A Real-Time Publish-Subscribe Control Plane for a COTM Node

HPEC

24 September 2008

J. Darby Mitchell
Software Architect
Wideband Tactical Networking
MIT Lincoln Laboratory

This research was sponsored by the Department of the Army under Air Force Contract FA8721-05-C-0002.
Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Outline

- Introduction
  - Assumptions
  - Requests
- Problem Statement
  - Project Vision and System Context
  - System Architecture
  - Software Architecture Problem
- Software Architecture
  - Quality Attributes and Architectural Styles
  - Architectural Reasoning
  - Quality Attribute Tradeoffs
  - Runtime View
- Design and Implementation
  - Designing Topics
  - Topic Mapping
  - Handling Exceptions
- Conclusion
  - Lessons Learned
  - Acknowledgements

Outline

- Introduction
  - Assumptions
  - Requests
- Problem Statement
  - Project Vision and System Context
  - System Architecture
  - Software Architecture Problem
- Software Architecture
  - Quality Attributes and Architectural Styles
  - Architectural Reasoning
  - Quality Attribute Tradeoffs
  - Runtime View
- Design and Implementation
  - Designing Topics
  - Topic Mapping
  - Handling Exceptions
- Conclusion
  - Lessons Learned
  - Acknowledgements

Assumptions and Requests

- **Assumptions:**
  - You know what MIT Lincoln Laboratory does
  - You recognize the value of buying vs. building software
  - You know that there's no such thing as a "silver bullet"
  - You are familiar with the concepts of call-return middleware
  - Many of you are familiar with real-time publish-subscribe
- **Requests**
  - If you'd like to discuss any of these assumptions, please talk with me offline
  - Please hold your questions until the end of the talk

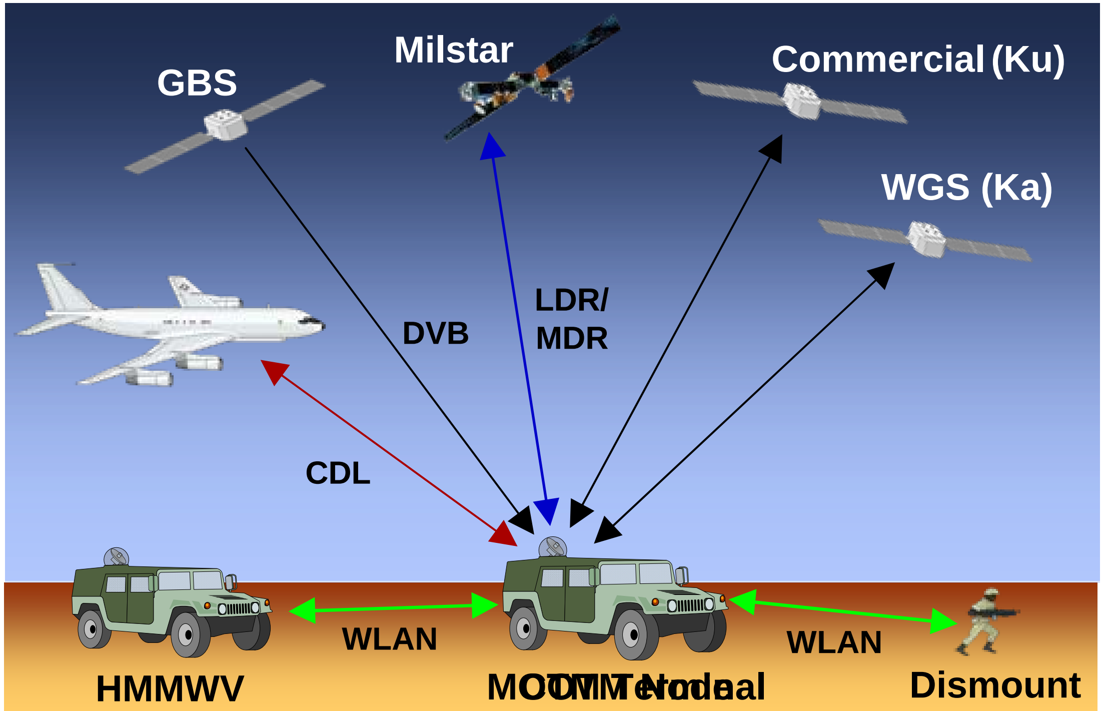
Outline

- Introduction
  - Assumptions
  - Requests
- **Problem Statement**
  - **Vision and System Context**
  - **System Architecture**
  - **Software Architecture Problem**
- Software Architecture
  - Quality Attributes and Architectural Styles
  - Architectural Reasoning
  - Quality Attribute Tradeoffs
  - Runtime View
- Design and Implementation
  - Designing Topics
  - Topic Mapping
  - Handling Exceptions
- Conclusion
  - Lessons Learned
  - Acknowledgements

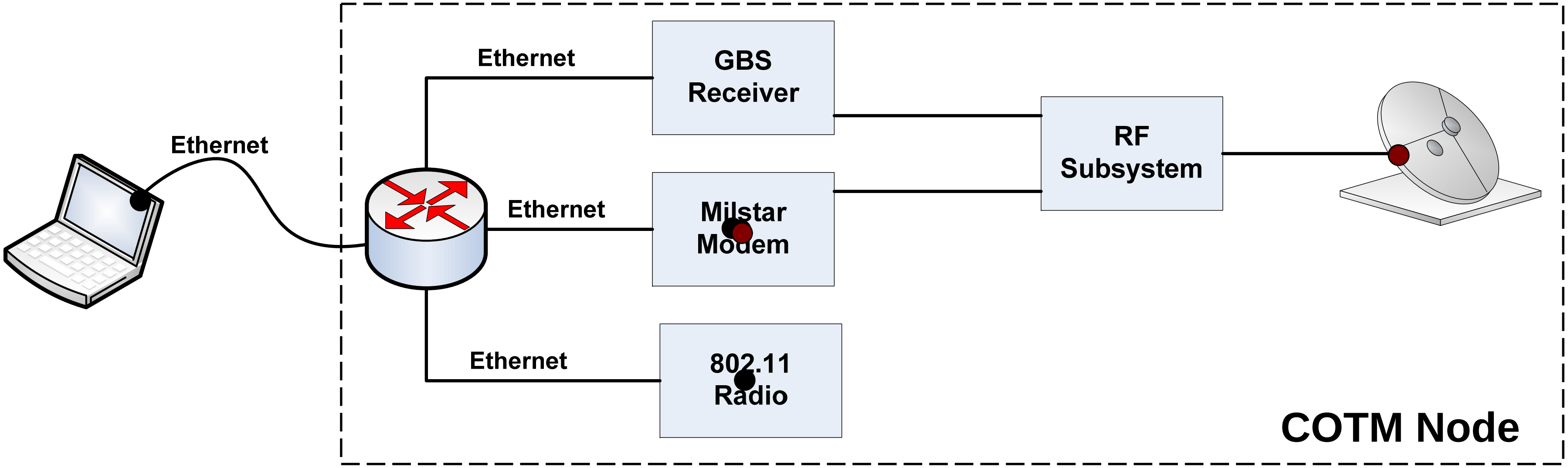
Vision: Evolution of Terminal to Node

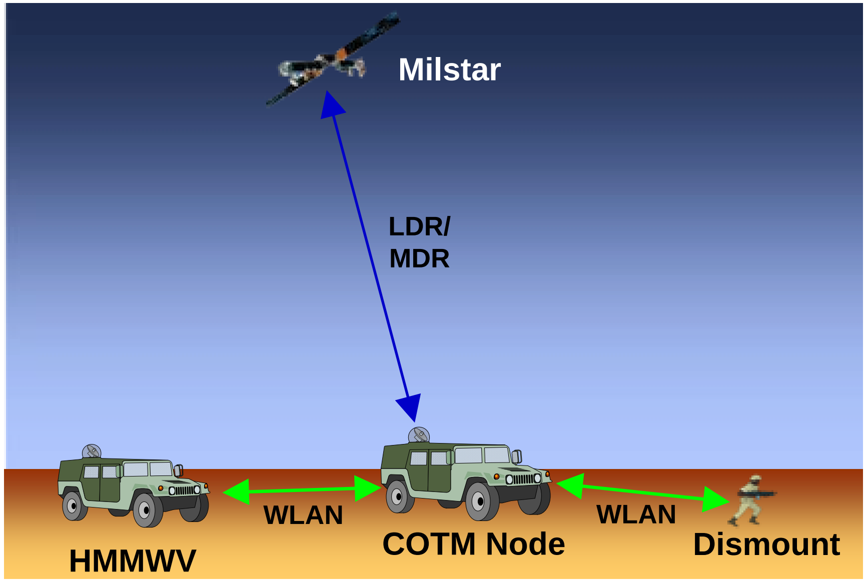
- Milstar On-The-Move (MOTM) Terminal
  - 3-axis positioner (MITLL)
  - Multi-band antenna and feed (44/30/20 GHz)
  - Blockage mitigation technology for COTM
  - IP over Milstar capability
  - Single link capability

