Interconnecting Heterogeneous Fieldbuses in Wide Area Industrial Environments

C. TRANORIS, K. THRAMBOULIDIS
Electrical & Computer Engineering Dept.,
University of Patras, 265 00 Patras,
GREECE
e-mail: {tranoris, thrambo}@ee.upatras.gr

Abstract: - The Industrial Process Measurement and Control Systems (IPMCS) architect has a plethora of fieldbuses to select. Many different proprietary fieldbuses, each one supporting different standards and targeting particular market requirements, have been installed in the industry in different time periods under different circumstances. There is an always-growing need to integrate the control applications of the above fieldbuses making interoperability in device and fieldbus level as well as real-timelines the key issues in the area of IPMCSs. In this paper we suggest an approach towards interoperability in the fieldbus level. This approach, that would make possible the interconnection of two or more distant fieldbuses over a common backbone, comes to assist process and systems engineers in the development and configuration of distributed IPMCSs. We also present the first step towards implementation, which was defined to include the interconnection of two fieldbuses of the same type. We chose as a case study PROFIBUS due to its popularity and wide acceptance.

Index Terms:-Fieldbus, Interoperability, Interconnection, distributed IPMCS, Profibus

1 Introduction

During the development of industrial automation and process control information systems (IPMCS), the engineer has to consider computer networks since they are an essential component of their architecture. Field buses have emerged as an effective way to overcome problems that prevent the accurate signal transmission such as electromagnetic pollution at the factory floor and high current switches. Fieldbus is an intelligent/smart transmitter protocol, which ensures very reliable and self-correcting data communication, provides control, alarm, trend and other services distributed to its devices. With its deterministic nature, fieldbus lets system architects distribute a process over the network even for applications with hard real-time requirements. A wide variety of fieldbus networks are available in the market, each one with its own special capabilities on the different levels of the hierarchical structure of Industrial Manufacturing [1].

The IPMCS architect has a plethora of fieldbuses to select. These come from many vendors, support different standards and target particular market requirements around the world. Among the most famous fieldbuses are PROFIBUS, Fiedbus Foundation, CAN, WorldFIP, P-NET and LonWorks. In spite of the technological advances, Industrial automation and process control industries lack the harmony that other areas of industry have. The final effort for an interoperable standard among the fieldbuses, which was carried by the ISA'S SP50 standards committee, ended last year and resulted in an 8-standards standard [2].

Currently, the standards that are promoted from different vendors, are International Electrotechnical Commission's IEC-61158 (supported from SP50), and Cenelec's EN-50170. Fieldbus foundation promotes 61158 but also other major vendors are interested including ABB, Honeywell, Siemens and Yokogawa. Lining up against 61158 is the Profibus International organization, whose technology originates in Germany. Senior Profibus vendors include Eurotherm, Mitsubishi and Siemens. Profibus has a large installed base with over 200,000 application mostly in Europe. Cenelec's EN-50170 embrace Profibus, P-NET and France's WorldFIP, but the mess already is apparent with Profibus-PA to be compliant with both 61158-2 and 50170!

However, the end users in the industrial automation business don't care about all this political disagreement and wrestling for market share. They

have a clear need for interoperability and they want proven solutions that can be deployed economically and maintained well into the future. Ideally these solutions would also be truly interoperable- the point of having any standard- but today's reality dictates otherwise[3].

In this paper we suggest an approach for the interoperability in the fieldbus level, that would make possible the interconnection of two or more distant fieldbuses over a common backbone. Our approach comes to assist process and systems engineers in the development and configuration of IPMCSs. During the design we had in mind and examined several issues such as different industrial buses, current standards, network architectures, industrial technologies and real-time constraints.

This paper is organized as follows: The second section examines industrial applications and introduces the function blocks from IEC 61499. The third section examines current status and solutions for interoperability in the fieldbus level. Next, the fourth section, presents our approach to Interoperability. We examine the Network topology, the interworking unit and the definition of a Virtual Fieldbus. The fifth section describes our prototype framework implementation, the equipment, technologies used and the difficulties we confronted for this first implementation.We close with our conclusions.

