Science & Technology Facilities Council
Rutherford Appleton Laboratory

Particle Physics

Update on IPbus v2

David Sankey

Friday, 21 June 2013

IPbus team:

Marc Magrans, Dave Newbold, Andy Rose, DS, Tom Williams

original incarnation from Jeremiah Mans, Erich Frahm and Eric Hazen

Much of this talk plundered from previous talks by the above and Rob Frazier

Overview of talk

Brief summary of IPbus

- Why IPbus
- Design Goals
- Why IP
- Why not TCP
- IPbus transactions
- IPbus components

Current status

- Reliable transport over UDP
- Multiple packets in flight for throughput
- IP address assignment with RARP

System tests at CERN and future work

Why IPbus

Need a mechanism to control xTCA hardware

Replacing the access over VME in historical systems

- and the need for VME SBC (ATLAS) and/or CAEN drivers/controllers (CMS)

General philosophy:

No reliance on proprietary systems

- LHC experiments operate on much longer timescales than industry

Based on an A32 D32 virtual bus

- Wishbone, that VME history…

Keep hardware and firmware as simple as possible

- push (any) complexity into software

Encapsulate complexity away from end-users

- simple API
- users should not be dealing with resource locks, threading *etc*.

Flexibility and scalability must be built in

- bench top to Point *n* deployment

Design Goals

Cover wide range of use cases

- single board through to 100s of xTCA modules

Good performance

- approach GB Ethernet wire speed?

Latency not an overarching priority

- but addressed to a degree by queuing and grouping requests in packets

Efficient FPGA resource usage

- fit into small FPGA or small % of large FPGA
- no requirement for CPU

Not dependent on xTCA features

- not tied to μTCA or ATCA
- can use IPMI for address assignment, RARP elsewhere

Work across many devices

- minimal vendor specific or family specific features
- essentially just the vendor cores for Ethernet MAC and FIFO

Simplicity!

Why IP

Target application

Medium speed control traffic

- perhaps some local DAQ function

Pervasive technology

IP over Ethernet is everywhere (including xTCA Base Interface)

- physical layer components are cheap
- every computing device can use it trivially

Key advantage: cost-effective scalability

- setup of a large (switched) distributed hardware control system is trivial
- setup of a many-to-many client/server system is trivial

Key advantage: no drivers required!

Comparison with other 'control bus' technologies

Not 'low performance' (*cf*. USB, SPI, VME)

Not really 'high performance' (*cf*. PCIe, RapidIO)

- but 10 GB Ethernet could change that…

Why not TCP

Design choice is UDP/IP

Essentially equivalent to ‘raw IP’

- but UDP allows user space drivers (at the expense of some latency)

Pros:

- simplicity, efficiency, stateless

Cons:

- no guaranteed delivery

Why not TCP

In practise, modern switched Ethernet is almost 100% reliable

- in tests, dropping 1 packet in 200 million, in test beam, 1 in 30 million
- UDP has significantly lower latency for small packets

TCP effectively requires a CPU at the device end

- FPGA soft CPUs not good for performance or logic requirements
- FPGA hard CPUs require external RAM

UDP easy to implement in firmware

IPbus transactions

UDP packet to hardware target contains an IPbus packet header

- used for reliability and streaming

And a series of requests

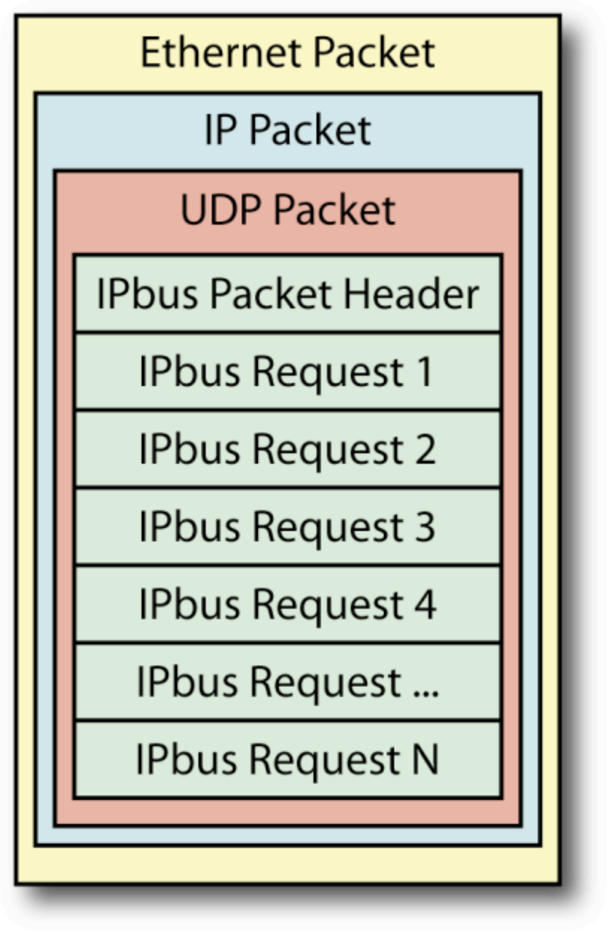
- Read
- Non-incrementing read (*i.e.* from FIFO)
- Write
- Non-incrementing write (*i.e.* to FIFO)
- Read-modify-write bits (*i.e.* manipulating bits in a register)
- Read-modify-write sum (*i.e.* adding to a register)