- COTM Node
  - Manage multiple links
  - Compose links from modular HW/SW components
  - Facilitate integration of “stovepipe” COTS radios
  - Dynamic routing for cooperative networking
  - Support for insertion of additional radios



COTM Node: Phase 0 Data Plane

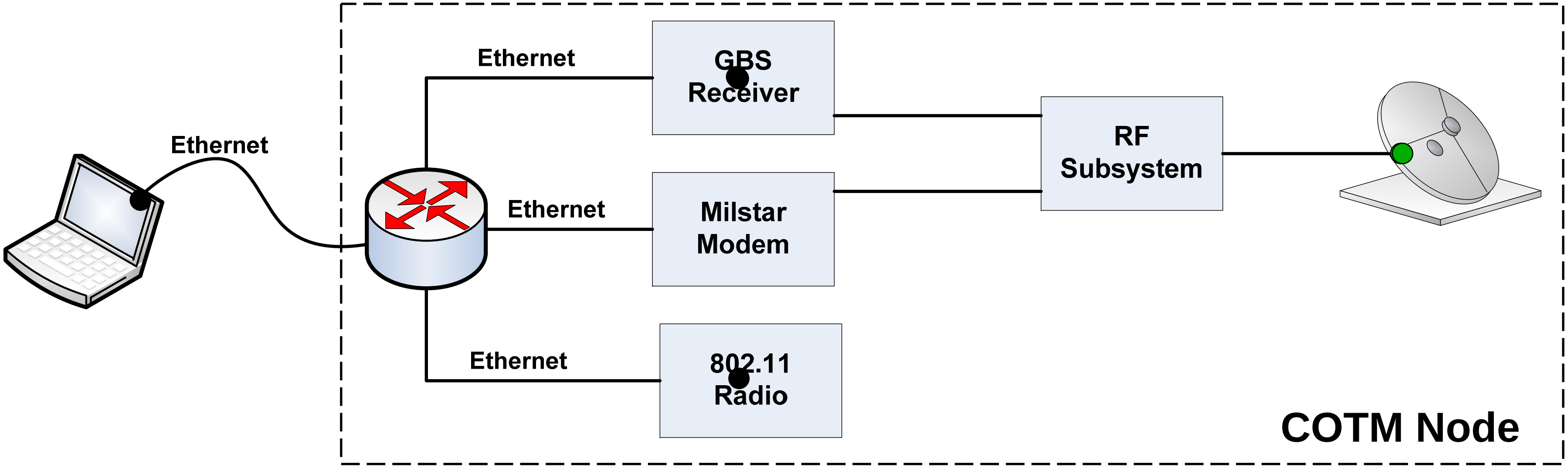


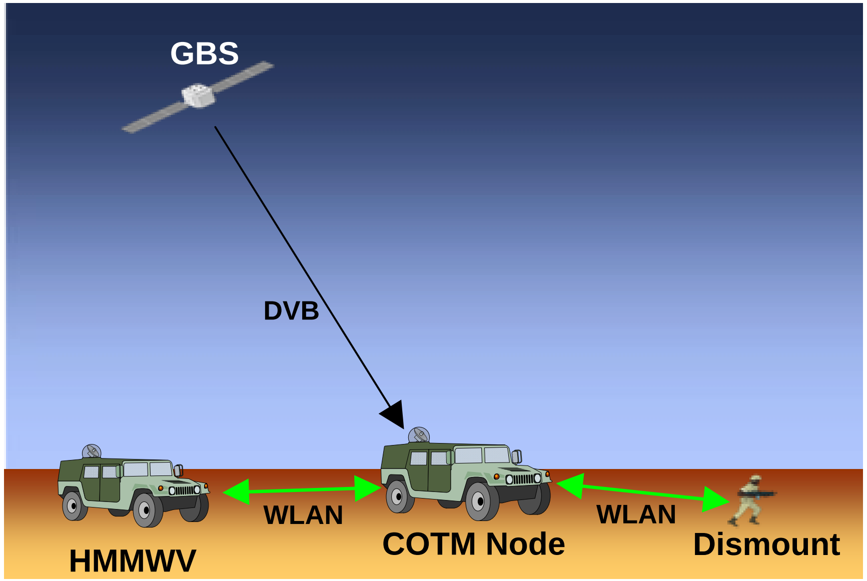


KEY

🔴 Milstar
🟢 DVB
⚫ IP

COTM Node: Phase 0 Data Plane

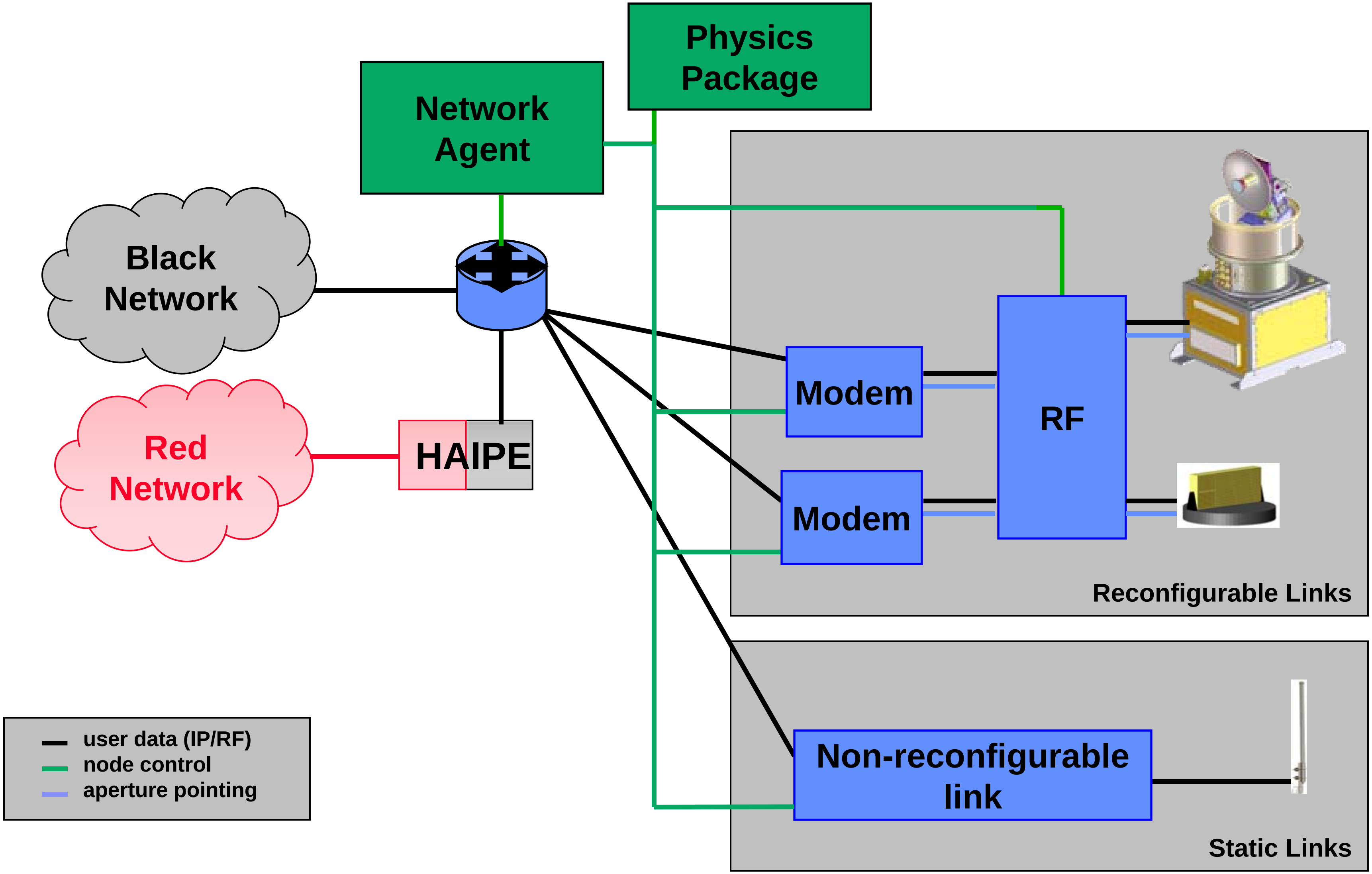




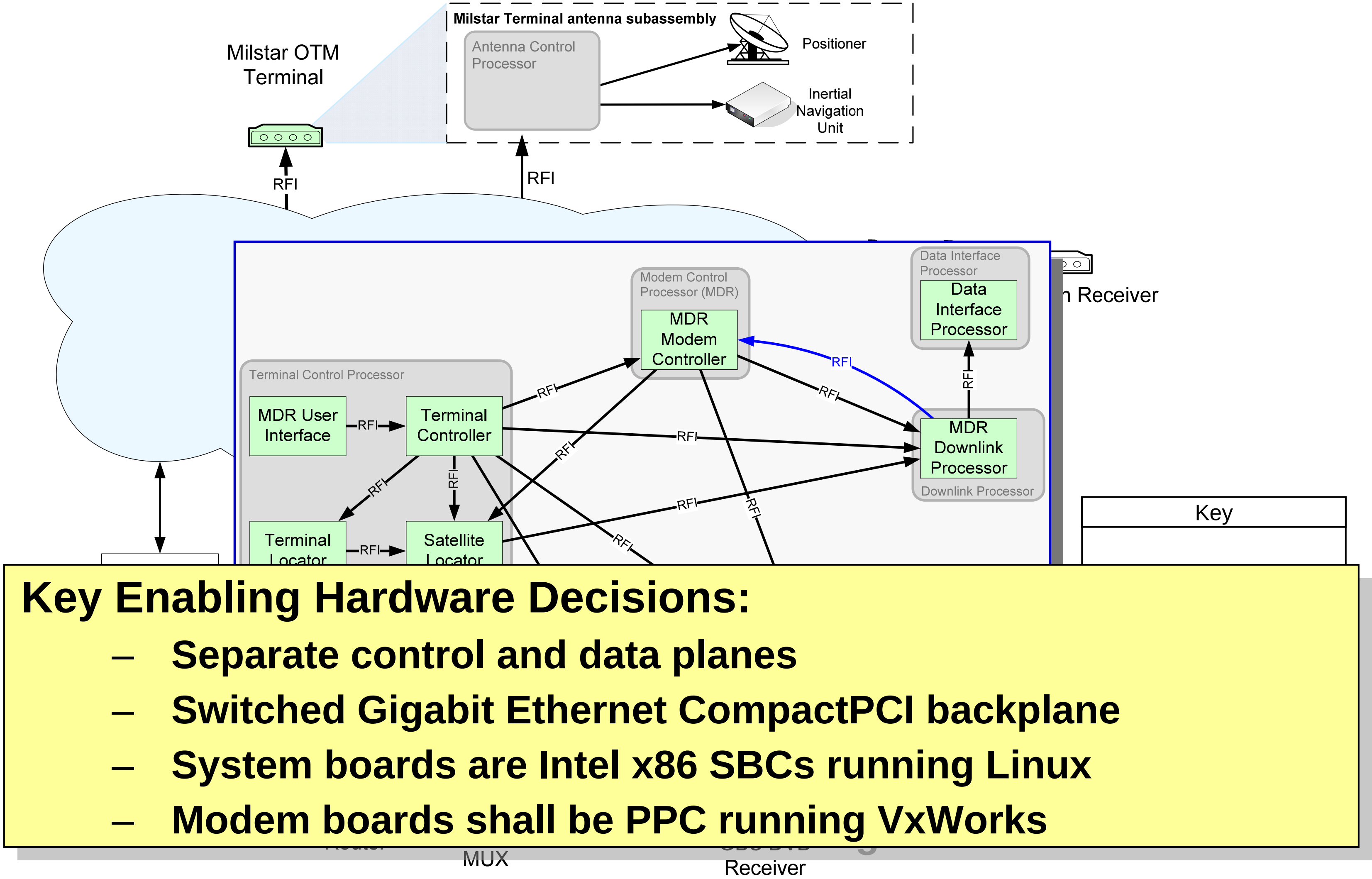
KEY

🔴 Milstar
🟢 DVB
● IP

System Architecture Concept



Software Architecture Problem



Key Enabling Hardware Decisions:

- Separate control and data planes
- Switched Gigabit Ethernet CompactPCI backplane
- System boards are Intel x86 SBCs running Linux
- Modem boards shall be PPC running VxWorks

Outline

- Introduction
  - Assumptions
  - Requests
- Problem Statement
  - Project Vision
  - System Context
  - System Architecture
  - Software Architecture Problem
- **Software Architecture**
  - **Quality Attributes and Architectural Styles**
  - **Architectural Reasoning**
  - **Quality Attribute Tradeoffs**
  - **Runtime View**
- Design and Implementation
  - Designing Topics
