TINE: An Integrated Control System for HERA

Philip Duval, DESY, Hamburg Germany

Abstract

Beginning with the 1998 run period, the PC-dominated control system for HERA is largely in place. Most elements of the machine are controlled via the TINE (Three-fold Integrated Network Environment) data protocol. TINE offers a multi-platform, multi-protocol, multi-architecture control system. It features distributed, object-based, plug-and-play, front-end devices and middle layer servers which communicate directly with console (and office) computers. Although the consoles in the HERA control room all run WINDOWS NT, the HERA front end computers and servers run a wide variety of operating systems and are not always PCs. TINE has been invaluable in integrating the HERA front ends into a working system. Furthermore, it offers a transparent way to progressively upgrade existing hardware (as opposed to replacing everything at once). We describe below some of the finer points and details concerning TINE as a control system.

1 INTRODUCTION

For the most part, TINE defines a data-exchange protocol, which can be used as the basis for a control system. As a control system proper, it is “do-it-yourself” oriented. That is, it is assumed that a hardware-near data acquisition system is in place on the relevant front-end devices. At DESY for instance, the in-house field-bus SEDAC is in common use, as are the CAN bus and GPIB. There are well-defined drivers and APIs for each of these, which provide a hardware IO layer for TINE here. Continuing the “do-it-yourself” discussion, we note that all redundancy and fault tolerance should be added where needed. Furthermore, as the ethernet is the data-exchange medium, it should be noted that mission-critical devices should never depend absolutely on communication with other elements on the ethernet. Likewise, it should be noted that TINE has been seen to scale to machines the size of HERA without any problems.

TINE follows the traditional “Client-Server” dichotomy, where the control system elements play either the role of (and have the characteristics of) a server or a client. Servers can be attached to hardware (Front End Computers – FECs) or can provide middle layer services. They have unique names and addresses and are known on the net via database or name server (i.e. they do not broadcast their services). Clients are “anonymous” and can exist in multiple instances anywhere on the network. By this, we mean that clients are nowhere entered in a database, and several clients can appear bearing the same name. (Each computer on the ethernet must of course have a unique address). Clients find Servers by querying the equipment name server. Finally, TINE is not database-driven, and does not require any central database for configuration.

2 THREE-FOLD INTEGRATION

Perhaps the most distinguishing feature about TINE is its integration of client and server components of vastly different networking environments. To begin with, TINE is a multi-platform system, running on MS-DOS, Win16 (Windows 3.X), Win32 (Windows 95,98, NT), most UNIX machines, VAX and ALPHA VMS, and VxWorks. TINE is also a multi-protocol system to the extent that IP and IPX are both supported as data exchange protocols. Finally TINE is a multi-control system architecture system, allowing client-server, publisher-subscriber, and producer-consumer data exchange in any variation. We shall describe these in more detail below.

2.1 Multi-Platform

TINE runs on a number of platforms as illustrated in the table below. At HERA, the de-facto console platform is Windows NT (Window 3.1 in earlier years). All manner of front-end platforms are in use, however the individual sub-system laboratories (e.g. the RF group) generally stick to their preferred platform for all sub-system components. Note also that by allowing a heterogeneous system, expensive front-end hardware can be used where warranted (for mission-critical devices) and inexpensive hardware used elsewhere. Furthermore a systematic, piecemeal upgrade of a control system is possible, since TINE will run fine on older systems such as VAX-VMS and MSDOS as well as the more modern systems such as Window NT, Solaris and VxWorks.

OS	**IP Stack**	**IPX Stack**
DOS	LWP or Client32	NOVELL
Win16	WINSOCK	NOVELL
Win32	WINSOCK	WINSOCK
Linux	Native BSD	Native BSD
Solaris	Native BSD	-
HP-UX	Native BSD	-
SGI	Native BSD	-
OSF	Native BSD	-
VAX-VMS	UCX	-
ALPHA-VMS	MULTI-NET	-
VxWorks	Native BSD	-

Table 1: TINE Platform and Protocol support

2.2 Multi-Protocol

TINE supports both IP and IPX ethernet protocols. As seen from the above table, the IP protocol is general available across all platforms, whereas IPX is available primarily on PC systems (DOS, Windows, Linux). Where necessary, IPX could also be ported to the other platforms. However, an IPX-stack must be obtained and installed separately, as it is not in the standard system kernels for these cases.

2.3 Multi-Architecture

TINE supports three modes of data exchange, each of which could be used to define the control system architecture.

Client-Server: A traditional data exchange mechanism available in most control systems is pure, synchronous client-server data exchange, where a client makes a request and waits for the completion of the request.

