EVLA Monitor & Control Software

Antenna Monitor and Control Subsystem (AMCS)

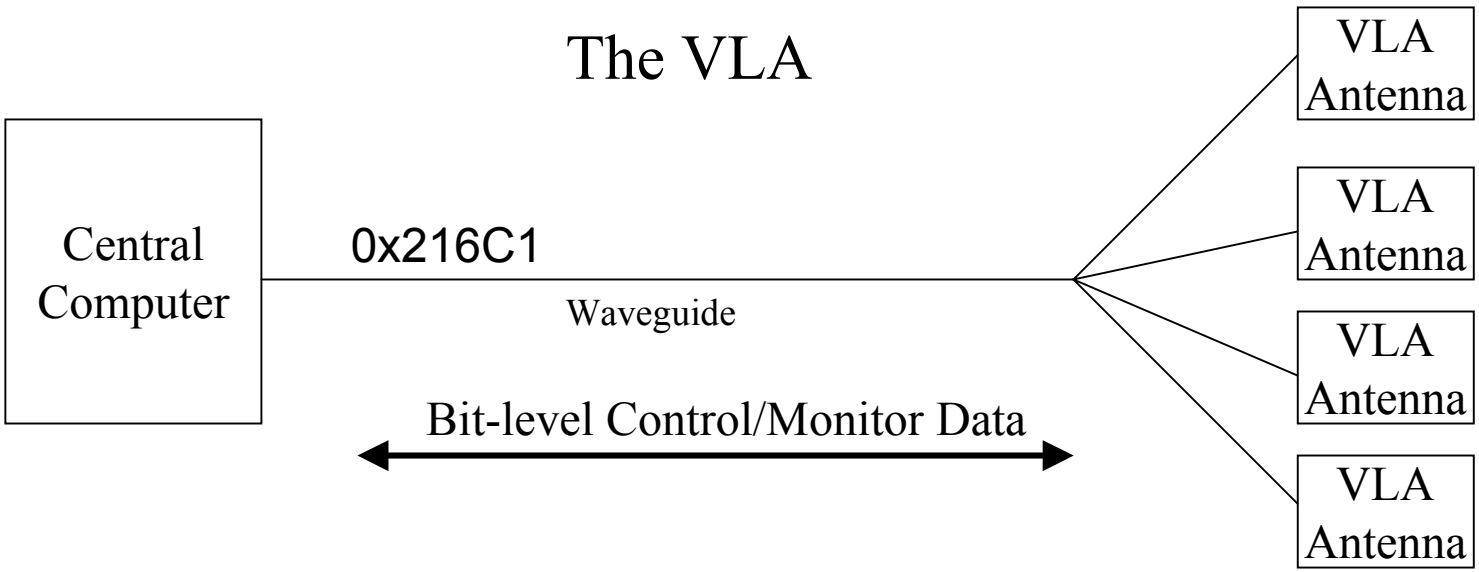
May 14-15, 2002

Kevin Ryan

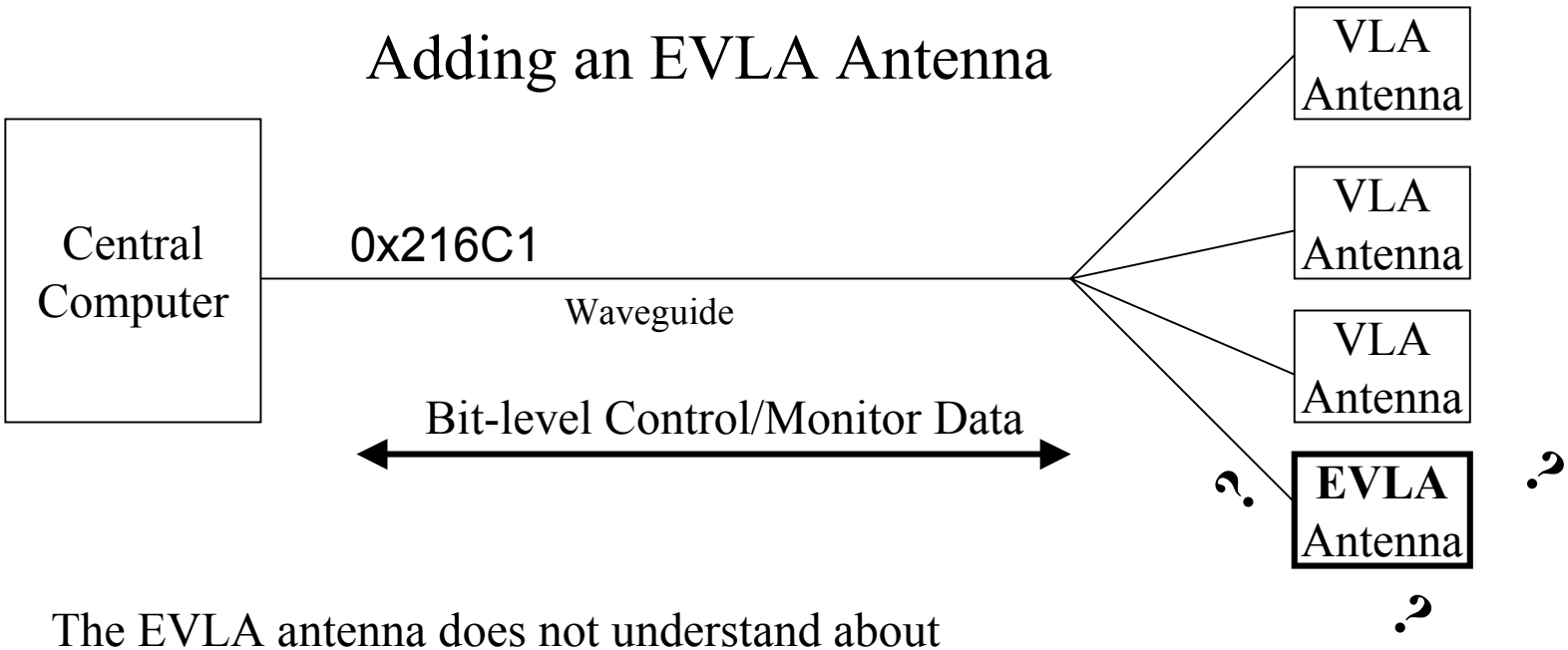
Contents

- Requirements that shape the design
  - (full requirements document can be found on EVLA web site)
- Preliminary design to satisfy them

- Factors that make EVLA different
  - Heterogeneous Array
  - Ethernet Based Communications
  - Widespread Operational Interface



To command the elevation to 47.0˚, set:
DS-0, MUX-0302 to 0x216C1
(Elevation Servo has 20-bit resolution, LSB = 1.236”)



The EVLA antenna does not understand about Data Sets, and MUXs and it may have a different kind of Elevation Servo (say a higher resolution).

Non-Identical Hardware

- Heterogeneous Array
  - From the onset will consist of both VLA and EVLA Antenna Types
  - Likely to add VLBA Antenna Types
  - If Phase II happens - add New Mexico Array (NMA) Antennas

