Partitioning and Protection
Breakout Session

John Rushby

Computer Science Laboratory

SRI International

Menlo Park, California, USA

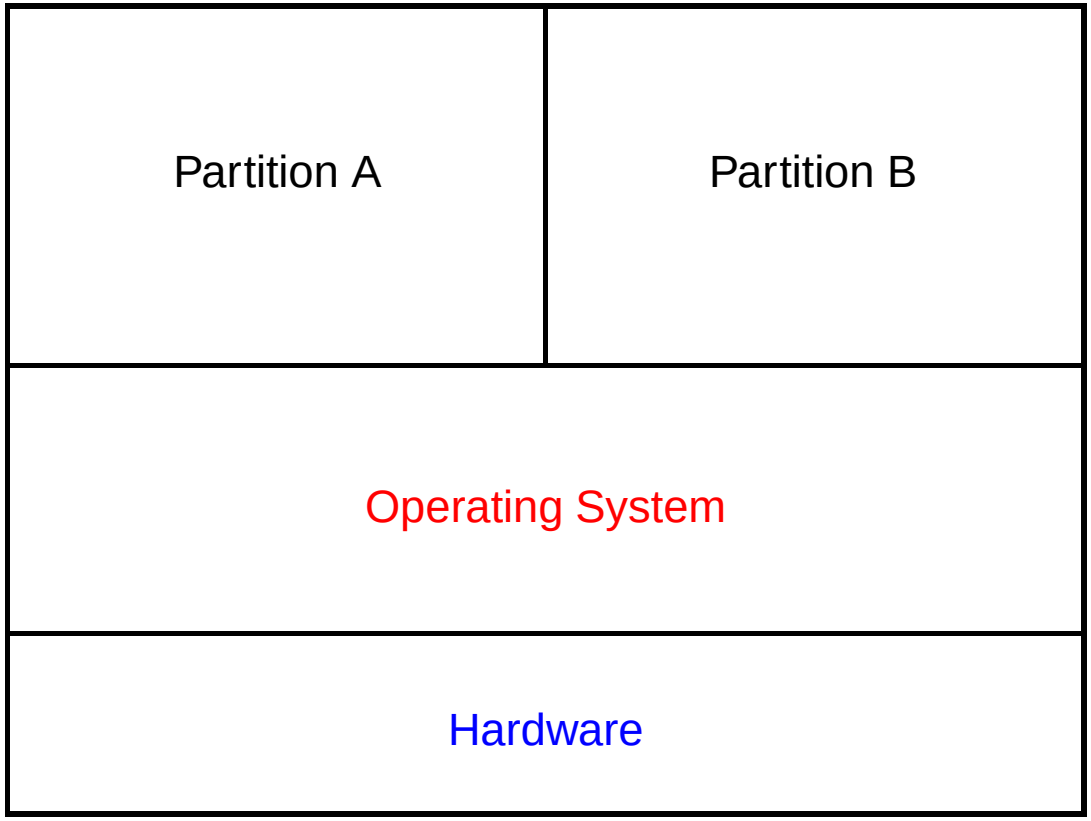
Overview

- This is an interactive session: pooling of knowledge, lessons learned, concerns from deveopers, certifiers, and researchers
  - I'm the author of NASA CR-99-209347 Partitioning in Avionics Architecture: Requirements, Mechanism, and Assurance, referenced in CAST-2
    Available at `http://techreports.larc.nasa.gov/ltrs/PDF/1999/cr/NASA-99-cr209347.pdf`
- I'll start off with a brief summary
  - Need for partitioning
  - Partitioning mechanisms in individual processors
  - Partitioning mechanisms in bus architectures
  - Requirements and assurance for partitioning
- Then it's over to you

CAST 2 Definitions

- Partitioning is just one means of implementing the general concept of protection
- Partitioning is method of separating components to ensure protection (section 2.3.1 of ED12B/DO-178B)
- The real issue is whether two or more components are protected from the actions of each other
- Component X can be said to be strictly protected from Y if any behavior of Y has no effect on the operation of X
- Component X can be said to be safely protected from Y if any behavior of Y has no effect on the safety properties of X