  - Topic Mapping
  - Handling Exceptions
- Conclusion
  - Lessons Learned
  - Acknowledgements

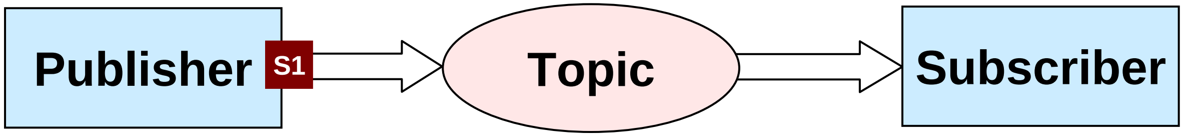
Quality Attributes and Architectural Styles

- **Essential Qualities**
  - **Predictability: Ability to anticipate task scheduling requirements**
  - **Timeliness: Ability to meet real-time constraints**
  - **Reliability: Ensures delivery of critical control data**
- **Desirable Qualities**
  - **Modularity: Facilitates decomposition and encapsulation**
  - **Extensibility: Facilitates addition of components (i.e. functionality)**
  - **Simplicity: Component development should be straightforward**
- **Architectural Styles**

Style	**Example Design Patterns**
Call-return	Client-server, forwarder-receiver
Implicit Invocation	GoF Observer, Publish-subscribe

Publish-Subscribe

- Subscribers register to collect *issues* to a particular Topic
- Publishers register to distribute *issues* to a particular Topic
- A Topic acts as a GoF Mediator to decouple Publishers and Subscribers



“VehiclePosition”

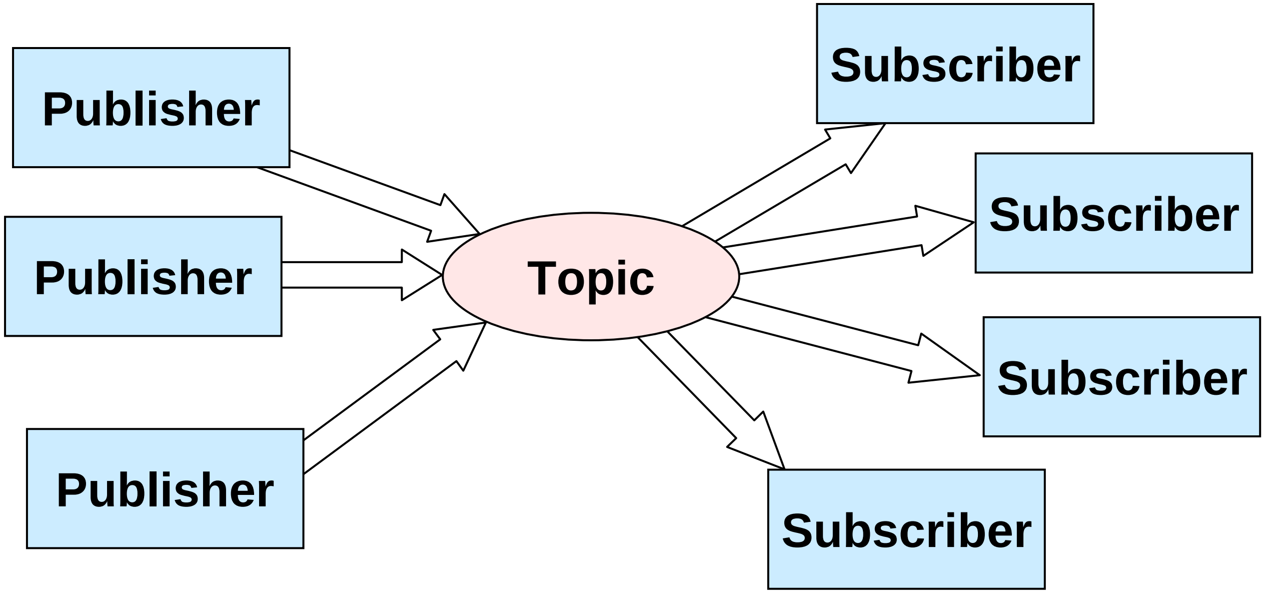
```
struct Position {
   double latitude;
   double longitude;
   double altitude;
}
```