VLA Antenna

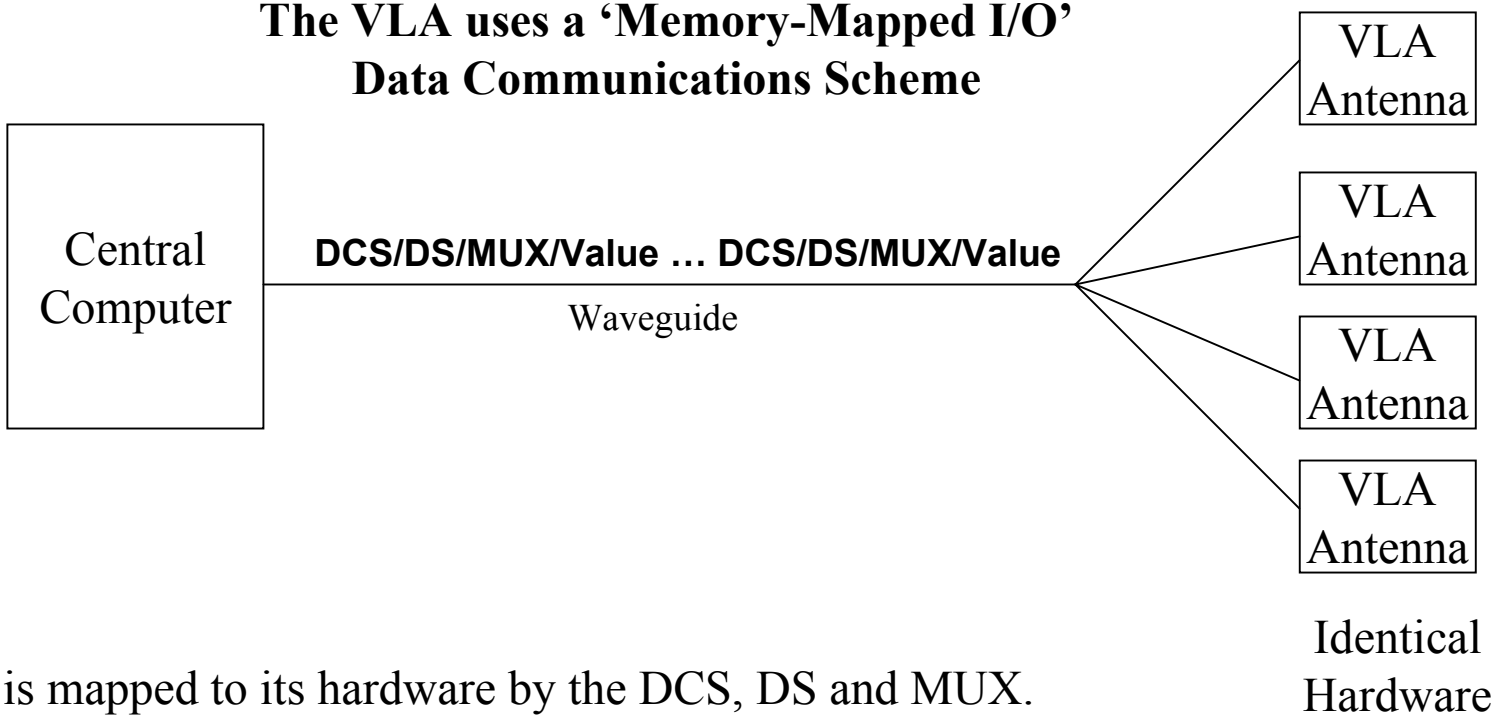
EVLA Antenna

NMA Antenna

VLBA Antenna

