6th International Symposium on AUTOMATIC CONTROL
Middleware-based Training System for Control and Automation Engineering

Cecil Bruce-Boye
University of Applied Sciences Lübeck
Bruce-Boye@fh-luebeck.de

Dmitry Kazakov
cbb software GmbH
Dmitry.Kazakov@cbb-software.com

Sascha Heinecke
cbb software GmbH
Sascha.Heinecke@cbb-software.com

Abstract

Studies at schools and universities are becoming ever more compact and compressed; therefore, the content of automatic control engineering, automation technology and control engineering must be transferred from theory into practice immediately efficiently, effectively and seamlessly.
With the middleware-based data acquisition, distribution and presentation tool such as a middleware experimental equipment, the theoretical treatment, simulation and testing of automation scenarios can be structured step by step as they are in industry so they are easy to understand and can be traced practically.
With the help of the middleware, each individual laboratory experiment on the network (Ethernet/Modbus) can be made accessible to several students. In principle, this works like a network printer.
The network of individual laboratory experiments enables several students in any individual laboratory experiments to access the network via middleware.

1. Introduction

Universities and schools identified the relevant parameter of a close combination of theoretically and practically education. The majority of educational institutions are willing to enhance and proof their student's education in both fields.
A middleware-based training system for control and automation offers the opportunity to enable university and school laboratories to combine theoretical education with state of the art technology. One of the biggest advantages is the use of standard industry components to also give the student

the chance to familiarize themselves with these technologies. Flexibility in hardware and software components utilized for the training enriched the freedom of education and the development in the regarding industries.

2. Learning Objectives

The middleware based training system for control and automation will focus on
Control Systems, in terms of theoretical understanding controller design, simulation and
automation technology, in terms of automated controller design and process automation.
Distributed and heterogeneous systems are common in nowadays industries data environment. Functionalities of the middleware based training system are to represent the automation technology and automatic control engineering data flow in typical industry processes. These functionalities are, but not limited, to process data acquisition, data distribution within the whole scope of automated processes as well as data presentation and simulations.

3. Immediate Knowledge Utilization

The compressed and compact studies at schools and universities create a need for immediate knowledge utilization – from theory to practical exercises. Especially the topics of automatic control engineering and automation technology have to be transferred efficiently and seamlessly.
Utilizing a middleware based physical experiment the theory can be applied immediately.

4. Basics of Middleware

The Abstraction of process layer and application layer is one of the main functions of a middleware. Independent ability of choice for the sensors or actuators as well as their regarding communication protocols on the one hand and application independence for e.g. graphical user interface, simulation tools, data acquisition tools and data analysis on the other hand is the common understanding of a middleware.

4.1. Middleware Architecture Overview

The middleware abstracts connections between nodes and represents them to the application as publisher/subscriber relations. However, the efficiency of this abstraction highly depends on the nature of the underlying connections.
Peer to peer connections, like TCP/IP sockets. The advantage of a peer to peer connection is that it allows straightforward packet filtering based on MAC (Media Access Control) addresses. An error correction mechanism is usually easy to implement, for example, by resending, because both sides are aware of each other´s state. The disadvantage of peer to peer connection is that in the case of 1-n and n-1 connections the overall overhead is a multiplicative of n.

Multicast connections like UDP datagram sent at the broadcast address. The advantage of multicasting lies in fixed overhead for 1-n connections. However using UDP might expose difficult Quality of Service problems and middleware management problems. This problem, which is not specific to middleware, was recognized by IETF (Internet Engineering Task Force) and a series of standards was designed to provide more efficient transport protocols suitable for 1-n connections. In particular IGMP (Internet Group Management Protocol) and PGM (Pragmatic General Multicast) are of special interest for middleware.

4.1.1. Test environment

The testbed network was comprised out of 6 identical computers running Microsoft® Windows® Server 2003 Standard Edition:

- Intel® Pentium® 4 Processor 519 (3.06 GHz);
- Mainboard with FSB 533 MHz;
- 750 MB DDR SDRAM PC400;
- 80 GB HDD;
- NET- 3COM 3C2000, gigabit network card;
- The computers were interconnected using CISCO Catalyst 3560 switch.

