Publication [P3]

M. Pohjola, L. Eriksson, V. Hölttä, T. Oksanen, “Platform for Monitoring and Controlling Educational Laboratory Processes over Internet”, in *Proc. 2005 Congress of the International Federation of Automatic Control (IFAC)*, Prague, Czech Republic, July 3-8, 2005, 6 p.

PLATFORM FOR MONITORING AND CONTROLLING EDUCATIONAL LABORATORY PROCESSES OVER INTERNET

Mikael Pohjola[1], Lasse Eriksson[1], Vesa Hölttä[1], Timo Oksanen[2]

[1]Control Engineering Laboratory, Helsinki University of Technology, P.O.Box 5500, FI-02015 TKK, Finland
[2]Automation Technology Laboratory, Helsinki University of Technology, P.O.Box 5500, FI-02015 TKK, Finland

Abstract: This paper presents a new platform which can be used for monitoring and controlling different educational laboratory processes over the Internet. The platform is built for a laboratory course at Helsinki University of Technology. The students can use the same platform for different experiments from distance or on the spot, which makes the use of the system easy, and the emphasis can be put on the content of the experiments. A special feature implemented in the platform is ability to pass the measurement and control signals via a network simulator, and thus the evaluation of the effect of networks in control can be studied. *Copyright © 2005 IFAC*

Keywords: Networks, control education, distributed control, simulators, software tools.

1. INTRODUCTION

Distance learning unties students from a given place and often also from a given time. The web has provided means to distribute teaching material in an efficient and more versatile form than a traditional textbook can offer. In addition to teaching and learning, the web offers new possibilities for control engineers. Nowadays, building automation systems, spare part dealers' warehouse balances, and even paper mill production statistics are monitored over the Internet. The advantages are the same as in learning: independency of time and place.

At the Helsinki University of Technology (TKK) students having automation and control engineering as their major or minor subject are required to pass a laboratory experiment course. During the course the students are familiarized with some practical cases similar to the problems that could arise in their future work as engineers. The course is offered in cooperation by Control Engineering and Automation Technology laboratories. The experiments were mostly built in the 1970's, but since then the world of automation and control has significantly changed. The development has been recognized, and renewal and update of some of the experiments has already been reported (Hölttä and Eriksson, 2003; Varso and Koivo, 2003).

The preceding work done in the field of distance learning at the Control Engineering Laboratory at TKK includes web based courses (Riihimäki *et al.*, 2003) and illustrating the properties of multivariate regression methods in the web using MATLAB Web Server (Hölttä and Hyötyniemi, 2003). Since the laboratory experiment course was being updated and the role of distance learning was becoming more important, a platform for monitoring and controlling laboratory experiments over the Internet was needed.

This paper describes the educational objectives (section 2) and technical details (section 4) of the developed platform. In section 3, the network delay distributions, which play an essential role in the system, are examined. The paper ends with conclusions.

2. EDUCATIONAL OBJECTIVES

The aim of this work was to develop a scalable monitoring and controlling platform that can be used for doing educational laboratory experiments on different processes over the Internet. In this way students can do laboratory experiments partially or completely at a distance, whenever it suits them best.

The platform was designed so that it is possible to monitor processes, and change the controllers and parameters. Several students can monitor a process at the same time, whereas only one can control it. Scalability implies also that new experiments can be added to the platform with little effort.

The fact that a common platform used in several laboratory experiments eases the students' workload. It is e.g. not necessary to learn a new user interface for each experiment. This becomes even more important when experiments are done from a distance. Any problem with the user interface may flatten the atmosphere and discontinue the experiment.

Besides distance learning, another new and developing area related more or less to automation is the use of networks in measuring and even in control. The really hot topic is the wireless sensor networks, although there are still many issues to be investigated before they can be used in real-time control. The measurement packets are delayed in networks, and the delays are time-varying. The analysis of control systems with time-varying delays is difficult, nevertheless, there are results available, see e.g. (Lincoln and Bernhardsson, 2000; Koivo and Reijonen, 2004).

The development of a web based monitoring system for the laboratory experiment course gave an excellent opportunity to demonstrate the students how the time-varying delays that are caused by different networks affect closed loop control. This aspect is treated in detail in section 3. In addition to the new platform that is used for measuring and visualization of results on the course, a new experiment focused on network delays was developed.

