Experience with LonWorks as a Fieldbus for the Light Source ANKA

B. Jeram, M. Juras, G. Mavric, M. Plesko, M. Smolej
J. Stefan Institute, P.O. Box 3000, 1001 Ljubljana, Slovenia
e-mail: mark.plesko@ijs.si

Abstract

ANKA[1] is a 2.5 GeV synchrotron radiation light source being built in Karlsruhe, Germany. The control system of ANKA is based on the standard model with a few modifications[2]: Instead of employing VME, we use a field bus network with intelligent nodes to connect the individual devices directly to PCs. Those can serve as consoles or as the middle layer of the control system.

Since a light source is a relatively static machine, there is no need for hard real-time control. Therefore we have decided to opt for the LonWorks[3] fieldbus, because it offers a complete network system in hardware and software in a single micro-controller (the Neuron chip) with all the necessary development and network management tools. LonWorks already implement layers 1-6 of the ISO/OSI model and thus transparently connect controlled devices and control room consoles without any need for network programming.

In this paper we report of experiences with LonWorks on a working prototype consisting of 10 nodes, each equipped with a Neuron chip.

1 Introduction

Exploiting the fact that over 30 drivers for different I/O models already exist in the firmware of the Neuron chip, e.g. digital, byte, serial, microwire, counter/timer, etc., we have designed and built in short time several modular I/O boards that connect to the Neuron chip:

- 8-channel 12-bit ADC
- 8 digital in
- 8 digital out
- RS-232 interface

and interfaced a Neuron chip to a commercial 16-bit DAC/ADC board.

The prototype implements the following functions in the nodes:

- complete magnet power supply control including state machine and alarms
- synchronous ramping of several power supplies where their current is increased bit-by-bit in 1 ms steps
- knobs: fine tuning with physical and virtual knobs through the LonWorks network generating over 25 actions/display pairs per second
- Vacuum pump, gauge and valve control and interlock
- beam position monitor acquisition and averaging

Due to the OLE and DLL interfaces existing for LonWorks, we are able to access the variables and structures used in the nodes directly at the PC console. Several applications to control and monitor the nodes were written using rapid application development tools like Visual Basic and Delphi. Through OLE, controlled variables can be accessed easily even from Excel. This makes local control of devices very easy, as any laptop PC can be just plugged to the LonWorks network as the need arises.

However, an accelerator control system should be reasonably scalable, therefore we use the OLE/DLL interface to write device servers [4] on the middle layer of the 3-layer standard model of ANKA's control system. As opposed to most control systems which use VME and a real-time operating system, our device servers run on PCs under Windows NT.

2 The elements of LonWorks

There are a number of control network technologies available today. None of them has emerged as a leading standard yet, which makes the choice very difficult. Many promise interoperability and plug-and-play in the close future. However, such promises have been given in the past ten years and never delivered. A notably exception is the LonWorks technology, developed and marketed by Echelon Corporation, which delivers both interoperability and plug-and-play today. There are currently around 2600 companies developing LonWorks into their products. With over 2.5 million installed nodes world-wide, LonWorks have one of the largest installed bases.

Several documents describing the basics of LonWorks technology are easily accessible over the World Wide Web[5]. Here, just a brief overview can be given. LonWorks comprises four elements:

2.1 LonTalk network protocol

LonTalk implements layers 1-6 of the ISO/OSI model and thus hides all details of the field bus protocol. Different network speeds and media are supported, among others also twisted pair wires. The protocol supports communication based on request/response or events, direct node-to-node communication, practically unlimited number of nodes, unicast, multicast and broadcast, message acknowledgement, authentication and message priorities.

Normally, network communication is done through network variables. A network variable is a data object that can be accessed by several nodes transparently through the network. Whenever the application program running on a given node writes into one of its output network variables, the new value is propagated across the network to all the nodes connected to that network variable.

2.2 The neuron chip

Each LonWorks node is controlled by the Neuron chip. It contains three microprocessors, LonTalk firmware and I/O channels all in one VLSI chip at a price of $5. The Neuron chip can be seen as an 8-bit micro controller with additional processors that handle network communications and I/O. Over 30 drivers for different I/O models exist in firmware, like digital, byte, serial, microwire, counter/timer, etc.. There are several versions of the Neuron chip available, the most powerful running at 10 MHz, allowing up to 64 kbyte external RAM for the firmware (16kbytes), application programs and data.

