Middleware for Partitioned Systems on Airborne Platforms

a briefing at the

AFRL Safe & Secure Systems & Software Symposium (S5)

15-17 June 2010
Dayton, Ohio

Outline

- **Why Partitioning for Airborne Platforms**
- **Composability, Partitioning, and Delayed Binding**
- **Realizing Delayed Binding**

Proliferation of UAV Roles

- **Increasing demand for small UAVs**
  - **Civil**
  - **Military**
- **Increasing demand for adaptable UAVs**
  - **Product families to handle multiple missions**
  - **Sensors and other payloads**
  - **Design time and run-time**
- **Contingency management**
  - **Fault tolerance**
  - **Long duration missions (e.g., Vulture)**
- **Increasing safety requirements**
  - **As UAVs move from the rare to the routine, more stringent safety and safety certification required**
  - **Access to civil airspace**

A Family of Boeing Small UAVs

ScanEagle

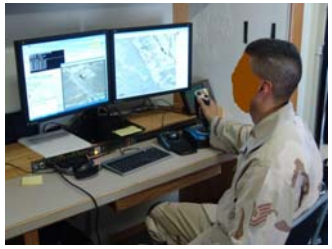
Over 320,000 Combat Flight Hours

Coalition Forces

Integrator

USMC

US Navy



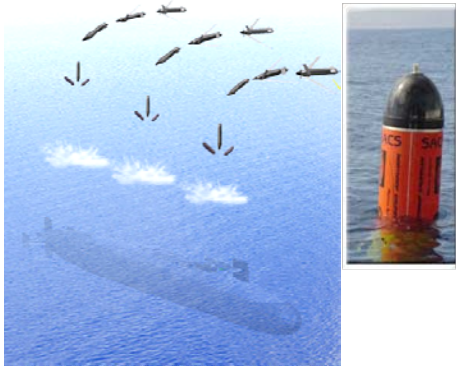
US Air Force 820th Security Forces

ScanEagle Compressed Carriage

Surfaces Deployed In Flight

Multiple SECCs in Air-Launch Palletized Containers

Underwater Launch



Multiple SECCs on Wing of a Strike Fighter

Why Partitioning for Airborne Platforms

- **Small UAVs in particular are already mixed criticality systems**
  - **At the limit, when you only have one CPU, it has to do everything**
- **Partitioning will be a *de facto* requirement**
  - **Mixing criticalities because of resource constraints**
  - **Increasing certification requirements will drive separating criticalities**
    - **Demand for demonstration of higher assurance for more critical functions**

Composability and Middleware

- **Composition is the most promising approach to spanning the space of desired UAVs**
  - **Compose UAV software from components**
  - **Product line reuse**
    - **Reuse software and certification**
- **Partitioning supports composition by providing a leak proof interface between components in different partitions**
- **Middleware provides a framework for composition and can provide shared functionality across a product family**

Composability, Partitioning, and Delayed Binding

- **Compositionality improves as binding times move later in the lifecycle**
  - **Design time**
  - **Implementation time**
  - **Compile time**
  - **Configuration time**
  - **Run time**
- **Explicit vs implicit**
  - **Rate monotonic schedule is implicitly defined at configuration time**
- **Scheduling**
- **Communications**

Delaying Binding Time for Communications

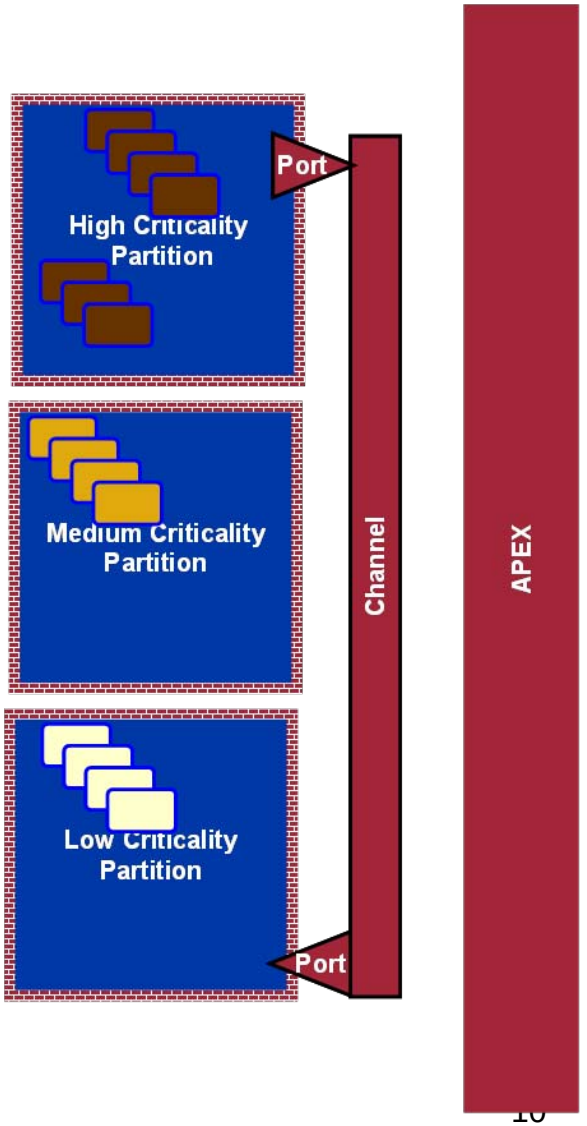
- **Traditional systems**
  - **Communication patterns fixed at design time**
- **Bold Stroke**
  - **Boeing mission avionics product line**
  - **Event channel used for communication**
  - **Configuration code generated from spreadsheet (initial)**
  - **XML configuration file read at run-time (CCM-like) (later)**
- **Future**
  - **Data centric approach that provides high levels of decoupling and independence**
    - **Data Distribution Service**