Issue 1

```
longitude = -71.225
latitude = 42.447
altitude = 44.8
```

Publish-Subscribe

- May be zero or more publishers per topic
- May be zero or more subscribers per topic

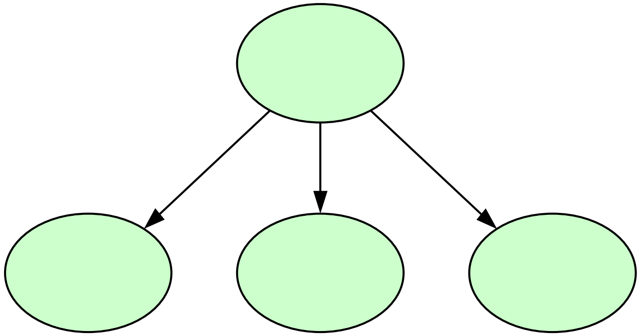


Architectural Reasoning

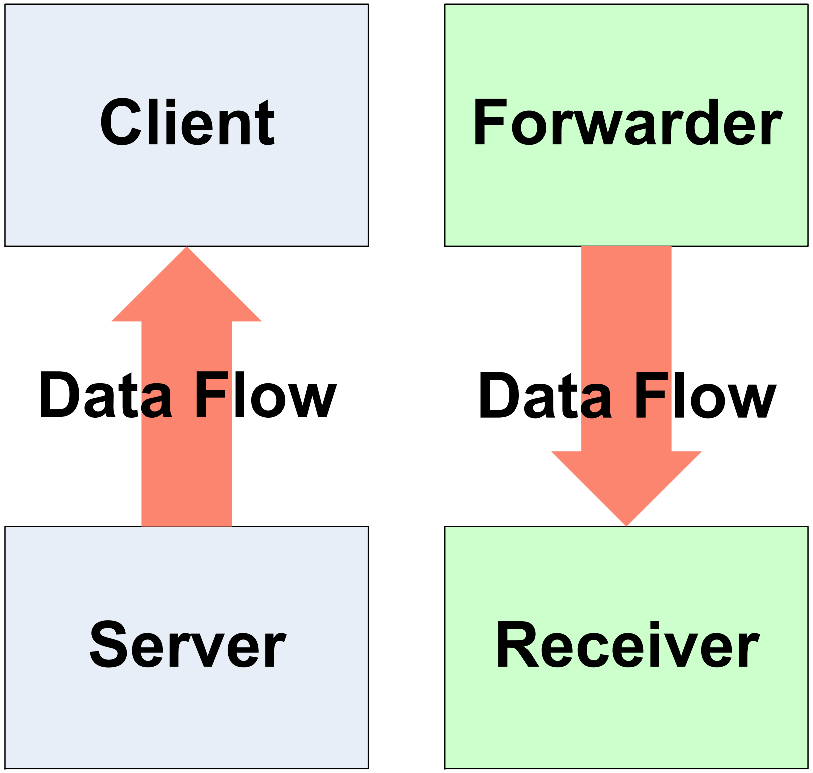
Publish-Subscribe

- Ideal for one-to-many or many-to-many relationships
- Promotes predictability
- Data-centric (data identifier)
- No assumption of existence
- Data source always initiates communication

Result: decoupled interaction



Call-return



Client

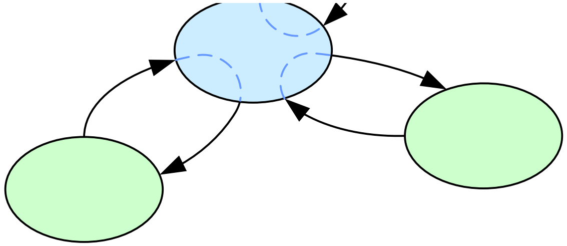
Forwarder

Data Flow

Data Flow

Server

Receiver



Quality Attribute Tradeoffs

Publish-Subscribe

- Timeliness
- Predictability
- Modularity
- Extensibility

Call-return

- Reliability
- Simplicity

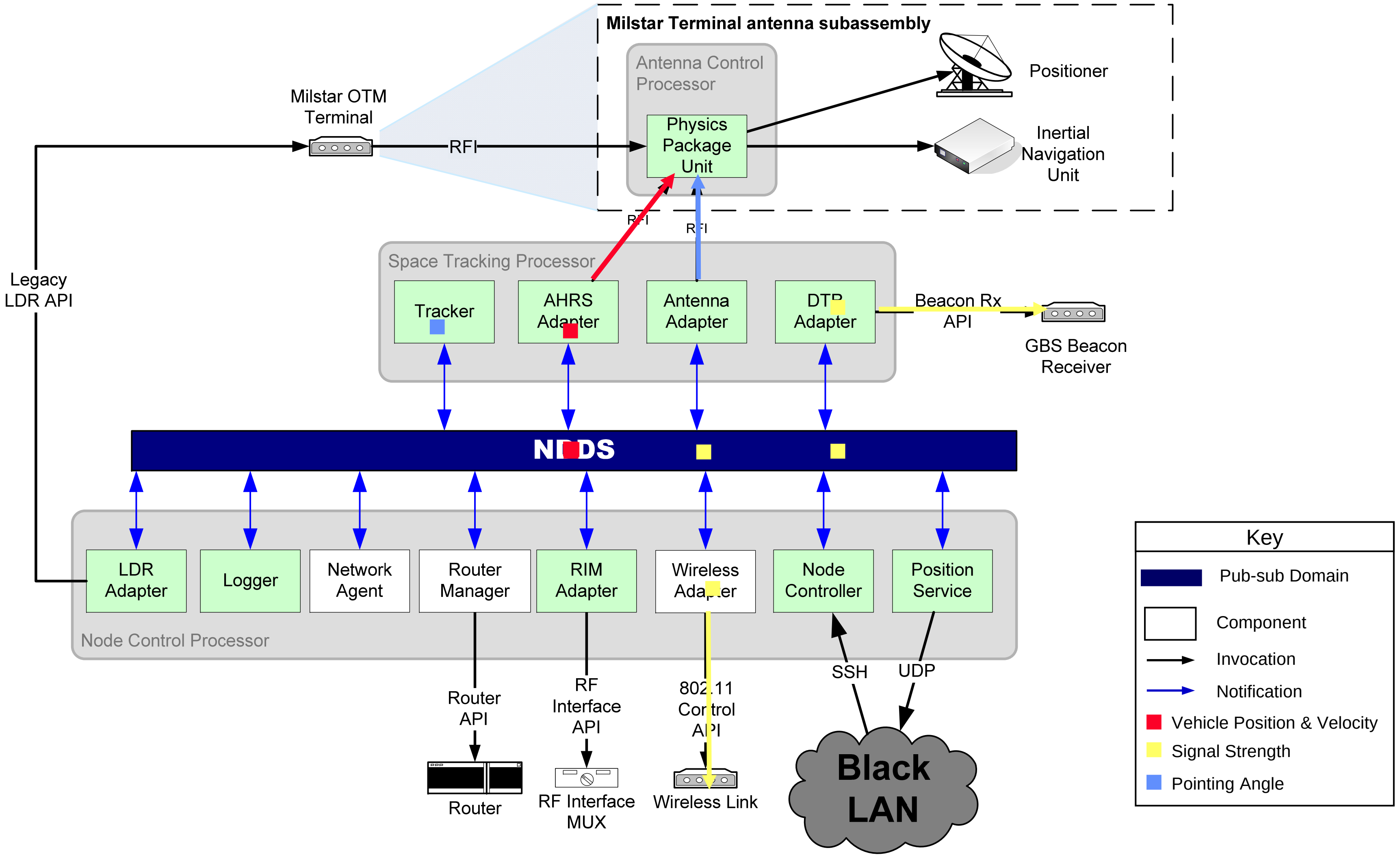
Real-time Publish-Subscribe with NDDS

- The Netw publish-
  subscrib vations, Inc.
- NDDS w ns
- Provides collection
  and dist
- At the ti pecification
  was still
- RTI was pecification
  effort
- RTI had DS spec
- RTI had

