrument are found. These virtual instruments contain instrument-specific commands and state-information for each instrument. The instrument-specific commands are used to obtain consistent instrument communication. The state-information can be used

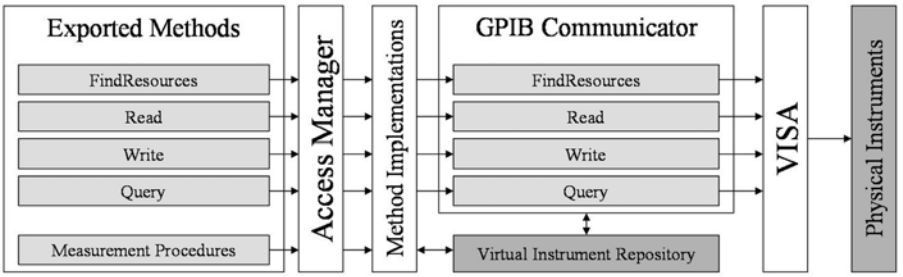

Fig. 2.13: The Customer Software Architecture. An external operator may call either one of the exported methods. Direct instrument operation happens through the four methods FindResources, Read, Write, and Query. The operator may also call more complex measurement procedures, preinstalled on the instrument computer.

to obtain cache-functionality, e.g when querying the instrument, it is often more efficient to query the virtual instrument than communicating with the physical equipment.

Initially, each customer instrument was represented as a remote object accessible to the server. This meant that the server had a proxy version of each of the instruments connected to the customer. Because there is only one GPIB controller at the customer side, managing the communication over the GPIB bus, this was changed to make synchronization, access control and security easier. Now the instruments are accessed via the customer's Read, Write and Query methods using an instrument identification string. This change can be seen in Fig. 2.14 and 2.15.

The authority may still use the instruments as if they were connected locally, but in stead of using proxy instruments, it is necessary to use the Read, Write and Query methods with an instrument identification string as input.

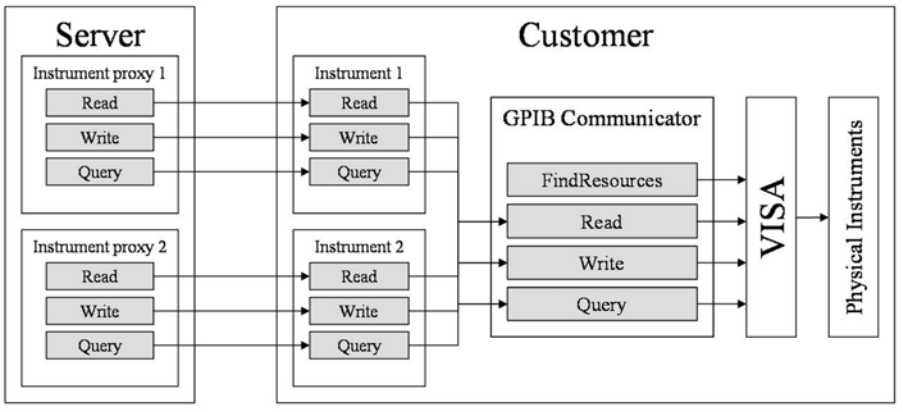

Fig. 2.14: Initial architecture. The server generates proxy versions of each customer instrument. The proxy instruments communicate directly with the corresponding instrument via the Internet.

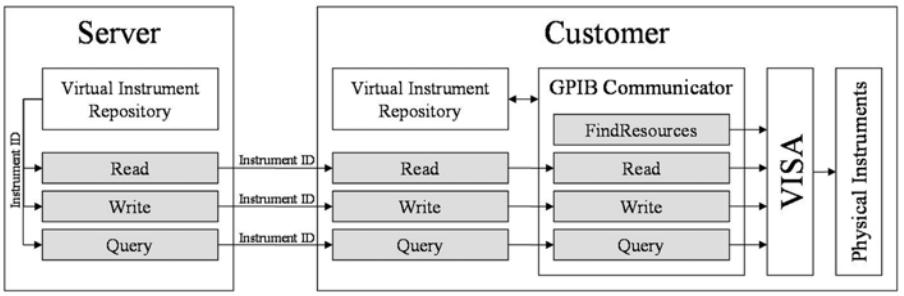

Fig. 2.15: Present architecture. The server only use the customer's Read, Write and Query methods to communicate with the instruments. An instrument identification string is used as input to the methods in order to route the method call to the correct instrument.

2.6.2 Server computer

The server computer implements both interfaces describing the authority and customer applications, in addition to a specific server interface.

To the authority the server looks like a customer, while it looks like an authority to the customer. This implementation of customer and authority interfaces, allows the server to act as a transparent proxy service between a customer and an authority. In addition to relaying data between the two, it can provide cache-functionality and security and consistency checks. This transparency can be seen in Fig. 2.16.

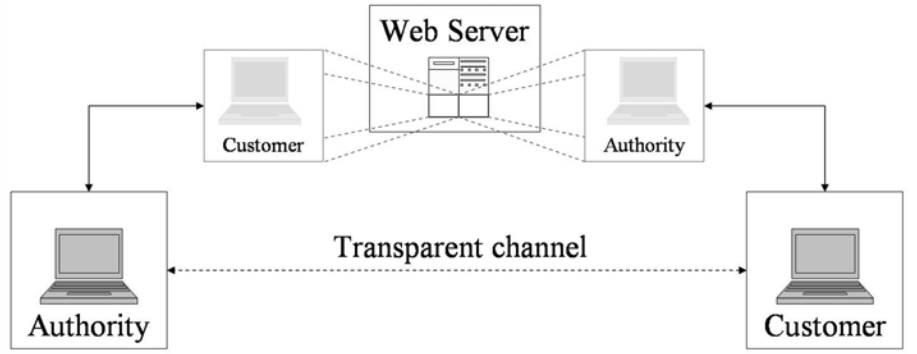

Fig. 2.16: The system server implements both the interfaces of the authority and the customer. To the authority the server looks like a customer, and to the customer it looks like an authority.

2.6.3 Authority computer

If the middleware system is running as described in 2.3.4, it is quite straightforward for an NMI authority to control a customer's instruments. The middleware system is transparent to both clients, and to the controlling authority it appears as if the customer's instruments are connected locally. The real and virtual communication paths are shown in Fig. 2.17 and Fig. 2.18.

In the system presented, the operator may either send individual instrument commands, by calling the client-side FindResources, Read, Write or Query methods, or he may call the measurement procedures, as shown in Fig. 2.13.

The NMI authority has access to the software interface implemented by the customer application. This interface describes how to call the customer's exported methods. Similarly, the customer knows the interface implemented

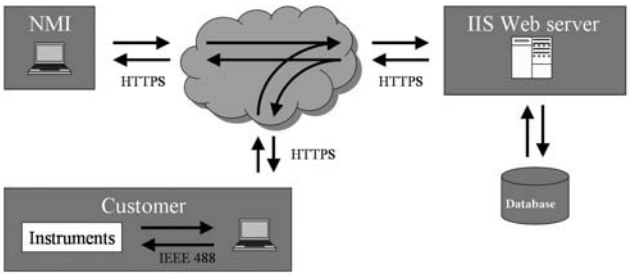

Fig. 2.17: Real Communication Path. The communication signals from the authority computer travels via the server to the customer's computer and the physical instruments.

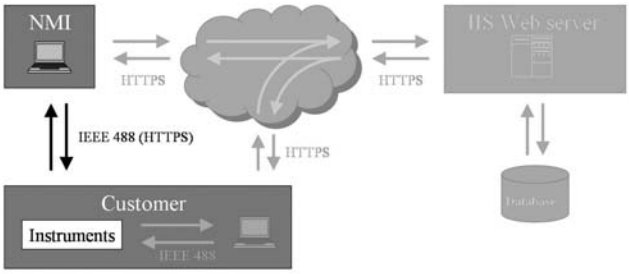

Fig. 2.18: Virtual Communication Path. To the authority the communication signals seem to travel directly to the physical instruments.

by the authority application, which enables it to call remote methods at the authority.

2.6.4 Interconnection

When a customer connects to the server, the customer's instrument bus is scanned and for each instrument found an instrument software object is added to the virtual instrument repository at the customer and on the server (and on the authority, if one is connected).

Measurement and calibration procedures, preinstalled on the instrument client, make use of the Read, Write, and Query methods to communicate with the physical instrument. These three methods are specified by the InstrumentCommunicator interface. The authority, server, and customer applications all implement this interface. They are interconnected such that calling e.g. the Write method on a host, using a specific instrument identification

string, will send the command to the correct instrument. This interconnection is shown in Fig. 2.19.

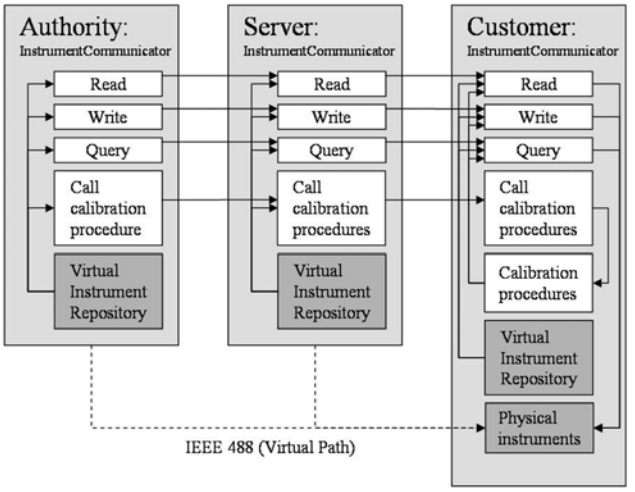

Fig. 2.19: The iMet system v.1.0 communication architecture. By implementing the InstrumentCommunicator interface correctly, an application may operate a customer's instruments using the Read, Write, and Query methods or run calibration procedures.

This means that the calibration procedures may run on any host implementing the InstrumentCommunicator interface. In the iMet system v.1.0, only a few measurement procedures were implemented, in the customer's application. These calibration procedures were defined in the interface implemented by the customer, and the authority could call them via the server as explained before.

2.6.5 Example

In an Internet-enabled calibration scenario, an authority chooses a specific calibration procedure, based on which instruments are in the virtual instrument repository associated with the specific customer. The measurement parameters used in the calibration are stored in the database as extensible markup language (XML). The calibration procedure is then called, and the measurement parameters are used as input to the procedure. When returned, the results are shown to the operator in a custom graph area, and stored in a text file as XML.

Another scenario would be to let a third-party expert, perhaps with extensive knowledge about a certain instrument type, gain access to a customer's instruments in order to guide in the measurement process or to bring his or her

expertise to help with problems or errors with the instruments. The expert would not need to sit by the instrument, but could be located wherever an access to Internet is available.

2.7 System test

A system test was performed were an operator at JV set up and controlled a measurement via a public web server at UniK. The instruments were located in a laboratory at JV, thus belonging to the same LAN as the operator. The setup is shown in Fig. 2.20.

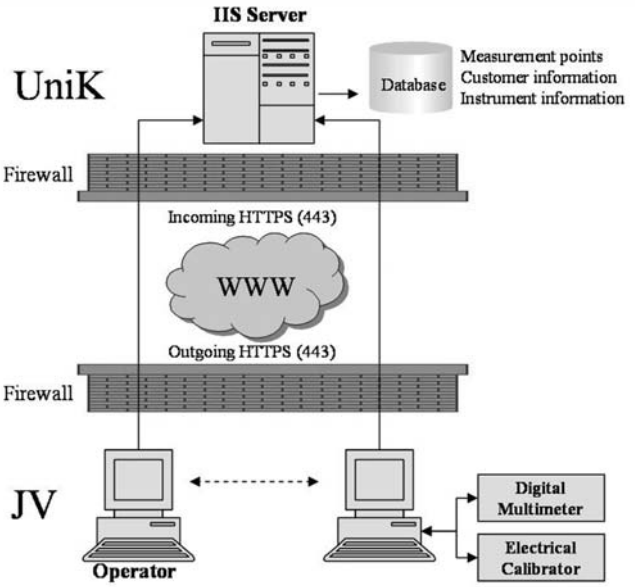

Fig. 2.20: System test of the iMet system v.1.0. The operator and instruments are connected to the same LAN at JV, but they communicate via a public web server placed at UniK.

This is equally challenging as if the operator and the instruments were at different LANs, because the communication signals travel via the server, which is on the outside of the firewall.

There were two instruments in use, an electrical calibrator and a digital multimeter. The measurement consisted of several steps:

1. Initialize instruments

2. Iteration process
   (a) Configure instruments
   (b) Generate signal with calibrator
   (c) Read signal with DMM
   (d) Store result
3. Return data to operator

On receiving the results, the measurement data was stored in a text file at the operator for future processing. The data was also shown in a graph.

The system test was quite satisfying. The measurements, even when set to run for half an hour, ran smoothly and returned data correctly to the operator. The process was fully asynchronous, so that the operator could spend time on other tasks, when the measurements were running on the instrument computer. The measurement results were returned to the authority by calling a specific authority method concerned with receiving measurement data.

2.8 Conclusions

The iMet system v.1.0 worked as planned. It allowed JV personnel to run measurements remotely, directly at a customer site, without the need to travel to the instruments involved. Tests showed that the system may be suited for Internet-enabled calibration as described in chapter 2.

The security of the system has been analyzed, as described in 2.4, and standardized security measures were used (SSL over HTTP with both certificate and client X.509 certificates, and password protection). It might be desirable to implement new or additional security measures in the future, because e.g. the field of non-repudiation was not treated. In the future digital signatures should be used to further ensure the validity of measurement configuration data and measurement results. The system is currently being tested using the attack tree methodology, and the results of the test may be used in future systems.

The measurement procedures were implemented directly in the customer application, which means that the system needs to be recompiled and redistributed when adding new measurement procedures. This is a quite tedious task, and it is therefore best to add all the measurement procedures needed before distributing the system. Later versions of the iMet system deals with this, by allowing custom measurement procedures to be downloaded from a database and run at runtime.

The system has not been tested on real customers, which could have produced important feedback regarding user experience. This was done in the next version of the system, which is described in chapter 4.

Chapter 3

A comparison of two different approaches

This chapter presents an extensive comparison between NPL's iGen system and JV's iMet system. The comparison parameters were agreed upon during the work, and advantages and disadvantages were found for both systems. Areas of improvement were investigated and implemented in the next system version.

3.1 Background

In 2004 it was decided to compare NPL's and JV's approaches to Internet-enabled calibration. Their approaches differed significantly, and the two institutes felt it would be advantageous to explore these differences.

The two systems involved were NPL's iGen, based on the iOTDR [90] system, and used for optical fiber measurements with Optical Time Domain Reflectometers (OTDR) and JV's iMet [19], used for remote calibration of electrical instruments via the Internet.

NPL have done a lot of work in the Internet-enabled calibration area, and in the last years several systems, like iPIMMS, iOTDR, iVR and iCOLOUR [24], have been developed. These systems all belong to the type 1 architecture, described in 1.6.1. The iGen system evolved from the iOTDR system, and thus inherited its type 1 architecture.

JV had done little work in the area before 2002, and had to choose one of the three architectures. The type 3 architecture, described in 1.6.3, seemed the better choice, since it was decided that the calibration process should be controlled by

personnel at JV without the need for traveling to the DUT. The iMet system was therefore of type 3.

The institutes' motivation for performing this comparison has been to show which possibilities each of the systems provide, and how they may solve the same challenges in different ways (e.g. two different ways to perform Internet-enabled calibrations). At the same time, it was important to see how the systems differ to enable the readers of the comparison report to decide which may be better for their own needs. The comparison also showed areas of improvement for each of the systems.

The work consisted of describing each system in equal terms, identifying comparison points, and then perform a thorough comparison based on these points.

The list in 1.7 was identified as the key properties on which the comparison was based, and sufficient to describe and compare the iGen and iMet systems.

3.2 iGen

3.2.1 System structure

The system architecture is shown in Fig. 3.1. The software is composed of a client, written in Microsoft Visual Basic (VB) [91], which runs on a computer located at the customer's site, and a server program, written in PHP [92], which runs on a server computer at the calibration laboratory.

The iGen server uses a MySQL database [93] for storage, which contains instrument information, calibration history data and measurement procedures.

3.2.2 System functionality

In the client, a Visual Basic control called INET [94] is used to handle communications with the server. The control calls a function "execute" to send messages to the server. This function takes the address of the server and the message to be sent, in Extensible Markup Language (XML) format, as parameters. The protocol in use, XML-RPC [95], uses HTTPS as transport, and is thus quite firewall-friendly. The XML-RPC protocol enables structurally complex and binary data to be exchanged.

When performing a calibration, the server controls the calibration process, while the client, using instructions received from the server (in the form of VB Script [96]), controls the equipment and interacts with the operator. Data collected from the instruments are transmitted back to the server for processing, and the results are stored in a database.

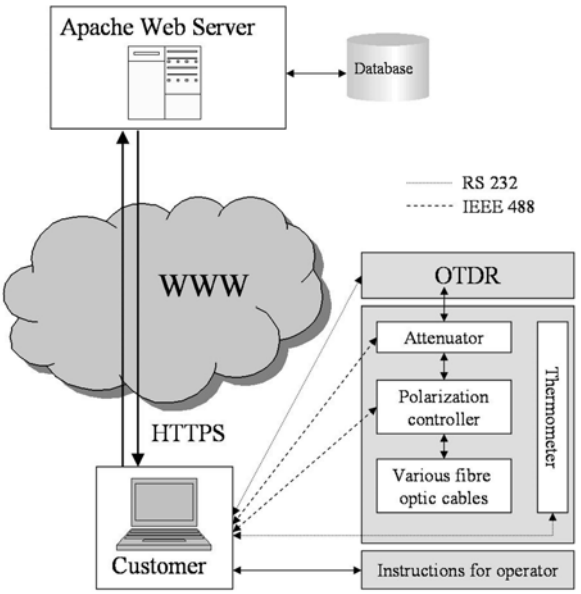

Fig. 3.1: iGen architecture.

Although the server software dictates how the calibration is conducted, the client software is in control throughout the process, and initiates all communication. The client sends a message to the server and waits for a reply. Upon receipt of each message, the server acts and then issues a reply.

A calibration is performed by executing a sequence of procedures on the server. Effectively, the server is state driven. Each message sent from the client includes the name of the function that the server is to execute, this name having been sent from the server to the client in the previous message. The first message from the client names the "loginpage" function that the server uses to log the user on to the system. The server returns an HTML page to the client enabling the user to enter the login name and password. It also returns the name of the function that must be called next, to start the calibration sequence. The client responds with a new communication that reactivates the server. The message includes the login name and the password, and the name of the function to be executed.

The client software has the capability to talk to up to four RS232 ports and as many IEEE488 [42] addresses as the National Instruments card can handle. A Visual Basic control is included for running VBScript. VBScript is a subset of Visual Basic, and it can be sent down from the server as an ASCII string to be run on the client. The VBScript can include calls to Visual Basic functions

on the client that read data from or write instructions to the hardware. These function calls can be embedded in loops.

3.2.3 Demonstration

The complete calibration of an OTDR consists of four different calibrations: the wavelength, the distance scale, the linearity and the loss scale factor. The iGen system has been used to perform two of these calibrations over the Internet, namely the distance scale and linearity. The customer is guided through each of these calibrations over the Internet including the set up of the instruments. The reference standard used is a calibrated fiber. Fig. 3.1 shows the instrument setup for calibration of linearity of the OTDR. Once the instruments are correctly set up the calibration is completely automatic (using a VBScript), with checks to see whether instruments are correctly set up, automatic adjustment of the instruments, loops to wait for instruments to complete their measurements and measurements of key environmental factors like temperature.

