IPbus

A method to communicate with cards over Ethernet

Communication Requirements

- Primary control path in MicroTCA is Ethernet
  - What protocol should we use UDP, TCP or something else?

- Requirements
  - Robust
  - Scalable
  - Reasonable bandwidth (make good use of 1Gb/s crate interface)
  - Relatively simple
    - Not too onerous (10% of design, not 90%).
    - Maintainable over 10 years with different versions of the tools and different people.
  - Portable from one card to the next

Communication Ideas

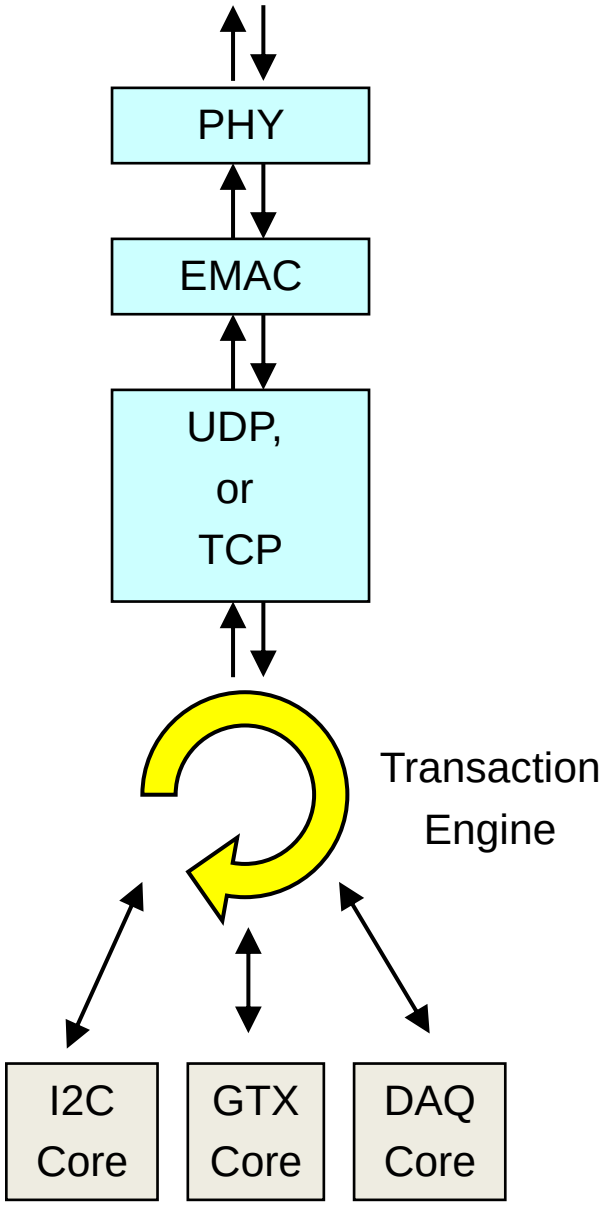
- Primary advantage of TCP is not reliability, but throughput
  - Imagine UDP with retry capability
  - Ethernet has a large latency (packet based, CRCs, etc)
    - i.e. single transactions will be relatively slow
  - TCP allows multiple packets in flight simultaneously and ensures that all packets arrive, and in the correct order
    - Ideal, but powerful CPU or complex firmware core to reach 1Gb/s
    - Separate commands still slow (i.e. do A, then B, then C)

Embedded CPU on the card either within FPGA or external
Can get quite complex (i.e require CPU, RAM, Flash, etc)