3. NETWORK DELAYS

One of the remote monitoring and controlling system objectives was to enable students to see the effect of varying network-caused delays in closed loop control. Instead of implementing a variety of networks with hardware, a network simulator called NetSim was built. The delay properties of different networks vary depending on the communication protocols and hardware implementation. Networks can be characterized by delay distributions. There are four delay distributions implemented in NetSim: Gaussian (Normal), Gamma, Lognormal, and "wireless network delay" distribution. Besides the delay, packet loss is possible, i.e. packets may disappear during transmissions. The packet loss probability is one of the parameters that can be adjusted in NetSim.

The simulation replicates the statistical properties of the simulated networks, not the sequential dependence of the packet delays. In other words, the delay distributions are similar to the actual delays, but packets sent quickly after one another experience different delays, though one would expect that on short time scale the delay would be constant. It has indeed been shown that the packet delays are correlated (Bolot, 1993).

3.1 Gaussian distribution

The Gaussian or normal distribution does not model any specific network, but it can be used as a generic delay describing some variance in the delay around a mean value. It is possible to obtain negative delays when using Gaussian distribution (by setting a sufficiently large variance), which in the implementation translates to no delay. If the expectation value of a Gaussian delay is set to zero, the distribution is actually turned from a Gaussian to one that has a large probability mass at zero, and above zero behaves like a normal distribution.

3.2 Gamma distribution

In a computer network such as the Internet, the delay distribution has been shown to resemble a shifted gamma distribution (Mukherjee, 2001). The measured delay distributions have had a positive minimum value from the shortest possible delay (therefore the shift), a peak and a long tail, similar to the gamma distribution. The gamma probability distribution function is given by

$$P(x) = \frac{\alpha}{\Gamma(n)}(\alpha x)^{n-1}e^{-\alpha x} \tag{1}$$

where Γ is the gamma function

$$\Gamma(n) = \int_0^\infty t^{n-1}e^{-t}\,dt \tag{2}$$

and n is the number of hops between first and last nodes, $\alpha = n / T$ where T is the mean delay. Fig. 1 shows the gamma probability distribution function for n = [1, 10] with mean delay T = 1. With n = 1 the distribution becomes exponential. The variance of the gamma distribution is

$$\sigma^2 = n/\alpha^2 = T^2/n \tag{3}$$

The shift is accomplished by moving the graph to the right. The result of the total network delay distribution consists of a *static* and a *stochastic* component

$$T_{tot} = T_{static} + T_{stochastic} \tag{4}$$

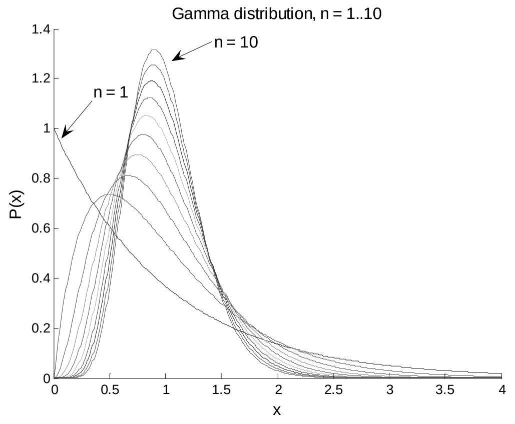


Fig. 1. Gamma distribution, n = [1, 10].

Table 1. Typical delays (milliseconds) in the Internet.

Connection quality	Static delay [ms]	Variance (stochastic delay)	Hops (n)	Loss %
Small	15	5	2	0
Fast	30	50	15	0
Normal	70	100	15-25	1
Slow	400-800	500-1200	20	2

where T_{static} is the minimum delay from the network and $T_{stochastic}$ the delay induced by other traffic, approximated with the gamma distribution.

3.3 Lognormal distribution

The lognormal distribution is similar to the gamma distribution, and it is derived from the normal distribution by defining a random variable x to have lognormal distribution if log(x) is normally distributed, i.e. $N(m,s^2) \sim \log(x)$. The lognormal distribution with mean μ and variance σ^2 is given in eq. (5).

$$P(x) = \frac{1}{xs\sqrt{2\pi}} e^{\left(-\frac{1}{2}\left(\frac{\log(x)-m}{s}\right)^2\right)},\ x > 0 \qquad (5)$$

m and s^2 are mean and variance of the normally distributed log(x), respectively, and are given by

$$m = \log\left(\frac{\mu^2}{\sqrt{\sigma^2+\mu^2}}\right),\ s = \sqrt{\log\left(\left(\frac{\sigma}{\mu}\right)^2+1\right)} \qquad (6)$$

The lognormal distribution is calculated from the normal distribution by taking exponents from both sides. Gamma and lognormal distributions with mean 1 and variance 0.5 are compared in Fig. 2.