Simple Picture



Need for Partitioning

- For modular certification
- Or to lower certification levels for some components
- Or just to simplify analysis
- We need to eliminate unintended interactions among components
- In particular, fault propagation
- Recently, new interest in security and protection
- Need to eliminate deliberate, unwanted interaction

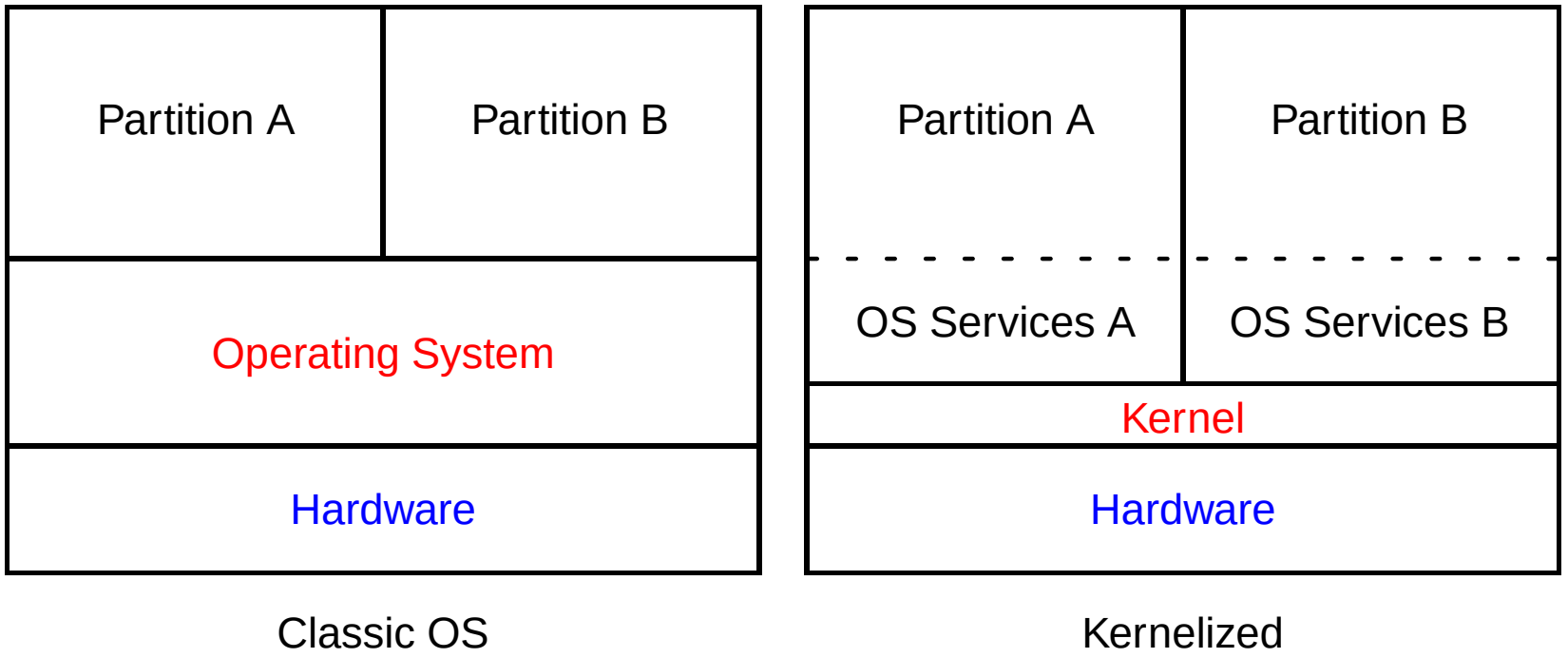
Basic Idea of Partitioning

- Shared resources are the channels for interaction
- Federated architecture has no shared resources, so provides the mental gold standard
- Partitioning architectures attempt to recreate same barriers to propagation