Quality of Service Parameters

Deadline	Presentation
Destination Order	Reliability
Durability	Resource Limits
Entity Factory	Time-Based Filter
History	Transport Priority
Latency Budget	Group Data
Lifespan	Topic Data
Liveliness	User Data
Ownership	Reader Data Lifecycle
Ownership Strength	Writer Data Lifecycle
Partition	

Software Architecture: Runtime View



Outline

- Introduction
  - Assumptions
  - Requests
- Problem Statement
  - Project Vision
  - System Context
  - System Architecture
  - Software Architecture Problem
- Software Architecture
  - Quality Attributes and Architectural Styles
  - Architectural Reasoning
  - Quality Attribute Tradeoffs
  - Runtime View
- **Design and Implementation**
  - **Designing Topics**
  - **Topic Mapping**
  - **Handling Exceptions**
- Conclusion
  - Lessons Learned
  - Acknowledgements

Designing Topics

- **Samples – periodic, independent measurements of the environment**

  Examples:
  - **Vehicle position and velocity**
  - **Link state**
  - **Modem signal strength**
  - **Satellite location and velocity**
  - **UTC Time**

RELIABILITY: BEST EFFORT

HISTORY: KEEP LAST

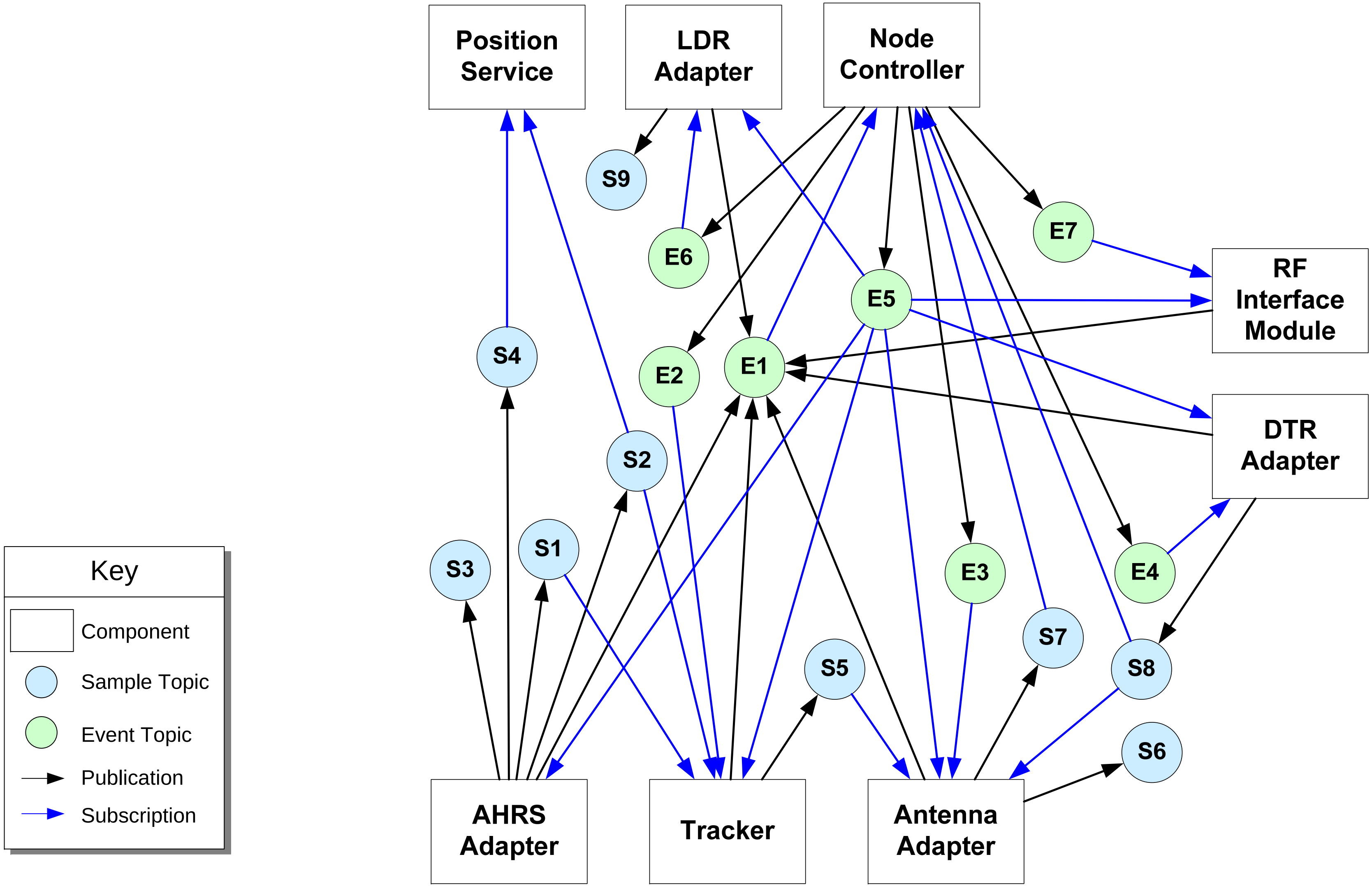
- **Events – sporadic, relative changes in system state**

  Examples:
  - **Link formation and teardown**
  - **Status messages**
  - **Error messages**
  - **Parameter changes**
  - **Routing changes**