IPbus

- Originally created by Jeremy Mans et al in 2009/2010*

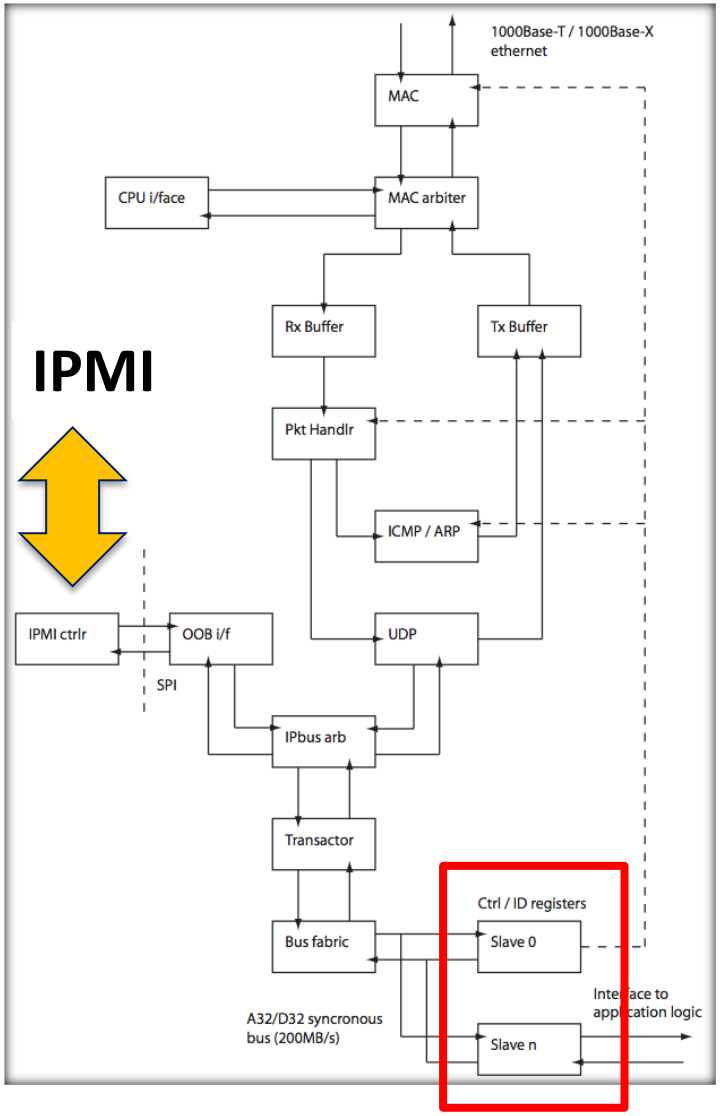
- Protocol describes basic transactions needed to control h/w
  - A32/D32 Read/Write, Block Transfers, Auto Address Incrementing
  - Simple concatenation of commands
  - Single packet may contain write followed by block read



***John Jones implemented something similar**

IPbus Firmware

- Resource usage:
  - Xilinx SP601 Demo board
    - Costs £200/$350
    - Small Spartan 6 (XC6LX16-CS324)
    - Uses 7% of registers, 18% of LUTs and 25% BRAM
  - Block RAM usage may increase slightly for v2.0 protocol
- Additional features:
  - Firmware also includes interface to IPMI controller

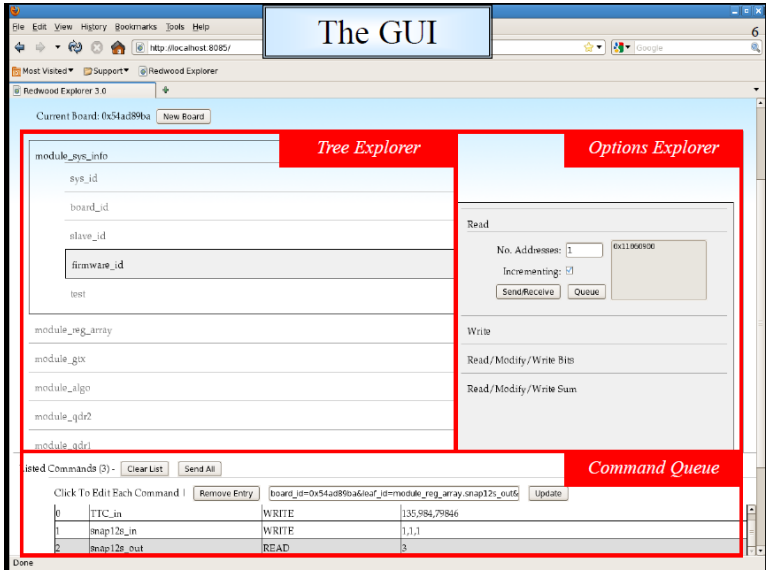


Firmware by Jeremy Mans Dave Newbold

The IPbus Suite Overview

Redwood
IPbus HAL

- MicroHAL (Based on Redwood)
  - C++ Hardware Access Library
  - Highly scalable and fast
  - Hierarchical with array capability
    - Mimic firmware structure
  - Automatic segmentation of large commands
    - e.g. block R split up
  - Software has full knowledge of registers
    - Map onto database