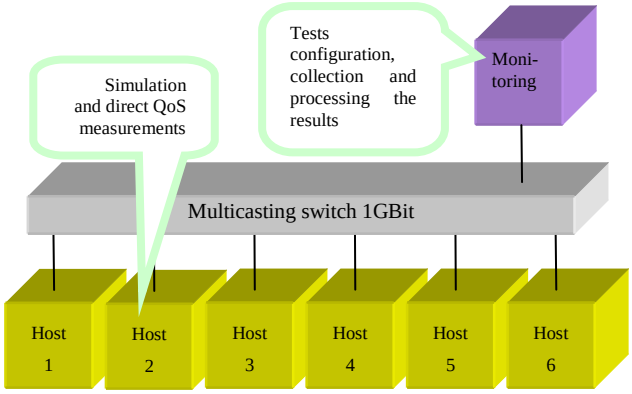

Figure 1

Figure 1 illustrates the simulation configuration. The hosts 1..6 simulated the nodes of a distributed control system connected via middleware. Each host ran the middleware, in this particular environment the middleware LabMap®, and a test application simulating data distribution and estimating Quality of Service.

Time Synchronization

The middleware nodes are responsible for time stamping of the values of the variables and the events related to them. Time stamping requires globally applicable time stamps, which can be stored, restored and transmitted over the network. This inevitability requires applying UTC (Universal Coordinated Time) timestamps rather than local political time.

The absolute accuracy of the time stamps is usually not required to be very exact. Normally ±1s would be suitable for all purposes. At the same time the relative accuracy within the distributed

system need to be much better. The relative clock reading on different nodes have to be far under 1ms accuracy margin.

There are two scenarios of synchronizing timestamps in the system:

Synchronization of the time sources. For example, by using synchronized atomic clocks, or by distributing time signals of the same clock among all participants.

Translation of the time stamps to one base.

The first approach is more universal but also more expensive and fragile. It also presents a difficult maintenance problem by requiring an access to all participants. With the second approach the clocks of data sources remain intact, but the time stamps are adjusted as they arrive at the subscriber side.

The utilized middleware offers both options.

4.2. Quality of Service Measurement (QoS)

To reduce the dimensionality of the result space we combined the number of variables n, and the publishing period Δt into an integral parameter n/Δt, which characterizes the number of middleware variables state changes per second. We discovered that the values of n/Δt is a key factor that influences QoS. Within wide bounds, the number of variables can be safely increased when the publishing period is prolonged and reverse.

Losses

Figure 2 and 3 represent losses by uni- and multicast correspondingly. As expected, multicast data distribution does not depend on the number of subscribers. The performance of unicast distribution degrades with the number of subscribers. The maximal changes frequency in both cases was about 104 state changes per second. This practically means, that for e.g. 500 variables could be published no more frequently then 50ms period. An important observation about multicast distribution was an abrupt rise of losses in the transition area. It means that a multicast based system should be planned more thoroughly to have more spare resources than a unicast- based system.

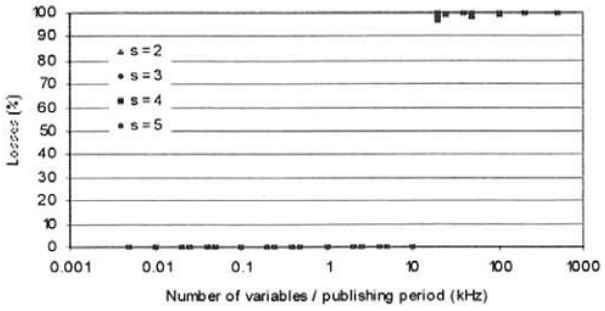

Figure 2 - Unicast losses under stress load

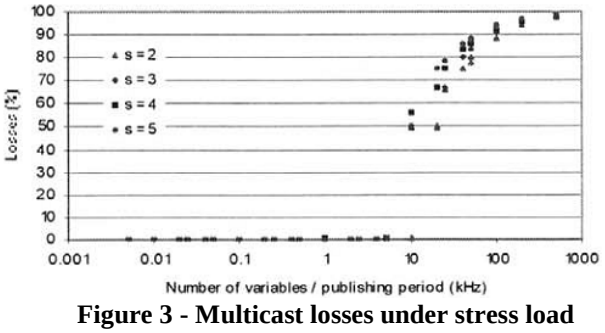

Figure 3 - Multicast losses under stress load

4.3. Application example – Applied Middleware

Within the scope of transferring theory of controller and automation engineering to practical experiments the didactics let the education occur seamlessly in the steps:

Theory (offline)

- Controller Design
- Simulation
- Practical component (online)
- Testing and Evaluation using the network experiment

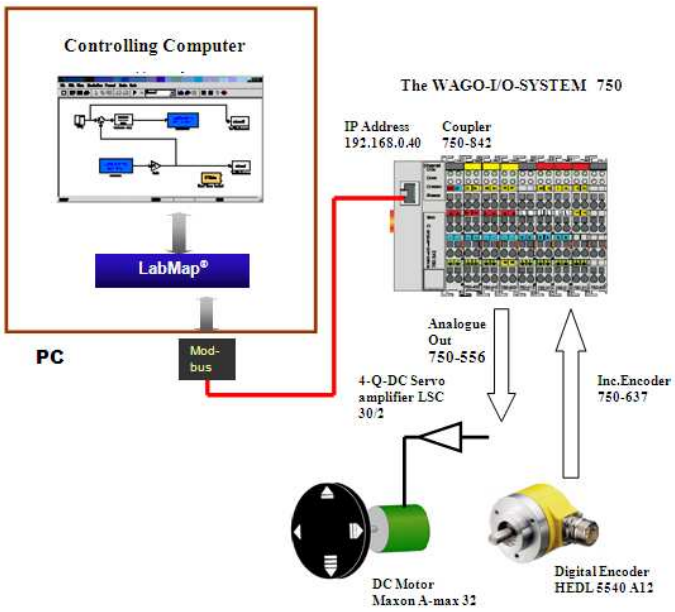

Figure 4 – system overview and hardware setting

Remark: Experiments with a flexible real time requirement (>10ms) can be operated under MS Windows without an additional real time environment.

4.4. Example control engineering

Digital PD Controller

Results from the M-File (MATLAB/Simulink)

The Connection between the Simulation tool MATLAB/Simulink and the underlying hardware representing the process to be controlled are connected per middleware. The connectivity is enabled by basically two main functionalities of a middleware, the ability to receive and send values to/from the simulation tool and the hardware.

Simulation Tool: MATLAB/Simulink

Hardware: DC Engine with amplifier and decentralized peripherals (digital and analog I/O) connected via Modbus/TCP.

The middleware - MATLAB/Simulink interface is established through a library of MATLAB s-functions, which encapsulates the standard middleware programming interface, written with the c programming language.

The two middleware functionalities are:

Get – Reads the value of the variable in a safe way. It is guaranteed that the read value bits and timestamp are consistent. The application is relieved from the burden of locking the value in presence of concurrent tasks accessing it. The application is unaware of the value source and the policy used to actualize or obtain the value.

Send – Initiates writing the value on the underlying hardware. The application is unaware of the actions the hardware undertakes upon send.

Remark: To achieve a comparison of simulated controller behavior and applied controller behavior we slow down MATLAB/Simulink to soft real time system.

4.5. Controller Design Theory

Within the calculation for the controller we utilize the following procedure and formula:

If the transfer function for the PD controller is denoted by

$$G_c = \frac{U(s)}{e(s)} = \frac{Kr(1+T_D s)}{1+T_{st} s} \quad (1)$$

The open loop transfer function yields

$$G_o = \frac{Kr(1+T_D s)}{1+T_{st} s} * \frac{k_e k_m}{(1+T_1 s)(1+T_2 s)} \quad (2)$$

If T2>T1, and we set TD=T1, we get for the controller design the following expression

d=closed loop damping ratio (3)

$$Kr = \frac{(T_2 + T_{st})^2}{4 T_2 T_{st} d^2} - 1$$

This leads to the digital PD controller with the coefficients

$$q_0 = \frac{Kr(T_0 + T_D)}{T_0 + T_{st}}; q_1 = -\frac{Kr T_D}{T_0 + T_{st}}; p_1 = \frac{T_{st}}{T_0 + T_{st}} \quad (4)$$