RELIABILITY: RELIABLE

HISTORY: KEEP ALL

Topic Mapping



Topic Mapping: Samples

Samples

S1 – UTCTime

S2 – AHRSLocation

S3 – AHRSDisplacement

S4 – AHRSVelocity

S5 – AntennaReferenceAngle

S6 – AcquisitionMetric

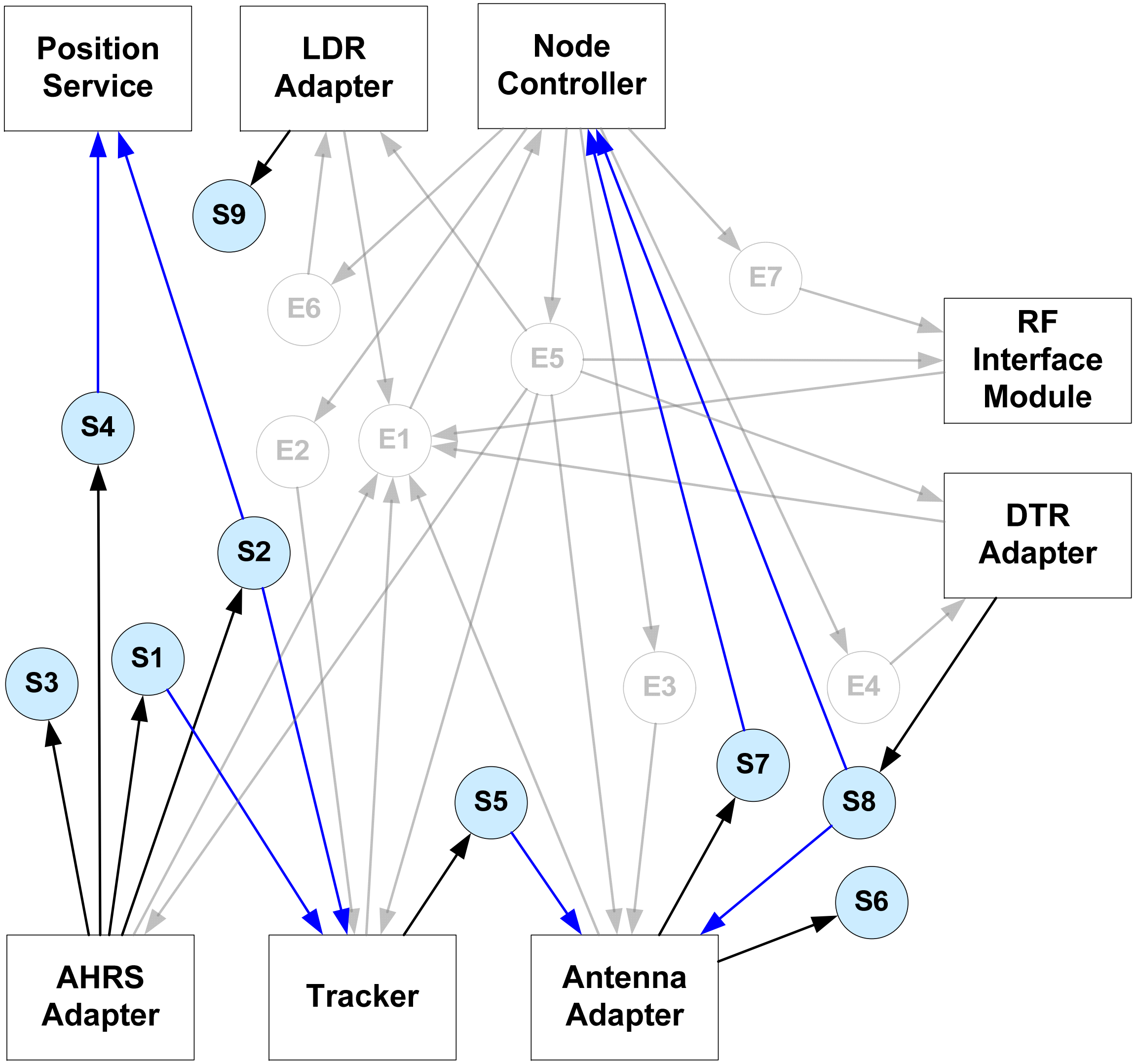
S7 – AntennaAngles

S8 – DTRSamples

S9 – LDREnergyMetric

Key

□	Component
○	Sample Topic
○	Event Topic
→	Publication
→	Subscription



Topic Mapping: Events

Events

E1 – DeviceStatus

E2 – TrackCommand

E3 – AntennaCommand

E4 – DTRParams

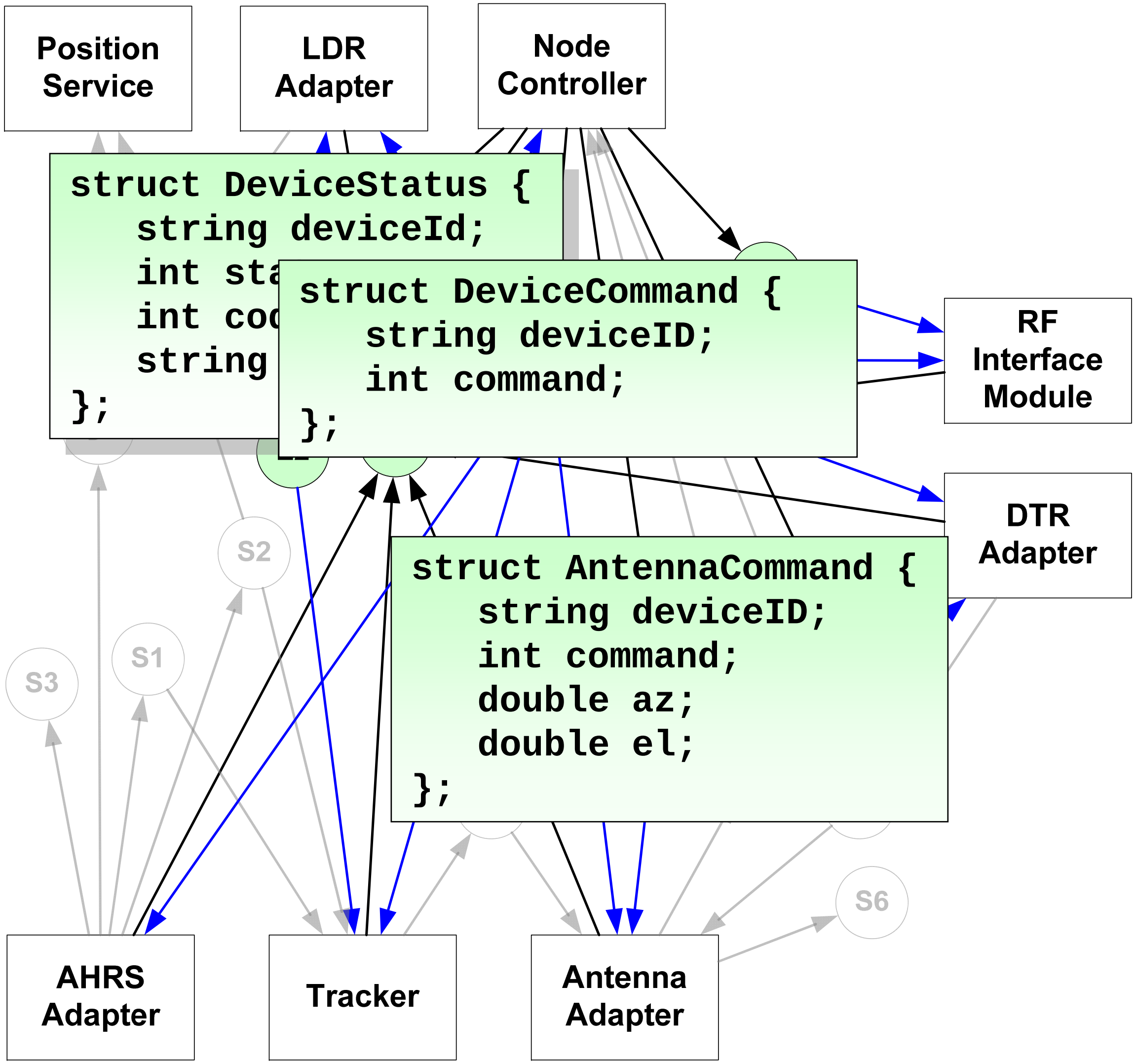
E5 – DeviceCommand

E6 – LDRCommand

E7 – RIMCommand

Key

- ▭ Component
- ● Sample Topic
- ● Event Topic
- → Publication
- → Subscription



```
struct DeviceStatus {
    string deviceId;
    int sta
    int coo
    string
};
```

```
struct DeviceCommand {
    string deviceID;
    int command;
};
```

```
struct AntennaCommand {
    string deviceID;
    int command;
    double az;
    double el;
};
```

Outline

- Introduction
  - Assumptions
  - Requests
- Problem Statement
  - Project Vision
  - System Context
  - System Architecture
  - Software Architecture Problem
- Software Architecture
  - Quality Attributes and Architectural Styles
  - Architectural Reasoning
  - Quality Attribute Tradeoffs
  - Runtime View
- Design and Implementation
  - Designing Topics
  - Topic Mapping
  - Handling Exceptions
- **Conclusion**
  - **Lessons Learned**
  - **Acknowledgements**

Lessons Learned

- Using publish-subscribe:
  - Made component development slightly more complicated
  - Greatly facilitated software integration
  - Enabled us to successfully defer some components, while still making progress on the project
  - Is not as straightforward when you are marshalling parameters with commands
- Respect the invariants of the architectural style:
  - NodeController could be killed and later restarted with no detrimental impact to system in steady state
  - Debug topics could be published for later use with negligible impact on system performance
- Actively managing consistency of QoS settings was essential
- Having a commercial vendor to delegate middleware support concerns to was very helpful

Acknowledgements

- Sponsor: PM WIN-T, Ft. Monmouth
- Group Leaders: Dr. Marc Zissman and Scott Sharp
- Systems Engineer: Dr. Andrew Worthen
- RF team: Dr. Jim Vian, John Murphy, Ted O’Connell
- Hardware team: Steve Pisuk, John Delisle, Jason Hillger
- Software team: Darby Mitchell, Curran (Nachbar) Schiefelbein, Marc Siegel, Marie Heath
- Testing team: Dr. Mark Smith, Ted O’Connell

Current Work

- TSAT Reference Terminal (TRT)
  - A joint project with Group 64 based on TRUST-T
  - A COTM Node that is based on the Software Communications Architecture (SCA) for software defined radios.
  - The SCA mandates the use of CORBA middleware, so DDS will not be used.
- Network and Link Emulation Testbed (NLET)
  - A distributed network emulation testbed
  - Uses DDS for a distributed real-time context simulation and real-time dynamic control of link emulation.

