

System Software and TinyAUTOSAR

Florian Kluge

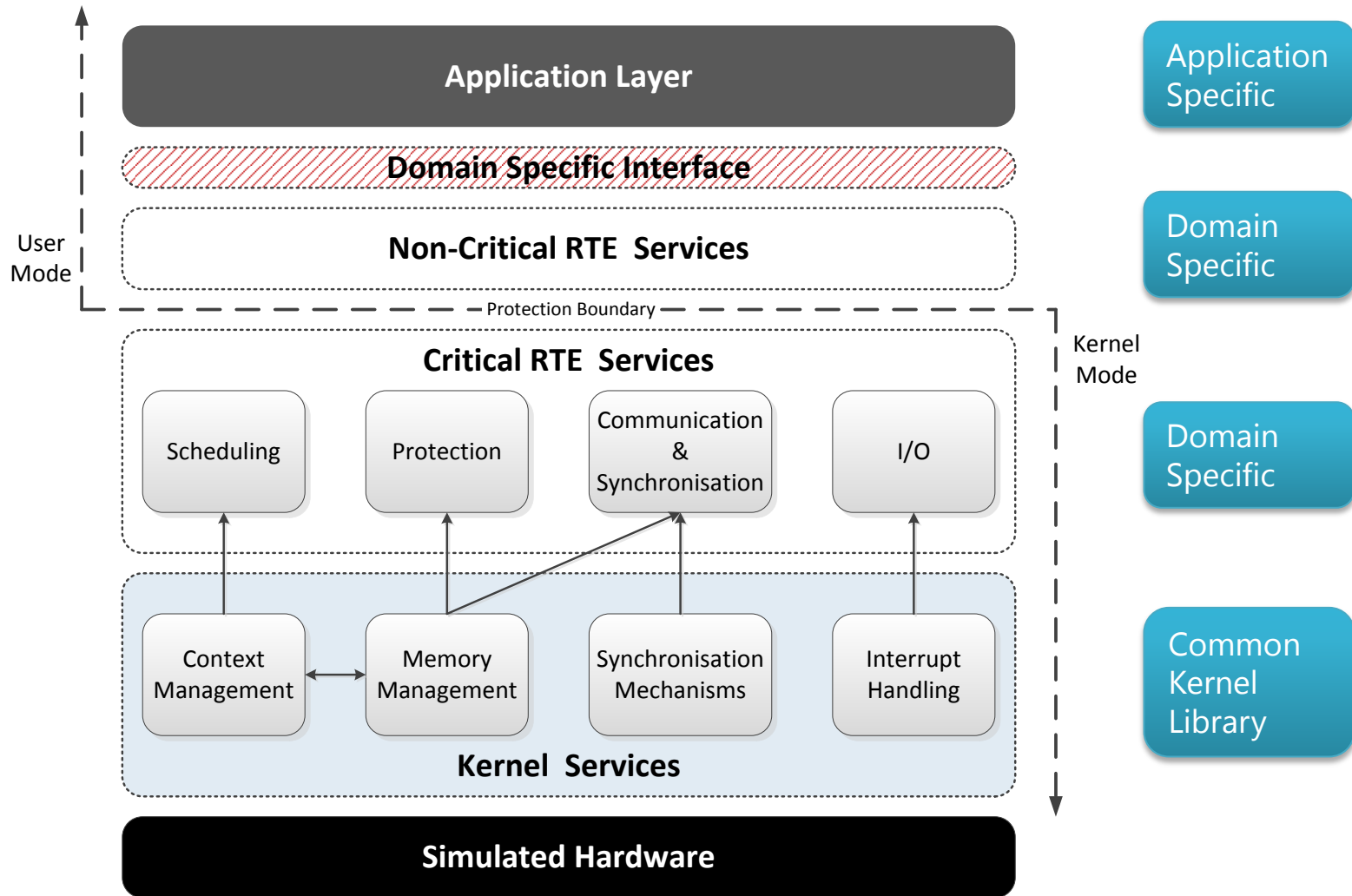
University of Augsburg, Germany

parMERASA Dissemination Event,
Barcelona, 2014-09-23

- parMERASA System Architecture
- Kernel Library
- RTE Implementations
 - TinyIMA
 - pBIOS4CM
 - TinyAUTOSAR
- Parallelisation techniques for AUTOSAR
 - Parallelised Interrupt Handler
 - Synchronisation Buffers
 - Performance Evaluation of Remote Service Calls