Of the digital PD controller

$$U_k = q_0 e_k + q_1 e_{k-1} + p_1 U_{k-1} \quad (5)$$

4.6. M-File for the controller design

```
% DC drive with PD controller in the lab
%Transfer function of the identified DC drive
num = 7224.07324                          % numerator of the drive
den = [1 13.63312 23.89440]               % denominator of the drive

p = roots(den)                            % real roots of den(s)
t1 = -1 / min(real(roots(den)))           % small time constant

t2 = -1 / max(real(roots(den)))           % large time constant

ke = num/den(3);                          % speed/voltage converter
km = 1/ke                                 % output/input for unit feedback

%design of PD controller
td = t1                                   % td = 0.0864
tst=8*t2                                  % tst = 3.8729
d=0.707                                   % damping ratio = 4%
kr=((t2+tst)^2/(t2*tst*4*d^2))-1          % find kr = 4.0640

%coefficient of digital PD controller
t0=0.01;                                  % sample time
q0=kr*(t0+td)/(t0+tst)                    %q0=0.10095
q1=-(kr*td)/(t0+tst)                      %q1=-0.090483
p1=tst/(t0+tst)                           %p1=0.9972
```

4.7. Practical Component

Executing the controller as calculated afore we create the following MATLAB/Simulink model. “Get” and “Send” are the interfacing to the utilized middleware. The comparison of the calculated controller by system identification via middleware (Get/Send) and the applied controller is done in this model and shown in figure 4 and 5.

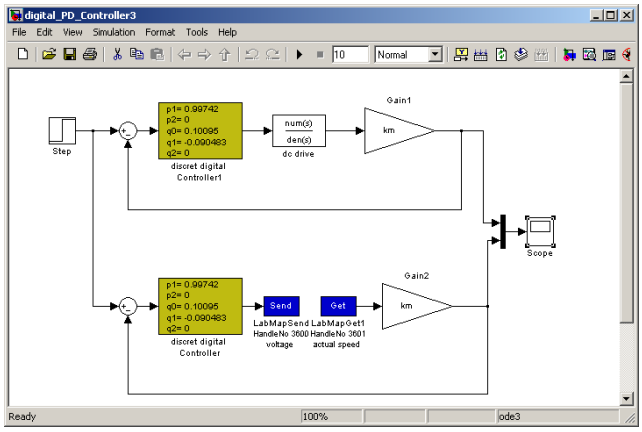

Figure 5 – simulated and real closed loop controller

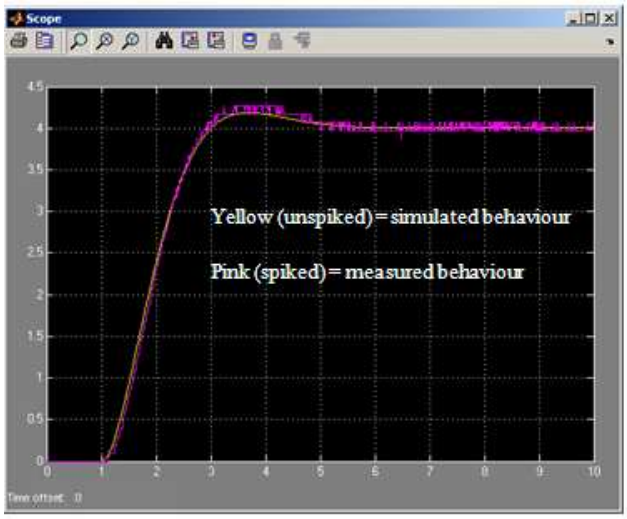

Figure 6 - simulated and real step response

5. Networking Experiment via Middleware

5.1. One for all – Networking

From the individual laboratory experiment to the network experiment.
The networking of individual laboratory experiments enables several students in any individual laboratory experiment to access the network via middleware.
Utilizing the middleware, each individual laboratory experiment on the network (Ethernet/Modbus) can be made accessible to several students. In principle, this works like a network printer.
The following figure 6 illustrates the network experiment in the so called setting “one for all” – meaning one hardware experimental unit far several students to be used. In this case the middleware is not only responsible for the linking of the simulation tool and the hardware but also for the networking functionality in a distributed laboratory environment connecting several workstations with one hardware experiment.