Redwood Explorer to access any register via simple web interface

Software by Andy Rose & Christopher Hunt

Just one example - hierarchical design

```
<module name="ttc" mask="0x000000FF" >
. <leaf name="CtrlReg" address="0x0006" >
. . <bitfield name="TtcIntL1aTime" offset="0" width="12" />
. . <bitfield name="TtcIntCmdVector" offset="12" width="8" />
. </leaf>
. <leaf name="PulseReg" address="0x0007" >
. . <bitfield name="TtcIntL1aStrobe" offset="0" width="1" />
. </leaf>
</module>
```

+

```
<module path="MiniT5ttc.xml" name="ttc" alias="module_ttc" address="0x00008000"/>
```

```
aRedwood.getLeaf(" module_ttc , CtrlReg ")->rmw_bits( "TtcIntL1aTime", 128);
aRedwood.getLeaf(" module_ttc , PulseReg ")->rmw_bits( "TtcIntL1aStrobe", 1);
aRedwood.getLeaf(" module_ttc , PulseReg ")->dispatch();
```

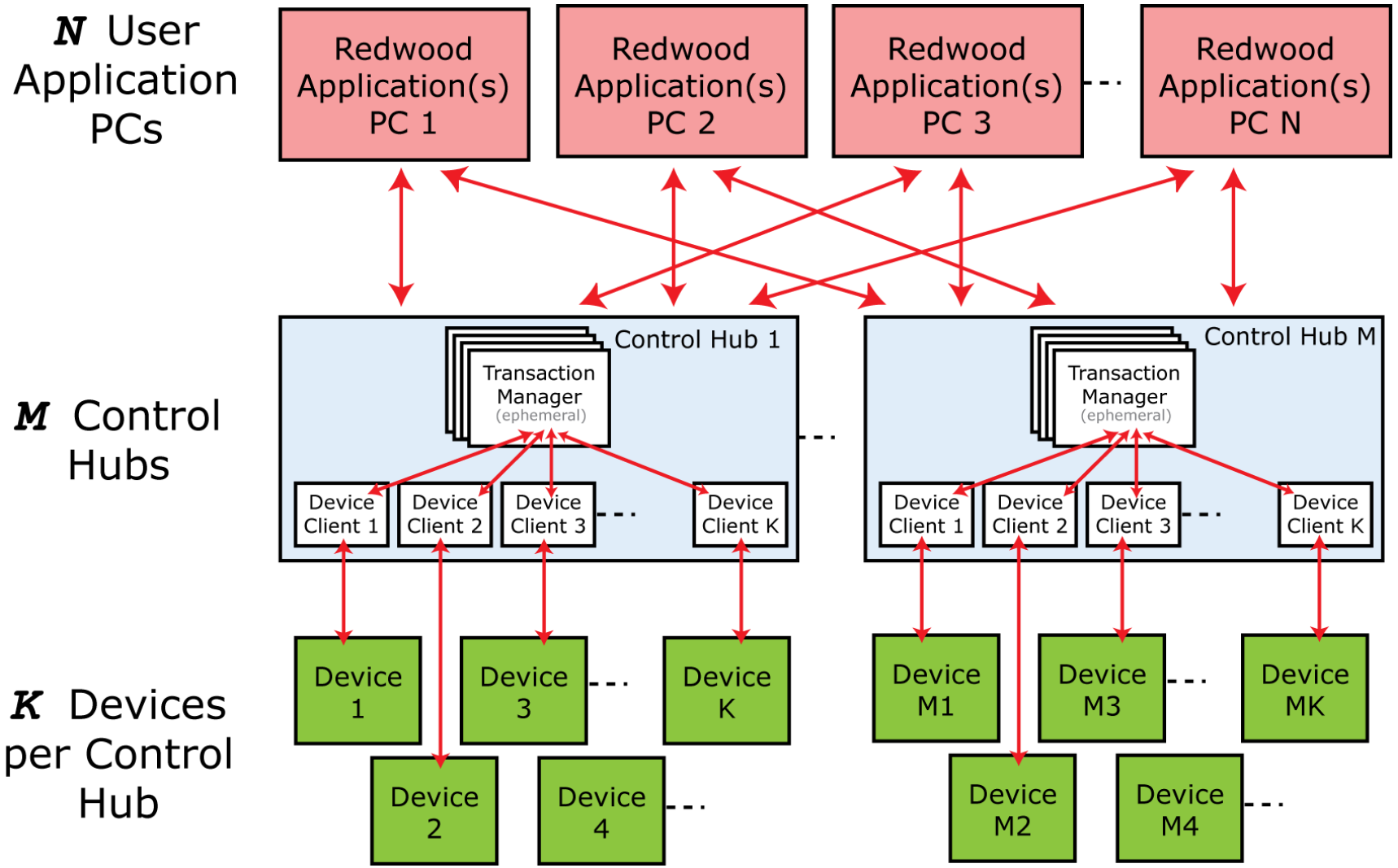
The IPbus Suite Overview

CONTROL hub

- Control Hub
  - Single point of contact with hardware