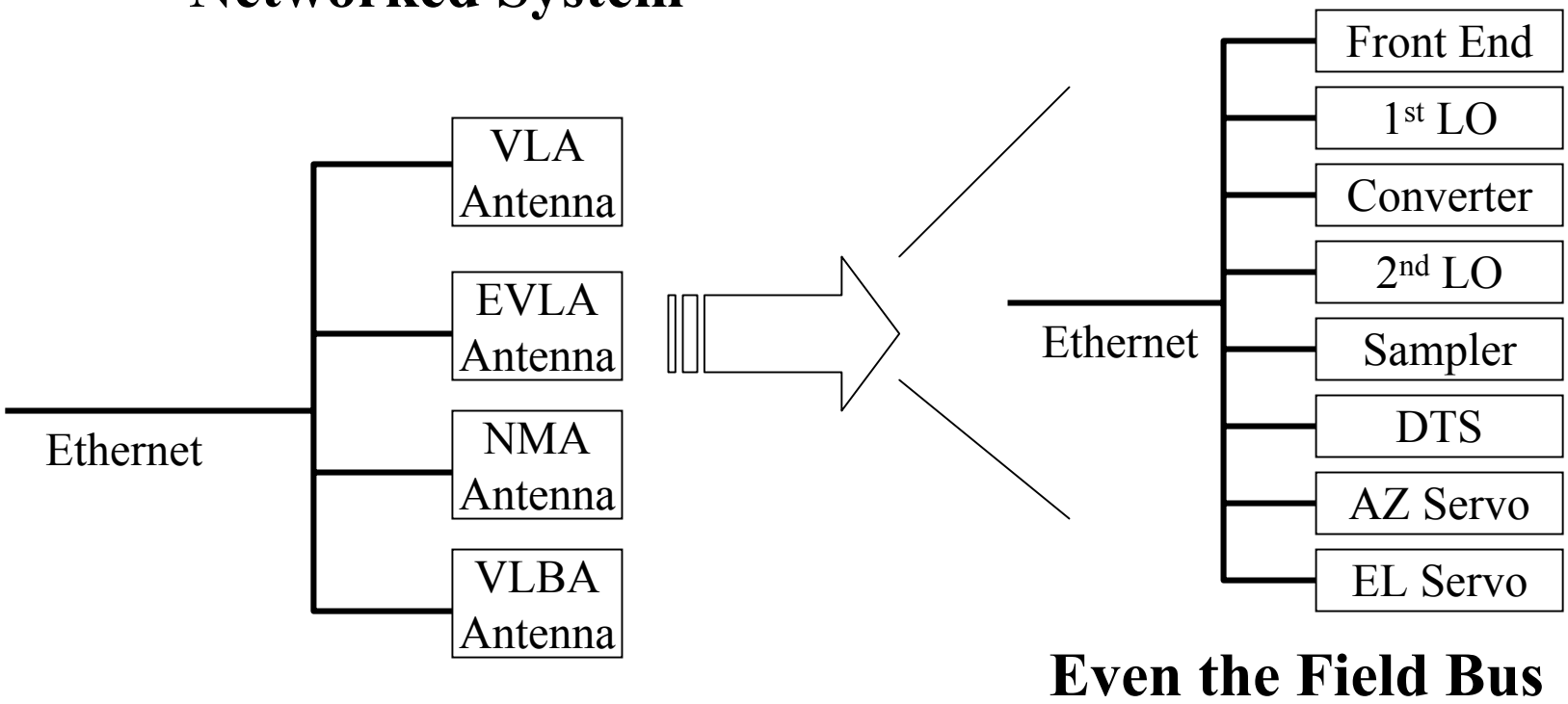
1. The New System must Accommodate Differences in Antenna Hardware.