References

- Mitchell et. al. "Applying Publish-Subscribe to COTM Node Control", *MIT Lincoln Laboratory Journal*, Volume 16, No. 2, December 2006.

  http://www.ll.mit.edu/news/journal/journal.html

- L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, Addison Wesley, 1998.
- Garlan, D. and M. Shaw, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

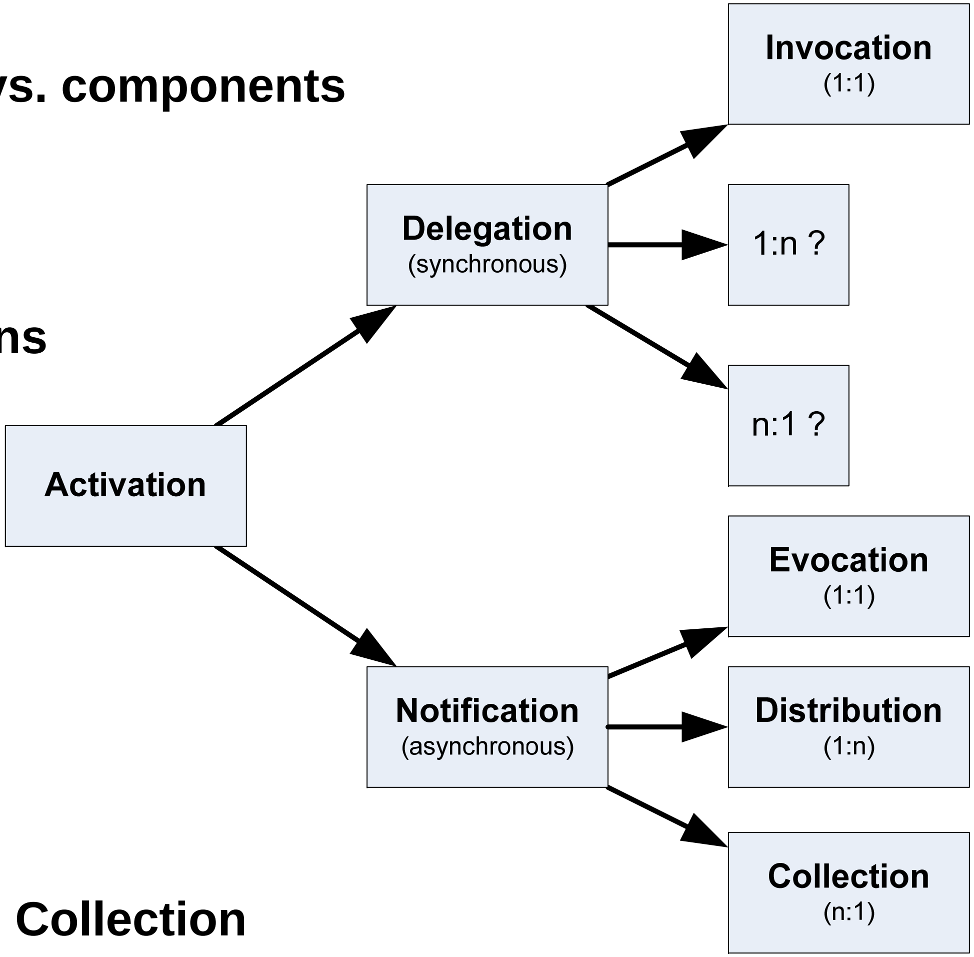
Questions?

mitchelljd@ll.mit.edu

Backup Slides

Reasoning About Connectors

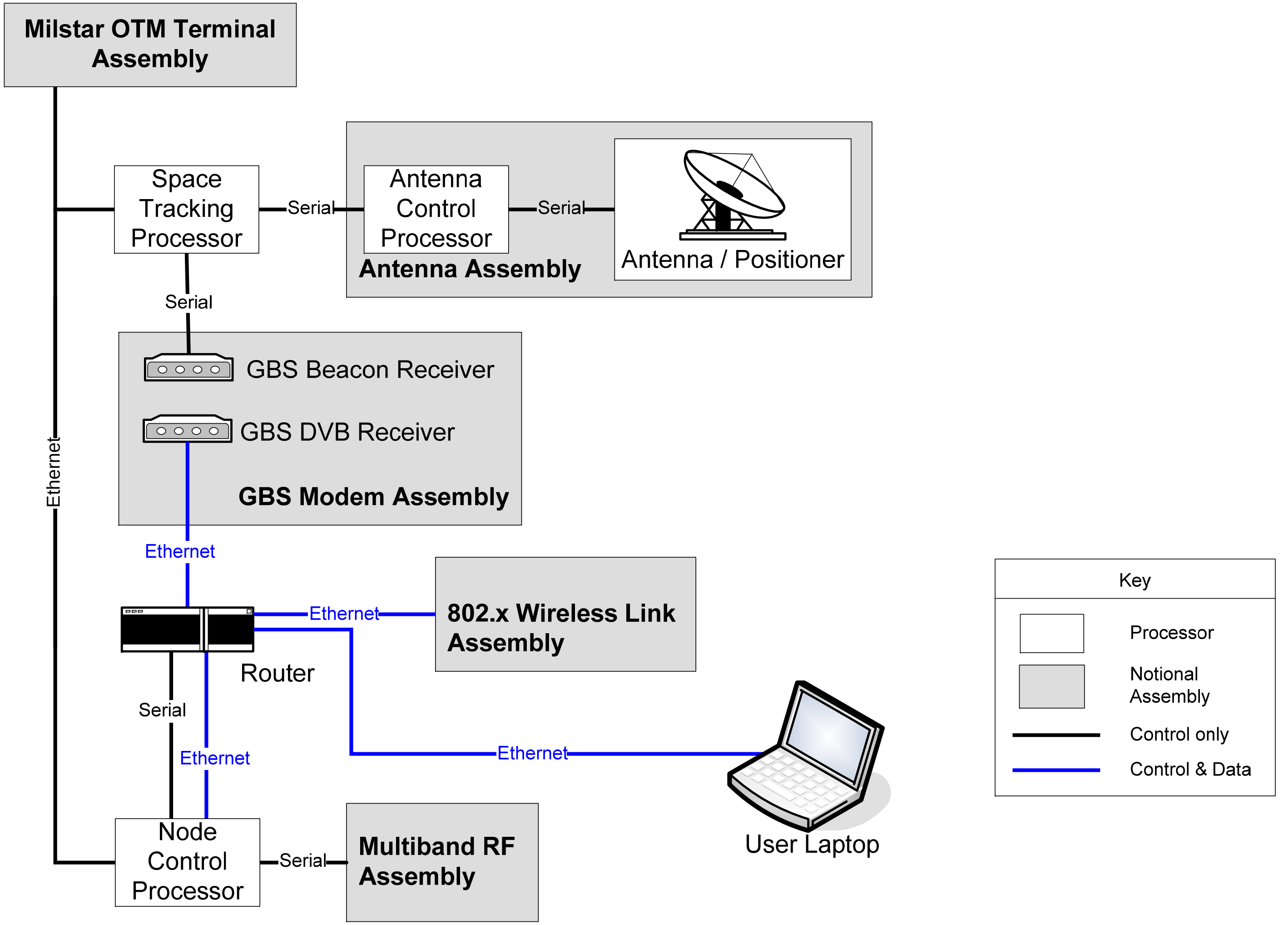
- Reasoning about connectors vs. components
- Consider several dimensions:
  - synchronous vs. asynch
  - cardinality (1 : 1 vs. 1 : n)
- Ignore implementation concerns