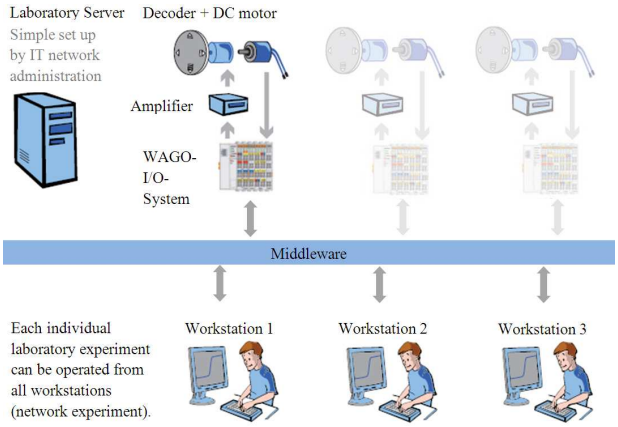

Figure 7 - one for all – Networking

5.2. All for all – Networking

The concept “one for all” is also well suited for modernizing existing laboratory experiments and making them network capable. Additionally the set up of a network capable distributed laboratory can be a matched with the setting of several hardware experiment sets and workstations, again all connected via middleware –"all for all”. This gives a new approach in distrusted control and automation.

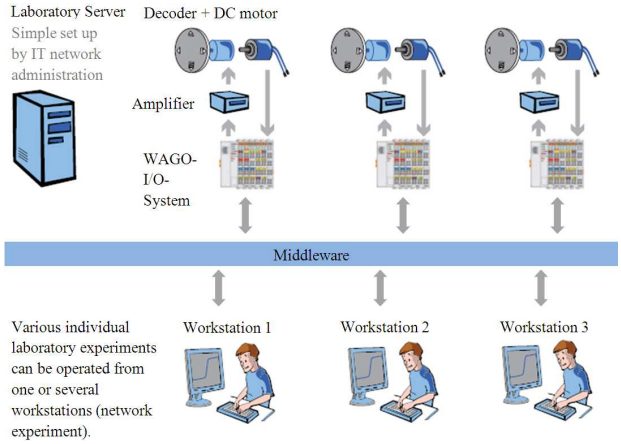

Figure 8 - all for all - Networking

5.3. Automation Technology

The third approach of a distributed middleware is the hardware and software simulation of real industry processes as e.g. the collaboration of several sensoring/actuating processes; all observed from one control station and centralized data acquisition (SCADA – Supervisory Control and Data Acqusition)