	Automotive	Avionic	Construction
Scheduling	Pre-emptive (Earliest Deadline First)	Fixed cyclic + Pre-emptive	Round-robin
Com. & Sync.	Resources, Events, buffered/unbuffered Messages	Messages; Events, Buffers, Blackboards, Semaphores	Events, Semaphores, Spin-Locks
I/O	Low latency	Predictability	Low latency
Protection	OS-Application, (Task)	Partition	Task

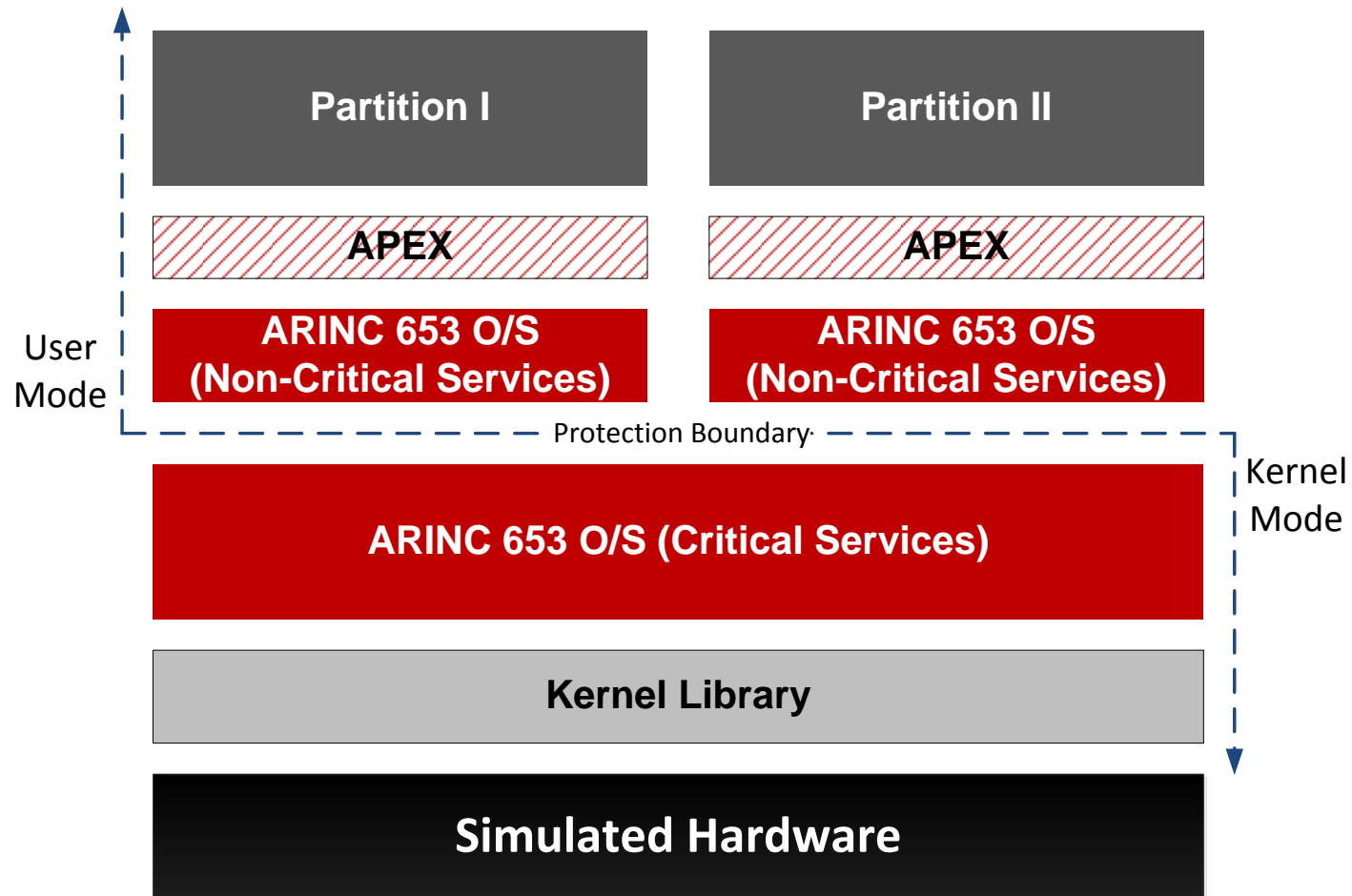
Blocking sync./com. techniques need interaction with system scheduler
↔ different scheduling algorithms