The VLA uses a 'Memory-Mapped I/O' Data Communications Scheme



- Data is mapped to its hardware by the DCS, DS and MUX.
- VLBA uses MCB/HCB for direct comms with hardware.

EVLA will be a Highly Ethernet Networked System



Even the Field Bus

Every component will have its own processor and IP Address

- Ethernet
  - Is more complex than ‘memory-mapped’ I/O
  - Will require (additional) processing to handle the various protocols used (IP/TCP/UDP etc.)
  - Not known for its real-time characteristics

1. The New System must Accommodate Differences in Antenna Hardware.
2. Must Accommodate Higher-Level Network Type Data Communications.

EVLA will have a Widespread Operational Interface

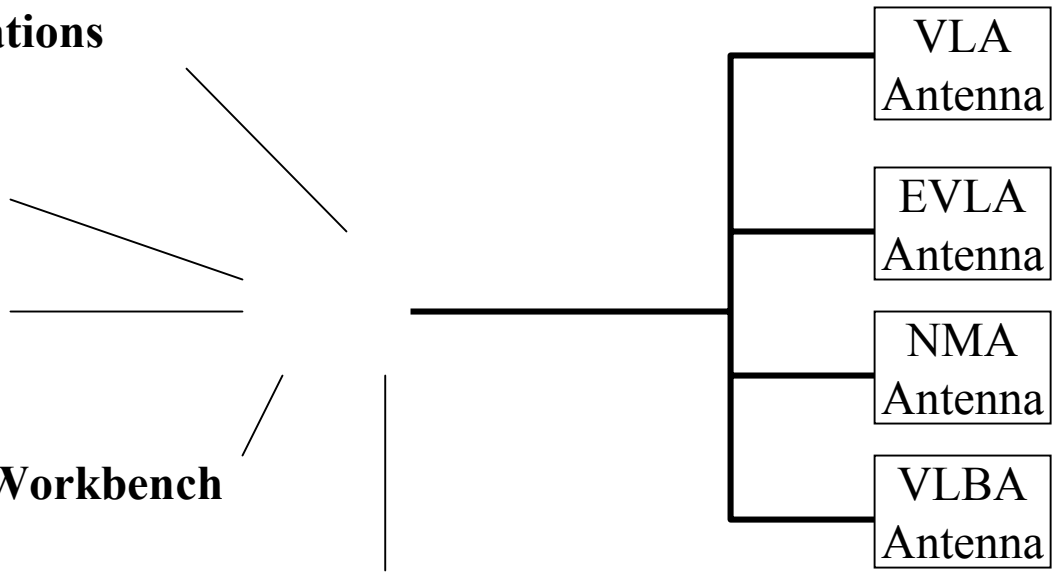
e2e Programmed Observations

Interactive Observing

Controlling from AOC

Controlling from Tech's Workbench

'Lurking' from a Browser over the Internet



1. The New System must Accommodate Differences in Antenna Hardware.
2. Must Accommodate Higher-Level Network Type Data Communications.
3. Must Serve a Variety of Users from a Variety of Physical Locations.

- Other Requirements – Performance
  - 100 μSec command start latency
  - Pointing updates every 50 mSec
  - Frequency change within band = 1 per sec
  - 'Nodding' source switch rate = 1 per 10 sec