3.3 iMet

3.3.1 System structure

The system architecture is shown in Fig. 3.2. The system consists of a LAN computer with some connected instruments, an operator's LAN computer and a public Microsoft IIS Web Server. The operator could be co-located with the instruments, and then only one LAN computer is needed. The instrument computer runs a specific .NET Remoting [73] client application, which handles communication with the server and the locally connected instruments. The operator's computer also runs a .NET Remoting client application, which handles communication with the server. The latter application is built to control the former, so that the operator may access the remote instruments as if they were connected locally. The client applications communicate via a software object hosted on the server. Both the clients and the object are written in Microsoft C# [51].

The server utilizes data stored in an SQL Server database [97], which contains instrument information, calibration history data and measurement configuration data.

3.3.2 System functionality

Both clients set up a full-duplex Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS) channel to the server, before logging on. This enables data to

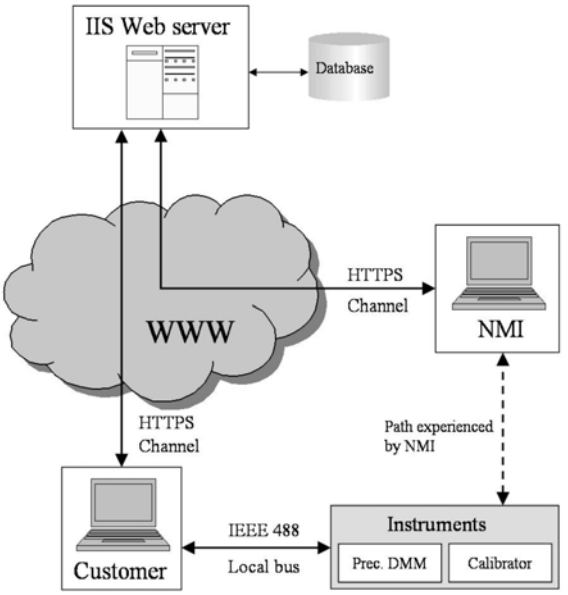

Fig. 3.2: iMet architecture.

be sent back and forth without inefficient client polling. The solution is based on the code provided by GenuineChannels [74]. The channel basically consists of two regular HTTPS connections, where one is in a pending HTTP request mode, enabling the server to contact a client instantly.

After logging in, the clients may communicate seamlessly via the server, without configuring local firewalls, proxy servers or Network Address Translators (NAT). This is due to the firewall-friendly nature of HTTPS. To the clients, the server is transparent, and it seems to them that the inter-client communication happens directly.

A calibration is performed by calling a preinstalled calibration procedure at the instrument computer. The operator has access to this procedure via the interface implemented by the client application at the instrument computer. The procedure may run for hours, and when finished the results are returned to the operator. The process is asynchronous, and the operator may perform other tasks while waiting.

There is no limitation as to where the operator may sit, as long as there is an Internet connection available.

The system may also be used for remote instrument control, where an operator sends individual instrument commands to the remote instruments. To the

operator, the remote instruments seem locally connected, except for possible network delays. Tests have shown typical delay times of up to two seconds back and forth. It is difficult to say if delay times of this order are acceptable or not, but there is a trade-off between this added functionality and the delay times. Allowing trusted experts to operate an instrument remotely (e.g. to try to find the source of odd behavior), one might have to accept delay times of some degree.

The .NET application at the instrument computer communicates with the connected instruments via the Virtual Instrumentation Software Architecture (VISA) interface [54]. The VISA interface makes it possible to communicate over several hardware interfaces, e.g. RS232 or IEEE488, without the need to change the control application.

3.3.3 Demonstration

As described in 2.7, Justervesenet tested the system in a realistic setup with two connected client computers, located at Justervesenet, and the web server located at UniK. Although the client computers were connected to the same LAN, they communicated via the server, and the complexity of the type 3 architecture was still preserved.

The structure of the procedure described in 2.7 allowed the system to perform calibrations, where one instrument was used as reference and the other was the Device-Under-Test (DUT). On calling the measurement procedure, the operator provided additional measurement configuration data from the database located at the server. This configuration data contained information about the iteration process, such as how many iterations to perform per data point, which data points to generate and measure, and the period of time to stabilize the signals.

3.4 Comparison

3.4.1 The Operator

The system design depends on who is supposed to initialize the measurement procedures or operate the instruments. The complexity of use depends on the operator's skill.

iGen

The iGen system is supposed to be operated by a customer. Because the customer often has little technical background, the system is intuitive to use, and provides step-by-step instructions. The system uses standard web pages to guide the user, which makes it possible to apply standard web design rules [98].

The system could also be used by expert users.

iMet

The iMet system is supposed to be operated by expert users at JV, which means that the system may require some education of users.

The system is more rigid with regard to giving feedback to the operator.

Conclusion

The iGen system seems easier to use than the iMet system, because the latter requires more user training. Because the iMet system is operated by certified NMI personnel, the system might be easier to accredit, which means being able to document that the system can perform certain actions, e.g. be used in Internet-enabled calibrations.

3.4.2 Operator localization

The localization of the operator is important for the design of the system. This can be seen from the three architecture designs given in 1.6.1, 1.6.2, and 1.6.3. When moving from type 1 to type 3, the complexity of the web communication increases along with added functionality.

iGen

The operator sits with the instruments when using the iGen system. This means that complex distributed communication solutions are not required, but the operator has to travel to the instruments.

The system was designed to be used by the customers, so the operator would normally be with the instrument.

If external expertise is needed, the expert must travel to the instruments.

iMet

When using the iMet system, the operator may sit wherever an Internet-connection is available. This means that an expert does not need to travel to the customer's instruments to perform an Internet-enabled calibration, but may remain in his or her own office.

This does not prevent an NMI customer from using the system. A customer may connect to the server using a specific *customer* role, which gives him or her access to control his or her own instruments.

The separation of the localization of the instruments and operators also means that it is easier to utilize external expertise. E.g. it is possible to let instrument experts on other continents help in the remote measurement process.

Conclusions

The two systems differ in architecture due to the choice of operator localization. The iGen system will be operated by the customers directly, while the iMet system can be operated by NMI personnel as well. If help from external expertize is required, the iMet system is more flexible in that the expert does not need to travel to the instrument.

3.4.3 Operator interaction

The error rate of the system partially depends on the degree of user interaction needed to operate the instruments or run measurements. When lengthy operations are performed it would be beneficial if the system could operate autonomously.

iGen

The iGen system is designed in such a way that the operator initiates each step in the calibration process. Lengthy measurements could be set to run for hours, without user interaction.

iMet

The iMet system is designed to have different degrees of user interaction.

The system could be used like the iGen system, where the operator initiates each step in the calibration process. There is only one calibration procedure, but the measurement could be configured to run for several hours.

The system could also be used for direct instrument operation. This means that the system autonomy is reduced to access control and error checking, while the operator is in charge of the high-level system operation.

Conclusions

Both systems may be configured to be self-operating, which means that they may run measurements for an extensive period of time with no user interaction. This may potentially reduce the error rate.

In addition, the iMet system is designed for full user control, which means that direct instrument operation is possible.

3.4.4 Data handling

The way data is handled and presented to the operator is important when analyzing the data and making decisions based on the results. When handling

calibration data it should be possible to tie it to a specific time, place, instrument, customer and operator.

iGen

The iGen system uses regular web pages to present the data. This means that the data could be presented as text in tables, or using dynamically generated graph images. At present the results are shown as text.

After performing a calibration, the calibration results are stored in text files. The system is likely to be rewritten so that the results can be stored directly in a database instead.

The data is not signed digitally or stored with a Message Authentication Code (MAC), which means that it is not possible to prove the integrity of the data.

iMet

When the operator receives measurement data, it is presented in a custom graph field. The graph field is developed in such a way that data sets of varying sizes can be shown. The somewhat rigid measurement procedure at the customer side, results in a somewhat rigid data presentation at the operator side.

The data is stored in a text file. As for the iGen system, it would not be challenging to store the data in a database at the server.

As for the iGen system, the data is not signed digitally or stored with a MAC, which means that the integrity of the data is not preserved.

Conclusions

The iGen system is potentially more flexible as to how measurement data is presented to the operator. This might be important when making decisions based on the measurement results.

The systems should add support for digitally signing measurement data or using MACs, so that the integrity of the data could be preserved. This would be especially important when performing Internet-enabled calibrations, where the calibration data must be available and verifiable for a long time. Using digital signing, the data could be bound to a specific instrument and customer at a specific point of time.

3.4.5 Security

When including the Internet in a measurement system, it is crucial to focus on security. If the security of a system is breached, damage control mechanisms should exist to reduce the effects of the breach.

iGen

The iGen uses a secure channel between the customer and the server, according to the HTTPS standard. The system uses certificates to authenticate the server to the customer, and the customer provides a username and a password to authenticate himself or herself to the server.

The server firewall may be set to only allow certain IP addresses to connect to it.

iMet

The iMet system also secures the communication using HTTPS. In addition to using server-certificates when authenticating the server to the instrument computer and operator computer, the system uses client-certificates, signed by the server, in the authentication of clients to the server. The clients also need to provide a username and password to fulfill the authentication process.

A user may log in to the system as *operator* or *customer*. Most customers are of the latter type, and have reduced access rights and privileges.

As for the iGen system, the server firewall could be configured to only allow certain IP addresses.

Conclusions

The two systems both secure the communication between the client and the server, although the iGen system is more vulnerable to lost usernames and passwords, due to the lack of certificate-based client-authentication.

3.4.6 Availability

Internet-enabled instrumentation systems should be accessible to all NMI customers, meaning that they should use firewall-friendly protocols and data formats. The services should be available to authorized users whenever a measurement is to be performed. The systems should be as robust as possible concerning changing network conditions.

iGen

Because the communication is done over HTTP, the system is quite firewall-friendly. When behind a non-transparent proxy server, as described in B.2.2 and B.2.3, the clients may not connect to the server.

Due to the choice of software, the clients must be run on the Windows platform.

The iGen server could be set to run continuously, so that it is always accessible to authorized users.

The system is quite robust to changing network conditions, since the instrument communication happens locally.

iMet

The system works with most firewalls, due to the choice of HTTP for communication. The system also works behind most types of proxy servers, because the clients may be configured with proxy authentication information.

The system must be run within the .NET Framework, which traditionally means that it is bound to the Windows platform. As seen in 2.3.5, solutions exist to port the framework to other platforms. The system has not been tested on other platforms, and it is not known how much work would be required to port the client applications.

The iMet server could also be set to run continuously, and it would thus always be accessible to customers and NMI operators.

The system can be quite robust with regard to changing network conditions, when running calibration procedures on the instrument computer. If operating instruments via the remote Read, Write, and Query methods, or implementing procedures on the operator's computer, the effects of changing network conditions can be severe.

Conclusions

Both systems are quite available on demand, though only the iMet system seems to works behind non-transparent proxies.

The iMet system is more sensitive to changing network conditions, especially when operating instruments directly.

Both systems are bound to the Windows platform, and solutions should be sought to broaden the platform support.

3.4.7 Scalability

An important feature of distributed software systems is how well they adapt to handling many users concurrently. For instrumentation systems its important to be able to add new measurement procedures and support for new instruments.

iGen

New measurement routines may be added as source code to the server database, while the system is running.

The system supports IEEE 488 and RS232, and may communicate with most instruments over these buses.

The server can handle several simultaneous users, because the it does not store much state information about each connected client.

iMet

New measurement routines may be added to the customer application only by rewriting and recompiling the application. The system was developed with a specific calibration in mind, involving a reference instrument and a DUT, with focus on configuration of the calibration routine. This configuration includes setting the reference and DUT, data points, signal durations, and waiting periods.

The system is only tested with IEEE 488, although it may potentially communicate with any hardware interface supported by the VISA standard.

Compared to the iGen system, the server stores more state information about each connected client. This is partially due to security reasons, but it is also necessary for the bidirectional inter-client communication. Thus, the server does not scale as well as the iGen server.

Conclusions

The iGen system is more scalable with regard to handling many users and adding new measurements.

The iMet system has an advantage when adding new instruments, because it works over VISA.

3.4.8 Development

The difficulty of further developing the systems or fix software bugs, depends on the choice of programming language, the development platform available and system documentation.

iGen

The iGen is developed using Visual Basic, VB Script, PHP and SQL, but lacks a good development platform. This, due to the heterogeneity of the software, makes debugging quite challenging, and error messages are often too general.

iMet

The iMet system is developed using C# and SQL, and good development platforms (Microsoft Visual Studio or SharpDevelop [99]) are available. Due to

C#'s object-oriented nature and it's exception handling capabilities, it is easier to create and maintain more complex systems.

The system also supports auto-generation of code documentation, a built-in feature of the Visual Studio development platform.

Conclusions

The iMet is easier to develop and maintain, because of the chosen programming language and development platform. The system also support auto-generation of source documentation.

Common to both systems is that only software experts may add new measurements or support for new instruments. The systems could be provided with management software enabling non-experts to construct and add measurements. This would require the development of well-designed and reusable components.

3.4.9 Configuration

The ability to configure parts of a system, without rewriting it, can sometimes be important. IP addresses or TCP ports may be changed, and new protocols may be introduced, which requires the systems to be changed. Security settings could also be made configurable. The configuration options could be hard-coded into the source code, it could be done using configuration files, or the configuration could happen in runtime with user input.

iGen

The iGen system uses XML-based configuration files to set up ports, addresses, and the protocol in use. The user provides login information at runtime.

iMet

As for the iGen system, the iMet system also uses XML-based configuration files to set up ports, addresses, and protocols in use. Login information is provided by the users at runtime.

Conclusions

Both systems use configuration files to set up the system, which make both systems quite flexible with regard to changing environments. They could add support for flexible layout configuration, so that users with special requirements could change font sizes, colors, or line thicknesses.

3.4.10 Usability

Usability means how easy it is to use a system for new users, and elements like guidance and feedback are important factors.

iGen

The user is guided through the use of web pages. These web pages may look different for different measurements, and standard web design techniques may be applied to make the pages more readable. User interaction involves picking elements from drop-down lists, reading text and pushing buttons, which most users are familiar with through the use of GUI OS's like Microsoft Windows, Linux or MAC.

When lengthy processes are performed, the user is told to wait through the use of progress bars.

iMet

The user can use the system in two ways, either for remote instrument operation, or for running remote measurements.

When operating instruments remotely, the operator has access to the instruments as software objects. Through these objects the operator can use instrument-specific commands. The commands are accessible from a drop-down list, and there are also free-text fields to enter custom instrument commands. Results from the instrument communication is shown in text boxes. For direct instrument operation, the usability of the system is somewhat low for first-time users.

Running remote measurements, the operator chooses the measurement from a list, the elements of which depend on the instruments connected to the customer. Lengthy operations are visualized through the use of progress bars. The operator is also notified by progress messages, which describe the measurement progress. When running remote measurements, the usability of the system may be compared to that of the iGen system.

Since the system is supposed to be used by skilled NMI personnel, the users will be trained before using the system.

Conclusions

For first-time users, the iGen system is easier to use than the iMet system. The systems should both focus on using standardized design techniques to enhance the system usability, thus reducing the error rate.

3.5 Conclusions

Two different approaches to Internet-enabled calibration have been analyzed and compared, NPL's iGen system and Justervesenet's iMet system.

There are pros and cons associated with both systems, and these may be used as guidelines for developing new systems. NMIs planning to develop an Internet-enabled measurement service must decide upon a few key points

- Who should be the operator?
- Where should the operator be located?
- Should the service be accredited?

The reason to focus on these points, is that they are crucial to the system and difficult to change after the system has been developed.

Looking at the two systems described, the most important difference appear to be the structure. For iGen, the customer is supposed to be the operator, and the system is constructed in such a way that the operator and equipment are placed together. For iMet, an NMI person is supposed to be the operator, and the system is constructed such that the operator and the equipment may be separated by the Internet. An iMet operator may remotely control the calibration process without the need to travel to the equipment under test. This adds to the complexity of the communication software, or the middleware, in use. This complexity makes the system more sensitive to changing network conditions, like response times, network speed and data transfer. It is important to program the system to act in predicted ways, when special situations arise, e.g. broken connections, timeouts, and software exceptions.

The difference in operator role also reflects the usability of the system. The iGen system is believed to be easier to use for new users, due to the flexibility and feedback capabilities of HTML. Detailed instructions are given to the operator by server-generated web pages. For the iMet system, the guidance of the user is not as flexible, and more user education is needed before using the system.

The development environment for iGen is more heterogeneous than the .NET facilities available to iMet, due to the use of multiple software languages. The complexity of a distributed software system often dictates the choice of software. It is easier to use one or a few programming languages to program a system, than to use many languages and depend on inter-language communication. It is also important to have a reliable development platform, on which to develop the major part of the program. This makes it easier to discover software bugs and to test the system.

If developing an accredited calibration service, the security of the system must be well-documented and follow standard recommendations. The security and

quality of the calibration should be as good as if the calibration was performed directly in an NMI laboratory. Both systems described are potential candidates for accreditation, but certain things need to be included or changed

- The NMI needs to be sure, when including customers in the calibration process, that enough user education is provided. Some kind of certificate should be issued to "accredited" customers.
- Equipment involved in the measurement process needs to be authenticated in a satisfactory way. As seen in 2.5, this can be quite challenging.
- The data collected in a measurement should maintain integrity during production, transportation and storage. This could be solved using digital signatures.

Chapter 4

The iMet system v.2.0

This chapter presents the second and improved version of the iMet system. In addition to utilizing the secure communication channel found in the first version, new measurement procedures may now be added in a consistent way and run when needed. The results from the comparison, described in chapter 3, suggested that measurement procedures should not be hard-coded in the control application, but in stead be available for download from a dedicated database server.

4.1 Introduction

The second version of the iMet system was developed with the comparison described in chapter 3 in mind. This comparison showed that the iMet system, though flexible with regard to the location of the operator, was not very scalable in adding support for new measurements. The system only supported one calibration procedure, preinstalled in the customer application. Although this procedure could be configured to work with a variety of instruments and different data points, it was desirable to allow authorized users to add custom measurement procedures, at runtime without recompilation, which could work with an indefinite number of instruments.

By running measurements installed at the customer, the operator also gained less insight into the measurement process than had it been run locally at the operator's computer. It was therefore important to check if it was practical to let the measurements run on the operator's computer, and which effects this had on the running time, network traffic and security. Luckily the original iMet system had been developed with this in mind, and by using the Read, Write, and Query

method, described in 2.6.1, a measurement procedure could be run on any host correctly implementing the InstrumentCommunicator interface. This process is described in 2.6.4.

It was also decided that the new system should be tested in a real calibration, in collaboration with one of Justervesenet's customers. The reasons were twofold. First, it was important to test the system on real customers to discover possible weaknesses in design and functionality from a real user's point of view. Second, it was a good opportunity to test which instruments could potentially be used as transfer standards. The transfer standards were to be sent by mail using the regular postal service.

4.2 iMet overview

Like the first version, the second version of the system consists of multiple client computers and one public web server. The clients may communicate seamlessly with each other via the server over secure, full-duplex channels, using the hypertext transfer protocol over secure sockets (HTTPS). As described in 2.3.4, potential problems concerning traversing firewalls, proxy servers and network address translators (NATs) are bypassed, because the HTTP packets are allowed through as long as the connections are initialized from the inside. Moreover, all clients are authenticated to the server with server-signed X.509 client certificates and a username and password. The server authenticates itself to the clients using an X.509 server certificate signed by a trusted third-party certificate authority.

After connecting to the server, a client may offer his or her instruments as services to an external operator. That means the operator may use the client's instruments as if they were connected locally to the operator's computer. The system architecture is shown in Fig. 4.1.

What differs in the second system version, is the ability to add measurements procedures to the system during runtime, without restarting or recompiling the system. The procedures are stored in a public database, and may be downloaded and run by authorized users whenever needed. The measurement procedures may either be run on the operator's computer or at the customer's computer. The procedures may potentially be run directly on the public relay server as well, though this was not tested.