Mechanisms for Partitioning in Individual Processors

- Traditionally divided into space partitioning
  - Memory, memory-mapped I/O devices
  - Uses hardware protection mechanisms (user/supervisor modes, memory management units)
  - And O/S principles (kernels, virtual machines, threads)
- And time partitioning
  - Relies on timeouts and scheduling
  - Static vs. dynamic (priority-based) scheduling
  - Static has simpler assurance, but may complicate application design
  - Dynamic scheduling requires knowledge of pitfalls
    - ⋆ Priority inversions (interaction of priorities and locks)
    - ⋆ Correct accounting (charging for process swaps)

Architectures for Partitioning in Individual Processors

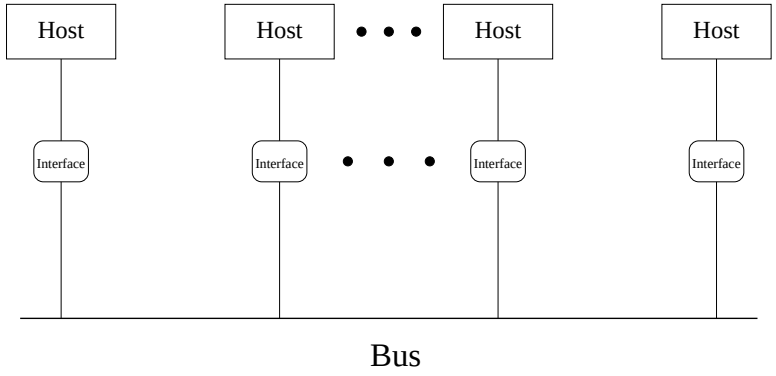


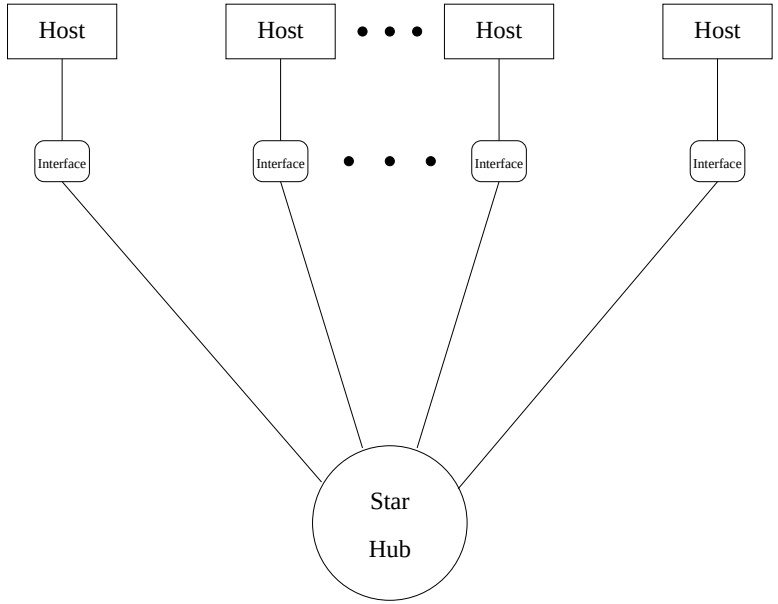
We can trust less software with the kernelized approach

Mechanisms For Partitioning in Bus Architectures

- It's a distributed system: no centralized control
- But we need to enforce a global property (partitioning)
- Intellectually difficult
- To protect against faults need separate fault containment units mediating access to shared resources
  - Paired BIUs in SAFEbus, Guardians in TTA
- Static schedule and synchronized clocks then allow mediation
- And hence time partitioning
- Addresses implicit in schedule: provides space partitioning
  - Explicit addresses are a partition violation-in-waiting
- Lock-free wait-free algs to move data across clock regions