- Other Requirements – Engineering
  - Flexibility and Maintainability
    - New observing modes supported with minimal software impact.
    - Hardware changes to have minimal software impact.
  - Use of Industry-wide technology
    - e.g. Avoid a single vendor specific middleware technology
    - Ethernet is example of non-vendor specific, industry wide technology

- Our design will take advantage of the fact that each subcomponent in the system will:
  - have ‘Intelligence’,
  - be networked with its own IP address.
- Distribute the functionality of the whole system amongst these intelligent subcomponents.
  - Hence, a true **distributed processing system**
    - As opposed to using the MIB as just memory mapped I/O gateway device to the hardware.

- Distributive Processing is:

“Putting the Intelligence where the Action is”

- This will allow us to design a **Modular** system
  - Separated into its various functional components
    - Arrays, sub-arrays,
    - Antennas (EVLA, VLA, NMA)
    - Antenna Subcomponent (Servo, LO, Front End, etc.)
  - Where each component contains its own processor
  - Which will present the ‘essence’ of that component to the rest of the system
    - Its ‘front panel’

    . . .

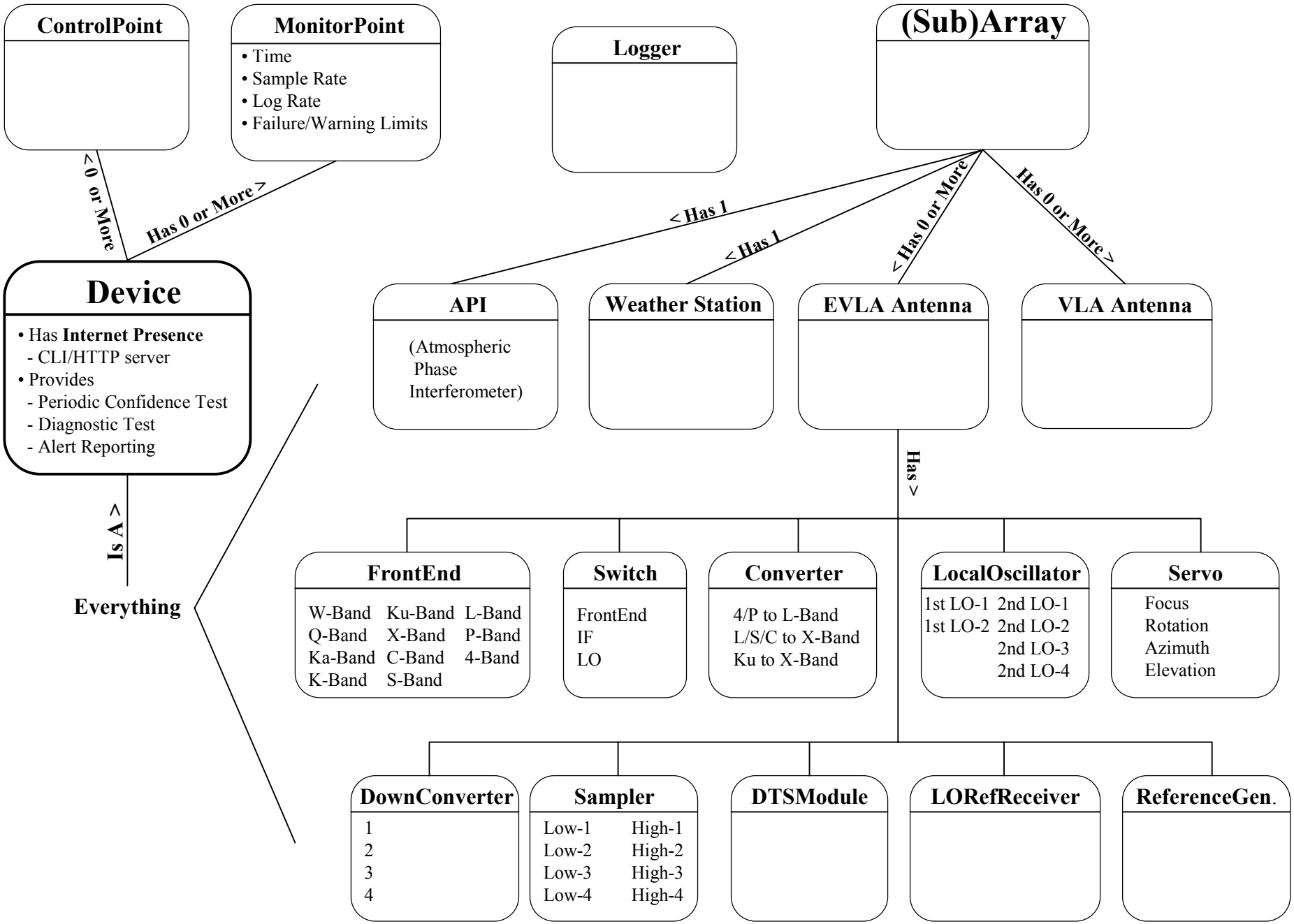
. . .