4.3 Measurement and calibration procedures

One important feature of the C# programming language, used to program most of the system, is the capability of compiling source code in memory during runtime. A running C# object may take a piece of source code, describing another C#

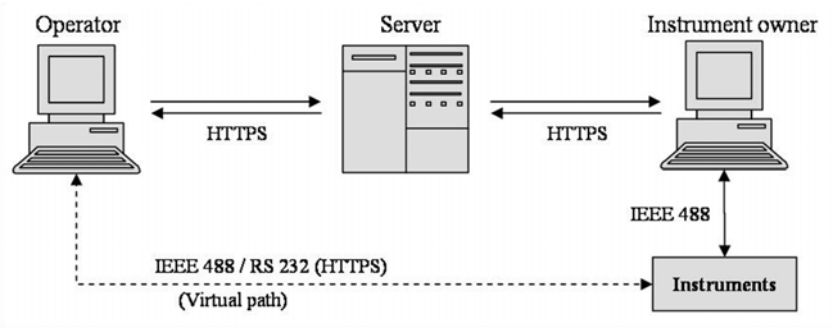

Fig. 4.1: The iMet v.2.0 architecture. The operator may operate instruments remotely via the public web server. The instruments seem locally connected to the operator's computer.

object, and compile and instantiate it. The new object becomes an effective part of the running object, and may be used as if it was compiled beforehand.

By utilizing this feature, the iMet system v.2.0 is able to download and dynamically compile and run measurement procedures available as source code. This means that the system could support an indefinite number of measurement procedures, as long as they are accessible from a database.

Instead of installing a procedure in the customer application, as was done in the first system version, the customer application now contains means to download and compile procedure source code and a container for running the compiled procedures. This is shown in Fig. 4.2.

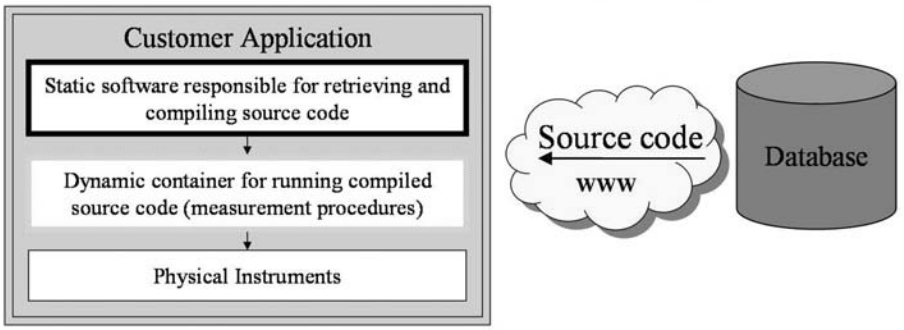

Fig. 4.2: The architecture of the new customer application. Instead of being hard-coded into the application, the measurement procedures are now downloaded as source code in runtime, and compiled and run.

The well-tested measurement procedure implemented in the previous customer application version, was stored as source code in the system database and used to do initial testing on the system. This was done to eliminate potential errors associated with the procedure. Instead, the focus was on the dynamic source code retrieval and runtime compilation.

The iMet system could still be used as a general instrument control system, where an operator operates an instrument remotely using single instrument commands and the Read, Write, and Query methods. More efficiently, one could construct more complex measurement or calibration procedures for automatically performing lengthy and complicated measurements, and store them in a database. The procedures may be downloaded, compiled in memory and run on request, and they may be run on any of the computers implementing the InstrumentCommunicator interface. This means that new procedures may be added to the database, without the need for recompilation of all system components. The procedures operate on instrument interfaces, thus allowing the same procedure to be used for different instruments implementing the same interface. This interconnection process, including downloading source code from the database, is visualized in Fig. 4.3.

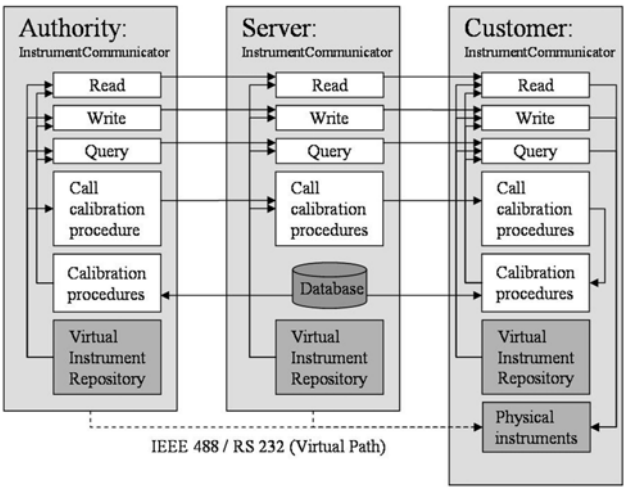

Fig. 4.3: The iMet system v.2.0 communication architecture. By implementing the InstrumentCommunicator interface correctly, an application may operate a customer's instruments using the Read, Write, and Query methods or run calibration procedures. The calibration procedures may be downloaded from a database and run at runtime, without system restart or manual recompilation.

4.4 Security

Most of the security discussion in 2.4 also applies to the iMet system v.2.0.

There is, however, one important difference, associated with the dynamic, in-memory compilation of downloaded source code. If malicious code could be inserted into the database by hackers, the result could be severe damage at the client side as the code is compiled and run, e.g. the deletion of system files. It is therefore important to check the validity of the code before compiling and running it.

Although not implemented in the system, the code could be signed by a trusted party before inserting it into the database. Practically, what would be signed is a small hash value, generated from the code. This way, the client who downloads the code could first check the signature against the public certificate of the trusted party. If the signature is valid, the client knows that the code is safe to run. More importantly, if the verification process fails, the code will not be run.

Clear agreements should be made beforehand, so that all clients have an understanding of who is responsible if some code is not running as expected.

4.5 Measurement scenarios

There are different ways the iMet system could be utilized. The operator may sit at any computer with an Internet connection, and he may control any number of instruments or measurement processes at the same time. The operator's location is completely independent of the location of the instruments.

4.5.1 Operator and instruments on the same computer

Inexperienced customer

For inexperienced customers, the system could be used as guidance when performing measurements and operating instruments. The system helps to chose from available experiments, with detailed descriptions of configuration possibilities, and it provides standard commands for operating connected instruments. This way, the operator do not need to know the details of the instrument communication, but could utilize higher-level tools.

Experienced customer

Experienced users could also benefit from using the system, as it provides consistency in the measurement process. Instead of writing their own measurement procedures or instrument commands, they may utilize the system's

well-tested measurement procedures and standard instrument commands. This way, the confidence of the measurement results is increased.

If all personnel in a company use the system, it would result in increased consistency and effectiveness.

4.5.2 Operator and instruments on different computers

Experienced customer

By using the system, experienced users could keep track of their ongoing measurements or configure and initiate measurements from remote (e.g. their own home).

Before leaving work, the measurement is started, and the measurement progress could then be monitored from home. The measurement could also be started from home, but the Internet connection would still need to be set up beforehand, due to availability issues discussed in 2.4.6.

NMI personnel

The original motivation for developing the iMet systems, was to allow NMI personnel to perform Internet-enabled calibrations directly at a customer.

By using the system, the operator may control and monitor the calibration process from his or her own office, instead of traveling to the customer. The benefits of doing calibrations this way was mentioned in 1.5.

External expert

Inexperienced customers may outsource entire experiments to be controlled by external experts. These experts could be located anywhere in the world. The experts could also be consulted when experiencing instrument problems, such that the experts have direct access to the instruments and can do the relevant error checking.

4.6 Time delays

When instruments are physically connected to an operator's computer, the instrument communication is instantaneous. Things change when including the Internet in the measurement process. Operating instruments via the Internet, one will always experience some time delays, usually in the area of a few seconds. This can feel quite unnatural to the operator, but there are a few ways to compensate for it.

Configuration	$03 \times 2s = 06s$
10 readings	$10 \times 2s = 20s$
Total delay	$26s$

Tab. 4.1: The accumulated network delay.

Configuring two instruments	$\approx 03s$
Stabilizing readings	$120 - 600s$
Make ten readings	$10 \times (5 - 15s) = 50 - 150s$
Total measurement time	$173 - 753s$

Tab. 4.2: The measurement time.

Generally, time could be saved by bundling commands together before sending. Instead of sending multiple commands one after the other, with network delays before and after each command, the operator could send all commands in one bulk. This would only add time delays before and after the whole operation, but would make the system act more like a "black box" to the operator.

4.6.1 Example

A typical example would be to operate a digital multimeter (DMM) remotely to do 10 high-resolution DCV readings from an electrical calibrator. The measurement procedure could run locally at the operator's computer.

Three bundled commands would be needed to set up the instruments, two for configuring each instrument and one for putting the calibrator in running mode. The instruments then need some time to stabilize. The multimeter might use a few seconds integration time per reading. Typically, the time delay is approximately one second each way on average, or two seconds per command sent. That means, it will take two seconds to send a command and receive confirmation when the command action has been performed. The total time for the delay and the measurement can be seen in Tab. 4.1 and 4.2.

If looking at single readings only, the worst case scenario is to use the minimum integration time of five seconds. Then the network delay would amount to about 2/(2+5)=28.6% of the total time, that is, the time to make one reading including delay.

If using the maximum integration time of fifteen seconds, the delay would be about 2/(2+15)=11.8% of the total time.

The formula for finding the delay-total time ratio is

$$\frac{Delay}{Delay + Measurement} = \frac{t_1 \times n_1 + t_1 \times n_2}{(t_1 \times n_1 + t_1 \times n_2) + (t_2 + t_3 + t_4 \times n_2)} \tag{4.1}$$

where t_1=time delay of sending one command, n_1=number of configuration commands, n_2=number of reading commands, t_2=total configuration time, t_3=stabilizing time (120-600 seconds), and t_4=integration time (5-15 seconds).

For the described measurement, the delay-total time ratio would lie in the range 3.3-13.1%.

When increasing the number of readings, the delay time will asymptotically converge to somewhere between the worst-case and best-case scenario when making one reading.

$$\lim_{n_2 \to \infty} \frac{t_1 \times n_1 + t_1 \times n_2}{(t_1 \times n_1 + t_1 \times n_2) + (t_2 + t_3 + t_4 \times n_2)} = \frac{t_1}{t_1 + t_4} = 11.8\% - 28.6\% \tag{4.2}$$

Typically, the stabilization time will count for a substantial fraction of the total measurement time, so the effective impact of the delay time will be lower than these limits.

It is important to notice that for direct instrument operation, the time to perform a single command could be in the area of milliseconds. Then the delay time would make up almost 100% of the total time.

4.6.2 Delay compensation

Utilizing waiting time

If there is some waiting time, T_{wait}, between each reading, it is possible to compensate for the delay times, as shown in Fig. 4.4, thus reducing the effect to almost zero.

If we look at Fig. 4.4 we see that

$$\Delta_1 = t_2 - t_1 \tag{4.3}$$

and

$$\Delta_2 = t_4 - t_3 \tag{4.4}$$

The total delay can be expressed as

$$\Delta_1 + \Delta_2 = (t_2 - t_1) + (t_4 - t_3) \tag{4.5}$$

or

$$\Delta_1 + \Delta_2 = t_4 - t_1 - T_{int} \tag{4.6}$$

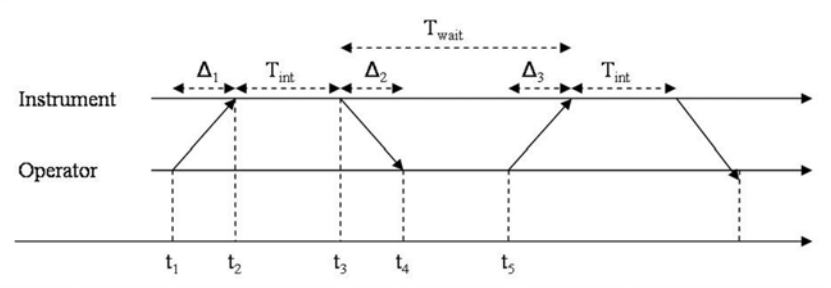

Fig. 4.4: Measurement time delay. t_1 = command sent from operator to instrument. t_2 = command received at instrument. t_3 = reply sent from instrument to operator. t_4 = reply received at operator. t_5 = new command sent from operator to instrument. T_{int} = integration time per reading.

The advantage of doing the latter calculation compared to the former, is that, given T_{int}, only one computer clock is involved.

As long as

$$T_{wait} \geq \Delta_2 + \Delta_3 \tag{4.7}$$

the time delay can be compensated for by sending the next reading command at t_5 such that the command arrives T_{wait} seconds after the last reading. Due to varying network conditions, this is quite challenging. To simplify, we assume that $\Delta_1 = \Delta_2 = \Delta_3 = \Delta$, so that

$$\Delta \approx \frac{t_4 - t_1 - T_{int}}{2} \tag{4.8}$$

which leads to

$$t_5 - t_4 \approx T_{wait} - 2\Delta \tag{4.9}$$

or

$$t_5 \approx t_4 + T_{wait} - 2 \times \frac{t_4 - t_1 - T_{int}}{2} = t_1 + T_{wait} + T_{int} \tag{4.10}$$

So for each reading the operator needs to calculate Δ (or *average* Δ based on the last delay calculations) and check to see if $T_{wait} \geq 2\Delta$. If it is, the next reading command is sent at t_5, according to Eq. 4.10. If not, the command is sent straight away.

Block readings

It is also possible to make block readings on some multimeters, where a number of readings are made consecutively and stored inside the instrument before returned

to the caller, and thereby effectively reducing the delay effect to almost zero. But this reduces the visibility of the process to the operator; the operator initiates the block reading, but he cannot see the details of the measurements before the block is finished.

Remote execution

One could also download the calibration procedure, or parts of it, to the computer connected to the instruments, and compile and run it there. This would also reduce the influence of delay to almost zero. The operator might then gain insight to the measurement process if the instrument computer sent him update information. The process would go as follows

1. The operator tells the instrument computer to download, compile and run a specified measurement procedure
2. For each command performed, the instrument computer notifies the operator (this could be done in separate threads, so that the performance of the measurement is not affected)
3. The operator may abort the measurement at any point (it is important to design the measurement procedures to handle such scenarios)

Due to the remote measurement execution, the operator has less control if an unpredicted exception occurs in the procedure code or in the data connections, and potential reading data could be lost.

4.7 System test

In June 2005, Justervesenet performed an extensive Internet-enabled calibration test in collaboration with one of it's customers in Stavanger, Norway. Two DMMs, DMM1 and DMM2, were calibrated and sent to the customer's laboratory, as seen in Fig. 4.5, where they were used to do comparative measurements of a multifunction calibrator. The process was initiated and controlled directly from JV.

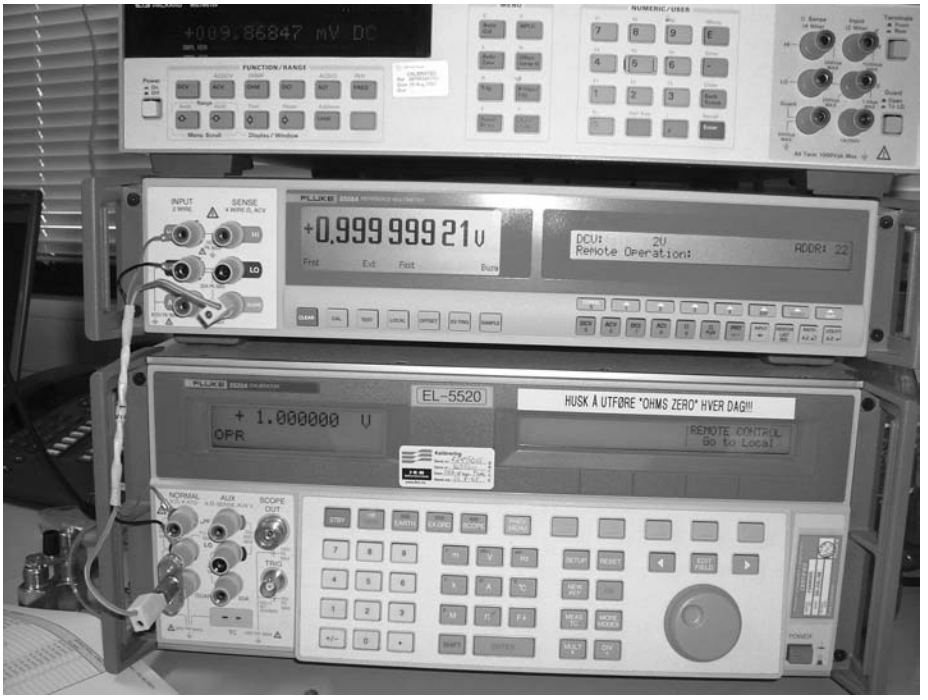

Fig. 4.5: Image of the instrument setup. Two multimeters were transported from Kjeller to Stavanger, then used to calibrate an electrical calibrator, after which they were returned to Kjeller for recalibration.

Five types of calibrations were performed, DCV, DCI, ACV, ACI and resistance. The same calibration procedure could be used for both multimeters, because they implemented the same instrument interface, as explained before. Only one procedure was used for all calibrations, and the logical structure of the procedure was as follows. The calibrator and the multimeter were set up to generate and measure the correct signal for each calibration point. A certain

amount of time was added between each point (for stabilization) and the readings were made equally spaced in time. The whole process was controlled directly from a laptop computer at Justervesenet. The operator first did some initial testing on the instruments to verify correct behavior. He then downloaded the correct calibration procedure, compiled it in memory and ran the procedure locally. The operator at Justervesenet and the customer used a phone to communicate verbally. The customer handled all instruments and needed guidance on how to connect them to each other and to the computer.

4.7.1 The results

The results were analyzed for all five types of calibration. The focus was on the behavior of the multimeters, before and after transport. The quality of the remote calibration at the customer is strongly dependent on the stability of the multimeters.

The communication between the operator and the instruments worked satisfactory. No significant time delays were observed when running the calibration procedure locally at Justervesenet. Because the procedures and the calibration points were stored in a database, it was easy to modify them without the need to restart the applications.

When the DMMs were returned to Justervesenet and recalibrated, the results showed that there were slight changes in their calibration values. From the analysis so far, it is difficult to say if this was due to natural drift or transportation. More analysis and experience with the DMMs is needed.

ACV

Before and after being used in the measurements at the customer, both multimeters were calibrated against a calibrated multifunction electrical calibrator.

Some changes were observed in both multimeters, as can be seen in Tab. 4.3 and 4.4. The changes can not be explained entirely from a drift only perspective. The voltages used in the calibration were 0.3V, 3V, 30V and 300V. These values are somewhat unfit for the voltage ranges of the two instruments, which were 2V, 200V and 1kV for DMM1 and 1V, 10V, 100V and 1kV for DMM2. This influences the uncertainties and measured values.

A (V)	B (Hz)	C ($\frac{\mu V}{V}$)	D ($\frac{\mu V}{V}$)	E ($\frac{\mu V}{V}$)	F (%)	G ($\frac{\mu V}{V}$)	H (%)	I (%)
0.3	1000	-85	-136	-51	-56	90	30	170
3	45	58	-6	-64	-116	100	40	160
3	1000	145	100	-45	-87	55	28	160
3	100000	-159	-172	-13	-8	155	2	590
30	1000	67	16	-51	-98	70	34	150
300	1000	-61	-93	-32	-43	75	20	160

Tab. 4.3: Change in DMM1 before and after transport. A = nominal voltage, B = frequency, C = relative aberration before transport, D = relative aberration after transport, E = Value difference (before-after), F = Difference relative to the uncertainty, G = JV uncertainty, H = change relative to the DUT specifications, I = DUT 90 days specification.

