INTEGRATING GIGABIT ETHERNET CAMERAS INTO EPICS AT DIAMOND LIGHT SOURCE

T. Cobb, Diamond Light Source, Oxfordshire, UK

Abstract

At Diamond Light Source we have selected Gigabit Ethernet cameras supporting GigE Vision for our new photon beamlines. GigE Vision is an interface standard for high speed Ethernet cameras which encourages interoperability between manufacturers. This paper describes the challenges encountered while integrating GigE Vision cameras from a range of vendors into EPICS[1].

INTRODUCTION

Diamond Light Source is a 3 GeV third-generation light source with a 561 m storage ring (SR), a full-energy booster (BR) and a 100 MeV pre-injector Linac[2]. The photon output is optimised for high brightness from undulators and high flux from multi-pole wigglers. The current operational state includes 19 photon beamlines, with a further three beamlines in an advanced stages of design and construction. A further phase of photon beamlines is now confirmed, detailed design and construction of these 10 beamlines started earlier this year.

A range of cameras are used to provide images for diagnostic purposes in both the accelerator and photon beamlines. The accelerator and existing beamlines use Point Grey Flea and Flea2 Firewire cameras. The disadvantage of Firewire is that it requires complex cabling with multiple repeaters chained together. Diamond has used both a licensed Firewire stack running under vxWorks and an open source Firewire stack running on Linux, but both implementations have been somewhat unreliable. This means that the bus occasionally needs to be power cycled, especially when running at 800Mbit/s over 10m cables.

For the next phase of photon beamlines a better solution was needed, so the decision was made to evaluate Gigabit Ethernet cameras. This would allow up to 100m data transfer lengths, with greatly simplified cabling.

INITIAL TESTS

Selecting a Camera

The most suitable replacement camera was the Prosilica GC1020[3], Fig. 1, as it uses the same sensor as the existing Flea2 cameras, and already had EPICS support in the areaDetector[4] module.

Figure 1: Prosilica GC1020 Camera.

Configuring EPICS Support

The areaDetector module provides a common interface for all supported 2d detectors. Integrating the camera into areaDetector allows image processing and analysis plug-ins to be chained together at runtime, such as the NDStats plug-in for statistics, or the ffmpegServer[5] plug-in to provide a compressed mjpg stream for visualisation (see Fig. 2).

The PvAPI[6] library is supplied by Prosilica (now part of Allied Vision Technologies) to control their cameras in the form of a software development kit (SDK) that works on Windows, Linux and OS X. The areaDetector Prosilica driver consists of a translation layer on top of the PvAPI library.

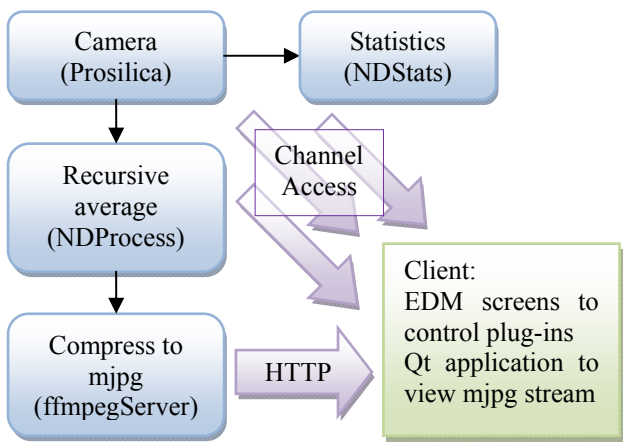

Figure 2: Typical areaDetector plug-in structure.

Conducting a Representative Test

A series of tests was conducted with 10 cameras on a typical beamline network consisting of a Nortel Gigabit Ethernet switch with servers, desktop machines, motor controllers and other Ethernet attached equipment. Different combinations of cameras were tested, and it was noted that the more cameras that were active, the less total bandwidth was achievable without dropped frames. With 3 cameras acquiring, 70MB/s was reliably achievable which was sufficient. Occasionally a camera would hang and stop acquiring, but this was not reproducible later in the lab.