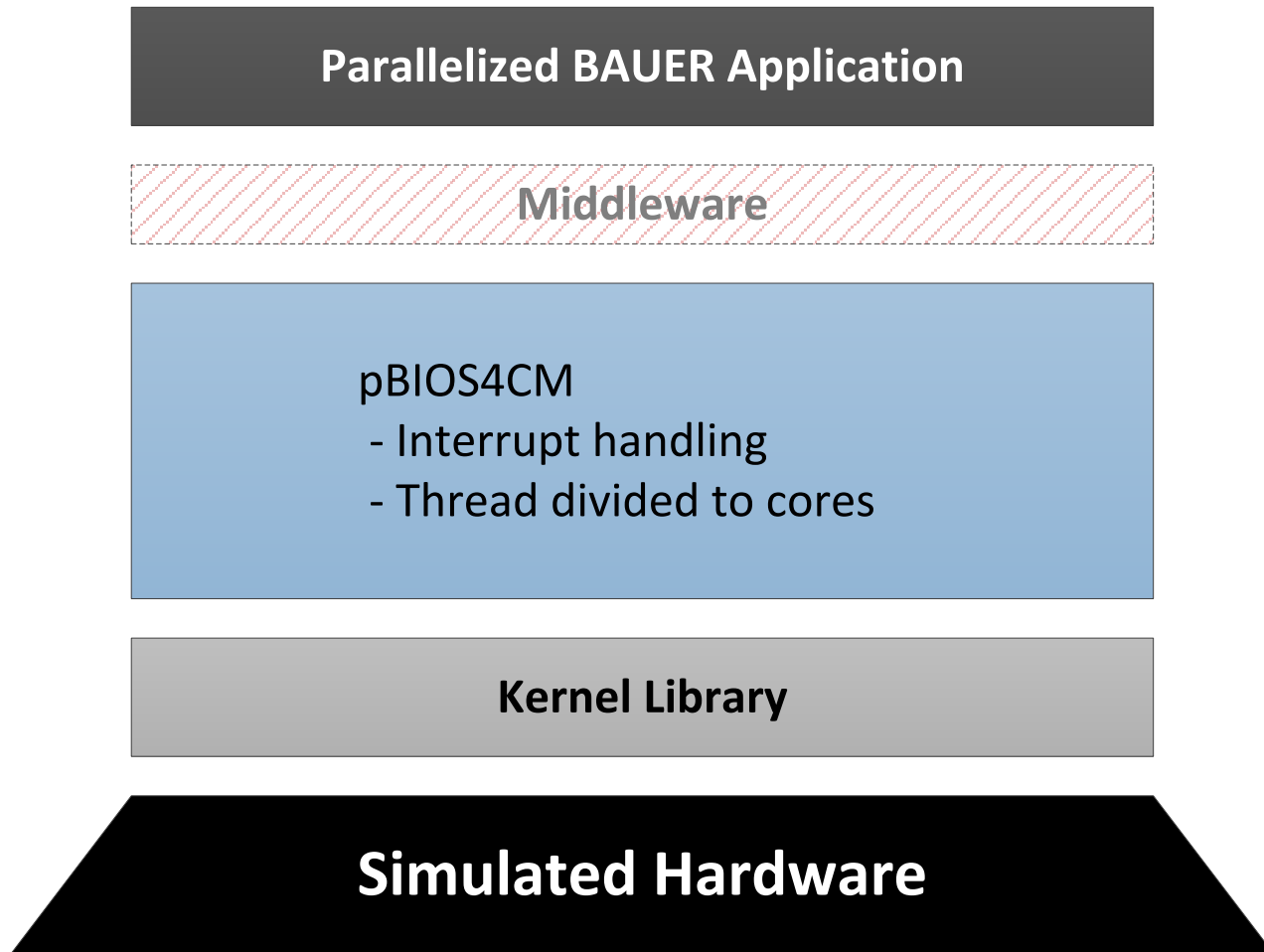
- Abstraction of underlying hardware
- Provides RTE-independent services
 - WCET-analysable primitives
 - Synchronisation functions for parallel applications
- Common base for RTE implementations

- OS Kernel consists of
 - Kernel Library
 - RTE-specific isolation-critical services