The lognormal distribution can also be shifted in the same manner as the gamma distribution to get a realistic network delay distribution.

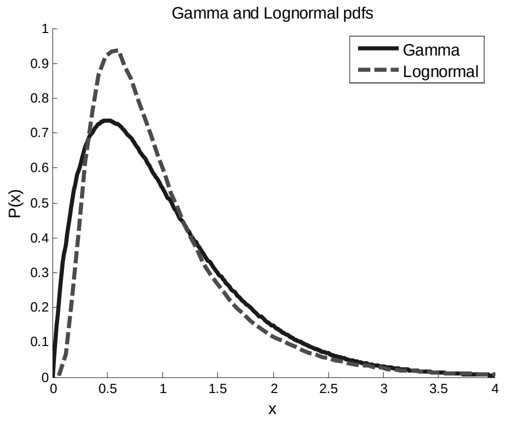


Fig. 2. Gamma and lognormal probability density functions with μ = 1 and σ^2 = 0.5.

3.4 Wireless network delay distribution

There is a delay implemented in NetSim mimicking the behaviour of two wireless nodes. The delay distribution resembles two gamma distributions, the second being smaller and shifted to later time corresponding to retransmissions (Lucan *et al.*, 2003).

3.5 Packet loss

Packet loss seems to be random, independent of the delay according to the investigation of (Bolot, 1993), though in some protocols (e.g. TCP/IP) packet retransmissions cause delays. The probability of packet loss is adjustable in NetSim. Table 1 shows typical values for static (minimum constant) and stochastic (variable) round-trip-times in the Internet. The network delays and variances were obtained by pinging several servers from all over the world. The means and variances were then calculated from the data. One could categorize the results into several classes:

- Small: Computers in a local area network (LAN)
- Fast: Servers from Europe
- Normal: Servers from Asia
- Slow and unreliable: Servers from Africa

4. SYSTEM COMPONENTS

The implemented platform for monitoring and controlling laboratory processes was given the name MoCoNet. System hardware consists of computers (of which one is a server), an I/O board attached to one of the computers, a network router, and process equipment. This section discusses the components (software and hardware) of the platform. The logical architecture of the platform is shown in Fig. 3.

The users monitoring and controlling processes are connected to the server of the system. The server takes care of the administrative issues of the platform. The measuring, actuators and control signal

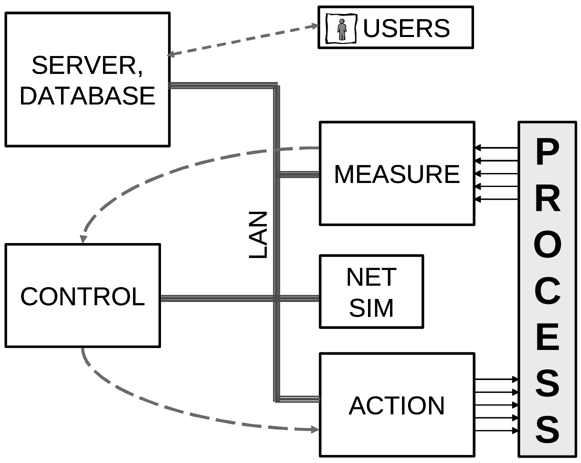


Fig. 3. Logical architecture of the platform.

computing are distributed in the network. All packets are transmitted through NetSim that can be used for delaying the packets. The server logs all network traffic and offers the data to users.

4.1 Software

For a user, the only visible part of the system is the graphical user interface that is implemented as a web page. Besides HTML, Java applets are employed e.g. in connecting to the control network and in displaying the collected data in a scope.

The server computer is responsible for maintaining connections between users and processes, for running a reservation system for controlling the processes (which user is allowed run which process and when), for collecting all the data into a database and for configuring the network simulator. It is used also for configuring the system.

The server side consists of three Java programs: *DatabaseServer*, *ProcessServer* and *SignalServer*. The DatabaseServer listens to process measurement and control messages that flow in the local network and stores them in a Microsoft Access database located on the server. The ProcessServer serves the user interface applet called ProcessControl, and enables the user to control processes from a remote location. The SignalServer transmits the process measurements to the remote location applet.