Multiple transactions in a packet

- optimise throughput
- *major difference compared to VME*

A32/D32 addressable by word

- 16GB addressable per IPbus end point
- can have multiple end points per target (different UDP ports)



IPbus components

IPbus firmware

VHDL, using UDP as transport protocol

- ARP, RARP, ping

μHAL

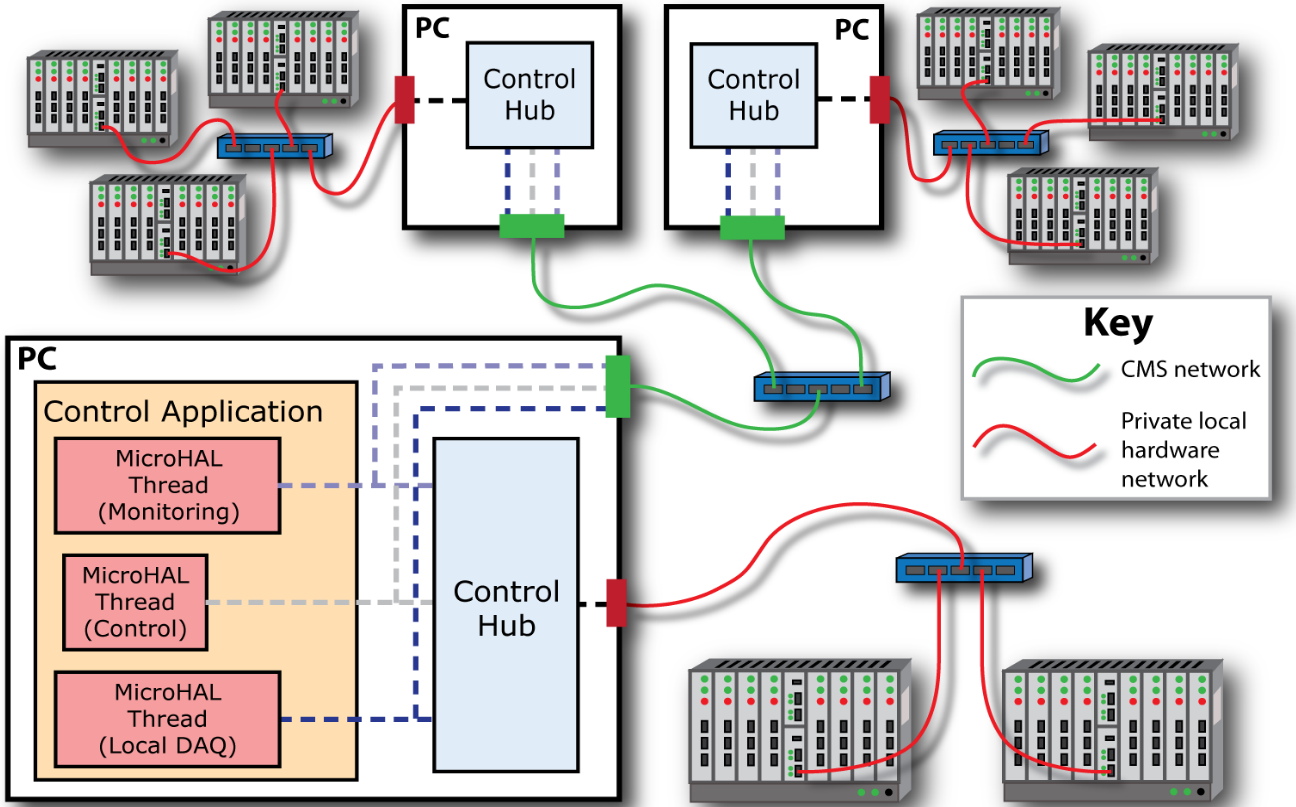
C++ Hardware Access Library

- Python bindings for easy scripting

ControlHub

Single point of contact with private hardware network

- serialises/routes IPbus transactions
- implements reliability over UDP
- communication between μHAL and remote ControlHub over TCP


PC
Control Hub
PC
Control Hub
Key
CMS network
Private local hardware network
PC
Control Application
MicroHAL Thread (Monitoring)
MicroHAL Thread (Control)
MicroHAL Thread (Local DAQ)
Control Hub