2.3 Neuron C

The Neuron chip applications are programmed with the high level language Neuron C. Being a subset of ANSI C, it adds several features that support control and communications such that assembler programming is necessary only for time critical applications. With Neuron C one can attach variables directly to I/O pins, use network variables and communicate through the I/O pins with high level protocols. Events are handled naturally with the when clause. Quasi real-time response is provided with the use of built-in timers.

2.4 The development environment and network management tools

Echelon provides a powerful cross-development and debugging environment for LonWorks. Hosted on a PC it allows the development and step-by-step debugging of single and multiple node applications as well as the communication between them. Complete network manage-ment functions are provided that install and configure nodes and monitor network activity down to each single packet. Network management with PCs creates both a dynamic database for network variable linking and a disk-based database for node and network link related data. Program downloading and invocation over the network is also possible.

A library of application programming interfaces (API) for PCs for network programming, management and control is also available. With the API library, Windows-based control applications can be built, exploiting both the convenience of LonWorks technology and the power of graphical user interfaces.

The whole picture is rounded up with excellent documentation and a long list of freely available tech notes and program examples.

2.5 Limits of LonWorks

The first target markets for LonWorks were building automation and large scale manufacturing facilities. For this reason, network throughput speed and Neuron performance have been sacrificed for long distance networking, reliability, security and low cost. Those design decisions show up as poor performances of the Neuron chip. Note that - as opposed to CAN or Profibus measurements, where network controller speeds are published - the following figures denote the time it takes that a message from one application reaches and is *processed* by another applications.

- an unacknowledged packet takes 3-7 ms;
- an acknowledged round-trip takes 15 ms;
- a when clause (event handling) takes 1 ms;
- a Neuron chip can transmit a maximal number of 100 acknowledged packets per second (a packet size can be up to 226 bytes).

LonWorks are thus not adequate for

- CPU-intensive tasks
- fast transfer of large data sizes
- sub-millisecond real-time operations

The network protocol, running at 1.25 Mbit/s, is more powerful, allowing a sustained rate of up to 600-800 packets per second, depending on the packet size.

3 The prototype built with LonWorks

First some tests were run at the storage ring ELSA of the University of Bonn, mainly to check the robustness of the protocol against noise but also to see how fast simple electronics and programs can be built around the Neuron chip. Based on those results, a functioning prototype has been built in order to see where the real problems lie. The discussion on the results and achievements follows. See figure 1 for the layout of the prototype.

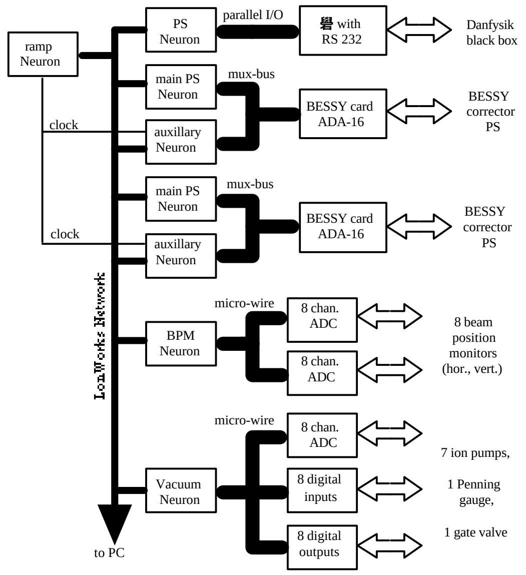

Figure 1: the layout of the fieldbus and I/O hardware of the prototype

3.1 Noise immunity

A small network was set up, such that a node was on a twisted pair cable about 30 m away from the others. Different situations were studied (see table 1), next to noisiest source in the ring (a kicker), under heavy network load and under unrealistically bad conditions (the network cable was hung around a 50 kV supply and looped in front a klystron). Although errors on the network such as collisions, timeouts and CRC errors were seen, not a single packet got lost, as the protocol automatically tries to re-send packets and compose messages.

3.2 The I/O combo

Originally planned as a multipurpose ADC/digital I/O board, this combo now consists of a backplane with up to 9 slots – one for the Neuron board and 8 for a mixture of the following half length 3U sized cards:

- an 8 channel digital TTL input
- an 8 channel digital TTL output
- an 8 channel 12-bit ADC

The Neuron board is a motherboard that takes "piggyback" the 1.25 Mbit/s transceiver and the LTM-10 module[6]. This module is a small board bought from Echelon that contains the Neuron chip, 32k RAM and 32k flash memory. All Neuron boards in the prototype are of this kind.