A (V)	B (Hz)	C ($\frac{\mu V}{V}$)	D ($\frac{\mu V}{V}$)	E ($\frac{\mu V}{V}$)	F (%)	G ($\frac{\mu V}{V}$)	H (%)	I (%)
0.3	1000	24	13	11	13	90	7	170
3	45	12	-12	24	44	65	15	160
3	1000	37	11	26	51	62	17	160
3	100000	-456	-501	45	29	155	8	590
30	1000	12	-14	25	49	70	17	150
300	1000	-41	-86	45	60	80	28	160

Tab. 4.4: Change in DMM2 before and after transport. A = nominal voltage, B = frequency, C = relative aberration before transport, D = relative aberration after transport, E = Value difference (before-after), F = Difference relative to the uncertainty, G = JV uncertainty, H = change relative to the DUT specifications, I = DUT 90 days specification.

The result of measuring the output signals from the DUT with the associated uncertainties are shown in Tab. 4.5.

The AC voltage results showed that the measured values were within the specifications of the calibrator. The AC voltage measurements compared to the 90 days specifications of the calibrator are shown in Fig. 4.6. The multimeters had changed too much to verify the specifications with a good margin. One of the calibration points (3V/45Hz) in the figure also have non-overlapping uncertainties for the two multimeters. Further work is needed to find the causes of the changes.

Set point		DMM1		DMM2	
Voltage (V)	Frequency (Hz)	Meas. value (V)	Uncert. (V)	Meas. value (V)	Uncert. (V)
0.3	1000	0.300019	0.000027	0.300026	0.000027
3	45	2.99962	0.00030	3.00027	0.00020
3	1000	3.00005	0.00017	3.00026	0.00019
3	100000	3.00061	0.00047	3.00069	0.00047
30	1000	30.0006	0.0021	30.0028	0.0021
300	1000	300.000	0.023	300.029	0.024

Tab. 4.5: AC voltage calibration results for the DUT with uncertainties.

To perform better in Internet-enabled calibrations, the uncertainties of the multimeters (transfer standards) need to be improved. The influence of transport and drift must be investigated, thus allowing lower uncertainties.

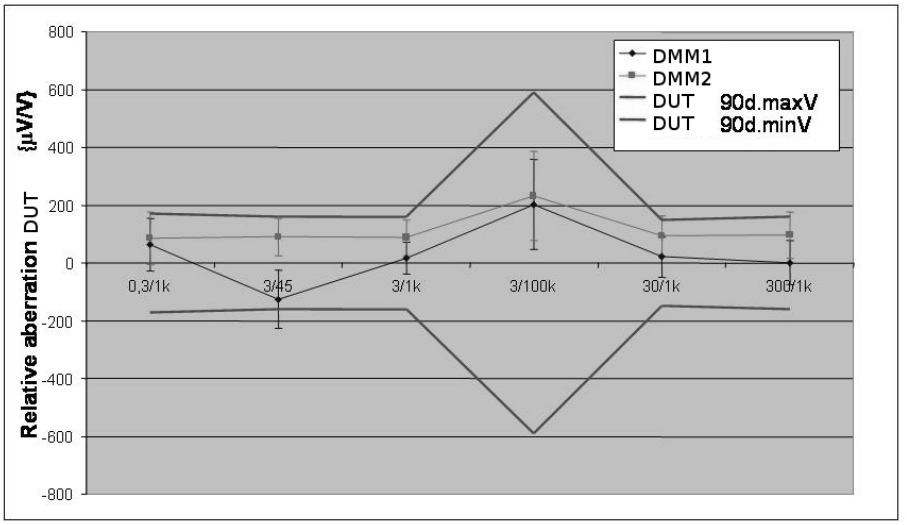

Fig. 4.6: Relative deviation with uncertainty for AC voltage measurements compared to the 90 days specifications for the calibrator.

ACI

As for AC voltage calibration, the two multimeters were calibrated against a reference standard, before and after being used in AC current calibrations at the customer. The values before and after are given in Tab. 4.6, and the AC current calibration results for the calibrator, using DMM2, is shown in Tab. 4.7. DMM2 showed better stability for AC current during transportation than DMM1. This can be seen in Fig. 4.7.

Set point		DMM2	DMM1
Current (mA)	Frequency (Hz)	Change (mA)	Change (mA)
3.29	45	-0.000150	-0.000100
3.29	1000	-0.000150	-0.000120
3.29	10000	-0.000190	-0.000120
0.19	1000	-0.000009	-0.000002
190.00	1000	-0.005700	-0.001600
1000.00	1000	-0.064000	-0.016000

Tab. 4.6: Change of ACI calibration values for DMM2 and DMM1 after transport.

Current Nominal (mA)	Frequency (Hz)	Meas. value (mA)	Corr. value (DMM2) (mA)	Total uncert. k = 2 (mA)
3.29	45	3.2901	3.2895	0.005
3.29	1000	3.2906	3.2903	0.005
3.29	10000	3.2906	3.2911	0.008
0.19	1000	0.1901	0.1902	0.00005
190	1000	190.0564	190.0999	0.05
1000	1000	1000.0582	1000.2022	0.5

Tab. 4.7: AC current calibration results for the DUT with extended uncertainty.

It is difficult to say anything about the transport uncertainties, due to lack of historical data for the two multimeters. But results could show that the multimeters are potential candidates as transfer standards for ACI. The changes in the multimeters are shown in Fig. 4.7.

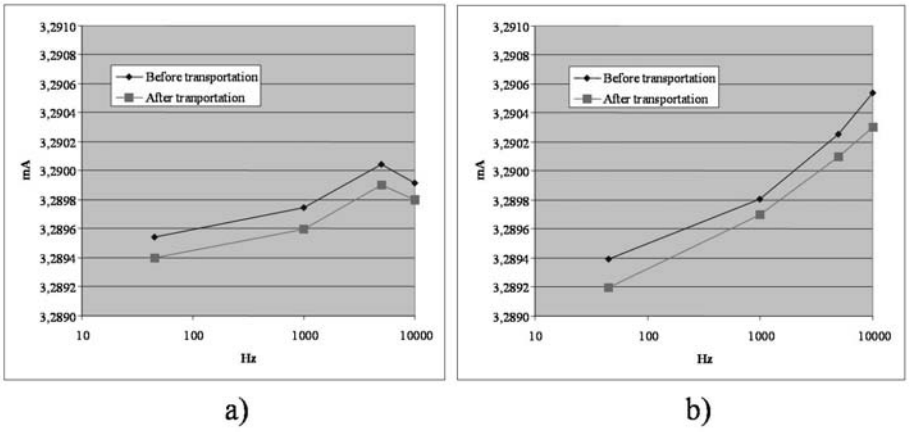

Fig. 4.7: AC current changes, for 3.29mA, before and after transportation. a) DMM1, b) DMM2.

DCV

The changes in the calibration values is given in Tab. 4.8. DMM2 showed greater change than DMM1, and the latter thus appears better suited for being used as a transfer standard for DCV. More work is needed before concluding finally.

The results from the remote DC voltage calibration of the DUT are shown in Tab. 4.9.

Voltage	DMM2		DMM1		DUT
Nominal (V)	Change (ppm)	Rel. change DUT spec. (%)	Change (ppm)	Rel. change DUT spec. (%)	90 days spec. (ppm)
1	1.49	14	1.02	9	11
-1	1.23	11	0.22	2	11
10	1.12	9	0.32	3	12
-10	1.30	11	0.41	3	12
1020	3.13	20	-0.03	0	16
-1020	3.00	19	0.06	0	16

Tab. 4.8: Change in DC voltage for DMM2 and DMM1 after transport.

Range (V)	Nominal voltage (V)	Measured value (V)	Total uncertainty k = 2 (V)
1	1	0.9999943	4.0μ
1	-1	-0.9999942	4.0μ
3	3	2.999983	12μ
3	-3	-2.999980	12μ
1020	1020	1019.9980	5.1m
1020	-1020	-1019.9979	5.1m

Tab. 4.9: DC voltage calibration results for the DUT with extended uncertainty.

DCI

The changes in the calibration values is given in Tab. 4.10. DMM2 showed much greater change than DMM1. DMM1 might therefore be a better candidate for being used as a transfer standard. More work is needed before concluding finally.

The results from the remote DC current calibration of the DUT is shown in Tab. 4.11.

Current	DMM2		DMM1		DUT
Nominal (A)	Change (ppm)	Rel. change DUT spec. (%)	Change (ppm)	Rel. change DUT spec. (%)	90 days spec. (ppm)
3.29m	-6.58	-7	3.34	4	95
-3.29m	-13.61	-14	-1.89	-2	95
32.9m	-47.51	-54	-0.18	0	88
-32.9m	-53.20	-60	0.49	1	88
1.09	-12.02	-6	-3.82	-2	200
-1.09	-13.95	-7	-1.75	-1	200

Tab. 4.10: Change in DC current for DMM2 and DMM1 after transport.

Range (A)	Nominal current (A)	Measured value (A)	Total uncertainty k = 2 (A)
3.29m	3.29m	3.289952m	66n
3.29m	-3.29m	-3.289932m	66n
32.9m	32.9m	32.89993m	0.66μ
32.9m	-32.9m	-32.89971m	0.66μ
1	1	0.999980	25μ
1	-1	-1.000027	25μ

Tab. 4.11: DC current calibration results for the DUT with extended uncertainty.

Resistance

Before and after being used in comparative resistance measurements, the two multimeters were calibrated against JV's resistance standards. The changes in the calibration values for the multimeters are shown in Tab. 4.12.

DMM2 showed some instabilities for 100 Ω at the customer, and exceeded the 90 days specification for the DUT.

The calibration results when using DMM1 are shown in Tab. 4.13. It looks like DMM1 is better suited to be used in Internet-enabled calibration, but more tests are needed.

Resistance	DMM2		DMM1		DUT
Nominal (Ω)	Change (ppm)	Rel. change DUT spec. (%)	Change (ppm)	Rel. change DUT spec. (%)	90 days spec. (ppm)
3.29m	-6.58	-7	3.34	4	95
-3.29m	-13.61	-14	-1.89	-2	95
32.9m	-47.51	-54	-0.18	0	88
-32.9m	-53.20	-60	0.49	1	88
1.09	-12.02	-6	-3.82	-2	200
-1.09	-13.95	-7	-1.75	-1	200

Tab. 4.12: Change in resistance for DMM2 and DMM1 after transport.

Range (Ω)	Nominal resistance (Ω)	Measured value (Ω)	Total uncertainty k = 2 (Ω)
3.29m	3.29m	3.289952m	66n
3.29m	-3.29m	-3.289932m	66n
32.9m	32.9m	32.89993m	0.66μ
32.9m	-32.9m	-32.89971m	0.66μ
1	1	0.999980	25μ
1	-1	-1.000027	25μ

Tab. 4.13: Resistance calibration results for the DUT with extended uncertainty.

4.8 Conclusions

The iMet system v.2.0 was developed with the comparative study, described in chapter 3, in mind. The new system differs from version 1.0, in that custom measurement procedures may be added to the system at runtime. The procedures are stored in a database as source code, and can be downloaded and compiled when needed. The compilation happens in memory, without the need for manual interaction. This makes the system very flexible and scalable.

The system also allows the operators to run measurement procedures locally, such that only individual instrument commands are sent across the Internet to the customers. This increases the operators' visibility of the measurement process, thus reducing the risk of errors. Tests showed that if the rate of commands or bundled commands sent to the instruments are moderate, the procedure is suited to be run locally at the operator's computer. If the rate is moderate to high, the procedure should be run at the instrument computer.

Analysis showed that the network time delay will increase the total time of a typical measurement by 10-30%, when run locally at an operator's computer. Several methods have been investigated to neutralize the effects of the network time delay. When doing direct instrument operation, network time delay will always have a certain effect.

A real calibration was performed to test the system. Two multimeters were sent to a customer to calibrate an electrical calibrator. Two instruments were used to test which were the better suited as transfer standard. Both multimeters were calibrated before and after, and the results indicate that the multimeters had changed too much during transport to be used in a high-precision calibration. Internet-enabled calibration requires detailed knowledge of the properties of the transfer standard used. If the properties of the transfer standard are known, it is possible, to some extent, to predict the behavior. This is important when calculating transport uncertainties.

The iMet system v.2.0 seems well suited for Internet-enabled metrology and for operating instruments remotely via the Internet. If using the system for Internet-enabled calibration, more work is needed to find suitable transfer standards.

Chapter 5

A generic instrumentation system

This chapter presents the joint efforts of NPL and JV to create a highly adaptable instrumentation system, which supports remote operation of potentially any device from anywhere using any measurement procedure. New middleware software was utilized to make the system more platform-independent. Information associated with measurement procedures and hardware was made available from a dedicated database server, accessible to all hosts.

5.1 Introduction

5.1.1 Background

In 2005, NPL and Justervesenet decided to engage in a joint effort to develop a generic instrumentation system, which would allow to dynamically add new measurement procedures and support for new hardware during runtime.

The operator should be able to communicate with potentially any device from anywhere, using either native commands or complex measurement procedures.

5.1.2 Motivation

A deficiency had been observed in some existing instrumentation systems. They only supported a limited number of hardware buses and hardware devices, and the cost of adding support for new hardware was relatively high.

It was also known that some instruments were delivered with manufacturer-specific control software, e.g. [100], [101], and [102], resulting in different control applications to operate different instruments. These instruments, and some others, were delivered with specific instrument drivers or application programming interfaces (API), with which the control application interfaced.

It was thus desirable to look into ways to dynamically add hardware and measurement procedures to a system during runtime. This would make it more scalable and it would reduce the number of control applications to one, by seamless integration of different manufacturer-specific instrument drivers into one custom-designed instrumentation system.

5.1.3 Approaches

Hardware scalability

Most PC-based instrument control applications are bound to specific hardware buses, like the GPIB and the RS232 bus, and they often only work with a limited number of physical devices. If these instrument control applications were to add support for more buses and devices, they would need to be rewritten, manually recompiled and redistributed (if used by many). This is a tedious, costly and error prone operation, and most instrumentation systems therefore remain limited to a few hardware buses and devices.

The VISA specification, described in 1.3.3, maintained by the IVI Foundation [103], solves part of this problem as it acts as a bridge between the control application and the hardware. It enables operation over several hardware buses without the need for changing the control application. VISA does not, however, support all hardware buses, and provides no high-level device drivers.

Sometimes the devices are delivered with their own application programming interface (API), which provides high-level device communication procedures. These procedures are mostly used by manufacturer-specific control software, but could also be utilized by custom control applications.

This chapter describes an instrumentation system, which obtains hardware communication information dynamically during runtime. It has the potential to communicate with all devices, either by using high-level device APIs, VISA, or native hardware drivers.

The information needed about the hardware is automatically added during runtime without the need for manual recompilation. The information is accessible from a dedicated database server. As all parties download this information at runtime, the consistency of the system is more easily maintained and there is no need to distribute the correct version to all users before activation. When downloaded by a user, the code is compiled in memory and instantiated, as described in 4.3.

The instrument control application only needs to add support for the currently connected devices, and therefore remains reasonably small during runtime.

Development is focused on code reuse, such that the same code may be used for several devices. Many devices perform the same actions, and there is often no need to write separate software wrappers for each one of them.

Support for new hardware may be developed and added after the system has been taken into use. The code should be tested extensively before making it available, and this may be done without closing down or restarting the system. Existing code could also be upgraded. There is no need to manually redistribute software to users, because the user applications remains static (the user applications only download, compile and run code that changes).

Measurement scalability

Similarly to the iMet system v.2.0, described in chapter 4, the presented system supports adding new measurement procedures during runtime, without system restart or recompilation. The measurement procedures are added to a database, and downloaded when needed.

The current system only supports running measurement procedures at the customer's computer. As seen in 4, it is also possible, and sometimes preferable, to run the procedures at the operator's computer. This would be fairly easy to implement later. The current system was developed to test the dynamic runtime-addition of hardware support.

Internet support

The ability to add support for new devices and measurement procedures during runtime may be important, especially when controlling instruments via Internet. More and more instrumentation systems also use Internet to transfer device control signals and measurement results. When the operator and the instruments are separated by Internet, it is convenient if the hardware support and measurement procedures could be added during runtime. Then the operator would not need to travel to the instruments and install them manually. The diversity of instruments is large among NMI customers, and it is not easy, nor desirable, to add support for all hardware interfaces or drivers beforehand.

As described in 2.3, specialized middleware software is needed to handle the network communication between clients. XmlBlaster [104] was used as middleware in this work. This technology will be presented in 5.3.

The ongoing joint project between NPL and JV aims to solve several issues with this approach.

To test the framework presented in this chapter, an Optical Time Domain Reflectometer (OTDR) [90] with an API was used.

5.2 System overview

NPL and JV propose a framework for a PC-based instrumentation system, which is able to load hardware-specific information during runtime, when needed. The solution utilizes some programming languages' ability to generate assembly-code from source-code during runtime, as explained in 4.3. Possible languages include Microsoft Visual C# [51] and Sun's Java [58]. Software source-code, describing different devices and measurements, are stored at a dedicated database server. When needed, these software proxies are downloaded, compiled, and instantiated during runtime.

5.2.1 System components

The system is split into four parts, as shown in Fig. 5.1.

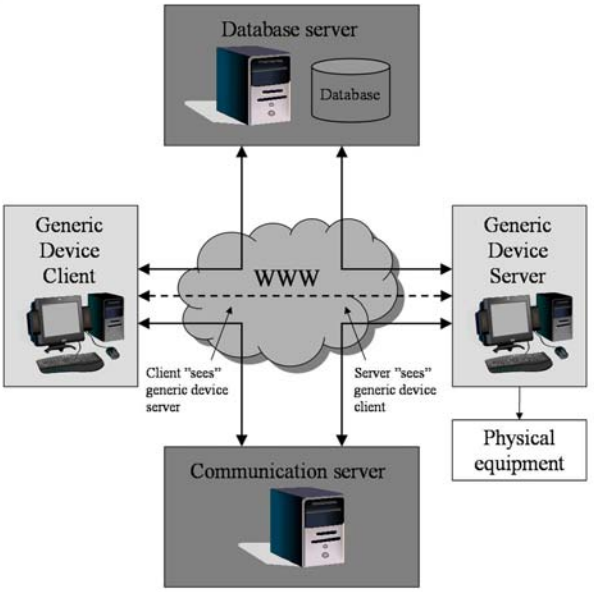

Fig. 5.1: Generic System Architecture. The generic client and the generic server communicate via a communication server, which relays the communication signals back and forth. Instrument and measurement software wrappers may be downloaded from the database server.

1. A generic device server (GDS) application, responsible for communicating with the devices and for providing services to control them.

2. A generic device client (GDC) application, acting as the device operator.
3. A database server, containing all device-related information and measurement procedures
4. A relay server, handling the GDS-GDC communication

The operator and the customer both run the same application, as it contains methods to instantiate both GDS and GDC objects. This is shown in Fig. 5.2. Only one GDS per computer is allowed, while there is no limitation to how many GDCs can be run on a single computer, since these often operate remote GDSs. When a GDC operates a GDS on the same computer, the communication happens between processes, as opposed to a GDC operating a remote GDS which uses TCP/IP communication.

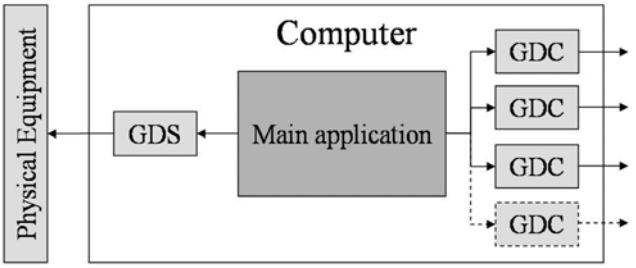

Fig. 5.2: The same application is run both by operators and customers. One instance of the GDS may be run on each computer, while there is no limitation to the number of GDCs.