IPbus status

Version 1.3

Last version in 1.x series, in extensive use

- not protected against packet loss
- only single packet in flight

PyChips (standalone only) replaced by pycohal (compatible with ControlHub)…

Version 2.0

Specification finalised

- https://svnweb.cern.ch/trac/cactus/browser/trunk/doc/ipbus_protocol_v2_0.pdf

First software release in April

- https://svnweb.cern.ch/trac/cactus/blog/released_cactus_ipbus_2_0_0

Control behaviour unchanged from C++/Python API perspective

Consolidation in back-end code of all components

- transport between ControlHub and firmware reliable
- packet streaming for throughput

Website: http://cactus.web.cern.ch

IPbus 2.0 firmware

Complete rewrite of transport layer

Full functionality for multiple packets in flight

Tested on SP605 and GLIB at Bristol, RAL and CERN (and other Xilinx, S6, V5, 6, 7)

- millions of pings
- millions of control packets to and from board
  i.e. 500k reads/writes (random sequence)

All responses correct

Number of packets configurable at VHDL compile time

Tweaks for ease of use

Cope with little-endian requests

- the one instance where the fix is in the firmware, not the software…

IP address assignment via RARP

- definitely not DHCP!

Support multiple IPbus endpoints on same node

- each on different UDP port

IPbus 2.0 μHAL

Only one or two additions since last release

- new protocol version already in, has been passing unit tests for last month
- support for streaming packets

Some changes to back end for consolidation/ ease of future maintenance

- again passing unit tests

Still to do (short term)

- add remaining IPbus 2.0 functionality to dummy hardware (software that replicates hardware IPbus interface)
- a few other smallish items

(and, although we don’t support it, PyChips still works)

IPbus 2.0 ControlHub

Large overhaul of one of the modules to implement packet loss recovery and multiple packets in flight

Testing

- force 0.1% packet loss to/from hardware
- far higher loss than on any sensible network

Sent 5 million packets to board

- *i.e*. 5000 packets dropped

All successfully recovered

Still to do (short term)

- performance tweaks

Longer term

- a handful of other changes for P5-style deployment

Packet loss recovery and multiple packet implementation

New in version 2 protocol is an IPbus packet header

This contains an incrementing packet ID

- hardware will only accept next packet ID (or 0), dropping any other packet

For *n* packets in flight, ControlHub and hardware buffer all *n* outbound packets

Packet loss on way to hardware

Hardware ignores subsequent packets

- ControlHub confirms next expected packet ID and resends from there

$\Rightarrow$ *communication resumed*

Packet loss on return from hardware

ControlHub spots missing response

- ControlHub confirms next expected packet ID and requests resend of missing packet

$\Rightarrow$ *communication resumed*

Multiple packets in flight $\Rightarrow$ streaming

IP address assignment with RARP

Need means of dynamically assigning IP address

- so can have single firmware image for multiple modules

In xTCA crate could use hardware address

- with some means of getting this from IPMI

More general address assignment needs to be network-based

Obvious candidate would be DHCP

- in xTCA HPM.3 would be interesting to give geographic IP address
- until you look at what's involved from a firmware perspective!

Far simpler protocol is RARP

- very similar to ARP
  what IP address for this MAC address
  rather than
  what MAC address for this IP address
- RARPD in Scientific Linux

It just works

- but even then 30% of logic usage in IPbus firmware is to support RARP…

Tests at CERN

Setup in Building 904

- test all IPbus system issues
- measure performance (bandwidth and latency) in Point 5-like system
- same network topology as currently planned for Point 5

So far concentrating on throughput

- full system commissioned in last couple of weeks

First results with multiple packet version of μHAL direct to module (not reliable)

- 500Mbps write bandwidth direct to single module with 1 μHAL client
- 450Mbps read

Initial results through ControlHub (reliable, multiple client)

- 280Mbps total write bandwidth to single module with 2 μHAL clients
- 500Mbps total write bandwidth to crate (1 μHAL client for each of 4 boards)

On-going plan of systematic set of measurements

- fully understand system
- identify bottlenecks and improve them
- first target ControlHub!

Conclusion

First version of IPbus 2.0 has been released

Implements reliable transport over UDP between hardware and ControlHub

Supports streaming multiple packets in flight over UDP

- improves block read/write bandwidth

Supports dynamic IP address assignment with RARP

Specification at

- https://svnweb.cern.ch/trac/cactus/browser/trunk/doc/ipbus_protocol_v2_0.pdf

Release notes

- https://svnweb.cern.ch/trac/cactus/blog/released_cactus_ipbus_2_0_0

Website

- http://cactus.web.cern.ch

Being used for integration testing at RAL and at CERN in Building 904