– But will **hide implementation details** of that component from the system
  - The Array Controller does not have to know the resolution of the Elevation Servo - it just sends a value of 47.0˚ in standard units.
    – The Elevation Servo will take care of converting the standard value to what it knows.
  - This means a component can change without affecting higher level software.
    – A servo with 24-bit resolution can replace the older 20-bit units.

- Modules will be fully autonomous
  - Components will control themselves
    - They just need to be told *what* to do, not *how* to do it.
  - Components will monitor themselves
    - They can do a much better job than a central computer
      - They alone 'know' themselves intimately
      - Can sample themselves at much higher rates than what could be done with a central computer over a bus.
        - » Transients such as voltage spikes can be detected and recorded
      - Won't clutter loggers and monitor displays with normal values

- Modules will be fully autonomous (cont.)
  - Components can operate independent of the system
    - On the workbench
      - Excepting physical limitations, an antenna component can be fully stimulated outside of the system.
  - Components will take on real-time responsibilities
    - Commands are preloaded across the not-so-deterministic Ethernet and executed at synchronous system tick times.

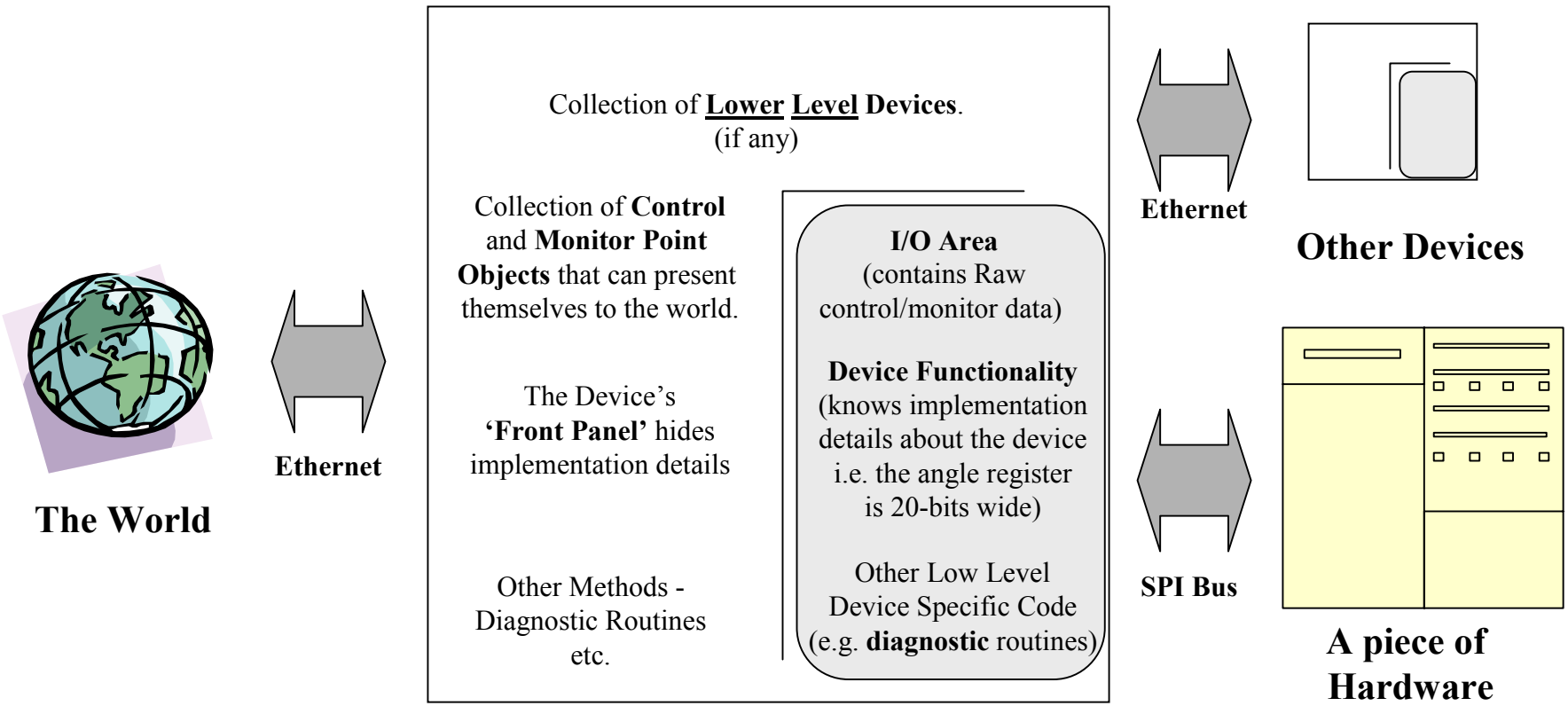
- A modular physical design maps nicely and naturally to an object-based software design.
  - Each physical component will become a software object.


ControlPoint
MonitorPoint
• Time
• Sample Rate
• Log Rate
• Failure/Warning Limits
Logger
(Sub)Array
< 0 or More
Has 0 or More >
< Has 1
< Has 1
< Has 0 or More
Has 0 or More >
Device
• Has Internet Presence
- CLI/HTTP server
• Provides
- Periodic Confidence Test
- Diagnostic Test
- Alert Reporting
API
(Atmospheric Phase Interferometer)
Weather Station
EVLA Antenna
VLA Antenna
Has <
Is A >
Everything
FrontEnd
W-Band Ku-Band L-Band
Q-Band X-Band P-Band
Ka-Band C-Band 4-Band
K-Band S-Band
Switch
FrontEnd
IF
LO
Converter
4/P to L-Band
L/S/C to X-Band
Ku to X-Band
LocalOscillator
1st LO-1 2nd LO-1
1st LO-2 2nd LO-2
2nd LO-3
2nd LO-4
Servo
Focus
Rotation
Azimuth
Elevation
DownConverter
1
2
3
4
Sampler
Low-1 High-1
Low-2 High-2
Low-3 High-3
Low-4 High-4
DTSModule
LORefReceiver
ReferenceGen.