The Generic Device Server

The GDS is shown in Fig. 5.3. It provides certain services to one or more GDCs, and consists of a service manager, which handles all client connections, and a control manager, which handles the device communication. The control manager may communicate with a device directly via a specific device object, or indirectly via measurement objects. A measurement object contains methods to perform a measurement involving one or more instruments connected to the server.

The GDS has no prior knowledge of the devices connected to the computer on which it is running. When setting up communication to a device it first needs to download this information from the database, in the form of source code, and compile and instantiate it at runtime. The remote database is effectively a part of the instrument control application, and enables a dynamic system that can adapt to changing environments.

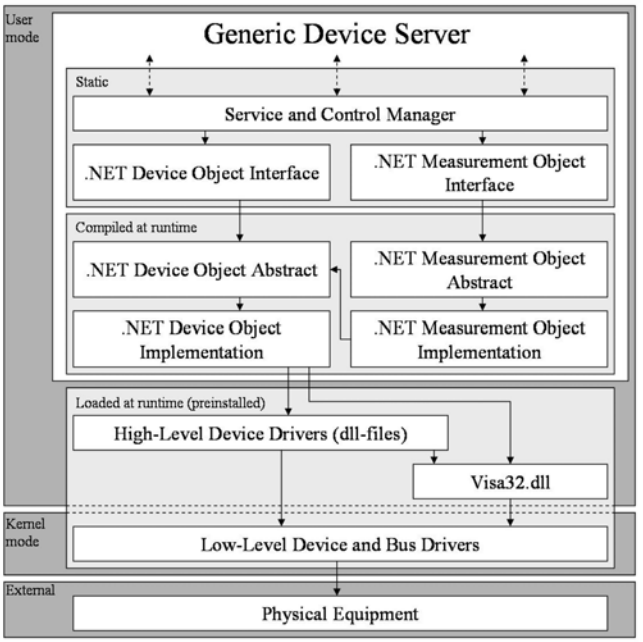

Fig. 5.3: Generic Device Server Architecture. The server communicates with the devices either directly or via a measurement object. All communication is interface based, as used in OO languages, which means that the server contains no prior implementation of the measurement and device objects.

The GDS implements a GDS interface containing several methods. The most important methods include

- **activateInstrument(...)**
  Activates and registers a specific instrument. This includes downloading hardware information, compiling the source code and providing a service to communicate with the instrument.

- **deActivateInstrument(...)**
  Deactivates and unregisters a specific instrument. This means closing the instrument connection and removing the service to communicate with the instrument. The compiled code is not removed, so that the instrument could be reloaded at a later stage without any recompilation.

- **runMeasurement(...)**
  Compiles and runs a specific measurement procedure. The procedure source

code is received from the calling operator (with measurement configuration data and measurement points).

The Generic Device Client

The GDC is shown in Fig. 5.4. It is used to operate devices connected to a GDS. The GDC may use inter-process communication, when run on the same machine as the GDS, or TCP/IP communication, when run on a different LAN computer. The GDC may control remote devices directly by using runtime-compiled device-specific graphical user interface (GUI) components, or indirectly using runtime-compiled measurement objects.

The *UML use case diagrams* for the GDC when running measurements and operating instruments directly are shown in Fig. 5.5 and 5.6. A UML use case diagram presents the primary elements and processes of a system. The elements are called actors, while the processes are called use cases. UML use case diagrams can be used to describe specific functionality of a system.

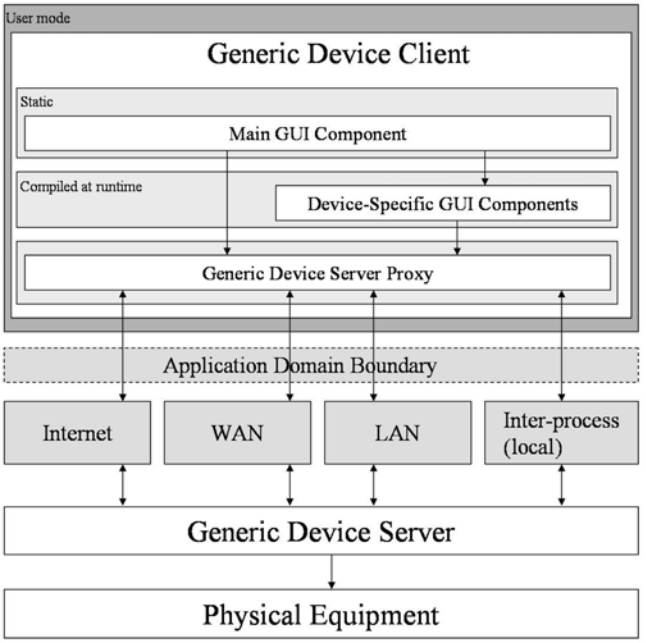

Fig. 5.4: Generic Device Client Architecture. The client may communicate with a generic server using several channels (Internet, WAN, LAN or locally). It may generate device-specific GUI controls to control a remote/local device directly.

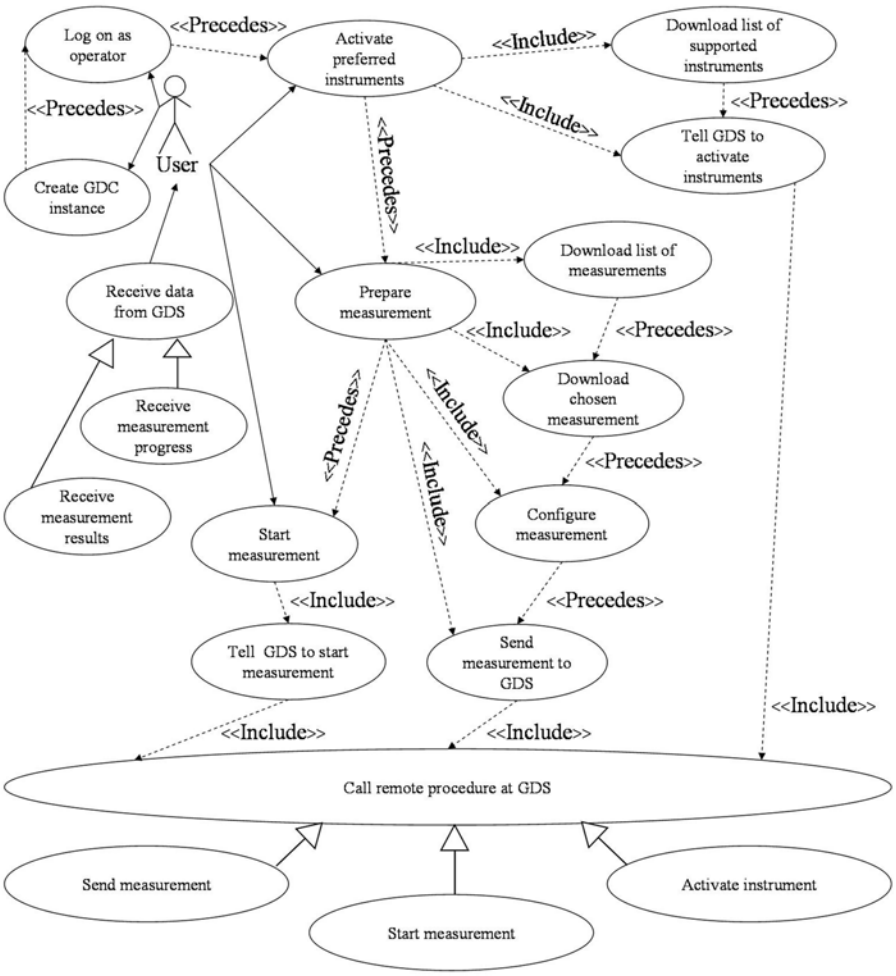

Fig. 5.5: The use case diagram shows the required actions for the GDC when running measurements.

5.2.2 Choice of technology

Microsoft Visual C# was used to program most of the system. Java could also have been used, but then an extra software layer would need to be added, as will be explained in 5.6. To run C# applications, .NET Framework needs to be installed on the computer, as was explained in 2.3.3.

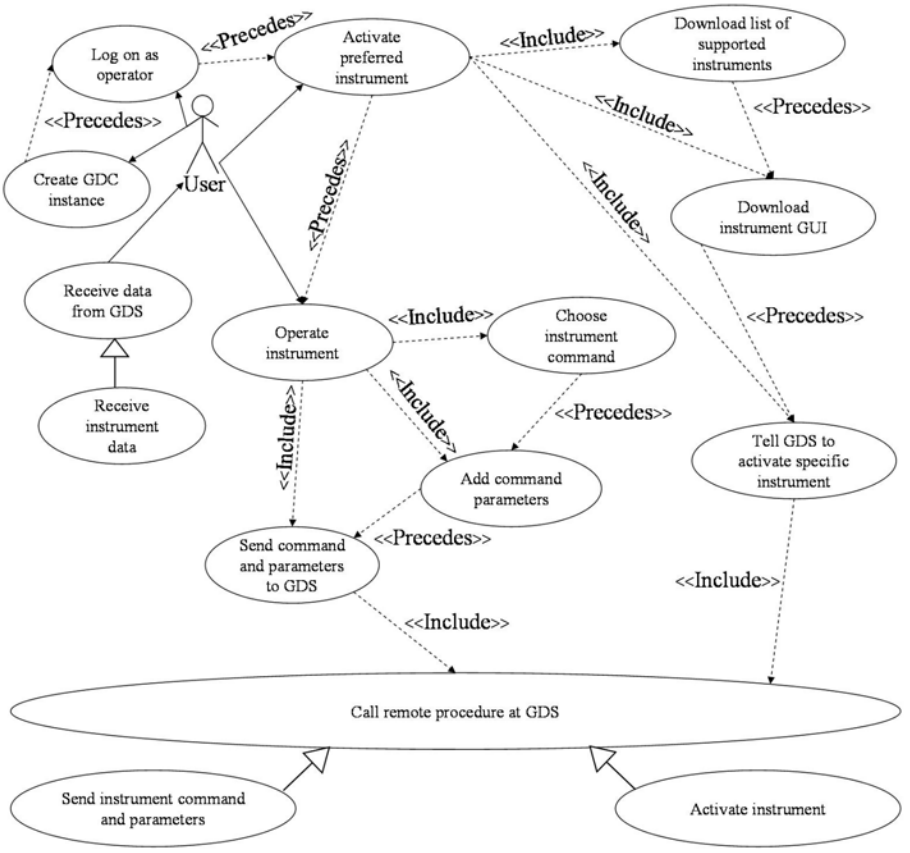

Fig. 5.6: The use case diagram shows the required actions for the GDC when operating instruments directly.

Although the operator and the devices may be co-located, the focus of the work presented has been on separating the operator and the devices using the Internet as a communication channel, as shown in Fig. 5.1.

As seen in 2.3, the GDS and the GDC generally cannot communicate directly when separated by the Internet, due to network protection components like firewalls and proxy servers. In the current work, XmlBlaster [104] was used to obtain bidirectional communication (though only tested for bidirectional TCP communication). The XmlBlaster project is still under development, and further testing is needed. The reason for choosing XmlBlaster was that this technology enables different programming languages to communicate (not just C#). If

implemented in Java, both the GDC and GDS could be run on any platform implementing the Java Runtime Environment, though an extra software layer would be needed, as will be explained in 5.6.

The system could also use .NET Remoting, as described in chapter 2. This would require few architectural changes to the existing system, as XmlBlaster and .NET Remoting could be set up to run in much the same way.

5.3 XmlBlaster

XmlBlaster is a message-oriented middleware (MOM), which exchanges messages between communicating computers, e.g. a client and a server. A message is described with XML-encoded meta information, and it may contain several objects, e.g. GIF images, Java objects, Python scripts, XML data, a word document, or plain text.

An XmlBlaster message consists of three components

1. **Header**
   XML-descriptions of the message content
2. **Content**
   The actual binary content, e.g. a picture, or a data object
3. **Quality of Service**
   Information regarding how XmlBlaster should handle the message, e.g. addressing information, and message expiration date

The XmlBlaster server is developed in Java, while the connecting clients may be developed in a variety of software languages, including PHP, Perl, Python, C, C++, C#, Visual Basic.net, Flash, J2ME, and Java.

The system supports several communication protocols, like direct socket, CORBA (using JacORB [105]), Java RMI, XmlRpc, HTTP, or email.

5.4 System operation

There are two ways of using the system. Either the GDC may operate local/remote devices directly using device-specific control GUIs or it may perform local/remote measurements.

5.4.1 Direct instrument operation

In direct instrument operation, the GDC downloads a device-specific GUI component from the database server. This GUI component is then compiled

in runtime and instantiated. The GUI component contains controls to operate the device directly, e.g. configuring a digital multimeter. The control signals are serialized into byte streams and sent to the remote GDS via the communication server. The GDS deserializes the byte stream, and sends the method call to the correct device software wrapper. This software wrapper is responsible for converting the method call into a viable device driver call. Measurement results are returned from the generic server to the client in much the same way.

Thus, for each device there are two software components, namely a GDC-side GUI control component and a GDS-side high-level device driver component. The operator does not need to know the details of the latter component, because it is handled by the GUI component (which is compiled and instantiated during runtime). When adding support for direct operation of a new device, both components need to be added to the database. If only used in measurements, the GUI component could be omitted or added at a later stage.

5.4.2 Indirect instrument operation

When using the system for remote measurements, the GDC does not need to know anything about the functionality of the devices involved. Instead it downloads the correct measurement description object from the database server, which contains methods to configure the measurement and add measurement points, in addition to the actual source code of the measurement procedure. After configuring the measurement, the GDC serializes it and sends it to the generic server. When received, the generic server deserializes the measurement byte stream, compiles the measurement source code, configures it, and runs it with the correct measurement points. If told to do so, the generic server notifies the operating client of the measurement progress during the measurement. When finished, the measurement results are sent to the client for further processing.

Measurements work on device interfaces, such that the same measurements could be used with different devices, if implementing the same interface. E.g. multimeters able to measure DC voltage, could both implement a method "measureDCV(double voltage)", which would allow different multimeters to be used in the same DC voltage measurement.

5.4.3 Preparations

When remotely operating devices connected to a computer, there are some actions that need to be done beforehand on that computer:

- The native hardware bus driver needs to be installed on the device computer
- If the device came with an installation CD containing a device driver and API, they need to be installed on the device computer

- Sometimes a little manual work needs to be done by a local human operator (simple shell commands) on the device computer

- A local human operator needs to start the GDS application, and connect it to the public communication server (when operating remotely)

5.5 Software drivers

Software drivers, or device drivers, are small extensions to an operating system (OS), which typically enable the OS to communicate with external, physical devices. These devices comprise both hardware buses, like GPIB, RS232 and USB, and specific devices connected to the buses, like digital cameras and printers. The drivers are often organized in a stack, such that a USB bus device driver makes use of the USB bus driver.

Drivers that communicate with hardware, need to run with privileges extending those for the traditional user software. Normal software, developed by regular users, usually run with *user mode* privilege, which among others prevents it from accessing memory directly. Hardware drivers, on the other hand, need to access memory directly, and therefore are usually set to run with *kernel mode* privilege. The user - kernel mode intercommunication is quite complicated, and most often the hardware drivers are delivered with a user mode interface. Normal software may access a driver through this interface, without knowledge of the underlying interrupt-based driver communication. The overview of the communication process is shown in Fig. 5.7.

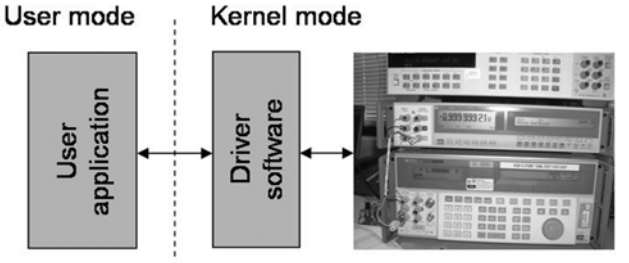

Fig. 5.7: The relation between user mode software and kernel mode software. The user mode application needs to communicate via a kernel mode driver to communicate with external hardware.

5.6 Platform independency

The system presented is dependent on the .NET Framework, which, traditionally, means using the Microsoft platform. The system could also be implemented using Java, which would make the system more available on different platforms. By implementing the GDC and GDS in Java, and using XmlBlaster as middleware, the system could be ported to any platform supporting the JRE.

The platform-independent nature of Java, makes hardware communication somewhat more challenging compared to using C# on the Microsoft platform. The standard Java class library does not support the platform-dependent features, like hardware access, and native driver communication, needed by the GDS. When using Java to access hardware, an extra software layer needs to be added between the Java application and the hardware driver, to handle the driver communication. This layer is usually programmed using C/C++, and it needs to be installed on the computer before running the Java application. The hardware communication architecture on the Microsoft platform using Java and C# is shown in Fig. 5.8.

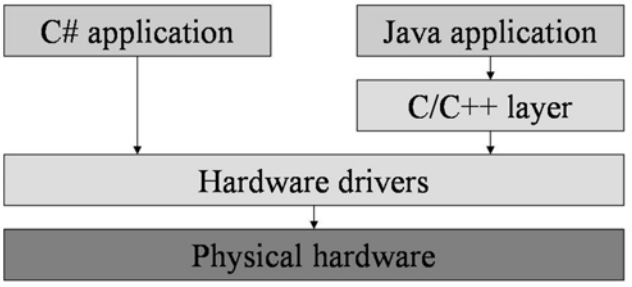

Fig. 5.8: Hardware communication using C# and Java on the Microsoft platform. As can be seen, it is easier to access hardware using C#, due to the extra software layer needed when using Java.

When a Java application must communicate with a platform-dependent component (like a native driver), the Java Native Interface (JNI [106]) is used to write *native methods*. These native methods handle the driver communication. Due to porting problems, this extra software layer would sometimes need to be compiled manually on the computer where it should be running. It is also often quite complicated to write (due to e.g. marshaling of custom types). Due to this extra manual work, and because most PC-based instrumentation is done on the Microsoft platform, C# was chosen as the main programming language for this implementation.

5.7 Results

A system test was performed in January 2005 using an OTDR. An OTDR is an optoelectronic instrument specifically designed for optical fibre testing. To test an optical fibre, a series of optical pulses is sent into one end and the reflected pulses are measured and analyzed. From the reflected pulses several attributes may be extracted, such as fibre attenuation, dispersion, non-uniformity, splice loss, fractures, and length. The instrument setup is shown in Fig. 5.9.

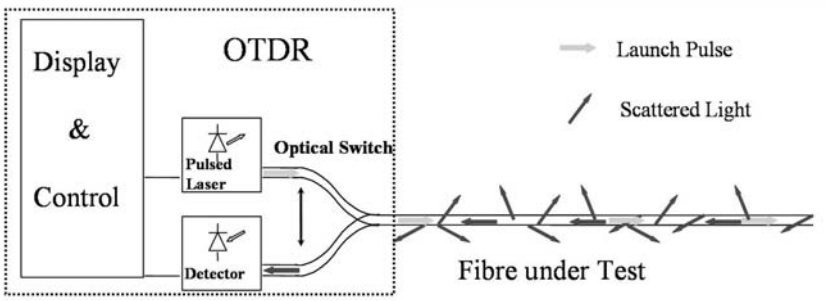

Fig. 5.9: OTDR Architecture. The fibre under test is connected to the OTDR, and a series of optical pulses is injected with a laser. The reflected pulses are measured with an optical detector.

In the experiment, an OTDR located at NPL in England was remotely operated from JV in Norway. Both computers were behind firewalls, and only outbound connections were allowed. Beforehand, a public web server was set up at NPL, and all communication signals between the operator and the OTDR were relayed through this.