Data Distribution Service

- **OMG standard publish subscribe service**
  - **Standard interface and wire protocol**
- **Based on topics**
  - **Topic defines a data flow**
  - **Data flow is made up of samples of instances**
  - **Publishers/writers**
  - **Subscribers/readers**
- **Provides for varying Quality of Service**
  - **Real-time**
  - **Fault tolerance**
- **Multiple vendors**
  - **RTI DDS**
    - **Collaboration supporting various development and R&D programs**

Data Communication and Partitioning

- **Current safety partitioning standard is ARINC-653**
- **Defines an Application Executive (APEX) providing time-space partitioning**
  - **Partition Scheduling**
  - **Intra-partition communication**
  - **Inter-partition communication**
    - **Communication at the partition level**
    - **Channels connecting partitions**
    - **Sending and Receiving ports link channels to partitions**
    - **All aspects of inter-partition communication defined at configuration time**

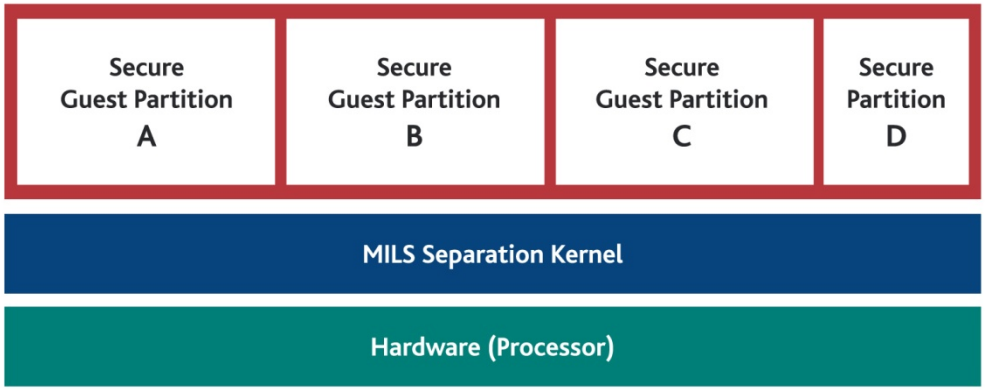


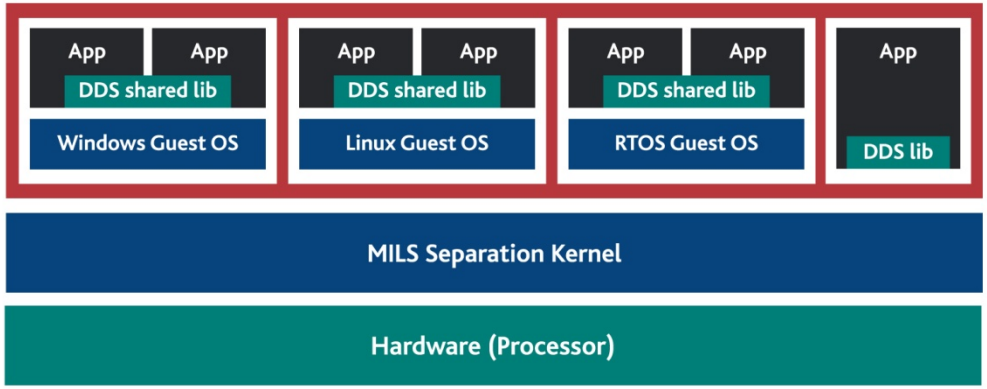
Realizing Delayed Binding

- **ARINC-653**
  - **Integrator completely defines the communications at the partition level at configuration time**
  - **Provides time/space isolation partitioning is there for**
- **DDS**
  - **Applications can define own data needs with application specific QoS constraints**
  - **Provides flexibility and decoupling required for compositionality and reuse**
- **How to reconcile DDS and ARINC-653 Inter-Partition Communication?**