From the ProcessControl applet the user can control the processes. The user can choose reference signals, controllers, and set their tuning, start and stop the processes etc. The ProcessControl applet spawns *ProcessMonitor* windows that are like scopes, displaying process measurements.

The control permission reservation system is a part of the ProcessServer. The reservation system maintains a list of all clients that have requested to control processes. If the user in control releases the control or stops the process, the control is passed to the next user on the reservation list.

The measuring and controlling software is run on a computer equipped with MATLAB xPC Target real-time operating system (RTOS). A RTOS is required since otherwise it would not be possible to guarantee that the next value of the control signal is computed fast enough. Since the measuring and real-time control of processes are done with C code running on MATLAB's RTOS, all uploads (configuration) must be done with MATLAB. The MATLAB Web Server (MWS) offers tools for updating and configuring executable code over Internet. With the MWS it is possible to call MATLAB functions from a web page and thus upload new parameters into controllers or change the measurement channels of the I/O board.

Process-end software is responsible for measuring process variables and for transmitting the measurements into the network. The controller is implemented on the same computer as the measurement unit, but the measurement unit, controller and actuator are virtually distributed in a network, since all packets are sent via NetSim. Controller and process-end software can be separated to two computers but, in order to save hardware, only one computer is used.

The server side Java programs are connected to MATLAB programs through the MATLAB Web Server interface, which is also used for uploading the process control models with tuned controllers into the xPC Target computer. Once the model is uploaded, start and stop messages can be transmitted through the MWS to run the real processes.

4.2 Hardware

The system contains three physical PC's, all connected to an Ethernet router. The router acts as a gateway to the Internet at the same time. Multicast messaging is used inside the LAN, and socket connections to remote user interfaces. Because the network is simulated, it would be possible to realize all the modules on a single physical computer as separate components or processes. However, the modules are separated both in hardware and in software to enable testing of different tools and allowing easy modification of the system. The physical architecture of the implemented system is presented in Fig. 4.

The process-end computer called xPC Target is equipped with an I/O board and the MATLAB xPC Target RTOS. The computer is used for executing compiled C code including among other things controller algorithm, UDP packet sending and receiving and I/O functions. The code is automatically generated from Simulink models using the Real-Time Workshop and the xPC Target Toolbox. This combination of extensions allows easy prototyping of control software.

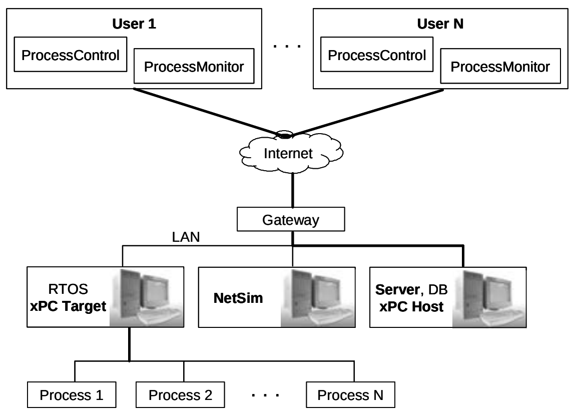


Fig. 4. Physical architecture of the platform.

The server computer is also called the xPC Host, because the xPC Target computer can be configured from this host. The host runs MATLAB with Simulink, Real-Time Workshop and xPC Target Toolboxes installed. The host PC can be used for compiling the Simulink models into executable code and the code can be uploaded into the target PC using MATLAB Web Server.

4.3 NetSim

Networked monitoring and control is usually implemented with some industrial network (Foundation Fieldbus, Profibus, CAN-bus). Each network has its pros and cons regarding performance and reliability.

In this work 100 Mbps Ethernet is used as a real network. Even if Ethernet is not an industrial network, it can be used in small solutions for controlling processes with quite high time constants over network.

One of the objectives of the presented platform was to show the control properties of a networked system with varying delays. The delay for control and measurement signals could be generated using long wires, for example in this case the messages could be sent back and forth in the Internet. However, for educational purposes the network properties should be easy to vary in order to show the effect of each property. For comparison it should also be possible to replicate the delays occurring during the experiments. Therefore the network visible to other modules is simulated with a device, NetSim, connected to the actual network. The delay properties of the used Ethernet network are considered insignificant with respect to the simulated variables.