The operator first queried the database server for a high-level OTDR driver, based on manufacturer name and model, which was then downloaded by the NPL computer, and instantly compiled and loaded. When loaded, the driver tested the communication with the physical OTDR. After this test was successfully completed, the operator queried the database server for all experiments involving a single OTDR, again based on manufacturer name and model. After finding the correct measurement, he downloaded a custom measurement object. Such measurement objects contain measures to configure the contained experiment and add measurement points. In the experiment described, the operator configured the measurement such that the pulse series lasted for 3 seconds, and each pulse consisted of laser light with a wavelength of 1310 nm.

The measurement results, in the form of reflected and acquired data, are shown in Fig. 5.10.

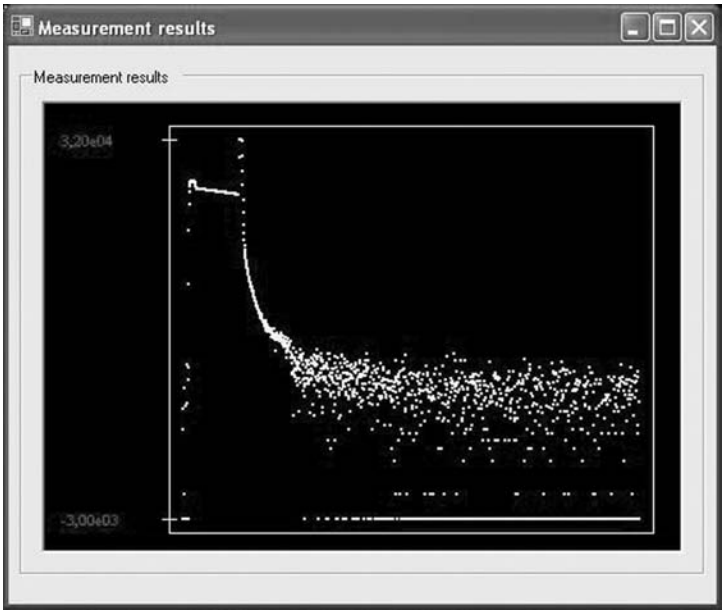

Fig. 5.10: Results from the data acquisition. The OTDR used laser pulses with a wavelength of 1310 nanometers. The length of the fibre was 1 kilometer.

Most of this data is noise, and only the far left-hand side of the graph is interesting. A cropped and magnified version of the upper left corner of Fig. 5.10 is shown in Fig. 5.11.

The focus of the experiment was to test the dynamic driver loading, and the measurement data is thus not interesting in this perspective. In a real OTDR measurement, several attributes may be drawn from the data (e.g. by looking at the gradient angle of the line in Fig. 5.11).

5.8 Conclusions

A dynamic instrumentation system has been developed, which may potentially operate any PC-connected device.

The system is designed such that all hardware-specific information is downloaded from a database and compiled during runtime. This makes the system very scalable, and hardware information may be changed or added to the central database even when the system is running.

The devices can be operated locally or remotely, depending on the location of

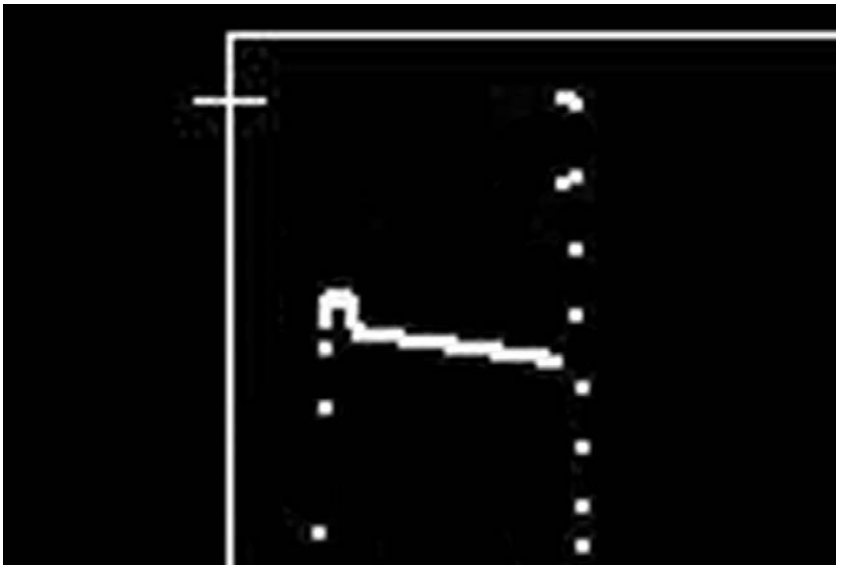

Fig. 5.11: Detail from the original results graph in Fig. 5.10

the operator. This means that third-party experts may be involved in the control process without the need for them to travel to the instruments. This requires the use of specialized middleware which handles the communication between the operator and the instruments, e.g. .NET Remoting or XmlBlaster.

The system also promotes consistent results as all measurements, controlled by different users, are done using the exact same measurement code downloaded at runtime. New measurements can be developed and tested, and then instantly be made available to all users of the system.

As only one system is needed to control all devices, it is very versatile and cost-effective. Training is only needed for new users, though in-depth knowledge of the physical devices is still needed. The users can focus on operating the devices instead of learning the application.

The security of the system depends on which middleware is used. Internet-traffic should be encrypted when sending sensitive data, and the use of MACs and digital signing of source code and data results is necessary to obtain data integrity. All users and computers should also be authenticated to reduce the risk of misuse.

As more and more measurements are done using a computer, the presented system could provide an cost-effective and scalable solution. This could result in large savings for companies that use many different instruments.

Chapter 6

Discussion and conclusions

In this chapter a thorough discussion of important areas regarding integrating the Internet into the metrology area will be given, after which will follow some recommendations for future work and some thoughts for the way forward. At last a short summary of the most important conclusions from the previous chapters is presented.

6.1 Introduction to discussion

Since the first IT-tools became available, the metrology community has utilized them in their everyday work, from writing reports to running experiments in their laboratories.

The interest to integrate the Internet in the metrology area is growing. Many NMIs across the world have done work in this field, but few or none have taken a broader approach of seamless integration of the Internet into their work.

The growing interest in Internet, has come as a result of the progress and development of ICT technologies, enabling secure and seamless communication over long distances at reduced costs. The ubiquity of Internet, combined with PC-based instrumentation, opens up new possibilities in the metrology area.

In this context, important questions arise regarding Internet-enabled instrumentation systems

- Can the Internet have a role in real calibrations?
- What is required to accredit Internet-enabled calibration systems?
- Are there suitable transfer standards that may be used in Internet-enabled calibrations?

- Can the operator and the instruments be separated in a secure and practical way?
- Which constraints do security requirements put on the functionality of an Internet-enabled instrumentation system?
- How can Internet-enabled instrumentation systems be designed to minimize the possibilities for errors?
- How does the architecture of an Internet-enabled instrumentation system influence the network dependability?
- What are the challenges for the instrument manufacturers in the future?
- How can Internet-enabled instrumentation systems be developed both with the present and the future in mind?
- To what degree will the Internet influence metrology in the long run?

Finding good answers to these questions will influence the future development of Internet-enabled instrumentation systems. Internet-enabled instrumentation is a relatively new field, and collaboration should be sought to develop standards to be used by NMIs, calibration laboratories and instrument manufacturers. Well-designed standard solutions would simplify the transition from ICT-supported instrumentation to ICT-based instrumentation.

6.2 Discussion

6.2.1 Can the Internet have a role in real calibrations?

Judging from all the work done up to now, both internationally and here, Internet seems to have the potential of becoming an important part in the future calibration and metrology area. Some of the challenges to be dealt with are security, availability, device authentication and dissemination of traceable calibration data. These challenges solved, real calibrations could be performed using the Internet as transport medium for control signals, calibration procedures and measurement results. It would also enable an instrument to be made available, as a service, to trusted parties across the world in a safe way. Instead of moving the physical instruments to the operators, the instrument buses would be virtually extended to the operators, as seen in Fig. 6.1. This would be important when dealing with direct instrument operation or when running more complex calibration procedures at the operator.

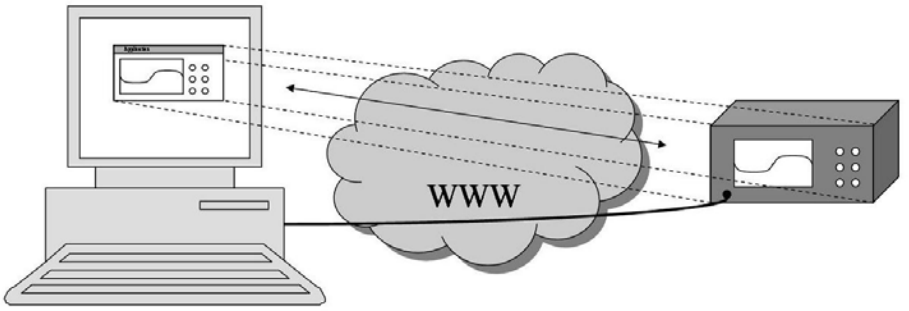

Fig. 6.1: Using ICT tools, an instrument bus may be extended across the Internet to a trusted party.

The results from the preceding chapters indicate that the software challenges with Internet-enabled instrumentation systems, can be dealt with. Authentication of devices, on the other hand, seems more tricky, as was discussed in 2.5. The availability of suitable transfer standards also presents a challenge, when working at an NMI-level.

Performing calibrations via Internet, traceable measurement values need to be disseminated to the DUT. The dissemination proceeds through the transfer of primary standards or more complex electrical instruments. When testing the iMet system v.2.0, the results (presented in 4.7) showed that the transfer standards in use had changed significantly during transport. The calibration process as such worked as planned. This suggests that the challenges might not lie in the software but in the hardware. To perform Internet-enabled calibrations at the level of an NMI, the biggest challenges could be to find and maintain a suitable transfer standard, in addition to sufficiently authenticating the devices in use.

In traditional calibrations, where the DUT is transported, the transport uncertainty is difficult to calculate. When "reversing" the calibration process, where the calibration values are disseminated to the DUT and the transfer standard is calibrated before and after transport, the transport uncertainty can be calculated more reliably. This means that, if suitable transfer standards are found, more precise calibrations than those available today may be performed.

Failing to find a suitable transfer standard would affect the level of calibration precision possible. The uncertainty introduced by the transfer of calibration values would influence the total calibration uncertainty. Internet-enabled calibration would thus only be suitable for low and medium-precision calibrations.

6.2.2 What is required to accredit Internet-enabled calibration systems

To answer this question, it is important to look at how such systems differ from traditional calibration systems (described in 1.4). Four important factors must be dealt with:

- **Network communication**
  Which security challenges are introduced by the Internet?
- **Device authentication**
  Can the devices involved be authenticated in a satisfactory manner?
- **Calibration data dissemination**
  Can a suitable transfer standard be found?
- **Customer trust and education**
  Can the NMI customer be trusted to handle the instruments and in performing small software tasks? Can the NMI customer be trusted not to intentionally misuse the system?

The technical challenges can be overcome. Technically, a system can be made as secure as needed. Using cryptographic techniques, control signals and measurement results can be secured during transport and users can be authenticated. Regarding availability, systems may be developed using firewall-friendly protocols, making them highly accessible to most user. Sometimes there is a choice between sufficient security and needed functionality, e.g. the need for speed. This may be solved through increased bandwidth and faster processing units.

Utilizing the Internet, a system can be based on at least three different architectures, as described in 1.6. Either the operator and the instruments may be co-located, or they may be separated by Internet, where the instruments are connected either to a public server or to a regular LAN computer.

Architecture 1 is used to obtain consistency, in that all users run the same downloaded code at runtime and exchange calibration data and measurement results with a shared dedicated server. If the operator is from an accredited calibration laboratory or an NMI and involved in every step of the process, most challenges concerned with customer trust and device authentication become irrelevant. NPL's iPIMMS system utilizes the Internet to obtain traceability for high precision measurements using microwave network analyzers (ANA). This accredited system is of architecture 1.

Architecture 2 can be used to set up online laboratories, where a public server gives access to a group of instruments and measurements. This architecture is

not suitable for Internet-enabled calibration, as described here, because it would require each customer to either set up their own dedicated web server or to open certain ports in their firewall. From an educational perspective, it is a well-suited architecture for letting users operate instruments and run measurements for practice. This could be a part of the user education needed when accrediting NMI customers. UniK's Lab-on-Web is a system used by students to run real and simulated experiments. It belongs to architecture 2.

Architecture 3 provides means to build general instrumentation systems, which enable remote operation of traditionally unavailable devices. This architecture eliminates the need for the operator to travel to the devices, and trusted experts can easily get involved directly in the measurement process. The systems described in chapter 2, 4 and 5 belong to architecture 3.

Whichever architecture is used, introducing human customers and operators opens up possibilities for breaches in the system security and reliability. Intentionally or unintentionally, the customer may perform actions which could make the system behave in a non-predictable way. Unintentional misuse should be prevented by sufficient testing before launching the systems. Intentional misuse could be much more difficult to prevent. To avoid this, a trusted relationship should be developed over time. Users of an accredited system could be provided with a special certificate, proving that they are trusted and accredited users. This could be combined with a contractual liability of misuse.

6.2.3 Are there suitable transfer standards that may be used in Internet-enabled calibrations?

Suitable transfer standards could be any high-precision device with well-known transport behaviors. The instruments tested here, two high-precision digital multimeters, showed considerable changes after transport. These changes could not be directly related to transport, because the drift behavior was not well-enough known.

Therefore, sufficient knowledge of the primary standard or complex instrument is needed before using them as transfer standards. Historical data is needed to be able to model the drift of the calibration values. When such models are available, with uncertainties, it is easier to identify the influence of transport.

To reduce the influence of transport, custom containers should be provided to ensure stability in temperature and humidity and to protect against impacts (during transport). Additionally, the containers should be equipped with sensors monitoring temperature, humidity and shock.

The Fluke 4950 Multifunction Transfer Standards System, previously available, was such a system. The Fluke 4950 was a digital multimeter, used in high-precision calibrations, and claimed to be suitable as a transfer standard. It

was used with a special transit case, including sensors for temperature, humidity and shock monitors. The Fluke 4950 was not tested in this work, because it was no longer available.

More work is needed to identify suitable transfer standards. It would be convenient if high-precision multifunction instruments could be used. Then several calibration values could be disseminated using one instrument, allowing calibrations to be performed effectively.

A potential candidate might be the Josephson Array Voltage Standard (JAVS), which could be used in combination with GPS. A JAVS can convert microwave radiation into DC voltage, and is often used to realize high-accuracy voltage values (the uncertainty is only dependent on the uncertainty of the microwave signal frequency). If each customer has their own JAVS, traceability could be obtained without using a transfer standard. A JAVS at the customer and a JAVS at the NMI could be used to measure the GPS signal simultaneously, and the signals would then be compared. This would effectively calibrate the customer's JAVS since the same GPS signal was used. The customer's JAVS could then be used to calibrate other devices at the customer. Another approach is to develop a transfer standard based on a JAVS, which would be less influenced by transport.

Programmable JAVS are normally used to generate DC voltage, but they can also be used to synthesize AC voltage signals [107], [108], and AC power [109]. Other candidates could be standards, where the measurement values are rooted in natural phenomena like Quantum Hall effects.

Time and frequency are two properties which can be calibrated directly without using a transfer standard. If these properties could be used to generate other physical properties, based on some physical phenomena, Internet-enabled calibration could be done without using transfer standards.

6.2.4 Can the operator and the instruments be separated in a secure and practical way?

There are several reasons why separating the operator and the instrument sometimes may be preferable. First, the operator saves time and costs, which is important in an increasingly competitive industry. Second, it enables the involvement of trusted experts in the measurement process, even if they are far away. Third, sometimes the conditions under which the instruments are working might not be suitable for the operator.

This work has shown that such separation could be obtained using specialized middleware software. One possibility is to use .NET Remoting, used in both iMet systems described in chapter 2 and 4. As seen in chapter 5, XmlBlaster was also tested, though with less focus on accessibility and security.

Due to security mechanisms like firewalls, proxy servers and NATs, a general approach is to let the operator and the remote instruments communicate via a public web server, an architecture supported both by .NET Remoting and XmlBlaster.

To further increase a system's scalability, it should be possible to add new measurement procedures and support for new hardware during runtime. This was tested successfully using the generic instrumentation system described in chapter 5.

6.2.5 Which constraints do security requirements put on the functionality of an Internet-enabled instrumentation system?

Normally, the more complex a system becomes, the more vulnerable it gets. When including the Internet into an instrumentation system, every Internet-connected PC becomes an entry gate for potentially misusing the system.

It is useful to list the most important effects of different security requirements

- Including the Internet into the instrumentation system increases the risks of unwanted intrusion.
- Separating the operator and the instruments increases the risks of identity forgery (either by the instrument owner or the operator).
- Often there is a trade-off between security requirements and the need for speed and availability.
- Allowing NMI customers to be more involved in the calibration process requires satisfactory training beforehand.

The technical aspects, concerning software, can in most cases be dealt with. User and device authentication, on the other hand, must to a large degree be built on trust. That is, it is possible to build a secure system, which allows only trusted customers access. However, it is impossible to prevent falsely trusted customers access. Logging mechanisms should be utilized to monitor user actions, to discover misuse and initiate counteractions.

Providing users with sufficient training before using the system, contributes to lowering the error rate associated with user actions. It would be convenient if the system security policy coincides with the company security policy of the customer.

When performing measurements where extensive amounts of data are communicated or precision timing requirements are needed, tests showed that the procedure must be run on the instrument computer, thus lowering the system's

feedback to the operator (if using architecture 2 or 3). There are few ways around this, other than increasing the available bandwidth and upgrading hardware.

6.2.6 How can Internet-enabled instrumentation systems be designed to minimize the possibilities for errors?

As mentioned, it is important to give users sufficient training before using the system. This would reduce error rates concerned with normal use. Well-trained people make fewer errors.

Through well-designed systems, using established object-oriented techniques, the error rate may also be lowered. Users should only have access to relevant information, and should only be allowed to perform actions according to their trust and authorization level. The fewer choices a user has, the less are the chances of making mistakes.

As described in chapter 5, it is possible to create single systems that can handle potentially all hardware and measurement procedures. By unifying and standardizing the design of GUI controls, used to operate different devices, users may grow accustomed to using the system, even when operating unfamiliar devices. This would reduce the error rate significantly.

It would be beneficial to integrate a web camera to monitor the remote location. This would be sent out with the transfer standard. It could be treated just as a normal device, where a software proxy gets downloaded and compiled when needed.

Easy-to-understand setup pictures should be provided to make the task of connecting the instruments easier and less error prone. Additionally, fixed plug connections could be provided for standardization.

Generally, user interactions should be held at a minimum, thus automatizing most of the process.

6.2.7 How does the architecture of an Internet-enabled instrumentation system influence the network dependability?

A system's network dependability is directly correlated to the rate of network communication. That means, a system which needs to access the network frequently is more dependent on the network than a system which accesses the network less often.

Of the three architectures presented in 1.6, the network dependability increases from architecture 1 to architecture 3. Architecture 1 only needs to access the Internet while downloading measurement procedures and uploading measurement results. Architecture 3 could require Internet-communication for

each instrument command in a measurement procedure if run on the operator's computer.

To reduce the dependence on the Internet, measurement procedures should run at the instrument computer. Generally, this would lower the operator's insight into the measurement process if not using architecture 1. The results could be handled in a store-and-forward operation, where the results are stored locally before being forwarded to the remote database. This would prevent loss of data even if the network connection is broken.

None of the systems presented here would work without access to the Internet. This is acceptable considering architecture 3. Using architecture 1, co-locating the operator and the instruments, it might be possible to develop cache functionality, such that measurement procedures and virtual device software are stored locally. When the Internet is available, the needed software would be downloaded (and stored for future use) to ensure consistency, while the locally stored versions could be used during network downtime.