Architectures for Partitioning in Distributed Systems

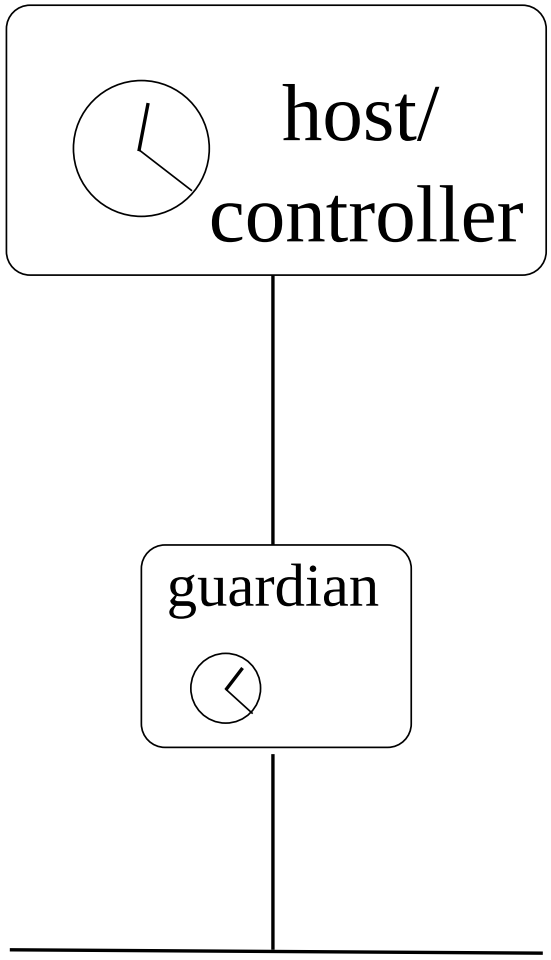




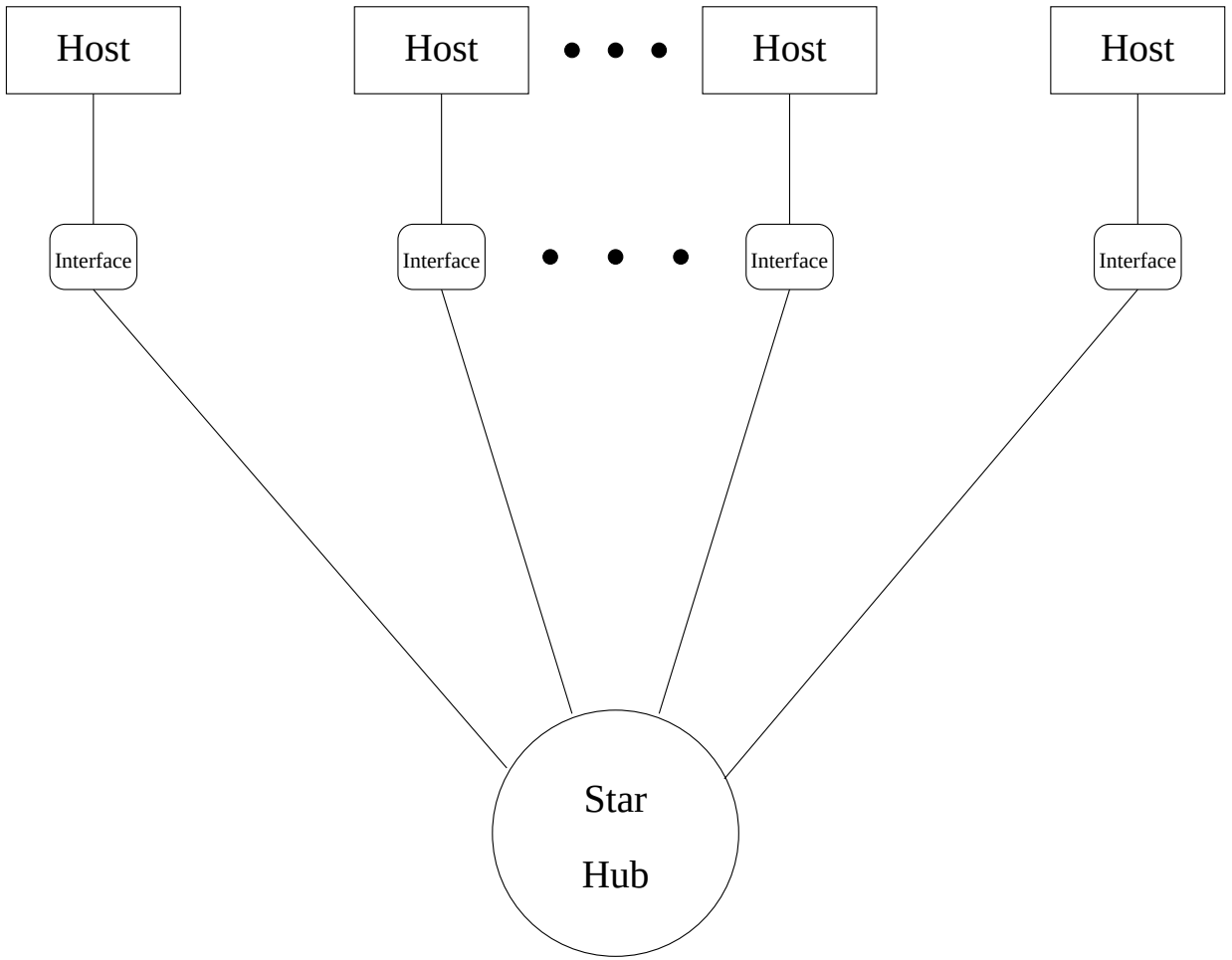
Bus/hub must be replicated; hub is a logical bus