KJR 5/6/2002

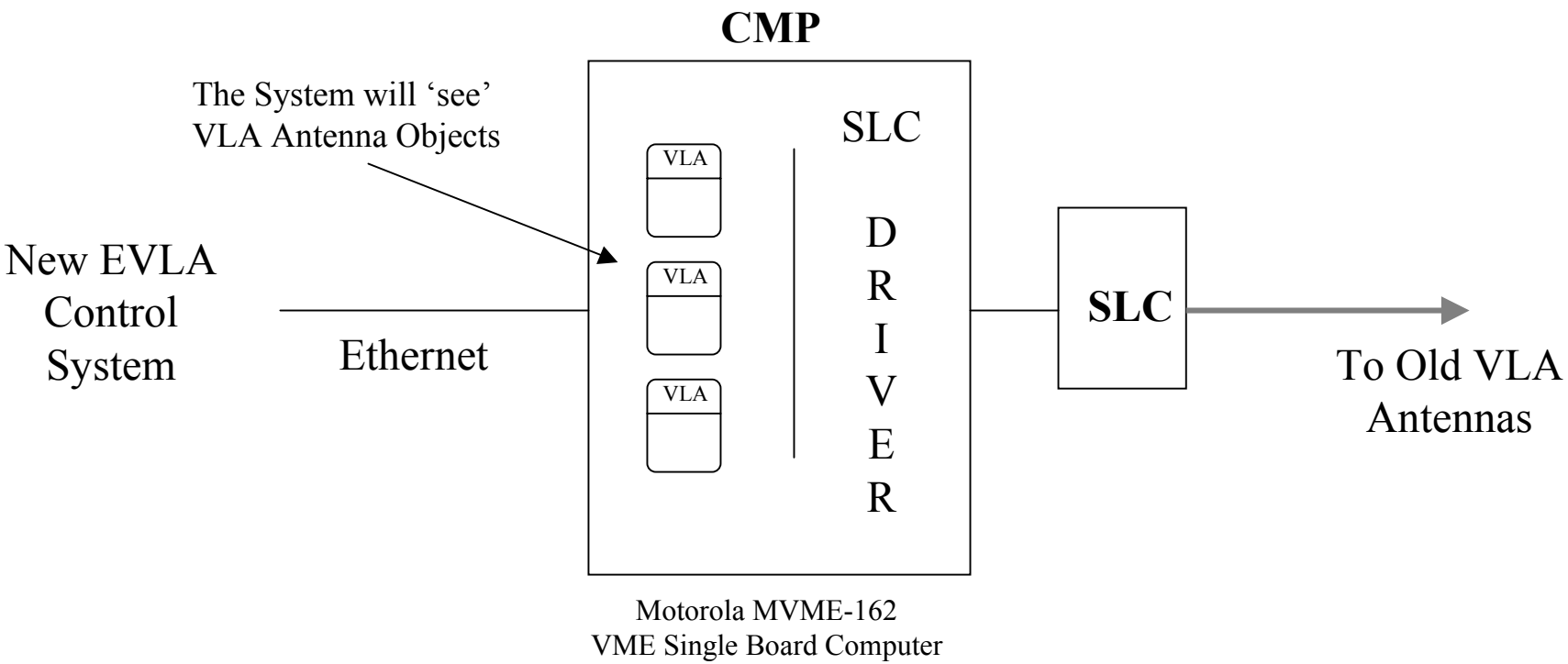
- The Device Class
  - Provides a common interface to outside world clients
    - Clients will only have to know about generic devices in order to 'discover' the rest of the system in detail.
  - Provides a natural hardware/software boundary
    - Method stubs for hardware engineering generated code
      - The hardware engineers will only write software that directly interfaces with the hardware component itself and will not have to concern themselves with integration into higher level system software.
      - This allows them to concentrate their expertise on the hardware itself.
  - Provides a great amount of code re-use.
    - Since every component inherits it general 'being' from it.

Antenna
Monitor & Control Software



- Devices contain ControlPoints and MonitorPoints
  - Also classes
  - Possibly independent threads
  - Can log themselves at desired intervals
  - Can alert the operator
- Devices provide Services
  - Clients use these services
    - e2e needs weather data feedback for Dynamic Scheduling. Instead of making AMCS aware of e2e and its specific needs, the AMCS just provides a weather service - whomever needs it just ‘takes’ it.

- An Antenna Device Class will be implemented for VLA antennas



- Risks
  - Very new technology (i.e. Web Services)
  - Puts a lot of responsibility on the MIB
- On the other hand …
  - Each MIB is probably more 'powerful' than a Modcomp
  - Industry supports this new technology
    - Java started out as a small footprint embedded language (but got temporarily off course when it was discovered how well it worked as an 'Internet Language'). It is now back on course as an embedded language that also does networking very well.
    - Internet Appliances and Web Services are currently a $Billion industry – they are not 'fad' technologies.