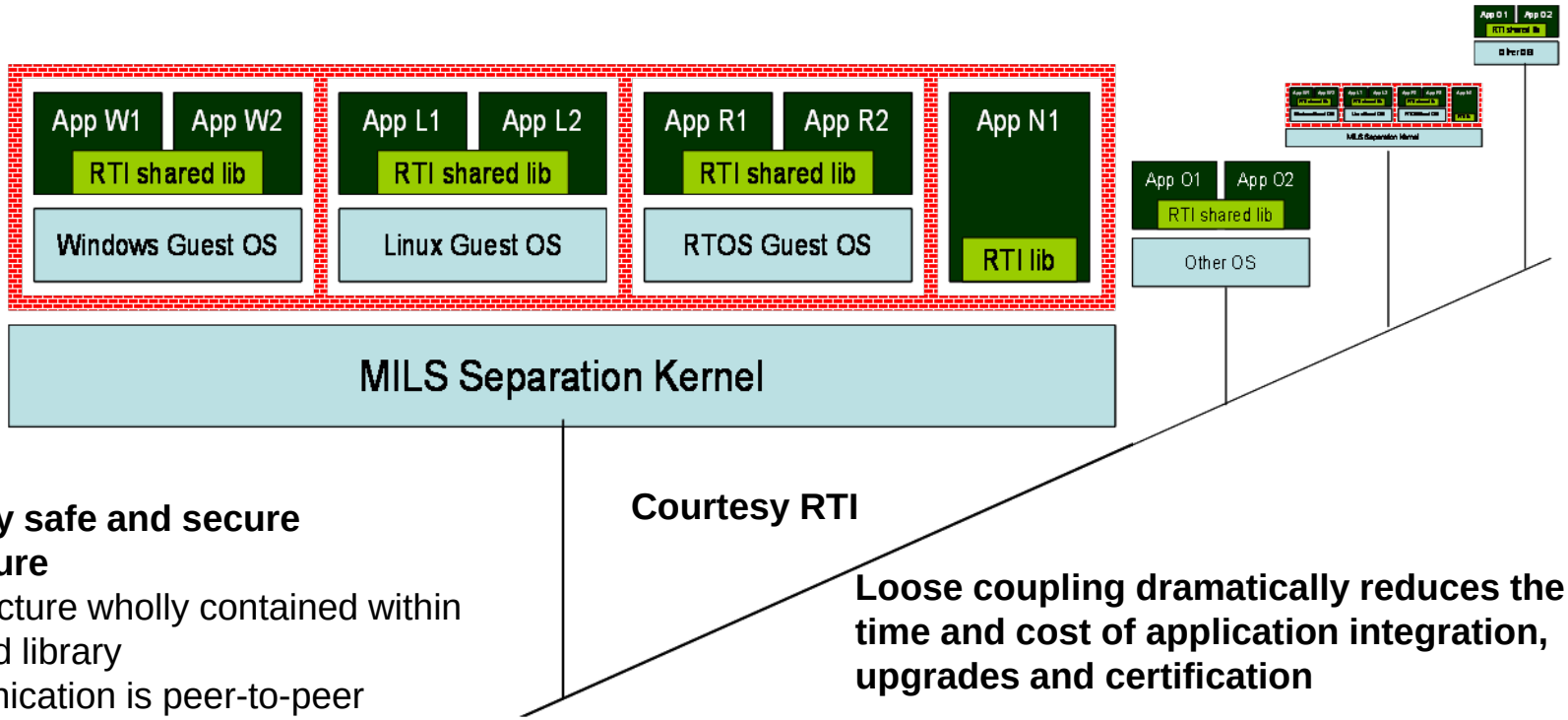
Realizing Delayed Binding

- **A slightly different problem and solution**
- **Partitioning for security**
  - **Security is faced with a similar partitioning problem**
  - **Information with different security restrictions present on the same system**
- **MILS (Multiple Independent Levels of Security) Separation Kernels developed to channel information flow**
  - **Separation kernel approach chosen to limit verification effort**





DDS for MILS



Courtesy RTI

Inherently safe and secure architecture

- Infrastructure wholly contained within embedded library
- Communication is peer-to-peer
- No shared broker, server or daemon processes
  - All security policies enforced by SK
  - Eliminates central vulnerabilities
  - Eases assurance
- Integrated authentication and encryption
- Small footprint for high assurance

Loose coupling dramatically reduces the time and cost of application integration, upgrades and certification

- Individual applications can be added or upgraded without requiring that others be changed or recertified
- Communication is seamless across programming languages, operating systems and processor architectures
- Communication is location independent
  - Within a partition
  - Between partitions
  - Between systems across LAN, WAN, wireless and satellite links

Verification and Delayed Binding

- **More is needed than implementation strategies**
- **Verification strategies are also needed**
- **Current approaches verify each configuration up front**
  - **Verify single nominal configuration**
  - **Verify predetermined backup configurations**
  - **Require extensive reverification for reuse (AC 20-148)**
- **Need to be able to support verification of configurations that emerge later in the lifecycle**
  - **Configurable/modular UAVs**
  - **Fault tolerance**
    - **Temporary degraded modes**
      - **Safe flight termination**
    - **Long term mode changes**
      - **Fault tolerance for very long missions (e.g., Vulture)**

Summary

- **Partitioning, Compositionality and Middleware are at the confluence of UAV evolution**
- **New approaches are needed to get us there**
- **Combining existing approaches in new ways can be a jumping off point**