All boards are protected through optocouplers. Here, the serial microwire (or SPI) interface is made use of. The Neuron already supports the SPI protocol, so we just had to buy an ADC which supports this protocol.

The digital I/O are designed such that 8 signals together form one byte - the unit of communication on the microwire. The microwire interface allows addressing of up to 8 devices, therefore we can mix an arbitrary number of ADCs, digital I/Os and DACs - the DAC board is not part of the prototype and will be done later. All those boards together have been designed and built in the shortest time. The prototype contained four I/O combos: 3 vacuum controller (with three cards, one of each type, to control 7 pumps, one gate valve and read out one Penning gauge) and one BPM controller (two ADC cards to read the horizontal and vertical position of 8 beam position monitors).

All software was written in Neuron C. The code for the vacuum is quite straightforward, even with the slow interlock scheme that runs in parallel to the read and set operations. The BPM code includes averaging of 32 consecutive readings. In order to send the averaged values once per second, a few tricks were necessary, as it takes 300-400 ms just for the averaging of 12-bit values. The 16 (8 BPM's times two planes) updates are sent in one message in order to save bandwidth and network address allocations. This is because a Neuron chip can send up to 15 different variables in an effective way. More variables take significantly more time - but several values can be composed into one message and sent as one variable.

3.3 Communicating with LonWorks from within Windows applications

The library of the LonWorks network services (LNS) can be incorporated into Windows software just by dragging their OLE control objects to the visual development form, e.g. in the Visual Basic integrated development environment, or in Visual C++. The simplest LNS functions are used to access and display network variables from within Excel or Visual Basic. A more power application is written in C++. This application is actually a CORBA[7] server that exports objects into the Internet, such that an application anywhere in the Internet can access certain network variables of a given device [4].

Although a powerful aspect of the LNS, it took 1-2 months to get it running properly. The main reason is lack of proper documentation and this is the main criticism we had with LonWorks. There is enough documentation including tutorials to establish simple tasks, but for certain features there exists only a definition of the API. Fortunately, customer support is quite helpful, even in Europe.

3.4 Control of power supplies

The prototype used the same card as BESSY to interface to a magnet power supply, the ADA-16 [8]. It has a high precision 16-bit ADC, 16-bit DAC, 8 digital inputs and 8 digital outputs. The card that usually communicates with a PC through the ISA interface was modified slightly by the manufacturer to support the muxbus interface instead - an 8-bit asynchronous parallel interface understood by the Neuron. With the addition of only one PAL, the Neuron was able to communicate with the ADA card.

A complete PS control node was created with sophisticated software that has the following functions:

- it constantly monitors the ADC and DAC settings of the ADA-16
- if a change occurs, the new values are sent over the network, but keeping a minimum time between two consecutive packets in order not to flood the network with spurious events
- if a maximum time period passes without any event, the values are sent nevertheless (heartbeat)
- if the values exceed certain limits, alarm messages are sent
- commands coming from the network (on, off, set, etc.) are executed and acknowledges sent back
- a state machine is implemented that blocks forbidden transitions

3.5 Synchronous bit-by-bit ramping

One of the requirements of the ANKA storage ring is that the power supplies must be ramped synchronously within 75 seconds, because the injected beam has not the full energy. The external synchronisation can come conveniently through one of the digital inputs of the ADA card. In order to keep the jitter between two different power supplies low, the ramp curve should be increased either by one bit or by none. This translates into the requirement that one step of the ramp is 1 millisecond. Under normal functioning, where the Neuron chip has to cycle through and event loop that cares about network messages and reads and writes to I/O, such a time scale is not reachable. Our tests have shown that a step of 12-15 ms would be needed.

However, if the Neuron is dedicated to ramping only, then the software can execute in a tight loop without ever going through the event loop. The ramping curve is loaded before the ramping starts and is converted into an array of bits in RAM. Again the slowness of arithmetic operations (even 16-bit integer) of the Neuron becomes obvious. To calculate a full ramping curve based on 10 intermediate points (corresponding to 8k of used RAM) takes close to 6 minutes. However, the curve is loaded only once, stored in flash memory and is never changed as long as the same machine physics optics is used.

