THE TINE CONTROL SYSTEM, OVERVIEW AND STATUS

Piotr Bartkiewicz, Philip Duval, Steve Herb, Honggong Wu, DESY, Hamburg, Germany
Stefan Weisse, DESY Zeuthen, Germany

Abstract

TINE (Three-fold Integrated Networking Environment) has been the Control System in use at HERA for some time, plays a major role in the Pre-accelerators at DESY, DORIS, FLASH, PITZ (Zeuthen), EMBL-Hamburg, GKSS-Hamburg, PF Beamline (KEK), and is the designated control system for the new third-generation light source PETRA3. TINE has always emphasized both performance and flexibility. For instance, using the multicast capabilities of TINE, state-of-the-art, near real-time video transmission is possible. At the same time, developers have a large toolkit and variety of software solutions at their disposal, and in general on their favorite platform and programming language. Code-generation wizards are available for rapid development of TINE servers, whereas intelligent GUI components such as ACOP aid in the development of either "rich" or "simple" client applications. The most recent major release brought with it a bundle of new features and improvements. We give here an overview of the TINE control system in general, what's new in particular, and focus on those features not available in other mainstream control systems, such as EPICS or TANGO.

INTRODUCTION

Historically, TINE [1] is a spin-off of the ISOLDE Control System [2] circa 1991/1992. Although originally PC-based it was soon ported to UNIX workstations, VxWorks, and VMS and has undergone continuous development (primarily at DESY, Hamburg) as well as ports to other platforms ever since. From the word "go" it was designed to handle the needs of a large and complicated machine, namely HERA. With well over 100,000 addressable control points and close to 200 device servers, HERA had to deal with issues of performance and scalability not usually seen in smaller machines. Although, HERA has been recently decommissioned, TINE has been shown to be a suitable choice for smaller accelerator facilities (either ring or linear) and for beam-line control, and will be the control system for the 3rd generation light source at PETRA3. TINE is a mature control system featuring an efficient control system protocol, a large number of central services, a device access layer, and a large set of rapid application development (RAD) tools for both client and server applications. At the same time, TINE is hardly "static" in any sense of the word. Indeed the next major release (4.0) is now available, with a bevy of new features and functionality.

A general overview of TINE as a control system as well as recent developments pursuant to TINE Release 4.0 will be presented in this article.

TINE KERNEL

The TINE kernel, as one might expect from its PC-based origins (where it fit snugly inside the 540 Kbytes allowed by DOS applications), has a small footprint. The kernel is written in the C programming language (as are most operating systems) with the exception of the TINE Java port. The data transport is based on Berkeley sockets, with the distinct advantage that no 3rd-party software installations (e.g. Sun RPC or CORBA) are required. TINE manages server name resolution itself via a TINE Equipment Name Server (ENS) or local caching files, obviating any need for a 3rd-party database (such as MySQL). Hence, installing TINE is a very easy and straightforward task on most operating systems, and brings with it no requirements for additional software.

Data transfer occurs via UDP by default, but can be configured to use TCP. If a client/server pair is on the same host, then pipes or messages are used. Data transfer payloads can generally be as large as can be allocated from the available memory (with double-buffering constraints).

The TINE Kernel can run either in single-threaded mode or in multi-threaded mode (where threads are available). Within a multi-threaded TINE kernel, internal processes can be configured to run on separate threads or not. This will include, calls to an equipment module handler, background i/o, schedulers, etc. The default configuration uses separate threads for all of these. (As a caveat however, note that, if properly programmed, a single-threaded application can frequently be more efficient, as there are then no context switches, thread synchronization, or issues of thread-safety to deal with).

TINE PROTOCOL

The TINE protocol allows data transfer according to one of four different paradigms, either separately or in combination. It is here that TINE distinguishes itself from other main-stream control systems. And it is here that TINE offers ready solutions for scaling the control system to large machines the size of HERA, without sacrificing accessibility. The first two data transport paradigms are familiar and are briefly listed here as "client-server" (i.e. synchronous transaction) and "publisher-subscriber" (i.e. asynchronous event notification). Both of these mechanisms inherently imply *uni*-cast (peer-to-peer) data transfer. In addition, TINE offers a "producer-consumer" mode of transfer, where control system data deemed of interest to most control system elements (e.g. beam energy) are *multi*-casted from a single source, regardless of the number of interested consumers. A fourth alternative is hybrid of the last two, a "producer-subscriber" mode, whereby subscribers

inform a server of their wishes, and the server (producer) *multi*-casts the results to his multicast group. Any number of clients can then make the identical request with no extra load on either the server or the network.

