

# An approach to distributed remote control based on middleware technology, MATLAB/Simulink - LabMap/LabNet framework

Cecil Bruce-Boye  
Fachhochschule Lübeck  
3 Stephensonstrasse  
Lübeck, 23562 Germany

Rüdiger zum Beck  
Fachhochschule Lübeck  
3 Stephensonstrasse  
Lübeck, 23562 Germany

Dmitry A. Kazakov  
cbb software GmbH  
1 Charlottenstrasse  
Lübeck, 23560 Germany

**Abstract-** In this paper we present an example of a distributed control system implementing state observer controller. Distribution is based on middleware approach. As the middleware for the hardware and network abstraction we used the LabMap/LabNet. For the controller platform we took MATLAB/Simulink. The approach illustrated in this work offers an outlook for the cases where computational resources are limited and distributed among several hardware components, especially in view on embedded application area.

## I. INTRODUCTION

Many control processes are not centralized at one particular location. The input and output operations are performed on different knots of the system distributed both physically and logically. Similarly the control activities can be bound not only to the locally connected inputs and outputs but also to the remote ones. Further they can be distributed as well. A middleware stretching through the whole system provides an access to distributed data, integrating the software and hardware components of the system. LabMap [1] represents an example of such bus middleware used in automotive area. LabMap provides abstraction of the application level from the hardware specific and decoupling the hardware interface modules from the application level. Another important advantage of LabMap is a smooth integration of numerous components with their variety of software and hardware protocols, supporting a component based software design [2, 3].

MATLAB/Simulink [4] is one of the most wide spread tools used for design, simulation, testing and final production of control systems. Though MATLAB/Simulink supports interaction with hardware in the loop, this requires relatively expensive plug-in cards. Furthermore the choice of the hardware one can communicate with it very limited.

LabMap interface to MATLAB/Simulink drastically enlarges the scope of application of this comfortable and popular engineering framework. The interface allows using MATLAB/Simulink in simulation mode, yet controlling hardware in real time. This requires no real-time workshop. The simulation time of MATLAB/Simulink is mapped to the real time by the interface.

As a software bus LabMap is natively networking. This opens a wide perspective for research and construction of controlling system on the fly, without even having any embedded target system in hand. Of course these opportunities come at some cost. In particular one cannot rely any more on the hard real time behavior of the system. In this paper we will show that soft real time might be an option.

## II. RELATED WORKS

The middleware architecture should respond to many, sometimes mutually exclusive, requirements. In the area of automation and control there exist numerous middleware products varying in their architecture.

OPC [7] is an initiative for open data connectivity. Similarly to LabMap OPC views data as variables. Though OPC is based on client-server architecture. Many hardware vendors provide OPC servers for their hardware. MATLAB OPC Toolbox 2 [8] offers an OPC client to Simulink, but it does not provide a server component essential for publishing the internal signals. A weakness of OPC is in relatively big overhead and its client-server architecture, which seems unsuitable for developing distributed controlling applications. Presently OPC is moving away from strict client-server approach by introducing OPC DX [11] for sever-to-server distribution of variables, but client-to-client communication is still impossible.

An alternative approach offers the MATLAB Distributed Computing Toolbox 2 [9], which though lacks Simulink support.

CORBA [10] is basically a software bus and does not suffer the problems of client-server approach, though it lacks description of timing constraints and is based on method invocation mechanism which does not well fit into MATLAB/Simulink framework.

## III. MIDDLEWARE ARCHITECTURE

The middleware architecture should respond to many, sometimes, mutually exclusive requirements. In the area of automation and control there exist various middleware products varying in their architecture. The OPC [7] is an initiative for open data connectivity. Similarly to LabMap OPC

views data as variables. The difference is though it has a client-server architecture. Though MATLAB OPC Toolbox 2 [8] offers an OPC client to Simulink, but it does not provide a server component essential for publishing the internal signals. In general a client-server architecture seems unsuitable for developing distributed controlling applications. An alternative approach offers the Distributed Computing Toolbox 2 [9], which though lacks Simulink support. CORBA [10] is basically a software bus and does not suffers the problems of client-server approach, though it lacks description of timing constraints and is based on method invocation mechanism which do not well fit to MATLAB/Simulink framework.