Network properties can be divided into two parts, delay and loss of packets. In deterministic networks the loss probability is zero. Both of these statistical properties can be time-varying or time-invariant.

NetSim hardware is a desktop PC with the QNX RTOS. All control and measurement packets in the control network are transmitted via NetSim. When a packet is received, the loss probability is calculated. If loss filter is passed, the delay is randomized from the selected distribution, a retransmitting timestamp is calculated and the packet is put to queue to wait until the real-time clock reaches the timestamp. Another thread of the software is responsible of transmitting packets back to the network.

UDP/IP packets are used in the local network. This is because UDP packets have low latency when compared to TCP communication, and there is no need to know what other modules belong to the network as multicast messaging is used. In practice it is impossible to use exactly the same messages for receiving and transmitting, so one UDP port is used for receiving messages and the other for transmitting. The network properties can be configured remotely, using a third UDP port.

4.4 Communication protocol

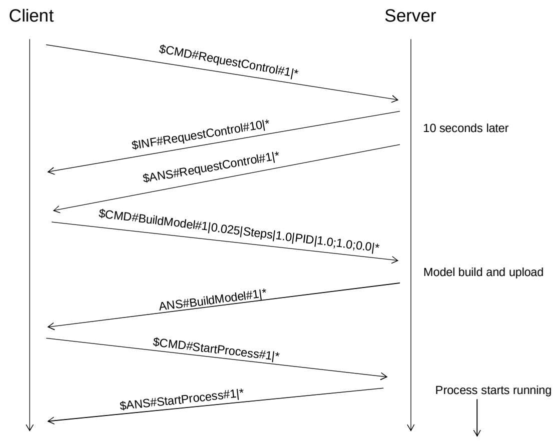
A specific inter-component communication protocol was needed. The *ProcessAscii*-protocol (PAP) was developed for asynchronous transmission of process information over a text based connection. The protocol supports queries of e.g. process properties, and it transmits control and measurement signals.

A PAP message consists of a message identifier, header and data. The messages are separated with a newline character. The format of the message is:

```
$MID#Header#Data|*
```

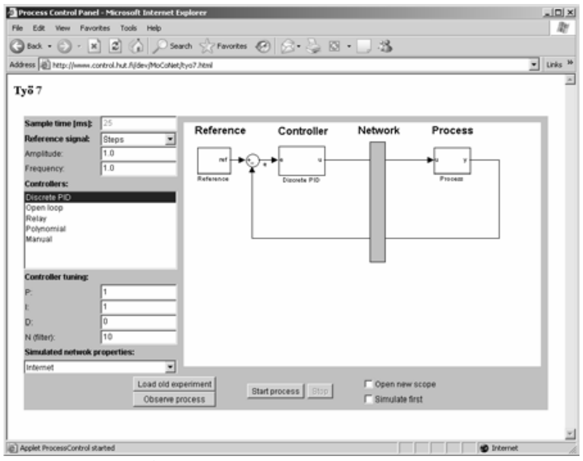
The message begins with a $ followed by a three character message identifier. The header is between the #-characters. If there is no header, only one #-character is used. The whole message is terminated by a star *. Multiple items in data are separated with a vertical bar character |.

The PAP includes several special messages and command-reply pairs. Fig. 5 shows an example of using the protocol when a new client contacts the server in order to control one of the processes.



```
$CMD#RequestControl#1|*
$INF#RequestControl#10|*
$ANS#RequestControl#1|*
$CMD#BuildModel#1|0.025|Steps|1.0|PID|1.0;1.0;0.0|*
ANS#BuildModel#1|*
$CMD#StartProcess#1|*
$ANS#StartProcess#1|*
```

Fig. 5. PAP messages during process start-up.