SUPPORTING CAMERAS FROM DIFFERENT VENDORS

All vendors supply an SDK of some form to allow end users to control their cameras, but using one of these SDKs has some disadvantages:

- Most vendors' SDKs are tied to the vendor's cameras, so that you cannot switch to a different vendor without changing SDKs
- Most vendors' SDKs are focussed on the Windows end user, while Diamond uses Linux for control system servers.
- Most vendors' SDKs are closed source, meaning that debugging problems is more difficult, and bugs must be fixed by the vendors.

These factors prompted a search for an open source library that could control any GigE Vision camera, would run on Linux, and could be integrated into EPICS

THE GIGE VISION STANDARD

Most Gigabit Ethernet cameras conform to the GigE Vision[7] standard, which uses the GenICam[8] standard to describe the features supported by the camera. The GenICam standard provides a common interface to many different types of cameras, across different vendors and even across different physical connections types. The camera provides an XML file which describes the features that it supports, and how they map to registers on the device. This means that supporting cameras from different vendors should be simple, as the XML file will describe how the camera should be controlled.

Unfortunately, the licensing conditions of the GigE Vision standard do not allow the licensee to reveal details of the standard. Any licensee wishing to publish software implementing the standard must make the part that is drawn from the standard closed source otherwise they risk breaking the licensing agreement. This would mean that if Diamond used the GigE Vision standard to produce any software implementation, its sources could not be published back to the EPICS community.

In order to be useful to the EPICS community, an open source library was needed, and the only way this library could exist is if it had been reverse engineered. Aravis[9] is one such library.

INTEGRATING ARAVIS INTO EPICS

A typical areaDetector driver consists of a number of parameters, both defined by the base class and specific to this detector, and a mechanism for publishing frames from the detector as NDArrays. As well as the driver, an EPICS database provides access to each of these parameters, and an EDM[10] screen to provide the user interface. The sections below describe the steps taken to wrap the Aravis library in an EPICS module called aravisGigE[11].

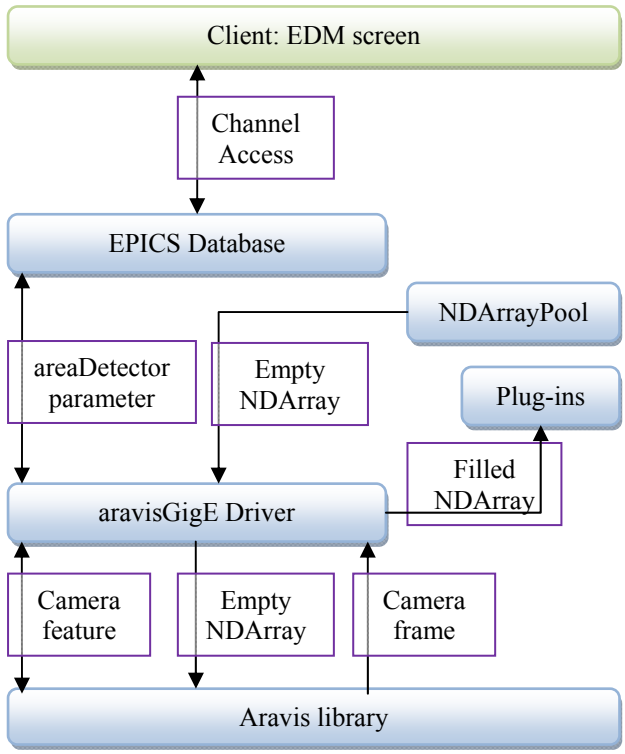

Figure 3: Structure of aravisGigE driver.

The Driver

One of the tasks done by the driver is mapping all the areaDetector common parameters to features supported by the camera. E.g. ADGain is mapped to the "Gain" or "GainRaw" feature. The mapping of areaDetector parameter to camera feature is placed in a hash table. For every other camera feature a new areaDetector parameter is created, and this is also placed in the hash table. When EPICS writes a parameter down to the driver, the corresponding feature is looked up in the hash table, and written to the camera. An update function also periodically updates the readback values of each of the parameters in a similar manner (see Fig. 3).