By an application the middleware LabMap is viewed as a set of variables. Each variable has a type, the current value and the current timestamp. No configuration is required. An application may access a value immediately after the system starts. Each variable responds to basic requests:

- *Get* reads the value of a variable in a safe way. It is guaranteed that the read value bits and the 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 source of the value and the policy used to actualize or obtain the value
- *Set* writes the value of a variable.
- *Request* queries a new value of a variable from underlying hardware. The actions necessary to undertake is the responsibility of the driver. Variable request is asynchronous and the application is not blocked until input / output completion.
- *Send* initiates writing the value on the underlying hardware. Like in the case of request the application is unaware of the actions the hardware undertakes upon send.

The middleware performs most of its operations asynchronously to the process invoking the operation. An I/O operation on the hardware is the most important example. To synchronize processes the middleware offers a variety of synchronization mechanisms:

- *Wait for I/O completion.* An application may enter time-limited waiting for completion of I/O involving a variable
- *Waiting for value change.* An application may enter time-limited waiting for the time moment the value of a variable is getting changed. This method is very often used for monitoring system state variables. The main advantage of this approach is that the application developer is relieved from the burden of polling the variable value and may concentrate on the domain objectives.
- *Blackboard.* Sometimes it is necessary to trace all changes of a variable. For instance an application visualizing signal waveforms would like to trace the signal. A usual approach for catching value changes involves some kind of notification mechanism between the source of the value and the application. Such point-to-point bias is hard to implement without overstraining the system resources and a danger of resource leaks when an application ends

abnormally. The software bus uses an alternative approach. All value changes are written onto the blackboard and remain for a certain time there. Any application may inspect the blackboard content in order to trace the changes of desired variables.

#### IV. NETWORK INTERFACE

The software bus provides abstraction mechanisms hiding the distributed nature of the bus from the applications and their components. Further the software bus itself abstracts networking as a hardware interface. LabMap provides a highly optimized implementation of the networking interface called

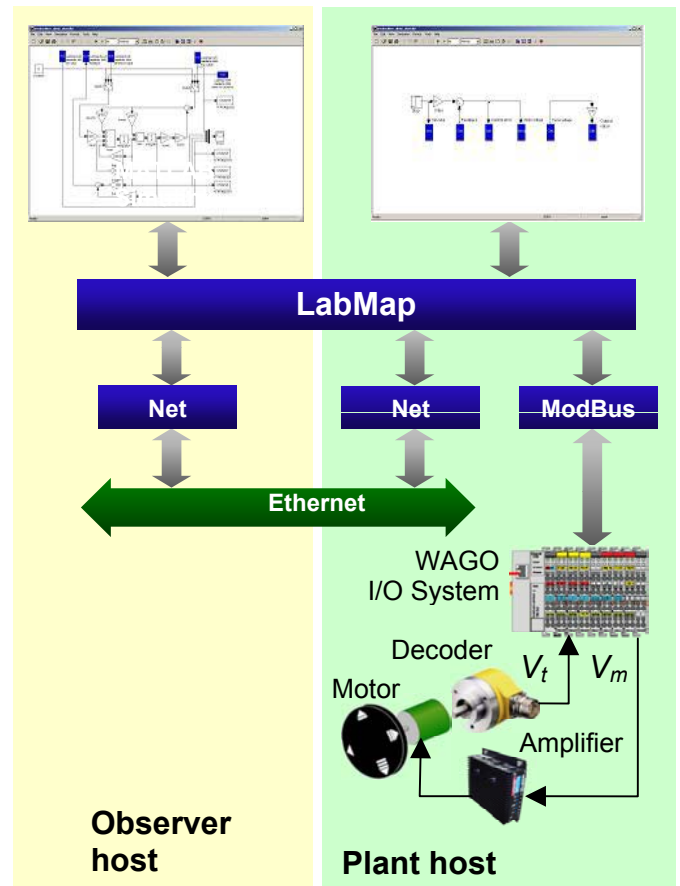


Fig 1. System setup

LabNet. LabNet corresponds to the presentation and application levels (6-7) within the OSI model. LabNet itself may function over different transport layers, but its most compelling use is based on Ethernet TCP/IP sockets.