Process Control Panel - Microsoft Internet Explorer

Address: http://www.control.hut.fi/dev/MoCoNet/tyo7.html

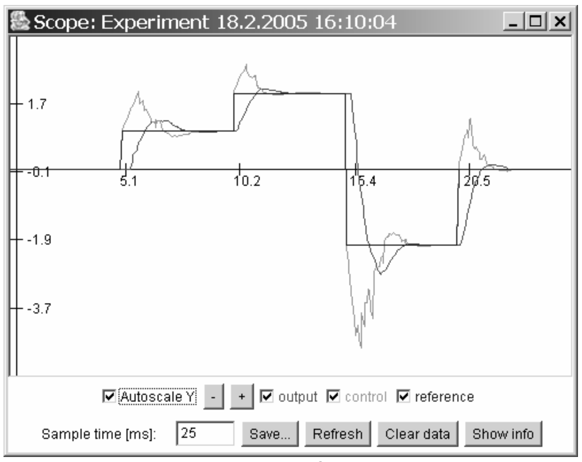
Työ 7

Sample time [ms]: 25
Reference signal: Steps
Amplitude: 1.0
Frequency: 1.0
Controllers: Discrete PID, Open loop, Relay, Polynomial, Manual
Controller tuning:
P: 1
I: 1
D: 0
N (filter): 10
Simulated netwok properties: Internet

☐ Open new scope
☐ Simulate first

Applet ProcessControl started

Fig. 6. GUI of monitoring and controlling system.



Scope: Experiment 18.2.2005 16:10:04

☑ Autoscale Y
☑ output
☑ control
☑ reference

Sample time [ms]: 25

Fig. 7. Observing control performance on a scope.

4.5 Graphical user interface

With the graphical user interface (GUI) of MoCoNet it is possible to monitor and control in real-time processes that are connected to the system. Besides the process, controller, network type and reference signal may be selected with the GUI. Also controller parameters can be changed (controller tuning) and previously collected measurements observed with GUI's scope. If the process is changed, a picture of it is updated on the GUI screen. The user interface seen in Fig. 6 is implemented using Java and HTML.

In Fig. 7, a laboratory process is controlled and monitored with the scope of the MoCoNet platform. The process is simple, but it is difficult to control because of a variable network delay. It can be seen that multiple signals can be tracked at once and that e.g. the scale of the scope can be adjusted.

5. CONCLUSIONS

A platform for remote monitoring and controlling of an educational laboratory processes was described in this paper. This kind of a system enables teaching more students with fewer resources, since it is possible to do an entire experiment remotely without help from the teaching staff. With cooperation between universities, students can use a laboratory process located in a different university, even in a different country. Further study using the platform is needed to evaluate the efficiency of such distance learning when compared to traditional laboratory courses.

The architecture of the platform enables easy addition of new processes and simultaneous access of several users. A simple and versatile protocol was developed for communication between the components of the system. The platform can also be used for simulating and demonstrating different types of delays that occur in computer networks – a highly relevant topic in future control systems.

REFERENCES

Bolot, J. C. (1993). Characterizing End-to-End Packet Delay and Loss in the Internet. *Journal of High-Speed Networks*, **2**, pp. 289 – 298.

Hölttä, V. and Eriksson, L. (2003). Teaching controller design and system identification: A laboratory experiment with real-time control. In: *Proceedings of the Nordic MATLAB Conference 2003* (L. Gregersen, Ed.), pp. 121 – 126. Copenhagen, Denmark.

Hölttä, V. and Hyötyniemi, H. (2003). Computer-aided education: Experiences with MATLAB Web Server and Java. In: *Preprints of the IFAC Symposium ACE 2003* (J. Lindfors, Ed.), pp. 63 – 68, Oulu, Finland.

Koivo, H. N. and Reijonen, A. (2004). Tuning of PID Controllers for Varying Time-Delay Systems. In: *Proceedings of the IEEE International Conference on Mechatronics ICM'04*, Istanbul, Turkey.

Lincoln, B. and Bernhardsson, B. (2000). Optimal Control over Networks with Long Random Delays. In: *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems*. Perpignan, France.

Lucan, V., Simacek, P., Seppälä, J. and Koivisto, H. (2003). Bluetooth and Wireless LAN Applicability for Real-Time Control. Automaatio2003, Helsinki, Finland.

Mukherjee, A. (2001). On the Dynamics and Significance of Low Frequency Components of the Internet Load. In: *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement*, pp. 281 – 293, San Francisco, USA.

Riihimäki, P., Ylöstalo, T., Zenger, K. and Maasalo, V. (2003). A new self-study course on the web: Basic mathematics of control. In: *Preprints of the IFAC Symposium ACE 2003* (J. Lindfors, Ed.), pp. 149 – 153, Oulu, Finland.

Varso, J. and Koivo, H. N. (2003). Multivariable PI-control of a lamp system. In: *Preprints of the IFAC Symposium ACE 2003* (J. Lindfors, Ed.), pp. 299 – 304, Oulu, Finland.