All memory management of frames in the driver is done using areaDetector's NDArrayPool, which maintain a pool of NDArrays that can be reused when all plug-ins have finished with them. The Aravis library is loaded with a number of NDArrays which it fills in when it gets a new frame.

The driver registers a call-back function with the Aravis library that is called whenever the camera produces a new

frame. This frame is annotated with information like timestamp, colour mode and data type before being published to any connected plug-ins.

Extracting the GenICam XML

Although the driver can query the camera to find out which features it has and create areaDetector parameters on the fly, the EPICS database and EDM screen must be generated at build time. As Aravis parses the GenICam XML from the camera in order to build its list of features, this XML is all that is needed to build a database and screen for the camera. Aravis provides a small utility that will allow the GenICam XML to be written to file.

The EPICS Database and EDM Screen

To parse the XML file, aravisGigE contains a Python script called makeDbAndEdl.py. The script creates a menu structure based on the Category nodes, and a list of feature nodes contained in it.

To create the EPICS database, for each feature node a suitable record is created, using stringin, longin, ai, and mbbi records for StringReg, Integer, Float, and Enumeration nodes respectively. If the field is editable the appropriate output record is created.

To create the EDM screen, a section is drawn for each category, then suitable widgets are created for the records that have been created. A tooltip is added with some suitable text, and a summary screen is created (see Fig. 4).

Figure 4: Section of Prosilica GC summary screen.

DISADVANTAGES OF REVERSE ENGINEERING

The main disadvantage of reverse engineering is that assumptions are made according to the cameras that are available. For instance, the author of Aravis only lists it as being tested on Prosilica and Basler cameras. It is no surprise then that the Prosilica cameras at Diamond work well with Aravis. Different vendors' cameras have been tested with varying degrees of success, the problems mainly being GenICam elements that are not supported in Aravis yet. There have however been some issues with dropped frames on some vendors' cameras that may point to a bug in the packet resend code.

CONCLUSION

The aravisGigE module is being rolled out on new beamlines, and GigE cameras have been retro-fitted on some existing beamlines. So far they have been more reliable than the Firewire cameras, and the simple cabling makes it much easier to move the cameras to different positions. Upcoming power over Ethernet versions of the cameras will reduce the number of cables still further.

It is very useful to have the ability to change manufacturer without the burden of having to write new software. Diamond's standard camera for beamlines is now the Allied Vision Technologies Manta, Fig. 5, but if a particular beamline has requirements that cannot be met by this range of cameras, then one can be sourced from another manufacturer. Aravis does an admirable job of controlling a range of cameras, given its reverse engineered origin.

Figure 5: Allied Vision Technologies Manta Camera.

REFERENCES

[1] Experimental Physics and Industrial Control System; http://www.aps.anl.gov/epics

[2] R. P. Walker, "Commissioning and Status of The Diamond Storage Ring", APAC 2007, Indore, India.

[3] Prosilica GC1020 Camera Specifications; http://www.alliedvisiontec.com/us/products/cameras/gigabit-ethernet/prosilica-gc/gc1020.html

[4] areaDetector: EPICS software for area detectors; http://cars9.uchicago.edu/software/epics/areaDetector.html

[5] ffmpegServer: video compression for areaDetector; http://controls.diamond.ac.uk/downloads/support/ffmpegServer

[6] AVT PvAPI SDK for GigE Vision® cameras; http://www.alliedvisiontec.com/us/products/software/windows/avt-pvapi-sdk.html

[7] GigE Vision® - True Plug and Play Connectivity; http://www.machinevisiononline.org/vision-standards-details.cfm?type=5
[8] The GenICam™ standard; http://www.emva.org/cms/index.php?idcat=27
[9] Aravis - A vision library for GenICam based cameras; http://live.gnome.org/Aravis
[10] EDM: an EPICS display manager; http://ics-web.sns.ornl.gov/edm
[11] aravisGigE: areaDetector GigE camera driver; http://controls.diamond.ac.uk/downloads/support/aravisGigE