What are the features of the networking interface LabNet, which make it attractive for distributed control?

- LabNet is fully transparent for applications using it.
- Data synchronization between distributed partitions is maintained in background, invisibly for applications.
- Middleware requests, such as I/O initialization etc are transported by LabNet and executed on the remote partitions

as if the hardware they control were directly connected to the local host.

- Measurement units are consistently handled, not compromising an ability to deal with different though compatible unit systems on different network hosts.
- An application may ensure that no other application accesses data for write. Further local and global access variables are supported. Local variables are seen by the applications running on the same network host where the variable is declared. Global variables are accessible from potentially any host. Access operations on global variables are also global.
- Any value is provided with a time stamp. A hardware driver is responsible for supplying correct time stamps. An application may figure out that the values of several are consistent by inspecting their time stamps.
- A peer-to-peer time synchronization mechanism is available to ensure consistency of the time stamps. Time stamps are transparently translated from one side clock reading to another. This feature can be used when there is no way to synchronize local clocks. At the same time any external time

synchronization, such as NTP, can be used.

- Local variables can be browsed from remote hosts.

### V. DISTRIBUTED STATE OBSERVER CONTROLLER SETUP

Fig 1. represents the test system we used for the controller. It consists of two hosts running MATLAB/Simulink. The hosts are connected via Ethernet. One of them has the WAGO I/O System [6] attached. It is used to acquire the revolution (via digital encoder HEDL 5540 A12) and set the voltage for the amplifier (4-Q-DC LSC 30/2) of an electric motor (DC Maxon A-max 32). The hosts run LabMap. This allows them to exchange data. LabMap natively provides time stamping for all values. The network interface of LabMap is shown as “Net” on the figure. It has a peer-to-peer time synchronization mechanism to ensure consistency of the time stamps. This does not require any synchronization of the partition’s clocks, because the time stamps are translated from one clock reading to other transparently. WAGO I/O System is attached using the ModBus interface of LabMap.