    - Allows multiple applications/clients to access a single board
  - Reliable and scalable
  - Built on Erlang.
    - Concurrent telecom programming
    - Scales transparently across multiple CPU cores
  - Automatic segmentation of large commands (e.g. block R split up)

Software by Rob Frazier

Scalability with Redwood and the Control Hub



IPbus Suite Overview

- PyChips
  - Python-based user-facing Hardware Access Library
  - Simple & easy interface
  - Great for very small or single-board projects
  - Cross-platform: Windows, Linux, OS X, etc
  - No dependencies except the Python interpreter itself

Software by Rob Frazier

IPbus Test System

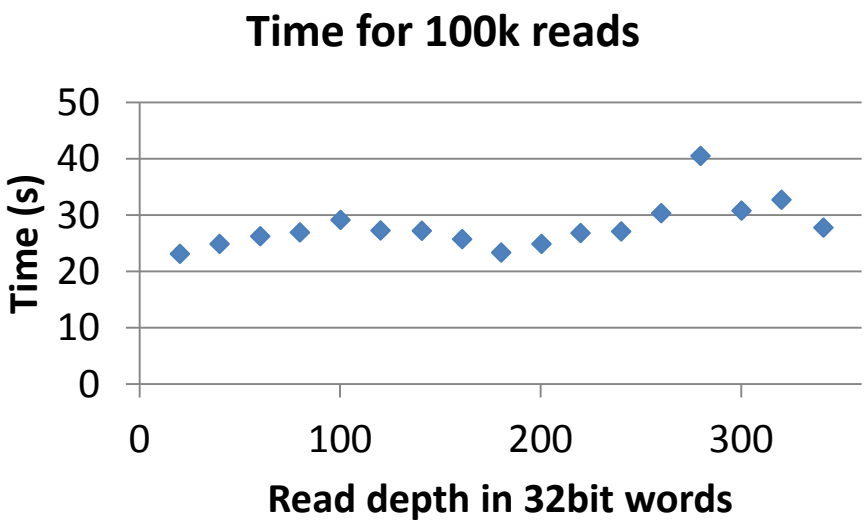
- 3 MicroHAL PCs
- 1 Control Hub PC
- 20 IPbus clients
  - 6 development boards:
  - 3 x Xilinx SP601 (Spartan 6)
  - 2 x Xilinx SP605 (Spartan 6)
  - 1 x Avnet AES-V5FXT-EV30 (Virtex 5)

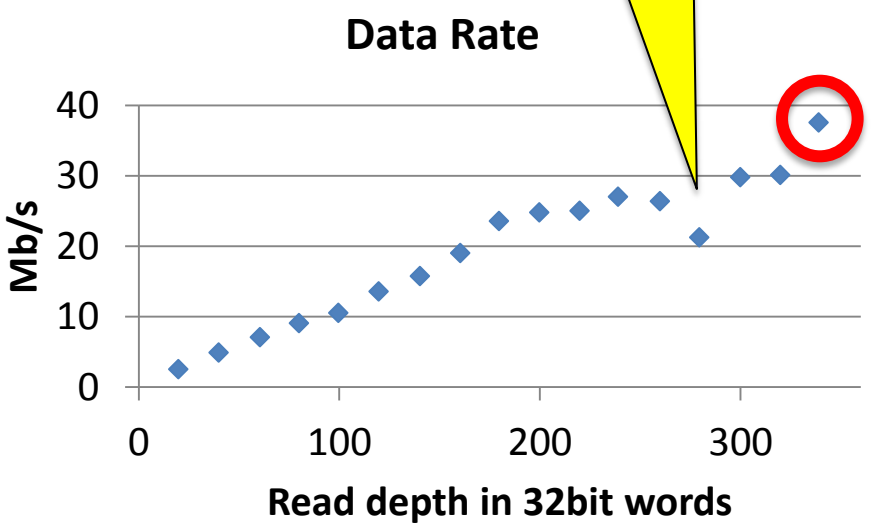
Allows for extended soak tests - Essential for verifying reliability