Partitioning Rests on Bus Guardians (or Equivalent)

- Guardian prevents controller writing to bus outside its slot
- Can be one guardian per node, or centralized
- Must be independent FCU from controller



Guardian in Central Hub



Fault Tolerance and Partitioning

- Consider Slightly Out of Specification (SOS) fault
  - Intermediate voltage
  - Message on frame edge

  Example of asymmetric or Byzantine fault
- Some nodes accept the message, others don't
  - Whole system has lost coordination
- So fault tolerance is needed for partitioning
  - And is also an application-level service
- Need mechanisms for Interactive Consistency
  - Consistent delivery in presence of faults

Design Issues in Bus Architectures

- What is the fault model?
  - Mask arbitrary single fault
  - Self-stabilize (reboot) with multiple benign faults (HIRF)
    - ⋆ How fast? How triggered?
- What are the fault containment units?
  - How independent are they?
- What services should be provided?
  - Fault diagnosis/group membership complicates bus
  - But simplifies applications

Both Kinds of Partitioning Together

- In some bus architectures, individual host processors run single applications (e.g., 777 AIMS)
  - Critical applications replicated across multiple hosts
- In others, have partitioning within the hosts as well
  - One Host might run one replica of autopilot, another of FMS, and some level C code

The Partitioning Property

- Gold Standard for Partitioning

  A partitioned system should provide fault containment equivalent to an idealized system in which each partition is allocated an independent processor and associated peripherals, and all inter-partition communications are carried on dedicated lines

  OK as mental benchmark, but how would you test applications?

- Alternative Gold Standard for Partitioning (Rockwell Collins)

  The behavior and performance of software in one partition must be unaffected by the software in other partitions

  Can test application by running it in presence of empty/dummy partitions

Assurance Issues

- Assurance for space partitioning in minimal kernel seems fairly straightforward
  - Security agencies face similar issues for separation
- But complex operating systems?
- And dynamic scheduling?
- Algorithms of bus architectures are difficult
- But they are otherwise minimal
  - That is, no extraneous services

Outstanding Issues

- Experiences?
- Lessons learned?
- Concerns?
- Over to you