INTEGRATING FIREWIRE CAMERAS INTO THE EPICS CONTROL SYSTEM

M. Dach, P. Jałocha, Paul Scherrer Institut, Switzerland

Abstract

A technical challenge in many scientific experiments is to capture and process images. There are potentially many solutions in this domain. One which seems to be cost effective, with high performance, is to use firewire (IEEE 1394) cameras. These types of cameras are easily connected to the PCs by mean of the firewire bus. We present a concrete solution based on firewire cameras' integration into the EPICS control system. Our solution allows for image capturing, processing and image distribution using Channel Access and HTTP protocols.

INTRODUCTION

Image capturing and processing has always been an important requirement in accelerator environments. Cameras are used mainly for diagnostic purpose at accelerators to observe the beam profile and beam position. There are also used in beam line experiments to setup, position and visualize samples.

For many years we have been using analog cameras at the SLS (Swiss Light Source in Switzerland). These cameras have had limited resolution. In addition, there was also noise from the analog cable linking the camera and a (remote) frame grabber. Only limited set of parameters could be remotely controlled for such cameras. To control, for example, the shutter time it was necessary to build dedicated hardware which was used to generate vertical and horizontal synchronization pulses. An attractive alternative for us was the introduction of firewire cameras which were much easier to interface and integrate with the EPICS control system.

Firewire cameras are connected to the system by means of the firewire (IEEE 1394) bus. The images captured by the cameras are digitized locally and transferred to the server in the digital form. The firewire bus is not only used for image distribution but as well for remote cameras configuration and control.

The firewire bus is an international standard. It supports data transfer rates of up to 400Mbps in IEEE 1394a, and up to 800Mbps in IEEE1394b. A single IEEE 1394 port can be used to connect up to 63 external devices. In addition to its high speed, IEEE1394 also supports isochronous data - delivering data at a guaranteed rate. This makes it ideal for devices that need to transfer high priority data in real-time, such as video devices. The IEEE1394 bus supports both Plug-and-Play and hot plugging, and also provides power to peripheral devices.

SLS IMPLEMENTATION

The goal at SLS was to integrate the firewire cameras with the EPICS based control system. We were searching for high performance and cost effective solutions. The EPICS control system is based on a client-server architecture and uses Ethernet as a communication medium. In order to connect the firewire cameras to such a control system it is required to introduce a bridge node which supports firewire and Ethernet connectivity.

We have finally converged upon an implementation using a PC computer with the Linux OS as a firewire server. The EPICS (toolkit) version 3.14.* onwards is well suited for Linux and PC computers. Linux operating system itself (kernel 2.6.*) has all required drivers for firewire bus. Appendix A lists the hardware and software components.

The preliminary version of the firewire server was using one camera per server. This solution was however not very effective and elegant so a multi-camera server has been developed. The firewire bus (like the USB bus) allocates the address for all connected nodes at run time. It is not possible a priori to know in advance which device has the given address on the bus. The only means to identify the firewire devices connected to the bus is to refer to them by the serial numbers which are unique.

The improved version of the server can theoretically deal with up to 63 cameras connected (Figure 1). Each camera is identified in the server by its given name. The camera names are associated with unique serial numbers using an ORACLE RDBMS. At start up the firewire server reads the serial numbers from all connected cameras and finds corresponding human readable names in the ORACLE RDBMS. Next, the server generates the EPICS DB to incorporate all connected cameras to the system. In addition a master client GUI application is created, in order to support control for the cameras.

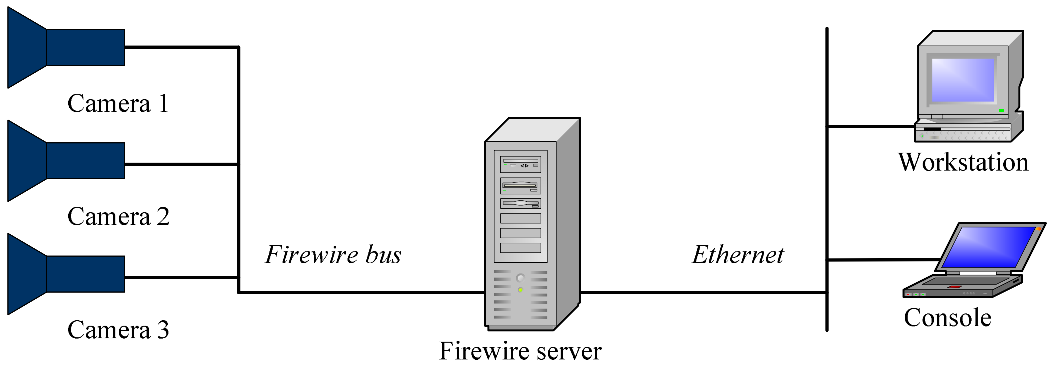

Figure 1: Integration of firewire cameras to EPICS based control system.