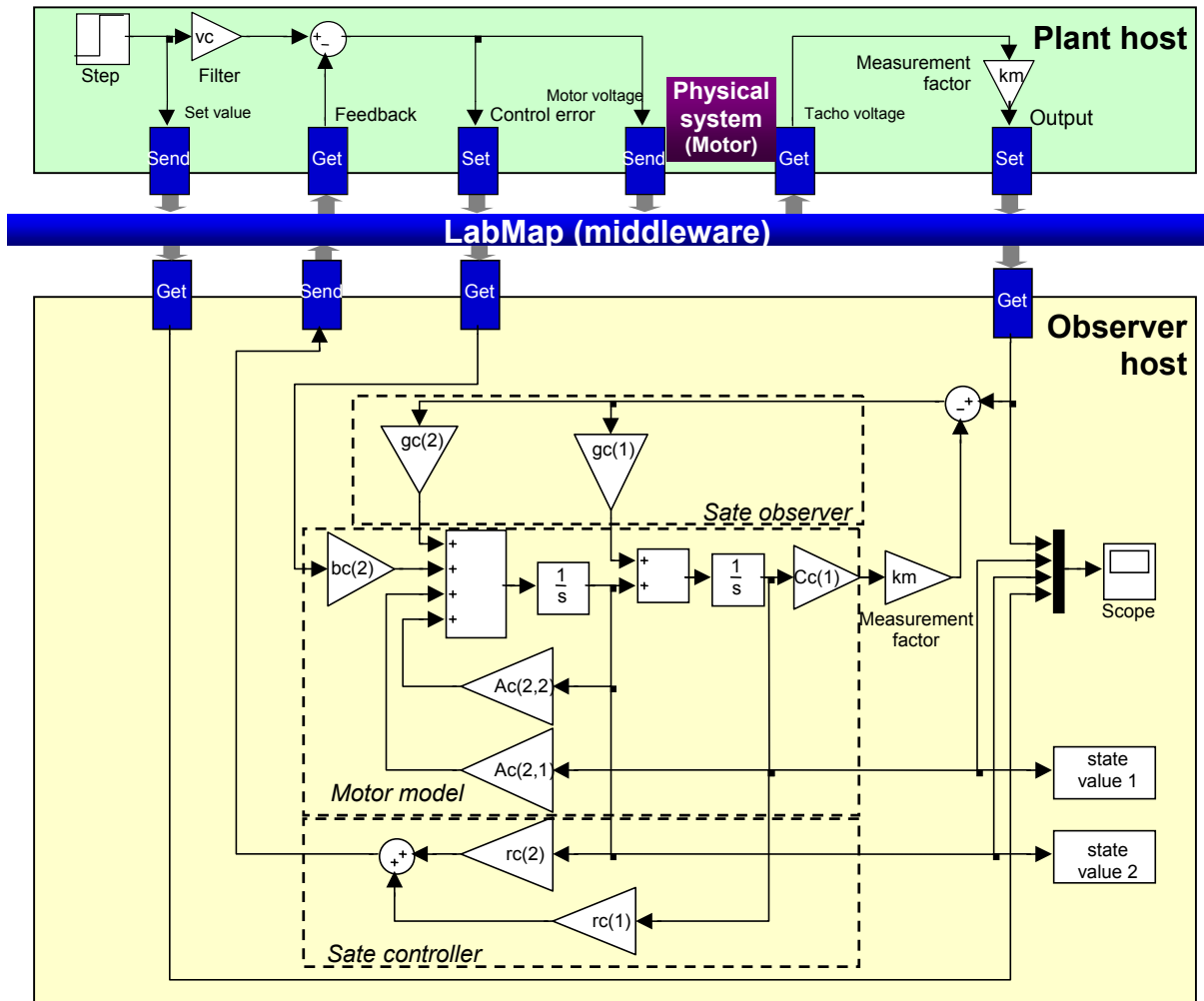


Fig. 2. MATLAB / Simulink model. Parts of the model running on different hosts shown on differently colored backgrounds

## VI. STATE OBSERVER AND CONTROLLER

For the test system shown on fig 1. we chose a Luenberger state observer controller [5]. The plant part and the observer part are running on two separate network hosts. Each host is a conventional PC. Fig 2. shows the controller outline. The state observer  $gc$  and the state controller  $rc$  are calculated by the Ackermann formula. In order to ensure input – output accuracy the filter  $vc$  is calculated with respect to the state controller  $rc$ , shown in the following MATLAB m-file:

```
ObserverCalc.m
%calculation of state space controller and
%luenberger observer
%the system identificatin was carried out
%by the system identification
%software toolbox IDICON
num = [3.2637] %result from system identification
den = [1 10.862 11.205] %frequency domain
Ac=[0 1; -den(3) -den(2)]; %state spaces discription
bc=[0;num];
Cc=[1 0];

km=den(3)/num %measurement factor
cm=km*Cc;
p=[-10 -20]; %pole placement observer
gc=(acker(Ac',Cc',p))'; %state observer
gc=[19.138; -19.082]
prc = [-2 -2.5]; %pole placement controller
rc=acker(Ac,bc,prc); %state controller
rc=[-1.9013 -1.9493]
vc=inv(cm*inv(bc*rc-Ac)*bc);%filter vc=0.44622
```

Fig 3. shows the control revolution, output values and two state variables. As seen from the figure the control performance is satisfactory despite distribution of the controller over the network and consequent delays imposed by the data transfer of the network.