Publish-subscribe =

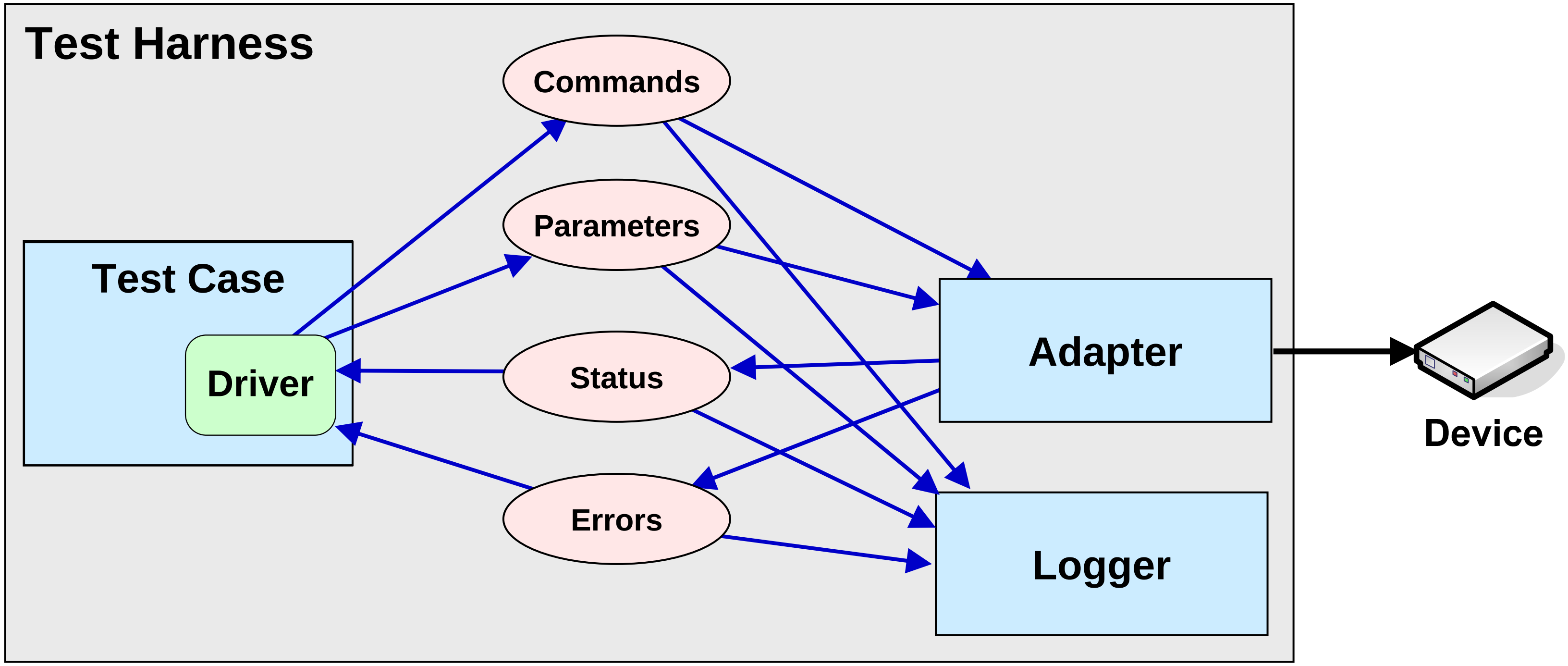
Distribution + Implicit Invocation + Collection

System Architecture: Connection View



Driver and Adapters

- There is a one to one relationship between Drivers and Adapters
- Node Controller only interacts with an Adapter through its Driver
- A Driver caches Status and Error updates from its Adapter
- Adapters may interact with other Adapters



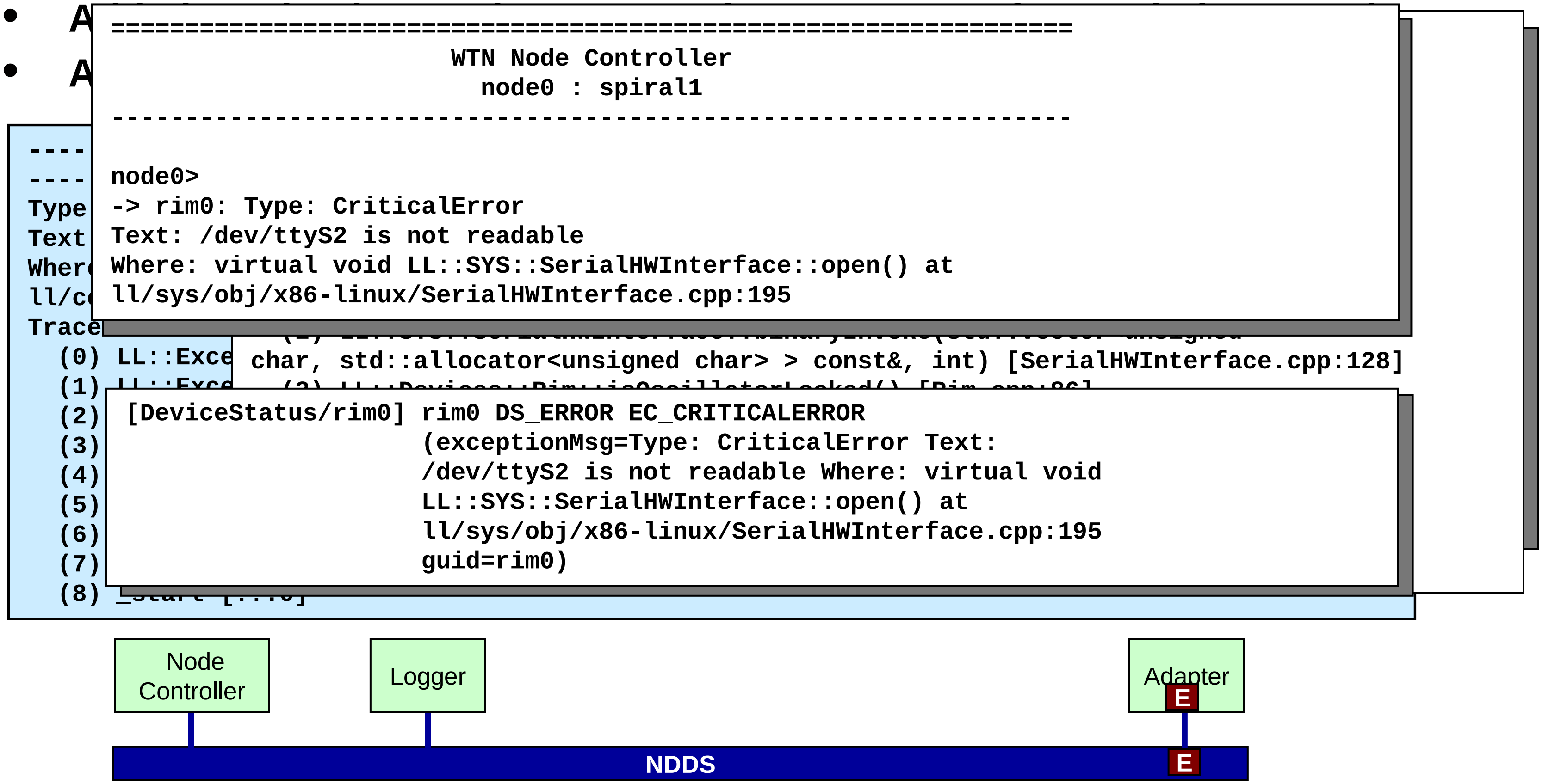
Exception Handling

- **Based on concepts from online article:**

  Agarwal, Sachin, “C++ Exception-Handling Tricks for Linux”, IBM Software Labs, Feb 2005.

  http://www-128.ibm.com/developerworks/linux/library/l-cppexcep.html?ca=dgr-lnxw1fExceptionTricks

- **A**
- **A**



```
================================================================
                      WTN Node Controller
                        node0 : spiral1
----------------------------------------------------------------

node0>
-> rim0: Type: CriticalError
Text: /dev/ttyS2 is not readable
Where: virtual void LL::SYS::SerialHWInterface::open() at
ll/sys/obj/x86-linux/SerialHWInterface.cpp:195
```

```
----
----
Type
Text
Where
ll/c
Trace
  (0) LL::Exce
  (1) LL::Exce
  (2)
  (3)
  (4)
  (5)
  (6)
  (7)
  (8) _start [...0]
```

```
  (1) LL::SYS::SerialHWInterface::binaryInvoke(std::vector<unsigned
char, std::allocator<unsigned char> > const&, int) [SerialHWInterface.cpp:128]
  (2) LL::Devices::Rim::isOscillatorLocked() [Rim.cpp:86]
```

```
[DeviceStatus/rim0] rim0 DS_ERROR EC_CRITICALERROR
                    (exceptionMsg=Type: CriticalError Text:
                    /dev/ttyS2 is not readable Where: virtual void
                    LL::SYS::SerialHWInterface::open() at
                    ll/sys/obj/x86-linux/SerialHWInterface.cpp:195
                    guid=rim0)
```