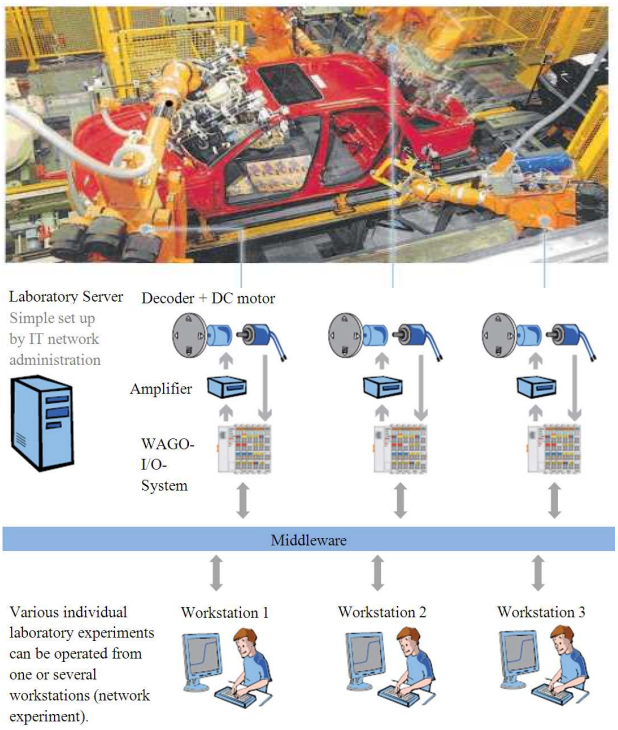

Figure 9 Assembly line in progress - simulation

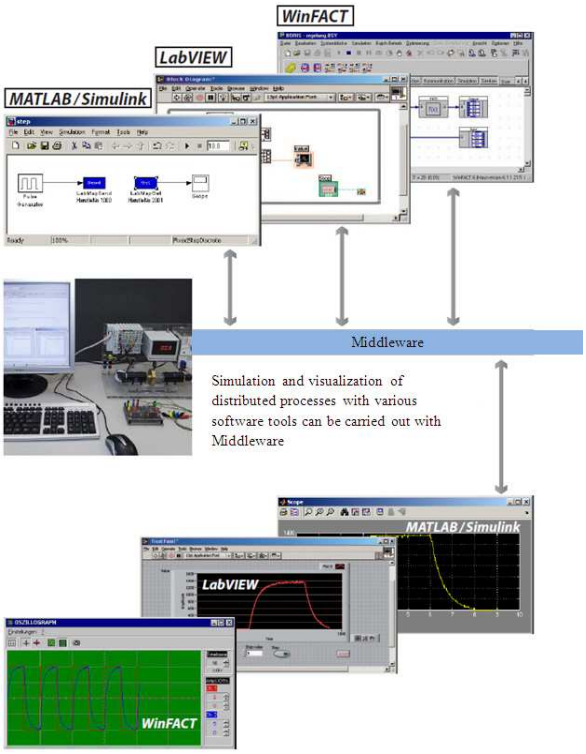

Figure 10 - multi tooling

5.4. Distributed, heterogeneous simulation by multi tooling

Further approaches of a middleware based system are the multi tooling. The ability of utilizing several tools, such as e.g. popular simulation tools like MATLAB/Simulink, LabView or WinFact, gives the opportunity of distributed control and tool comparison in one network. As described in figure 8 simulation and visualization is done synchronously and well arranged, even with various settings of hardware.

5.5. Reference Middleware

The middleware utilized for the afore described purposes as for Training systems for control and automation engineering is the well recommended middleware LabMap® (see also VDI/VDE 2657 Middleware in Automation)

References

[1] C. Bruce-Boye and D. Kazakov, *"Distributed data acquisition and control via software bus,"* Proceedings CSMITA'04, pp. 153–156, Sep 2004.

[2] C. Bruce-Boye, D. Kazakov, "*Quality of Uni- and Multicast Services in a Middleware. LabMap Study Case"* Conference CIS2E 06 (International Joint Conference on Computer, Information and System, Science and Engineering") 2006, IEEE, 4-14 December 2006

[3] C. Bruce-Boye, D. Kazakov; Rüdiger zum Beck, "*An approach to distributed remote control based on middleware technology, MATLAB/Simulink-LabMap/LabNet framework"*, Conference CIS2E (International Joint Conference on Computer, Information and System, Science and Engineering") 2005, IEEE, 10-20 December 2005.

[4] C. Bruce-Boye, D. Kazakov, *"Distributed data acquisition and control via software bus"*, International Industrial Ethernet Development High Level Forum 2004 (IEHF 2004) in Peking, Automation Panorama No. 5

[5] D Luenberger, "*An introduction to observers", IEEE Trans. Automatic Control*", AC-16 1971

[6] Ashish Tewari, *"Modern Control Design with MATLAB and Simulink"* John Wiley and Sons, Inc. 2002.

[7] *"LabVIEW Simulation Module User Manual*", National Instruments, April 2004.

[8] http://www.wago.com

[9] C. Dorf, R. Bishop, „*Modern Control Systems (9th Edition)"*, Science Press and Pearson Education North Asia Ltd. 2002, ISBN 7-03-010133-2

[10] D. Bakken, "*Middleware*", Washington State University

[11] G. Coulouis, J. Dollimore, T. Kindberg, *"Distributed systems. Concepts and design"*, Addison-Wesley, fourth edition 2005, ISBN-10: 0-321-26354-5.

[11] cbb software/Wago, *"CATS – Control and Automation Training System"*, cbb software, Germany, 2009