2 Interoperability in the fieldbus level

What is meant by interoperability in the fieldbus level, is the ability of interconnecting independent fieldbus segments, even from different vendors, in the enterprise's intranet backbone. For example, if an enterprise wants to transfer data among three buildings where every one has its own fieldbus, it has to interconnect the three heterogeneous fieldbuses. It is also a requirement from the corporate level to have monitor and control of these buses. The solution is to use interworking units between each fieldbus and the enterprise intranet, leaving unchanged in this way the already defined process of each bus, and transferring only the necessary values through the intranet. This would also result, in the least effort that the enterprise should spend over the configuration of the new process. Here, though, the most dominant requirement is the real-time transmission of data between the fieldbuses, how efficiently the interworking units can handle such requests and how the underlying intranet can provide the required quality of service. With the availability of cheap memory and processing power, it is far more practical to make a whole range of devices work with any bus through an interworking unit, without the need for converting the entire product line to operate with an “alien” bus and all its attendant hidden proprietary features. Interworking units will theoretically appear to have intrinsic delays and performance limitations - but practically, can be made to operate with very little disadvantage.[1]

Industry's solution to this problem is not very clear. For example, Softing's Profigate is an Ethernet-Profibus Gateway which consists of a PROFIBUS connector, CPU board and an Ethernet TCP/IP software interface[4]. The extra software that is offered for the access to the gateway, can be used for telemonitoring, remote diagnosis and remote access and it could also be used for interconnecting two or more PROFIBUSes. The same company also offers an OPC server for open connectivity. Another example is Siemens' SIMATIC OPC SERVER[5], which offers libraries for distant connectivity. Two major reasons, though, put these solutions in question: first the real time transmission between two fieldbuses, due to the Ethernet backbone(CSMA/CD), cannot be guaranteed and second these gateways offer only PROFIBUS connections, except OPC connectivity through a client. Another commercial product that promises connectivity between fieldbuses is SST-X-Link from SST Network gateways[6]. This product, a combination of hardware and software, is based on a custom kernel. It seems to provide connectivity between two independent fieldbuses, provided that the user has certain drivers for each network. Although this product seems promising, needs great effort from user's part for configuring networks, cards, etc, and additionally it doesn't offer the capability for remote configuring, diagnostics, monitoring, etc.

Academic approach to this problem is limited. Recent paperwork focuses on issues for interoperability on the device level, Java devices, making Ethernet real-time, OPC and remote administration, fieldbus standardization etc. [7][8][9]

3 Industrial Applications

The latest trends in the development of control systems are concentrated in the development of the

evolving international standard IEC 61499 on function blocks for IPMCSs. The standard is especially dedicated to the software reusability issue, modularity, encapsulation and inheritance. It follows the previous standard on programming languages for programmable logic controllers IEC1131-3 extending it to the new requirements of distributed control systems.
According to the voting draft of IEC61499 [10], the general concept of a distributed system is displayed in figure 1. *Devices* may communicate with each other over one or more communication links, and may interface to controlled machines and processes. The a*pplications* may be distributed among one or more devices.

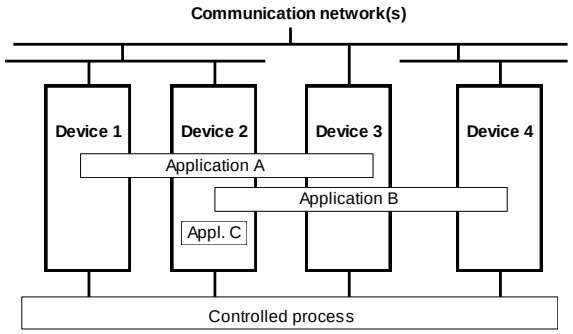

Figure 1. Distributed System Model

An application can be distributed among several *resources* in the same or different *devices*. A *resource* uses the causal relationships specified by the application to determine the appropriate responses to *events* which may arise from communication and process interfaces or from other functions of the resource.
An application consists of one or more function block *instances*, interconnected by *event connections* and *data connections* as shown in figure 2. The function block instances may be distributed among devices. Functionality of a function block is provided by means of algorithms, which process input and internal data and generate output data. Figure 3 shows a general diagram of a function block. The block consists of a head and a body, where the head is connected to the event flow and the body to the data flow.

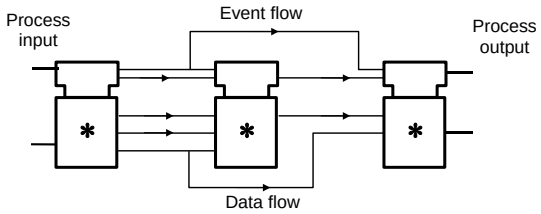

Figure 2. The Function Block model of an Application

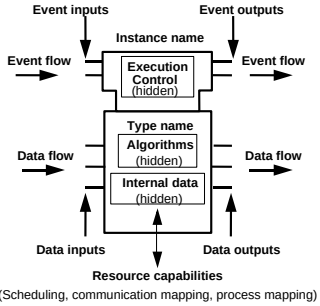

Figure 3. The structure of a Function Block

4 Our approach to Interoperability

Our approach to the interoperability in the fieldbus level, suggests a framework that would make possible the interconnection of two or more distant fieldbuses over a common backbone. What we want to achieve is the distribution of an application beyond the range of a single fieldbus network even in great distances as is shown in Figure 4.