This fourth alternative is an excellent way to send large payloads (such as video frames) to a large number of listening clients.

Furthermore, TINE offers the ability for a server to send data upon event by explicitly calling the kernel scheduler. Typically, asynchronous event notification on the client-side involves polling the data channel of interest locally on the server, and transmitting the results on change or timer. This server-side polling can itself be made redundant if the server calls the scheduler when it knows that there is a reason to examine the channel's data (e.g. the server has just finished its i/o cycle or it has been triggered externally).

Data Types

TINE offers a wide variety of data types including all primitive types, as well as compound types consisting of data pairs (e.g. a float-int pair), data triplets, data quadruplets, fixed-length names, and again distinguishing itself from other control systems, TINE offers user-defined "tagged" structures. Tagged structures are defined by the developer, given a name (the "tag") and registered at the server and client. Tagged structures provide a very useful way of transmitting simple data objects, whose fields belong together, as an atomic unit.

Data types new to TINE Release 4.0 include an XML type, a BITFIELD type (where individual bits or groups of bits can be addressed by name), and a VIDEO data type (whereby a system-defined video header is sent with each video frame).

Application Programmers Interface

TINE offers several APIs depending on the application platform in question and the skills of the application programmer. Fundamentally, TINE is once again distinct from other control systems in that it recognizes at the API level that the data transfer is occurring between remote endpoints and offers parameters to the programmer for adjusting performance and scalability. For instance in the asynchronous interface, the 'kind' of monitor (on 'change', on 'timer', on 'event', etc.) requested can be specified, along with the 'scope' of the monitor (send as multicast, use TCP, etc.). And as of TINE Release 4.0, the transfer reason can determined upon receipt by a client (e.g. 'response', 'event', 'heartbeat', 'timer', 'data stale', etc.).

To be sure, many of these capabilities are of interest only to the professional programmer. TINE also offers far simpler client and server interfaces (which make default decisions) for part-time programmers.

In addition, TINE offers client and server APIs for LabView and MatLab, as well as a command line interface, which can be used in scripting languages, ActiveX controls, which can be used in Windows applications supporting ActiveX, and Java beans, namely the ACOP family of beans [3]. .Net controls are in progress.

TINE NAMING SERVICES

TINE has a hierarchical naming scheme common to other control systems, consisting of 3 levels of identification in specifying a device. In TINE these are known as 'context', 'device server' (or 'device group') and 'device name'. The attribute, command, or action requested from the device is given by a 'property'. In addition, a server's subsystem is also known and browseable from the ENS, but is not part of the name space. What is worth noting here is the distinction between 'device server' and 'device group'.

In a purely engineering point of view, the device server refers to a single host computer connected to its devices. In a machine physicist/operator point of view, one tends to think of a device group, say, Magnets, where the devices refer to all the magnet power supply controllers, regardless of whether they are attached to a single host computer or not.

In TINE one can effectively have both views by making use of device redirection, where a device group will appear in any control system browser and be addressable as a logical server. Accessing a particular device will then be redirected to the correct physical host managing the device. Accessing a range of devices from a device group (via wildcards, for instance) will also correctly determine where the constituents reside.

As of release 4.0, TINE allows context and server/group names up to 32 bytes and device and property names up to 64 bytes. This is usually well beyond the needs of typical ansi-character encoded names and should also easily be sufficient to handle uni-coded names. The TINE names used in this hierarchy are now case insensitive.

Also new in release 4.0 concerning naming services is a local dynamic address cache on all client machines with a disk. Typically a TINE site will have 2 ENS servers running, a primary and a secondary. In a scenario where neither ENS can be reached for whatever reason, a starting client application has in the past looked for a static and possibly dated address file located on disk as a fallback. With Release 4.0, all client applications will update a dynamic address cache upon acquiring a server's address from the ENS.

HARDWARE DEVICE ACCESS

TINE offers hardware device access via several means. In many cases, the server side API is easy enough to learn that a simple 'do-it-yourself' approach is a good way to go, for instance when a piece of hardware is purchased which comes with drivers and API. One can also leverage the LabView drivers by using them and making

use of the TINE LabView server VIs. Likewise EPICS [4] drivers can be used by running Epics2Tine [5] on an EPICS ioc.