E.g. when adding a device to the instrument computer

1. If an Internet-connection is available
   (a) The instrument computer first checks the remote database for a device proxy
   (b) If found, the proxy is downloaded and added to the local cache
2. If an Internet-connection is unavailable
   (a) The instrument computer checks the local cache for a device proxy

If there is no connection to the Internet, the above will work as long as a version exist in the local cache. The consistency of the procedure can not be guarantied, but at least the device operation will work.

6.2.8 What are the challenges for the instrument manufacturers in the future?

More and more new devices come with built-in Ethernet cards, and may be accessed directly from a LAN or Internet. To prevent unwanted users to access, some security measures have to be established.

The devices could be equipped with features to help with the authorization of users, either by using certificates and/or usernames and passwords. The devices could also utilize *user groups*, and associate each group with certain functionality. E.g. the group "Administrator" could have access to all functionality, while "Basic user" could have access to simple configuration and reading functionality.

When integrating the Internet into the system, firewalls should also play a part in the authorization process.

As already mentioned in 2.5, it may be a challenge to securely authenticate devices. The devices range from some not having computer-interfaces, to others being equipped with computer-interfaces and built-in identification strings. These identification strings are sometimes not "permanent" and can be altered. New devices should have permanent and unique identification tags, which may be accessed by a computer. Possibly, all devices could be equipped with unique MAC-addresses (through their Ethernet cards).

Traditionally, computer-based instrumentation has been a local process, where a computer communicates with a device directly over a local bus. There is now a shift from computer-based toward network-based instrumentation. This might suggest the need for the development of a new instrumentation protocol, for example based on SCPI, which takes into account network issues like quality of service, addressing, and security. If manufacturers could agree on a standardized instrumentation protocol, thus obtaining interoperability, it would be very advantageous. This new protocol could use for example SOAP (XML over HTTP), such that it would work with firewalls and proxy servers.

6.2.9 How can Internet-enabled instrumentation systems be developed both with the present and the future in mind?

Developing systems for the future, flexibility and modularity in components are important properties. That is, when new standards or technologies are introduced, new modules may be added or old modules replaced or removed *without* affecting the rest of the system. If a new communication protocol is introduced, it should be possible to make the system work with this protocol without affecting the instrumentation system as such. It should be possible to add, replace or remove measurement procedures or support for hardware, without rewriting or recompiling the rest of the system. Methods to make a system scalable has been discussed in chapter 5.

In the future, an instrumentation system should be able to provide remote control services to external users, e.g. via the Internet. Moreover, the system should ideally be configurable to fit every user's needs. Many users requires many different configuration options. *Standardization* is then again important. If the metrology community could agree upon certain ways of integrating the Internet into their work, *interoperability* and *cooperation* would be much less challenging. In the future, there might be a shift toward cooperation over the Internet between NMIs, especially within the European Union (EU). Though somewhat reluctant to make use of the Internet in their everyday instrumentation work today, as seen

in appendix A, the metrologists might change their minds in the future.

6.2.10 To what degree will the Internet influence metrology in the long run?

In many areas, e.g. business and education, Internet is becoming increasingly important. Business-to-business transactions are being done online, and students may attend online classes. The metrology community will most probably also be influenced more and more.

In the future it could be much more common to run or monitor measurements via the Internet. Initially, companies might utilize Internet to run or monitor their own measurements or operate their own instruments. If or when standards become available, it would be possible to cooperate and coordinate actions more easily between NMIs.

Instruments could be utilized more efficiently by more people, because they may be operated from remote. Sharing instrument could be much more common.

When in need of performing a special measurement, involving both local and remote instruments, an analysis must be performed to check which instruments have to be shipped where. Instruments with well-known transport properties should be sent instead of instruments with less well-known transport properties. This must be weighted against the costs of the instruments, company policy, and the skill of the people handling the instruments. It also requires all involved parties to agree upon the terms of such inter-laboratory cooperation.

Public databases could be used to obtain consistent measurements. If several NMIs run the same software, standardization may be implemented more easily. This would also reduce costs, in that individual NMIs do not need to write their own software. Well-tested software could instead be shared among many laboratories.

IPv6 is an abbreviation for "Internet Protocol Version 6", and is the next generation Internet protocol. It will succeed IPv4, which is most common today. An important difference between the two protocols is the size of their address space. While IPv4 may address only 4294967296 devices, IPv6 may address 3,8e+38 devices. This could lead to more devices being equipped with a network card and made available via the Internet. This may include lottery machines, gas pumps, mass balances and household electricity meters etc., all of which need to be monitored, calibrated, or checked on a regular basis. Due to the inconvenience of transporting many of these devices, the experiences from the work on Internet-enabled metrology may be utilized.

There is a trend towards a convergence of different everyday applications into fewer and fewer devices. A good example would be new cell phones with integrated multimedia players and digital cameras. They connect to other devices

via several communication channels, and soon they will likely include GPS receivers. Becoming more connected, these multifunction devices will perhaps be the device-of-choice when using Internet in the future. In a few years measurements could be controlled and run using regular cell phones.

6.3 Recommendations

Some recommendations may be proposed, based on the experiences from this work. Several areas have been discussed and some conclusions have been drawn.

6.3.1 Architecture

It is believed that architecture 3, introduced in 1.6.3, will have the best potential in the future. It's advantages clearly surpasses it's weaknesses. In a way, the other architectures are sub-versions of this.

It might be a good idea to let a system be configurable to work with any of the architectures. Developing with architecture 3 in mind, it would be manageable to let users choose between any of the three architectures. That is, when operating local instruments, the operator should use architecture 1. If operating remote instruments, the operator should use architectures 2 or 3, depending on the accessibility of the instruments.

6.3.2 Communication

The communication used in an instrumentation system is dependent on which architecture is used and the accessibility-level of the operator and instruments. If the operator and the instruments are separated, and the operator and/or the instruments are behind firewalls, proxy servers and/or NATs, flexible middleware software is required, to enable full-duplex communication channels through the network protection entities. Sometimes this communication channel needs to support bidirectional HTTP, when dedicated HTTP proxy servers are involved. .NET Remoting was the only middleware found and tested in this work, supporting bidirectional HTTP channels. .NET Remoting is practical to use and integrate into a system, but requires to be run within the .NET Framework.

When architecture 2 is used, the same middleware software as for architecture 3 might be needed. This is due to potential asynchronous operations, which requires the server to communicate directly with a client. It is somewhat less complicated than for architecture 3, in that only one channel to the server is needed.

When using architecture 1, no complex middleware software is needed. Often regular HTTP calls to the server are sufficient, containing measurement data or

requests for measurement procedures.

If implementing all architectures into a system as described in 6.3.1, it would also require the implementation of all communication types. The system should handle different communication protocols, though always have a strong focus on security. E.g. if no HTTP proxy servers are involved, direct TCP communication could be used instead.

6.3.3 Security

When developing an instrumentation system to be used with Internet, a clear understanding of the security requirements is needed.

1. Operators and instrument handlers must be able to use systems without the risk of harm.

2. Instrumentation systems sometimes comprise expensive instruments and devices, and it is therefore important to protect them from intentional or unintentional misuse.

3. Graded information must not be readable or accessible to unauthorized users. If using the Internet to transfer sensitive information, cryptographic techniques should be used to protect the traffic.

In both versions of the iMet system, described in chapters 2 and 4, SSL over HTTP was used. This protected the traffic during transport. In addition, the use of digital signatures and MACs would preserve information integrity after transport. E.g. when performing a remote calibration, the calibration results should be signed both by the customer and by the NMI operator. This would make it easier to verify the results at a later stage.

6.3.4 Scalability

To make an instrumentation system as scalable as possible, it should be easy and practical to add or upgrade measurement procedures and support for new hardware. As seen in chapter 5, this was solved by adding the information to a database, after which it could be downloaded and run when needed. Initially, the control application as such did not support any devices or measurements.

For direct instrument operation, two inter-communicating components could be developed for each device, one dynamically added to the operator's application and the other to the application running at the instrument computer. This would make the main control applications independent of the connected instruments. This was tested for the generic instrumentation system described in chapter 5.

6.4 Way forward

Only the principles are developed and tested here. Quite a lot of work remains to make a complete system. The properties of the complete system will be strongly dependent upon the content of the database, which should contain all measurement and hardware information. Work could be saved if developing with reuse in mind, such that devices having similar functionality, utilize the same software proxies.

JV will continue to work for increased NMI collaboration and coordination in the ICT field, among others through the EUROMET network.

Device authentication is a challenging field, which could be satisfactory obtained in the near future. JV has recently started an MSc work to look into this. In that work, analysis of electrical properties of different devices plays an important role in the authentication process.

It is of utter importance to find suitable multifunction transfer standards. Work will continue to search and test different instruments. Internet-enabled calibration without transfer standards will also be looked into.

It is important to look for new ways to utilize the Internet in the metrology area, and this search will continue at Justervesenet.

6.5 Summary of conclusions

During the current work, three systems were developed, which comprised different functions. The first system was also compared and tested against NPL's Internet-enabled instrumentation system, iGen.

6.5.1 First system: iMet v.1.0

What separated the first system from existing instrumentation systems, was the ability to let an operator remotely control an instrument located behind strict firewalls or proxy servers in a secure way. As seen in 2.3.2, this was made possible by utilizing specialized middleware software, which supported full-duplex HTTP channels.

In addition, preinstalled calibration procedures allowed more complex measurements to be run. Because the procedures where preinstalled, the system needed to be rewritten, recompiled, and restarted whenever adding a new procedure. Tests showed that the system was well-suited for performing remote calibrations.

Regarding the security of the system, the focus has been on using standardized security measures. The results from a security analysis showed that the system seems quite secure regarding the Internet-communication, but more work is

needed to ensure long-term non-repudiation of data, e.g. by using digital signatures.

See 2.8 for further conclusions.

6.5.2 Comparison of iGen and iMet

Two instrumentation systems were compared to identify potential areas of improvement. This was NPL's iGen system and JV's iMet system.

Several pros and cons were discovered for both systems. The two main differences were the differences in architectural design and the degree of measurement scalability. The iMet system could be used to operate remote instruments, while the iGen system assumed locally connected instruments. On the other hand, the iGen system was more robust with regards to changing network conditions, since the instrument communication happened locally.

The iGen system was much more flexible with regards to adding and running new measurement procedures, since it interpreted and ran source-code at runtime.

See 3.5 for further conclusions.

6.5.3 Second system: iMet v.2.0

The second system was constructed with the comparison in mind. It utilized the same middleware software as the first system, but the focus was now on scalability. Measurement procedures could be downloaded and run when needed, and new procedures could be added to the system without system recompilation or restart.

Some object-oriented programming languages, like Java and C#, can dynamically compile source-code in memory and run it at runtime. This ability was utilized in the second system, such that source-code measurements could be downloaded by control applications at runtime, and compiled and run in memory. The source-code was added to a dedicated database server beforehand.

Tests showed that the system can be used to perform remote calibrations, though more work is needed to identity well-suited transfer standards.

See 4.8 for further conclusions.

6.5.4 Third system: Generic instrumentation system

The third system was a joint project between NPL and JV. The focus was now on scalable hardware support. Several hardware devices had been identified, which used non-standard PC-cards and control applications. This traditionally resulted in the use of multiple control applications.

The aim for the third system was to construct a system which could dynamically add support for potentially any new hardware at runtime. The

solutions chosen in the second system where further developed to comprise software components used to communicate with hardware devices. This meant adding support for new measurements and new hardware devices to a dedicated database server at runtime, and downloading, compiling, and running this source-code when needed.

The system was tested, and the results showed that the system was able to dynamically add support for instruments with non-standard PC-cards. It is then possible to gather all instrument control applications into one single application, which would lead to less errors, less user training and ease of maintenance.

See 5.8 for further conclusions.

Appendix A

European survey on ICT tools usage

A.1 Background

In 2005, an electronic survey was conducted to explore the current status of ICT-tools usage among different NMIs[1] in Europe through the EUROMET[110] network. The work was part of EUROMET's iMERA[23] project which builds on the MERA[111] project. The results of the survey revealed, to a certain degree, the attitude of NMI employees toward the use of ICT-tools and the Internet in the metrology area. The complete report is available on the Internet [112].

The survey comprised several areas like network topologies and technologies in use, and the opinions of the NMIs employees concerning ICT tools usage.

A total of 81 people participated in the survey, which is somewhat less than expected. Conclusions drawn from the survey results are therefore limited to qualitative trends.

The survey looked at how NMI employees deal with ICT-related issues in their everyday lives. Are they utilizing ICT tools to solve instrumentation problems in a new way, or are they using traditional tools. If the latter, is it because they do not have access to the correct tools, is it because they do not have the proper education, or are they just stuck in a traditional way of thinking? Is there general interest in co-operating with other NMIs using new communication methods? Could instrumentation procedures be improved by allowing several NMIs to co-operate and co-ordinate their actions in a seamless way? What

[1] UK, Germany, France, Italy, Sweden, Denmark, Republic of Slovenia, The Netherlands, Norway, Czech Republic, Poland, Switzerland and Finland

changes are needed to allow different NMIs to further co-operate and co-ordinate their actions? Is there a need for standardization of certain elements like file formats, data formats, or software in order to obtain compatibility between the NMIs? Which economic issues must be dealt with, and who should be responsible for dealing with them?

A.2 Survey Participants

From each NMI, the following persons were selected to attend the survey:

- 1 IT administrator
- 5 Metrologists
- 5 Managers

The relative proportions of each user group to the total number of survey participants are shown in Fig. A.1.

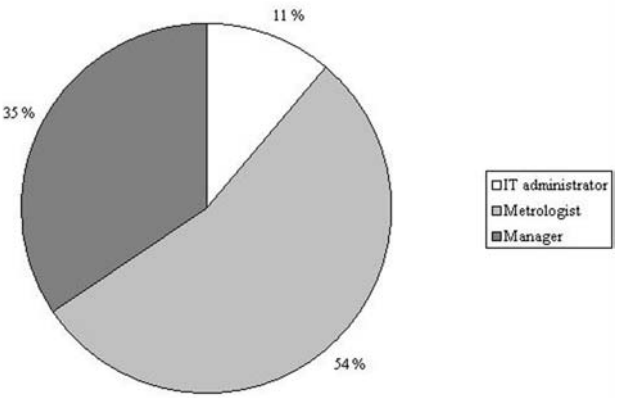

Fig. A.1: Question: What kind of user are you?

A.3 Technology

One of the aspects of the survey was to investigate the degree of inter-compatibility of the European Measurement Institutes, that is to find what needs to be done to let the NMIs co-operate and co-ordinate their actions in a seamless

way. The nature of the co-operation would comprise e.g. remote instrumentation systems, database solutions, and co-operative project management.

The participants were asked what platform the PCs the used at work was running, and the results are shown in Fig. A.2.

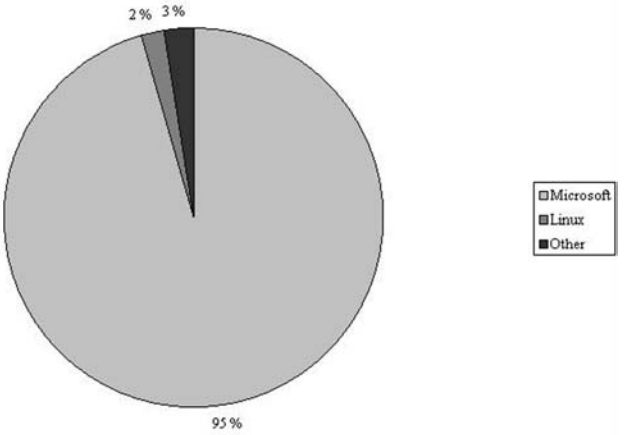

Fig. A.2: Question: Which operating system is running on the PCs you normally use at your organization?

A.4 Trends and Opinions

A.4.1 Communications

The participants were asked if they wanted to use different means of communication, other than email and phone, to collaborate with other NMI employees. The results for video conferencing (VC) and text chat (TC) are shown in Fig. A.3.

As is seen, a majority (67%) of the metrologists would like to use VC tools when collaborating with other NMI employees, while only 51% are interested in using TC tools. The reasons why people answered no can be seen in the following list:

- No added value
- Email is sufficient
- Requires all communicating parties to be online at the same time
- Face-to-face meetings are preferred

- Infrastructural problems
- Too much technical knowledge is required
- Too much preparations
- Emails are easier to understand (listening to low-quality speech is hard)

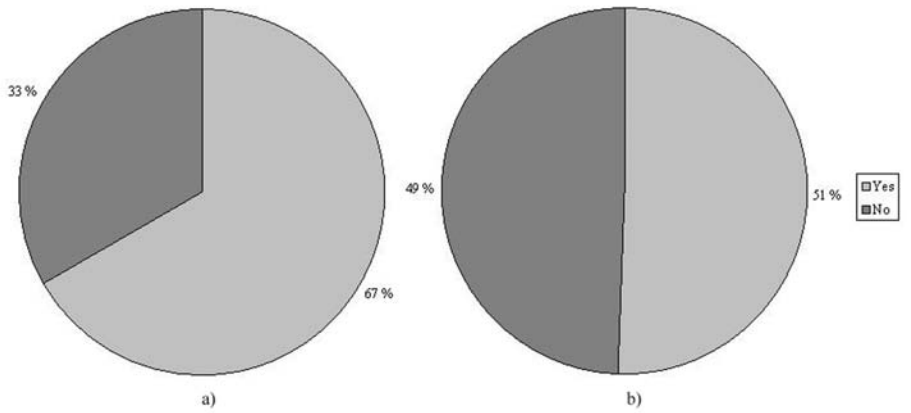

Fig. A.3: Questions: a) Would you like to use VC software to communicate with people at other measurement institutes in Europe? b) Would you like to use electronic TC software to communicate with people at other measurement institutes in Europe?

A.4.2 Measurements

The survey tried to high-light the opinions among metrologists toward remote control and monitoring of measurements. The results found are shown in Fig. A.4.

A.4.3 Templates

The survey tried to find if the NMIs were using templates for different tasks, like document templates and measurement data templates, and the results can be seen in Fig. A.5 and Fig. A.6.

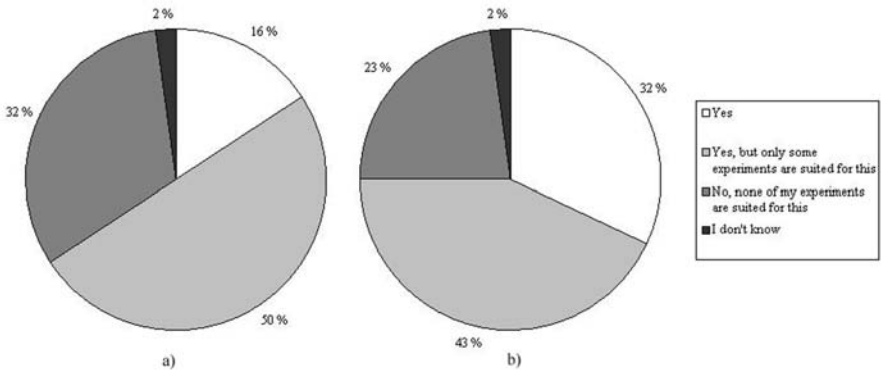

Fig. A.4: Questions: a) If the communication challenges are dealt with, would you like to be able to run your experiments via the Internet? (e.g. from your own home) b) If the communication challenges are dealt with, would you like to be able to monitor your experiments via the Internet? (e.g. from your own home)

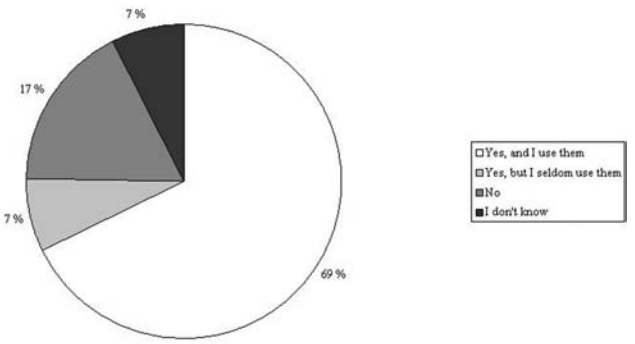

Fig. A.5: Question: Does your NMI provide for word processor templates that may be used by all employees?