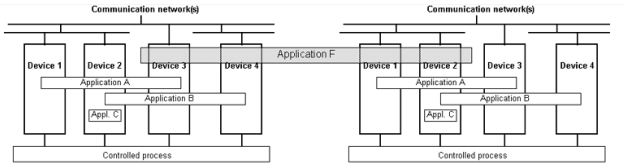

Figure 4 - Distributing an application over distant fieldbuses

For the design of this framework, we examined a series of issues:

- different industrial fieldbuses like PROFIBUS, Lonworks and WorldFIP. Their topologies, operation and configuration,
- fieldbus devices such as Gateways, Master/Slaves, PLCs, Lon Nodes, etc, that can be connected to such networks
- network architectures such as Gigabit Ethernet, ATM, switched Fast Ethernet , topologies, and examination of which would be the optimal for playing the role of the backbone
- technologies like OLE for Process Control, Jini, CORBA, RMI, that we could use for the design, configuration and monitoring of the whole process.

The proposed framework consists of the following:

- A network architecture
- An interworking unit
- The definition of a Virtual Fieldbus

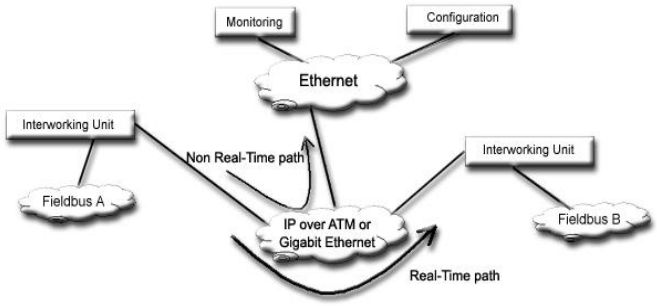

Figure 5. Network Architecture

Figure 5, shows the proposed network architecture. The two fieldbus networks A and B are connected through an interworking unit to the ATM backbone where the enterprise's Ethernet is also attached. Packages can be transmitted from one fieldbus to the other, through the real-time path. Non real-time packages that has to do with monitoring, trending and configuration of the interworking units are transmitted by means of the non real-time path. Generally the operation is the following: A value from a device will be encoded based on some rules from the propriety representation (integer, double, etc) to a more generic representation (temperature, pressure), processed, routed and transmitted by means of IP over ATM, to an alien fieldbus.

A strong examination of several network technologies, for the backbone, brought ATM to be the most attractive one. Although ATM has lost the battle for the desktop, is the only technology today, that can comprehensively handle the QoS routing, essential to supporting traffic engineering and maintaining service quality in large networks. A possible solution could be Gigabit Ethernet for its low cost and Ethernet 's wide acceptance, but since it's a new technology we avoided it[11]. Additionally, ATM proved to be a good solution for interconnecting fieldbuses[12] and we had already a description of how to install such a network under Linux[13]. Another promising solution that is currently under examination is switched Fast-Ethernet.

Figure 6 shows the interworking unit's high level architecture. The interworking unit is capable of accepting an autonomous "island" of fieldbus devices and interfacing it to the enterprise backbone, making possible the connection to other autonomous fieldbus "islands". By autonomous fieldbus we mean that this fieldbus is already operating, and the application process has now the requirement that some values must be transmitted to a distant fieldbus. This interworking unit is remotely configurable and gives the capability of remote monitoring and trending. The overall design must meet the hard real-time constraints of such environment.

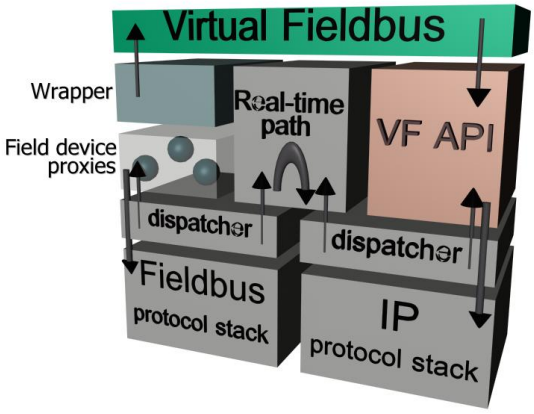

Figure 6. Architecture of our interworking unit.