TINE itself makes use of the Common Device Interface (CDI) [6] which itself is a powerful, plug-and-play device layer offering a portal to the attached hardware. From a server process, the hardware is essentially accessed using the TINE client API, using 'localhost' and 'cdi' for context and device group. The devices are the named devices from the CDI database and properties refer to either bus actions or information. For each new kind of hardware, a CDI bus plug needs to be provided, which defines the interface routines to the bus.

Also available are a bare-bones Network Queue for interfacing to low level PLC or FPGA controllers, and a dedicated TINE-CANOpen interface [7].

TINE runs well on embedded controllers such as PC104 and Altera NIOS, and a port to Windows CE is well underway.

VIDEO APPLICATIONS

The TINE video system makes use of two of the TINE-specific protocol features mentioned above to provide state-of-the art video transmission, namely the publisher-consumer paradigm (multicasting) to allow as many client applications as desired and event scheduling to send video frames immediately after they are grabbed. At the PITZ facility in Zeuthen, for instance, where rather stringent video requirements are in play, ½ MByte video frames are sent in this fashion over a 100 MBit Ethernet at 10 Hertz to as many clients as desired.

HIGH LEVEL APPLICATIONS

TINE provides a number of Rapid Application Development (RAD) tools to facilitate writing or configuring client applications. Most recently the ACOP family of Java beans has been expanded to incorporate a large set of displayer widgets capable of browsing the control system at run-time as well as design-time through the use of bean customizers [3]. In this way, rich client Java applications can be easily created via a Java Integrated Developers Environment (IDE) such as Eclipse or NetBeans, or simple clients can be assembled at run-time without the need of a heavy framework. Similarly, there is a large set of TINE Virtual Instruments (VIs) which can be used in LabView applications. Although the ACOP ActiveX [8] control can also be used in .NET applications, it represents a legacy technology and to this end, an effort to provide ACOP .NET controls is also underway. A MatLab API is also available.

TINE web applications can likewise be assembled quickly via the Web-based Controls Client (Web2c) toolkit [9], which is an AJAXian framework for creating thin clients. Such web applications have the advantage of being accessible anywhere, without firewall considerations.

CONNECTIVITY TO OTHER SYSTEMS

To make use of all the server-side TINE features alluded to above (e.g. event scheduling) it is prudent to use native TINE servers. Nonetheless, most features (such as multicasting) are readily available by providing a TINE server acting as a translation service for other control system elements. In the case of EPICS elements, the epics2tine [5] translator can either run practically embedded on any EPICS ioc, or run in parallel on a soft-ioc. Essentially, it bypasses the channel access protocol completely and runs EPICS over TINE. In the case of TANGO [10] elements, the tango2tine translator runs only as a separate process. This is also true for the STARS/COACK bridge [11].

DOOCS [12] elements on the other hand can be seen as pure TINE elements, as TINE is embedded in DOOCS.

CONCLUSION

TINE is a mature control system which is at the same time keeping pace with technological advances in many areas, including web technology and RAD tools. The TINE kernel has a very small footprint compared to other systems, has no 3rd party dependencies, and can provide high performance data transmission under extreme scenarios. To this end it has set the standard for video transmission via the control system.

REFERENCES

[1] http://tine.desy.de
[2] "A PC Based Control System for the CERN ISOLDE Separators", R. Billnge et al, ICALEPCS '91.
[3] "The ACOP Family of Beans: A Framework Independent Approach," J.Bobnar, et al, these proceedings.
[4] http://aps.anl.gov/epics.
[5] "EPICS to TINE Translator Release 2.0," P. Duval, et al., PCaPAC '05.
[6] "Using the Common Device Interface in TINE," P. Duval, et al., PCaPAC '06.
[7] "Integration of CANOpen-Based Controllers with the TINE Control System for Petra III", P. Bartkiewicz, et al., these proceedings.
[8] "The Use of ACOP Tools in Writing Control System Software", I.Deloose, et al. ICALEPS'97.
[9] "A Web Based Control Client Toolkit," R. Bacher, these proceedings.
[10] http://www.tango-controls.org
[11] "The Interconnection of TINE and STARS", T. Kosuge, PCaPAC '06.
[12] http://doocs.desy.de