The purpose of the firewire server is multifold:

- control all cameras connected to the firewire bus:
  - switching the image acquisition on/off
  - selecting camera mode of operation
  - setting various camera parameters such as: exposure time, gain, external/internal trigger and many other.
- provide end users with:
  - raw or compressed images and
  - beam parameters such as: the beam profile and beam position.

An algorithm is implemented in the firewire server to calculate the beam profile and beam position, enabling the end user to obtain a few parameters of the beam instead of the entire image. This approach reduces network bandwidth requirements. The firewire server is multithreaded. Each camera connected to the bus is served by the acquisition thread. An acquisition thread is dynamically launched when the camera is put to the image acquisition mode. When the acquisition is stopped then the dedicated acquisition thread is terminated.

The firewire server is fully integrated with the EPICS control system; every camera is controlled in terms of EPICS channels. The grabbed (raw) images are stored in EPICS waveform channels. EPICS, however, is not well suited to transfer big chunks of data like images. The EPICS waveform channels which are used to hold images have statically defined sizes. The possibility to adjust the EPICS channel size would be especially useful when dealing with the compressed images or video streaming. The EPICS channel access protocol does not allow that unfortunately.

The firewire server has a built-in thread which acts as web server (Figure 2). The web thread is used to deal with jpeg compression which significantly reduces the network traffic. The firewire server can then provide the end user with:

- raw images using EPICS channel access protocol or
- compressed images using http protocol.

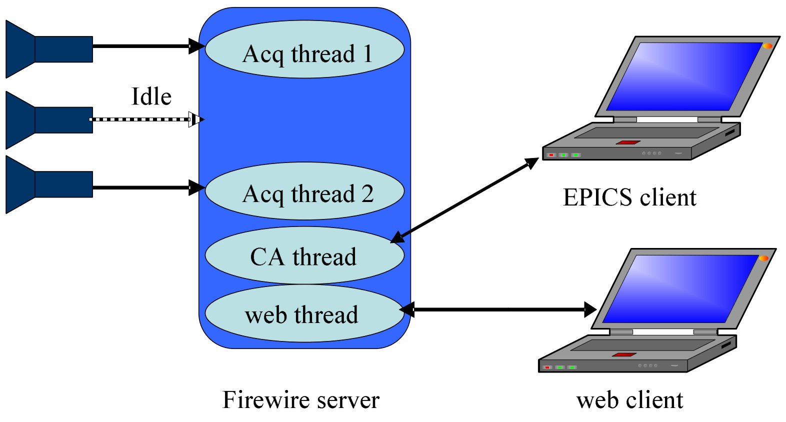

Figure 2: Firewire server internal functionality.

SUMMARY

At the SLS is used the firewire server implementation based on PC computers under Linux OS. Each firewire server can deal with up to 63 cameras connected to the firewire bus. The firewire server runs as a multithread process which is fully integrated to the EPICS control system. It is capable of performing image processing to calculate the beam profile, beam position and also image compression. The firewire server uses software and hardware components which are listed in the Appendix below:

Appendix A: Hardware and software utilized

Hardware:

Cameras:	Flea and Flea2 cameras by Point Grey (Resolution 0.8 or 2.0 Mega Pixels)
Servers:	PC computer: Pundit 2.8 GHz

Software components:

Server side:

- operating system: Scientific Linux 5 (kernel 2.6.18)

  Linux drivers:
  - ieee1394: Core of the IEEE1394 subsystem.
  - raw1394: Higher level driver module for bus access.
  - video1394: Fast DMA frame transfer driver.
  - ohci1394: Low level host card driver.

  Linux libraries
  - libraw 1.3.0: Low level user interface to access raw1394 module
  - libdc 2.0.0 rc-5: API to access firewire cameras
- Oracle 9.2
- EPICS infrastructure 3.14.8.2
  - Auto save and restore 4.2.1

Client side:

- medm based GUIs for EPICS raw data images
- any type of the web browser to view compressed images or for video streaming.