To provide interoperability in the field bus level a Virtual Fieldbus (VF) must be defined. This VF will allow:
the interconnection of field bus device islands of the various commercial field buses,
the similes interface to the enterprise intranet and
the development of a field bus independent engineering tools.
An object-oriented API is provided to support the development of the engineering tool as well as to allow the connection of SCADA client systems. Since the most of the commercial SCADA systems have adopted the OPC interface we decided to provide an OPC compliant interface. From the view point of developing independent engineering tools the VF may be considered as an aggregation of device proxies, industrial process parameters and application specific parameters.

5 A Prototype implementation

5.1 General

Here we present the first step towards implementation, which was defined to include the interconnection of two fieldbuses of the same type. We chose PROFIBUS as a case study due to its popularity and wide acceptance. The most related work, towards interoperability in the fieldbus level, comes in [13]. The author examines the

interconnection of two or more PROFIBUS "islands", interconnected over an ATM backbone and presents the operation of the network in a simulation tool.

PROFIBUS is a vendor-independent, open field bus standard for a wide range of applications in manufacturing and process automation. PROFIBUS is a multi-master system and thus allows the joint operation of several automation, engineering or visualization systems with their distributed peripherals on one bus. PROFIBUS distinguishes between the following types of field device:
a)Master device. This type of device controls the data communication on the bus. It sends messages without an external request when it holds the bus access rights (the token). Masters are also called active stations and are devices such as PLCs, PCs, etc.
b)Slave device. Slave devices do not have bus access rights and they can only acknowledge received messages or send messages to the master when requested to do so. Slaves are called passive stations and are peripherals such as I/O devices, valves, drivers and measuring transducers.

For a PROFIBUS system to be installed, its devices must be initialized with global parameters, i.e., station address, token target rotation time and control timers. A connection list, residing on each master station, contains its pre-defined communication relations to other devices, including both masters and slaves. This allows for a deterministic behavior of PROFIBUS. During a message cycle, a slave responds first in a diagnostic package request, then the master sends the data request and the slave responds with the actual information.

For the implementation of the interworking unit we used a Pentium based PC running RTLinux. The PC was equipped with a PROFIBUS master card and IBM's Turboways 25 Mbps ATM adapter and it was connected to an IBM's 8285 ATM switch. Fortunately, drivers were available under Linux for all our hardware. As PROFIBUS card we selected Softing's PROFIBOARD, that offers a dualport RAM as an interface mechanism to the processing unit.

Linux is a freely distributed, 32-bit, pre-emptive multi-tasking, multi-user operating system. Once developed for educational purposes it has become an attractive counterpart to its commercial competitors. Nowadays, Linux has been expanded from a simple UNIX kernel to a huge suite of applications and utilities developed by means of GNU, C/C++, JDK, TCP/IP and X-Windows. RTLinux is a hard real-time operating system following the POSIX specification for RTOSes. It handles time-critical tasks and runs Linux as its lowest priority execution thread. In RTLinux, a small *hard-realtime* kernel shares one or more processors with standard Linux. This allows the system to run accurately timed applications to perform data acquisition and systems control, while still serving as a standard Linux workstation [14]. Worst case between an interrupt being asserted and a realtime handler starting is about 15 μsecs for a generic x86. Interprocess communication is provided through real-time FIFOs or via shared memory. We chose RTLinux because of its stability, networking via Linux, user community, extensive tools and price. It's a solution that offers the advantages of real time OS on a low-end Pentium, close in price, with low maintenance and not a special software/hardware.

5.2 Interworking Unit's functionality

Figure 7, shows two PROFIBUSes interconnected through an Inteworking Unit. ATM is used as the network backbone where IP packages will be exchanged between the Interworking units. The Interworking Unit is acting as another master on the existing network. During PROFIBUS configuration, we should assign the slaves (read/write operations) that the Interworking Unit has access.