In such a set up, two consecutive trigger pulses can be as close as 0.48 ms. The whole time from the trigger pulse to the actual set is 0.21 ± 0.05 ms, where half of the time is spent on the communication with the ADA card. More important is the jitter between the set of two different power supplies. That was measured to be 0.06 ± 0.04 ms, which exceeds the specifications by far.

In order to allow both controls of the power supply: normal operation (set/read) and ramping, two Neuron boards have to be connected to one ADA card (see figure 1). This is possible with the additional help of PAL's that take care of the timing of the muxbus interface.

3.6 Knob control

Old analog control systems allowed to control parameters smoothly by the use of knobs from the control room. Most of the current control systems don't support this, because of slow network communications. This is changing now as even the object oriented CORBA takes only between 2-4 ms to execute a remote method. A fieldbus alone should definitely be fast enough. For the operator to get an analog feel, about 25 set/read pairs are necessary, just like on a TV screen.

Our measurements on the complete control system[2,4], which starts with a Java application at the operator console, communicates through CORBA with the server, which in turn uses the LNS to communicate over LonWorks with the Neuron chip, show a turn-around time of 22 to 40 ms, depending on how much we have optimized the parameters of the LonTalk protocol for our needs. Table 1 shows measurements of PC to Neuron communications under different conditions.

Table 1: sustained number of executed power supply commands per second

command	no traffic	simulated traffic
read only	46	44
set only	55	42
smooth set only	26	19
both read and set	36	29
read and smooth set	15	10

4 Conclusions

LonWorks are a commercial fielbus and devicebus system that can significantly reduce cost and save development time of a control system. LonWorks simplify the tedious work of programming the I/O and the network communication, putting all of its essentials in form of firmware into the Neuron chip. Our experience have not shown any severe limitations of the LonWorks and so far in no case exclude the idea of using them for the control system.

Apart from being simple and suited for quick starts, LonWorks also have the power of a very complex system. Of course, the details get complicated and then the learning curve becomes steep. However, these details can be left to one or two experts while other people still work with the basic tools. That is ideal for equipment experts who need to interact with the fieldbus but do not want to learn the "secrets" of microcontrollers or network communications.

Due to the relatively modest requirements of ANKA, we have found that instead of having relatively dumb devices and a strong middle layer, we bring intelligence to the devices and avoid the complexity of VME. This is possible also because most equipment has already some intelligent control. It is just necessary to combine them with a proper field bus.

In principle, a layer of PCs and the LonWorks fieldbus is enough to control an accelerator. Such a setup is acceptable for small systems. It reduces complexity and price. Large-scale scalability can still be achieved at moderate cost: connect LonWorks to a simple fall-through middle layer which can be anything from a TACO device server to an EPICS VME-based real-time databank. Such a hybrid solution (VME/LonWorks) might well be suited for a more demanding accelerator.

Acknowledgements

We thank the ANKA-team at the Forschungszentrum Karlsruhe and D. Einfeld and H. Schieler in particular for creating a pleasant atmosphere of collaboration and for their hospitality during our visits. One part of the LonWorks test were performed at the storage ring ELSA of the University of Bonn at the chair of D. Husmann, whose help is greatly appreciated. Many important questions were raised and suggestion given from the collegues in the TACO collaboration, notably A. Goetz, S. Hunt and W.D. Klotz. Some hardware for the prototype tests were lent by BESSY (corrector magnets power supplies and magnets) and Danfysik (a power supply simulator). We thank them all.

References

[1] D. Einfeld et al., ANKA - Status of the 2.5 GeV Synchrotron Light Source at Forschungszentrum Karlsruhe, Proc. PAC 97, Vancouver 1997.

[2] S. Avsec et al., The Design of the Control System for ANKA, Proc. PAC 97, Vancouver 1997.

[3] The '95-'96 Echelon LonWorks Product Databook, Echelon Corporation, 1995.

[4] B. Jeram et al., A Control System Based on WWW-Technologies, this conference.

[5] http://www.mot.com/SPS/MCTG/MDAD/ lon_overview.html, http://www.jged.com/web_pages/LonPaper.html

[6] LTM-10 User's Guide, Rev. 2, Echelon Corp., 078-0132-01B.

[7] S. Hunt, B. Jeram, M. Plesko, The Implementation of the OO Control System API with CORBA, this conference. I-ADA16-IO8 Version 1.0 Anwenderhandbuch, EuKontroll, 15.0.4.1996.