Our experiments with the system showed robust

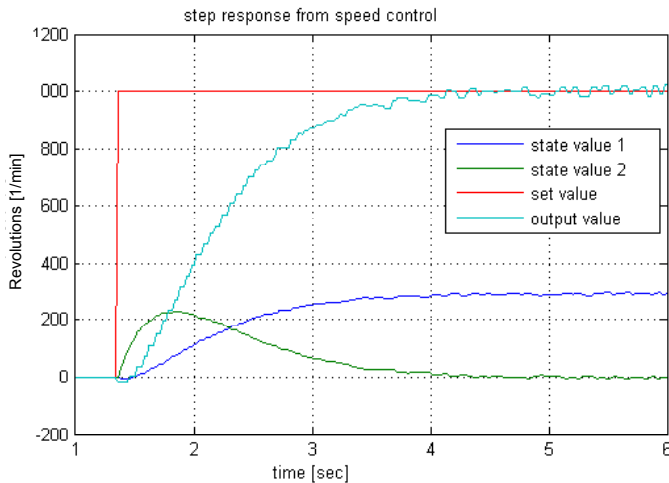


Fig. 3. State-controller behavior

characteristic of the distributed controller against external disturbances.

In fig 4. the open-loop control and the close-loop control by the previously discussed state-controller with observer is illustrated. In the first 5.5s the DC-motor is running in an open-

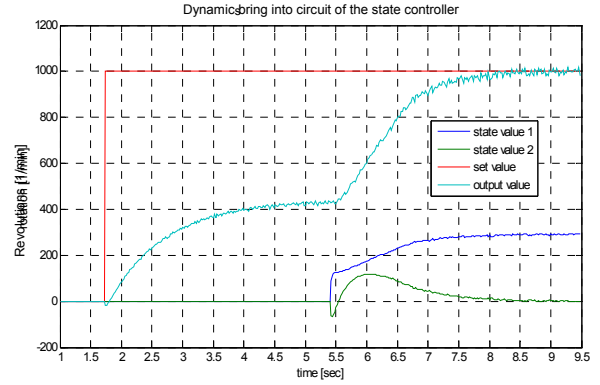


Fig. 4. Open-loop and close-loop performance

loop mode without any feedback signal. Instead of running with the reference value of 1000 revolutions per minute the output is approximately 400 revolutions per minute. After switching the state controller into the circuit the close-loop performance is satisfactory, i.e. the output value matches the reference value.

## VII. CONCLUSION

The approach illustrated in this work shows feasibility of distributed control for the cases where computational resources could be limited and distributed among several hardware components. In such cases it is essential to be able to balance the resources to ensure meeting functional and non-functional requirements of control system, especially in embedded application area.

On the other side, the approach provides a framework for engineers accustomed to MATLAB/Simulink, who might wish to use the hardware unsupported directly by the MATLAB/Simulink or simulated by the software.

LabMap provides an interface to decouple the control application development from the hardware layer. A distributed application communicates with the hardware through and its partitions via variables and the middleware carries out the actual sending of the data through the appropriate protocol and interfaces for the given hardware.

In our future work we plan to investigate applicability of the approach for achieving transferability of the distributed controlling components.

The presented framework is successfully applied for educational use in the control laboratories of the Universities of Applied Sciences in Luebeck and Bremen [12]. It allows to share laboratory resources between groups of students. Students can design their controller on their own observer hosts offline. After completing their task they gain access to the laboratory plant host through the middleware as shown in fig.1 to perform experiments.

## REFERENCES

- [1] <http://www.cbb-software.com/labmap.html>

- [2] C. Bruce-Boye, D.Kazakov, A. Fechner, "The hard and soft option LabMap the ultimate auto test platform", Testing Technology International. May 2001
- [3] 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
- [4] M. Schetzen, V.K. Ingle, "Discrete systems laboratory in MATLAB," Thompson Engineering, 2000
- [5] D.Luenberger, "An introduction to observers." IEEE Trans. Automatic Control, AC-16 1971
- [6] <http://www.wago.com>
- [7] F. Iwanitz, J. Lange "OPC - Fundamentals, Implementation and Application", 2002
- [8] MATLAB OPC Toolbox 2 <http://www.mathworks.com>
- [9] MATLAB Distributed Computing Toolbox 2, <http://www.mathworks.com>
- [10] "The Common Object Request Broker: Architecture and Specification", OMG Document 99-10
- [11] "OPC DX 1.00 Specification 2003-03-11"
- [12] H.-W. Philippsen „Einstieg in die Regelungstechnik“, 2003, ISBN 3-446-22377-0