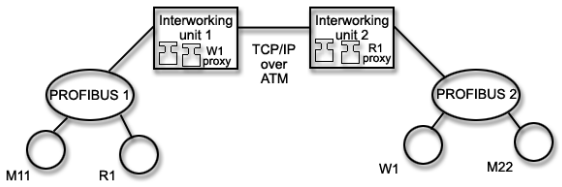

Figure 7. Interconnection of two PROFIBUS networks

As an example, we consider the data items from slave R1 must be processed and written to slave W1 (see figure 3). To accomplish this, we must assign R1 slave to interworking unit 1, for reading and W1 slave of PROFIBUS2 to interworking unit 2 for writing. In PROFIBUS1 the token is shared between the two masters i.e the M11 master and the interworking unit 1 that acts as a master. When interworking unit 1 acquires the token it reads the data item from slave R1. In more detail: the master card on the interworking unit reads the data item

from slave R1; a task running on the interworking unit will get, through card's API, the data item and write it to W1's proxy on interworking unit 1. The proxy then is responsible for the transmission and will use the routing functionality of our interworking unit to send the value to the correct destination. On interworking unit 2 similar things happen: When the token comes to interworking unit 2, the interworking unit should write the data item to W1.
The following calculation, describes the times for a data item to be read from slave R1 until the write to slave W1 take place. We use the terms that have been introduced in [15].

Interworking Unit with master functionality

$T_{TOTAL}=$	Read of Inteeerworking Unit from slave R1
$T_{MCread}+$	
T_{BD+}	Bridge Delay
$T_{TDIP}+$	IP transfer delay
$T_{BD}+$	Bridge Delay
$T_{TR}+$	Token rotation (until token comes to Interworkin Unit)
$T_{MCwrite}$	Write To Slave

The proxy pattern is described as one which "decouples clients from their servers by creating a local proxy, or stand-in, for the less accessible server. When the client needs to request a service from the server, such as retrieving a value, it asks its local proxy. The proxy can then marshal the request to the original server…." One use for this pattern in the IEC 61499 context is to utilize *service interface function blocks (SIFBs)* as proxies for functions of devices that may not themselves be compliant to the standard[16].

6 Summary

Currently, there is a great interest in the automation industry for distributed architectures, especially for these that give solutions to the problem of interoperability. In this paper, we presented our approach for interoperability and a first implementation of interconnecting two independent fieldbus segments. Our approach attempts to facilitate the interconnection of two or more distant fieldbuses over a common backbone. We have examined the network architecture, the interworking unit and we defined a virtual fieldbus. We proceeded a step further and made a prototype implementation, by interconnecting two PROFIBUS fieldbuses over ATM. We used new technologies such as RT-Linux and examined how they perform in such environments. We are currently planning to use switched Fast Ethernet instead of ATM that results in a more simple and low cost communication system.

Acknowledgments
This work has been funded in part by the Greek General Secretariat for Research and Technology in the context of the ARTIO project. We gratefully thank Chris Koulamas and Stefanos Aslanis, members of the ARTIO development team, for their helpful discussions.

References

[1] Jim Pinto, *Fieldbus - Conflicting "Standards" Emerge, but Interoperability is Still Elusive* , Design Engineering magazine, UK, Oct.99
[2] Jim Pinto, *The great Fieldbus debate is Over!*, proceedings of Industrial Controls Intelligence Nov.1999
[3] David March, *Surviving the fieldbus wars*, EDN Magazine, April 1999
[4] Softing's Profigate Datasheet, http://www.softing.com
[5] SIMATIC OPC Server, User guide http://www.siemens.com
[6] SST-X-Link gateway, http://www.mysst.com
[7] Marti P. et al.,*A java -based framework for distributed supervision and control of industrial processes*, Proceedings ETFA'99
[8] Pritty D.W. et al., *A real-time upgrade for Ethernet based factory networking*, Proceedings IECON 1995
[9] Thomas Lumpp et al.,*Virtual Java devices. Integration of fieldbus based systems in the Internet*, Proceedings IECON 1998
[10] IEC61499
[11] Riezenman, M.J.; Sweet, W, *Technology 1999, Analysis & Forecast, Communications*, IEEE Spectrum, Jan 1999
[12] Cseh, C. et al., *ATM networks for factory communication*, Proceedings ETFA '99
[13] Jepsen, T.C.et al., *Linux Update: An experimental ATM Network*, IEEE Software, Sept/Oct 1999
[14] Victor Yodaiken, *Real-time Linux, A short position paper*, http://www.rtlinux.org
[15] O. Künert, *Kopplung von PROFIBUS - Systemen über ATM*, Dissertation, 17/12/1999 (Connection of PROFIBUS systems over ATM)
[16] J. H. Christensen, *Basic Concepts of IEC 61499*, Fachtagung Verteilte Automatisierung, Magdeburg DE, March 2000