Publisher-Subscriber: For many cases, a much better approach is the publisher-subscriber data exchange. Here a client (the subscriber) communicates its request to a server (the publisher) and does not wait for a response. Instead it expects to receive a notification within the timeout period. This can be a single command, or for regular data acquisition it can be a request for data at periodic intervals or upon change of data contents. In this format, the server maintains a list of the clients it has and what they are interested in.

Producer-Consumer: A third alternative for data exchange is the Producer-Consumer model. In this case a server is the producer. It transmits its data via broadcast on the control system network. Clients (i.e. consumers) simply listen for the broadcasts. This is frequently the appropriate data transfer mechanism. For most control systems, there are certain parameters of system-wide interest. At HERA for instance, the Electron and Proton beam-energies, beam-currents, beam-lifetimes, etc. are made available via system broadcast at 1 Hz. In the next major release of TINE, a multi-cast alternative will be offered.

3 MECHANICS

A detailed description of the functionality and operability of TINE is given in reference [1]. Below we present a synopsis of some of its features.

3.1 Equipment Modules

TINE is object-based, where TINE servers contain one or more equipment modules, which are designed to present an object-view of the equipment being controlled. These equipment modules have system-wide unique export names and contain one or more instances of the equipment, defined by device names. Furthermore equipment modules support device properties, which reflect the equipment operation.

TINE clients contact a particular device by specifying the equipment module’s export name and the device name. The nature of the request is specified by the device property.

In Producer-Consumer mode, on the other hand, a server makes registered data available via broadcast. In this case a client simply listens.

3.2 API

TINE offers a common, intuitive Application Programmers Interface (API) in C across all supported platforms. In addition, a Visual Basic API is provided in the Win16 and Win32 environment. Recently, a JAVA (client-side) API has been made available for applications on host machines, where JAVA has been installed.

3.3 Plug-and-Play

A TINE server registers its address, server name and its equipment module names with the equipment name server upon startup. A TINE client can then query the equipment name server for names and addresses (and cache the results). As persistent timeouts have the effect of forcing an address resolution, one can take a TINE server down, and bring it up on another machine (and protocol!), without the necessity of restarting a client.

3.4 Data Exchange

TINE data requests can consist of up to 64 Kilobytes of data in either direction (client to server and/or server to client), both in synchronous as well as asynchronous transfer modes. There are currently 38 defined system data formats, and it is also possible to specify user-defined structures. The latter must be registered at both the client and server if automatic byte-swapping and alignment is to take place.

3.5 Security

As noted above, TINE servers are registered with the equipment name server, which constitutes the control system database. TINE clients on the other hand are anonymous in this respect. A TINE server is fundamentally open to access from any client anywhere on the ethernet. A specific data request can however carry a WRITE access bit, which can in turn be filtered against at the server side. Namely a TINE server can allow WRITE access only to specific users, and/or only from specific networks, or network addresses.

3.6 Alarm System

As most alarms ultimately originate during hardware or service IO operations, it is most natural to locate the first layer of alarm processing directly on the front-end. A Tine server maintains a local alarm list containing all relevant information about each alarm. Clients can then at any time query a server, i.e. the Local Alarm Server (LAS), as to its alarm state. The primary client for all local alarm servers is the Central Alarm Server (CAS), where the next stage of alarm processing takes place. See reference [2].

3.7 Archive System

The implementation of TINE at DESY uses an external archive system, which acquires and provides both machine data and event-driven post-mortem based on the TINE protocol. A future release of TINE will offer a local archive server, whereby short term histories of selected data can be maintained at the device server.

3.8 Remote Control

TINE offers a good set of remote control and restart of server processes for most platforms.

4 CONCLUSIONS

The flexibility of TINE has been invaluable in integrating the HERA front ends into a working system. Just as important, it has demonstrated a transparent way to progressively upgrade existing hardware. Where for practical reasons the latter must remain on “older” platforms and operating systems, TINE servers can nonetheless be installed and maintained. When it becomes practical to “modernize” front-end elements, this can be achieved piecemeal, without any blanket restructuring. The implementation of TINE at DESY is PC-dominated. Although TINE works fine in say a pure UNIX world, the number of GUI tools developed for PC consoles running Win32 make the latter (currently) the most attractive platform on the client-side. On the server-side at DESY, TINE has also been shown to run on EPICS front ends offering “bilingual” data access (either via Channel Access or TINE). It is also planned that the next release of DOOCS[3] will also offer data exchange via TINE.

REFERENCES

[1] P.Duval, “The TINE Users Manual”, DESY internal document.
[2] M.Bieler, et al., “PC Based Alarm System for the HERA machine”, Proceedings PCaPAC’99, 1999.
[3] G.Gygierl, O.Hensler, and K.Rehlich, “Experience with an Object Oriented Control System at DESY”, Proceedings PCaPAC’99, 1999.