A.4.4 Measurement Data Format

As a step toward standardization in order to increase compatibility between different NMIs, the metrologists were asked if they thought it was a good idea for all metrologists, within the same discipline, to use the same measurement data formats. The results are shown in Fig. A.7.

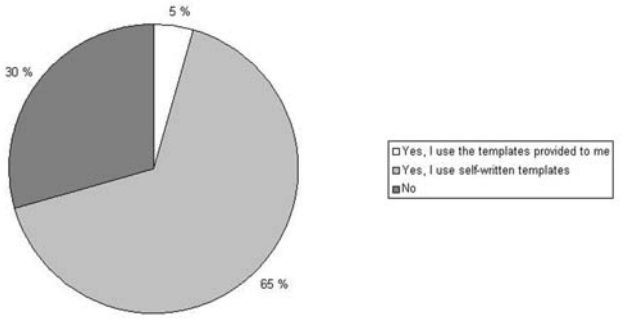

Fig. A.6: Question: Do you save your measurement results in a standardized format, allowing easy exchange of data? (e.g. use the same software spreadsheet template for all your measurement)

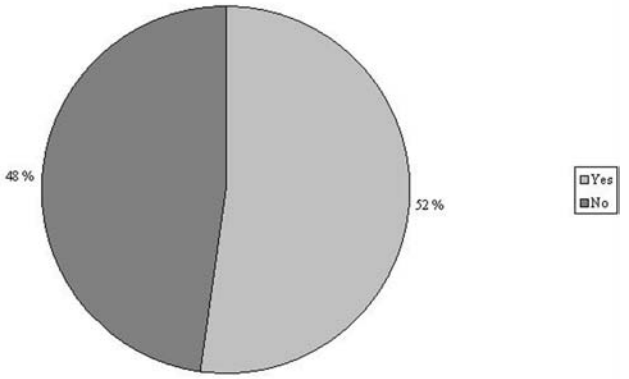

Fig. A.7: Question: Do you think it is a good idea for metrologists, within the same discipline, to use the same measurement data format?

A.5 Conclusions

When looking at the results from the survey, there are a few conclusions that can be drawn:

- About Technology
  - The IT platforms on the client side is mostly Microsoft-based
  - The IT platforms on the server side is shared between Microsoft and Apache

  - Tendency of different solutions and lack of standardization

- About Opinions
  - Metrologists are somewhat reluctant to change formats of existing templates
  - Metrologists are generally interested in monitoring and controlling measurements remotely via the Internet

Appendix B

Network security

In order to protect a company's sensitive information accessible to their LAN users, it is crucial to install some sort of security protection systems like firewalls, proxy servers or network address translators (NAT).

B.1 Firewalls

A firewall is used to analyze and filter all packets exchanged between a LAN and the Internet. If data packets arrive at the firewall, which do not meet certain requirements, they are usually dropped. A firewall is used to protect it's LAN computers from external hackers, and to restrict access to certain domains on the Internet.

B.2 Proxy servers

A proxy server is used to avoid direct communication between a LAN computer and an external host. The proxy will act on behalf of the internal client, and all traffic between the client and the Internet passes through the proxy server. A proxy server is often used to hide the internal address space from the Internet, due to security policy or lack of public addresses. A proxy server is often equipped with firewall and NAT capabilities.

B.2.1 Transparent proxies

Transparent proxies operate so that an internal client is unaware of the proxy server.

B.2.2 Generic proxies

Generic proxies may require user identification before allowing access to certain services. Thus, the internal client needs to be aware of the proxy server.

B.2.3 Dedicated proxies

Dedicated proxies are associated with certain services. E.g. a HTTP proxy server is used to only allow HTTP traffic.

B.3 Network address translators

Due to the shortage of public IP addresses, NATs are used to translate private LAN addresses to public addresses. It changes the sender field in each packet coming from the internal LAN to the NAT's public address. External hosts will believe that the packet originated from the NAT server. The NAT keeps track of each packet sent, so that responses from external hosts are forwarded to the correct internal hosts. NATs may be used in combination with routers and proxy servers.

References

[1] *International Vocabulary of Basic and General Terms in Metrology.* International Organization for Standardization, 1993, ISBN: 92-67-01075-1.

[2] "University Graduate Center," http://www.unik.no.

[3] "Justervesenet," http://www.justervesenet.no/.

[4] H. Shen, Z. Xu, B. Dalager, V. Kristiansen, Ø. Strøm, M. S. Shur, T. A. Fjeldly, J. Lu, and T. Ytterdal, "Conducting Laboratory Experiments over the Internet," *IEEE Transactions on Education*, vol. 42, no. 3, pp. 180–185, 1999.

[5] T. A. Fjeldly, M. S. Shur, H. Shen, and T. Ytterdal, "AIM-Lab: A System for Remote Characterization of Electronic Devices and Circuits over the Internet," in *Proc. 3rd IEEE Int. Caracas Conf. on Devices, Circuits and Systems (ICCDCS-2000)*, Cancun, Mexico, 2000, pp. I43.1–I43.6, iEEE Catalog No. 00TH8474C.

[6] T. A. Fjeldly, J. O. Strandman, R. Berntzen, and M. S. Shur, "Advanced Solutions for Performing Laboratory Experiments over the Internet," in *Engineering Education and Research 2001, A Chronicle of Worldwide Innovations*, W. Aung, P. Hicks, L. Scavarda, V. Roubicek, and C.-H. Wei, Eds. iNEER in cooperation with Begell House Publishers, 2002, pp. 135–146.

[7] J. O. Strandman, R. Berntzen, T. A. Fjeldly, T. Ytterdal, and M. S. Shur, "LAB-on-WEB: Performing Device Characterization via Internet Using Modern Web Technology," in *Proc. Int. IEEE Caracas Conf. on Devices, Circuits and Systems (ICCDCS-2002)*, Aruba, April 2002, pp. I022.1–I022.6, iEEE Catalog No. 02TH8611C.

[8] T. A. Fjeldly, J. O. Strandman, and R. Berntzen, "LAB-on-WEB - a Comprehensive Electronic Device Laboratory on a Chip Accessible via Internet," in *Proc. Int. Conf. on Engineering Education (ICEE 2002)*, Manchester, UK, August 2002, paper no. O337.

[9] T. A. Fjeldly and M. S. Shur, *Lab on the Web, Running Real Electronics Experiments via the Internet.* New York: John Wiley & Sons, 2003.

[10] S. Kolberg and T. A. Fjeldly, "Internet Laboratory with Web Services Accessibility," in *Proc. 2nd Int. Conf. on Multimedia ICTs in Education*, vol. 3, 2003, pp. 1700–1704.

[11] J. Martinez, F. Gmez, T. Zimmer, M. Billaud, D. Geoffroy, H. Effinger, W. Seifert, R. S. Jaeger, T. A. Fjeldly, K. Jeppson, H. Mann, N. Asimopoulos, Z. German-Sallo, R. Cabello, and I. Gonzalez, "eMerge, a European Educational Network for Dissemination of Online Laboratory Experiments," in *Engineering Education and Research 2003, A Chronicle of Worldwide Innovations*, W. Aung, T. Cermak, R. King, and L. M. S. Ruiz, Eds. iNEER in cooperation with Begell House Publishers, 2004.

[12] S. Kolberg and T. A. Fjeldly, "Remote Educational Laboratory System Based on Web Services Standards," in *Engineering Education and Research 2004, A Chronicle of Worldwide Innovations*, W. Aung, R. King, J. Moscinski, S.-H. Ou, and L. M. S. Ruiz, Eds. iNEER in cooperation with Begell House Publishers, 2005.

[13] ——, "Integration of Remote Laboratory Exercises into Learning Management Systems," in *Proc. IASTED Int. Conf. Education and Technology, ICET 2005*, Calgary, Canada, July 2005, pp. 96–101.

[14] "Conference on Precision Electromagnetic Measurements," London, England, http://www.cpem2004.npl.co.uk/, 2004.

[15] "International Workshop - From Data Aquisition to Data Processing and Retrieval," Ljubljana, Slovenia, http://www.amctm.org/nview.asp?id=19&src=home.asp, 2004.

[16] "NMIJ-BIPM Workshop on the Impact of IT in Metrology," Tsukuba, Japan, http://www.bipm.fr/en/bipm/nmij-bipm_workshop/, 2005.

[17] "International Conference - AMCTM 2005," Lisbon, Portugal, http://www.amctm.org/nview.asp?id=24&src=home.asp, 2005.

[18] "2006 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems," La Coruna, Spain, http://ewh.ieee.org/soc/im/vecims/vecims2006/index.html, 2006.

[19] Å. Sand and H. Slinde, "IMet - A Secure and Flexible Approach to Internet-Enabled Calibration at Justervesenet," in *Series on Advances in Mathematics for Applied Sciences - Advanced Mathematical and Computational Tools in Metrology VII.* World Scientific, 2006, vol. 72, pp. 229–236.

[20] Å. Sand, M. Stevens, and G. Parkin, "A Dynamic Instrumentation Framework for Remote Operation of PC-Connected Devices," in *Proceedings of*

2006 IEEE International Conference on Virtual Environment, Human-Computer Interfaces and Measurement Systems, La Coruna, Spain, 2006, pp. 8–12.

[21] Å. Sand, H. Slinde, and T. A. Fjeldly, "A Secure Approach to Distributed Internet-Enabled Metrology," *IEEE Transactions on Instrumentation and Measurement*, accepted for publication.

[22] Å. Sand, M. Stevens, and G. Parkin, "Internet-Enabled Calibration: An Analysis of Different Topologies and a Comparison of Two Different Approaches," *IEEE Transactions on Instrumentation and Measurement*, accepted for publication.

[23] "implementing Metrology in the European Research Area," http://www.euromet.org/projects/imera/.

[24] D. Rayner, "Survey of International Activities in Internet-Enabled Metrology," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, pp. 7–11, 2003.

[25] "National Physical Laboratory," http://www.npl.co.uk.

[26] R. A. Dudley and N. M. Ridler, "Traceability via the Internet for Microwave Measurements Using Vector Network Analyzers," *IEEE Trans. Instrum. Meas.*, vol. 52, pp. 130–134, February 2003.

[27] D. Rayner, "iVR," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 9, 2003.

[28] ——, "iCOLOUR," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, pp. 9–11, 2003.

[29] ——, "iOTDR," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 11, 2003.

[30] "National Institute of Standards and Technology," http://www.nist.gov.

[31] D. Rayner, "SIMnet," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 12, 2003.

[32] ——, "MeasureNet," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 13, 2003.

[33] ——, "Electrical Calibration," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 13, 2003.

[34] ——, "Dosimetry Calibration Services," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, pp. 13–14, 2003.

[35] "Physikalisch-Technische Bundesanstalt," http://www.ptb.de.

[36] D. Rayner, "Coordinate measuring machine calibration and monitoring," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, pp. 15–17, 2003.

[37] ——, "Electrical Calibration," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 18, 2003.

[38] ——, "Collaboration with NPL on AC Josephson voltage standard," http://www.npl.co.uk/ssfm/download/documents/cmsc21_03.pdf, p. 18, 2003.

[39] "Nederlands Meetinstituut," http://www.nmi.nl.

[40] "National Metrology Institute of Japan," http://www.nmij.jp/index_en.html.

[41] "Hewlett-Packard Development Company," http://www.hp.com/.

[42] "IEEE 488," http://www.ni.com/gpib/.

[43] "USB," http://www.usb.org/home.

[44] "Ethernet," http://en.wikipedia.org/wiki/Ethernet.

[45] "IEEE 1394," http://developer.apple.com/firewire/.

[46] "VXIbus Consortium," http://www.vxibus.org/.

[47] "PXI Systems Alliance," http://www.pxisa.org/.

[48] "RS-232," http://en.wikipedia.org/wiki/RS232.

[49] "National Instruments," http://www.ni.com/.

[50] "LabVIEW," http://www.ni.com/labview/.

[51] "Microsoft Visual C#," http://msdn.microsoft.com/vcsharp/.

[52] "SCPI Consortium," http://www.scpiconsortium.org/.

[53] "Standard Commands for Programmable Instrumentation," http://www.scpiconsortium.org/scpiinfo2.htm.

[54] "Virtual Instrumentation Software Architecture," http://www.ni.com/visa/.

[55] "Labwindows/CVI," http://www.ni.com/lwcvi/.

[56] "Measurement Studio," http://www.ni.com/mstudio/.

[57] "Microsoft Visual Studio," http://msdn.microsoft.com/vstudio/.

[58] "Java," http://java.sun.com.

[59] F. Pianegiani, D. Macii, and P. Carbone, "An Open Distributed Measurement System Based on an Abstract Client-Server Architecture," *IEEE Trans. Instrum. Meas.*, vol. 52, pp. 686–692, June 2003.

[60] M. Bertocco, F. Ferraris, C. Offelli, and M. Parvis, "A Client-Server Architecture for Distributed Measurement Systems," *IEEE Trans. Instrum. Meas.*, vol. 47, pp. 1143–1148, October 1998.

[61] W. Winiecki and M. Karkowski, "A New Java-Based Software Environment for Distributed Measuring Systems Design," *IEEE Trans. Instrum. Meas.*, vol. 51, pp. 1340–1346, December 2002.

[62] M. Bertocco and M. Parvis, "Platform Independent Architecture for Distributed Measurement Systems," *IMTC*, vol. 2, pp. 648–651, May 2000.

[63] K. Michal and W. Wieslaw, "A New Java-Based Software Environment for Distributed Measurement Systems Designing," *IMTC*, vol. 1, pp. 397–402, May 2001.

[64] H. Ewald and G. Page, "Client-Server and Gateway-Systems for Remote Control," *IMTC*, vol. 2, pp. 1427–1430, May 2003.

[65] D. Ives, G. Parkin, J. Smith, M. Stevens, J. Taylor, and M. Wicks, "Report to the National Measurement System Directorate, Department of Trade and Industry - Use of Internet by Calibration Services: Demonstration of Technology," *NPL Report CMSC*, vol. 49/04, pp. 1–10, Mars 2004.

[66] A. Carullo, M. Parvis, and A. Vallan, "Security Issues for Internet-Based Calibration Activities," *IMTC*, vol. 1, pp. 817–822, May 2002.

[67] "Common Criteria," http://www.commoncriteriaportal.org/.

[68] G. Sindre and A. L. Opdahl, "Eliciting security requirements by misuse cases," in *Proceedings of TOOLS Pacific 2000*, 20-23 November 2000, pp. 120–131.

[69] B. Schneier, "Attack Trees," http://www.schneier.com/paper-attacktrees-ddj-ft.html, December 1999, Dr. Dobb's Journal.

[70] "CCITT Recommendation X.800 - Security Architecture for Open Systems Interconnection for CCITT Applications," http://fag.grm.hia.no/IKT7000/litteratur/paper/x800.pdf, 1991.

[71] "The Common Object Request Broker Architecture," http://www.corba.org/.

[72] "Java Remote Method Invocation," http://java.sun.com/products/jdk/rmi/.

[73] "Microsoft .NET Remoting," http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/hawkremoting.asp.

[74] "GenuineChannels," http://www.genuinechannels.com.

[75] "The Mono Project," http://www.go-mono.com.

[76] "The DotGNU Project," http://www.dotgnu.org/.

[77] "The Object Management Group," http://www.omg.org/.

[78] "OMG Interface Definition Language," http://www.omg.org/gettingstarted/omg_idl.htm.

[79] "The Remoting.CORBA Project," http://remoting-corba.sourceforge.net/.

[80] ".NET," http://www.microsoft.com/net/basics.mspx.

[81] ".NET Framework," http://msdn.microsoft.com/netframework/.

[82] "Programming languages supporting the .NET Framework," http://www.dotnetpowered.com/languages.aspx.

[83] "Common Language Infrastructure," http://www.ecma-international.org/publications/standards/Ecma-335.htm.

[84] "Ecma," http://msdn.microsoft.com/netframework/ecma/.

[85] "Common Language Runtime," http://msdn.microsoft.com/netframework/programming/clr/.

[86] "RFC 2616 (HTTP 1.1)," http://www.w3.org/Protocols/rfc2616/rfc2616.html.

[87] "Novell," http://www.novell.com/.

[88] M. Day, Lotus, J. Rosenberg, H. Sugano, and Fujitsu, "A Model for Presence and Instant Messaging," http://www.ietf.org/rfc/rfc2778.txt, 2000.

[89] "The SELMA Project," http://www.selma-project.de/.

[90] "Measuring Optical Time-Domain Reflectometers over the Internet," http://www.npl.co.uk/scientific_software/case_studies/software_dev/iotdr/.

[91] "Microsoft Visual Basic Developer Center," http://msdn.microsoft.com/vbasic/.

[92] "PHP: Hypertext Preprocessor," http://www.php.net.

[93] "MySQL," http://www.mysql.com/.

[94] "Internet Transfer Control," http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon98/html/vbconusinginternettransfercontrol.asp.

[95] "XML-RPC," http://www.xmlrpc.com/.

[96] "VB Script," http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/0a8270d7-7d8f-4368-b2a7-065acb52fc54.asp.

[97] "Microsoft SQL Server," http://www.microsoft.com/sql/default.mspx.

[98] J. Nielsen, *Designing Web Usability: The Practice of Simplicity*, 1st ed., ser. VOICES, Unknown, Ed. New Riders, December 1999, iSBN-10: 1-56205-810-X; ISBN-13: 978-1-56205-810-4.

[99] "Sharp Develop," http://sourceforge.net/projects/sharpdevelop.

[100] "Apex-Alpha^{TM} - Alpha Spectroscopy Software Suite," http://www.canberra.com/products/481.asp.

[101] "Mettler-Toledo - BalanceLink Software," http://us.mt.com/mt/products/products-applications_industrial-weighing_sqc-spc-systems/BalanceLink_0x000245f900026ee40007d15c.jsp.

[102] "Leica Geosystems - Axyz Software," http://www.leica-geosystems.com/corporate/en/products/laser_tracker/lgs_1667.htm.

[103] "IVI Foundation," http://www.ivifoundation.org/.

[104] "XmlBlaster," http://www.xmlblaster.org.

[105] "JacORB," http://www.jacorb.org/.

[106] "Java Native Interface," http://java.sun.com/j2se/1.4.2/docs/guide/jni/index.html.

[107] O. Chevtchenko, E. Houtzager, H. Brom, G. Wende, M. Schubert, T. May, H. G. Meyer, O. Kieler, J. Kohlmann, R. Behr, and J. Williamsl, "Synthesis of a Sinusoidal Voltage with Josephson Arbitrary Waveform Synthesizer at NMi," in *2006 Conference on Precision Electromagnetic Measurements*, July 2006, pp. 376–377.

[108] R. Behr, L. Palafox, T. Funck, J. M. Williams, P. Patel, and A. Katkov, "Synthesis of Precision Calculable AC Waveforms," in *2006 Conference on Precision Electromagnetic Measurements*, July 2006, pp. 440–441.

[109] L. Palafox, G. Ramm, R. Behr, W. G. K. Ihlenfeld, and H. Moser, "Primary AC Power Standard Based on Programmable Josephson Junction Arrays," in *2006 Conference on Precision Electromagnetic Measurements*, July 2006, pp. 450–451.

[110] "EUROMET," http://www.euromet.org/.

[111] "Metrology in the European Research Area," http://www.euromet.org/projects/mera/.

[112] H. Slinde, "European Report on ICT Tools Usage - Results Report," http://imera.npl.co.uk/iCohere/default.cfm (username and password needed).