Performance

- Currently
  - 40Mb/s with full structure albeit with multiple MicroHAL instances on same PC and the ControlHub (i.e. likely scenario in CMS)
  - Scales linearly with number of cards (i.e. 480Mb/s) for a crate

- Target of 100+ Mb/s with UDP (Require TCP/IP for 1Gb/s)
  - Reducing copy stages in firmware from 5 to 3
  - Moving to Jumbo frames 1.5kB to 9kB – **x6**



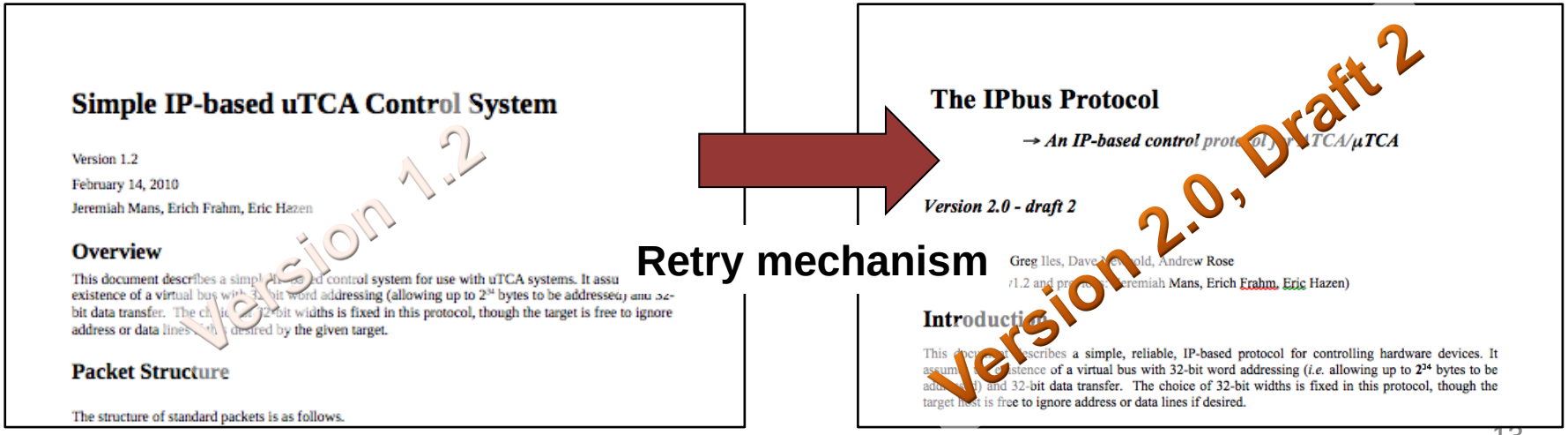


PC used to synthesise firmware

Reliability

- Private network
  - All unnecessary network protocols switched off (spanning tree, etc)

- Sent 5 billion block read requests
  - 10 billion packets total, 53 went missing.
  - 350 * 32-bit block read
  - 7 Terabytes IPbus payload data received
  - 19 IPbus clients used in test
  - Packet loss averages at **1 in 189 million** UDP packets



Links

- IPbus SVN & Wiki hosted on CACTUS project
  - Website
    - http://projects.hepforge.org/cactus/index.php
  - HepForge repository
    - http://projects.hepforge.org/cactus/trac/browser/trunk
  - MicroHAL
    - The Software User Manual, Instant Start Tutorials and Developers Guide
    - http://projects.hepforge.org/cactus/trac/browser/trunk/doc/user_manual/Redwood.pdf?rev=head&format=txt

- Firmware Chief: contact Dave Newbold: dave.newbold@cern.ch
- Software Chief: Rob Frazier: robert.frazier@cern.ch
- MicroHAL & Redwood: andrew.rose01@imperial.ac.uk