Dt100 Software Architecture: 2 Layer

Client

3rd Party Client

dt100rc

MDS+

Server

TCP/IP Stack

Web Server Httpd

Dtacq server dt100d

MDS server mdsipd

acq32-driv
kernel device driver

AcqXX boards on local pci bus

Ethernet

PCI BUS
DtNasDaq Software Architecture: 3 Layer

**Client**
- 3rd Party Client
- dt100rc
- MDS+

**Middle Layer Server**
- TCP/IP Stack
- Web Server Httpd
- Dtacq server dt100d
- MDS server mdsipd

**Existing Server Processes**
- see the same device interface, local or remote

**D-TACQ**
- Network Attached Satellite Data Acquisition
  - networked autonomous data acquisition unit

**Model**
- PCI BUS
- AcqXX boards on local pci bus
- Local PCI Boards (optional/legacy)

**Ethernet**
- dt100-hub
- dt100-proxy
- ACQ196CPCI/ACQ216CPCI

**Map**
- slot=ip-addr
Data Flow Detail

Device Driver [1]:
*dt100-hub*

Server (Perl) [1]:
*dt100-proxy*

data mover (C)[n]:
*dt100-connect*

HOST

HOST

2G ARM

User App

Device Driver [1]:
*dt100-hub*

Server (Perl) [1]:
*dt100-proxy*

data mover (C)[n]:
*dt100-connect*

HOST

HOST

2G ARM

Client: calls open() on local device

*dt100-hub*: commands *dt100-proxy* to start connection
*dt100-proxy* spawns dedicated *dt100-connect*
process to manage connection.

*dt100-connect* links back to *dt100-hub* to provide
command or data link back to client.

*dt100-proxy* cleans up on connection teardown
(timeout or explicit close()).

There may be many concurrent connections, each
with a dedicated *dt100-connect* process
Data Flow Detail: Possible simplification

NB: this configuration is NOT recommended because dt100-proxy has to select() on many (hundreds) of open sockets.
Two Tier Architecture

Client

- dt100rc
- Custom User App
- Web browser

2G ARM

- dt100d
- acq200control
- Webserver thttpd
- 2G Device Driver

Status Viewing, Control Possibilities
D-TACQ recommends Mozilla