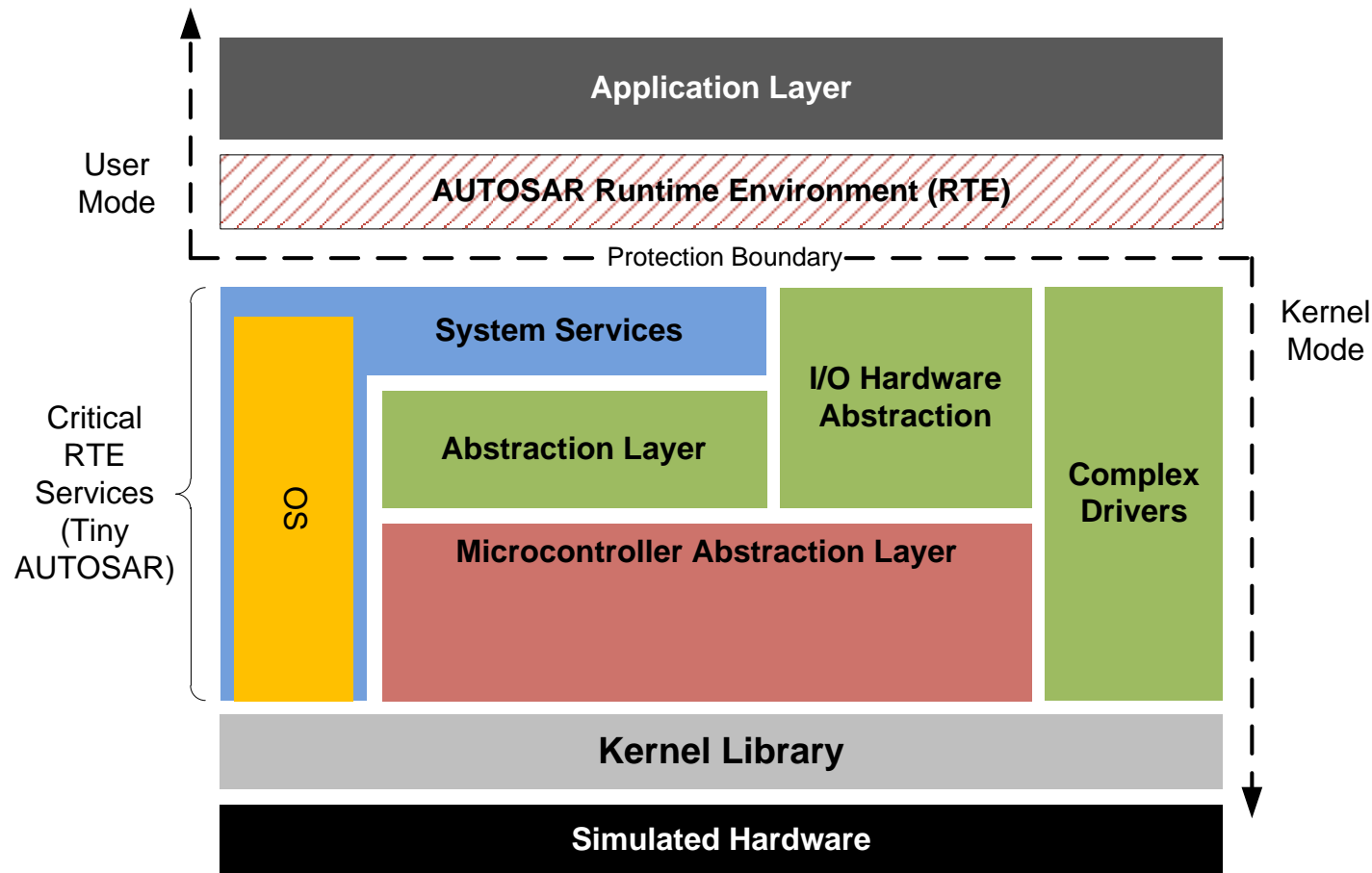
- Non-critical RTE services: no influence on other partitions



- Minimal subset of ARINC 653 API required by Honeywell application to guarantee **time isolation**
 - Partition and process management
 - Communication and synchronisation primitives
 - Buffers, Queueing Ports, Semaphores
 - ARINC XML system definition replaced with C functions
- **Extend** API to support cluster-based many-cores
 - Partitions → Clusters
 - *CONNECT_SRC_DST_QUEUEING_PORTS*
 - Processes → Cores
 - *BIND_PROCESS_TO_CORE*
- No parallelisation at TinyIMA level has been provided



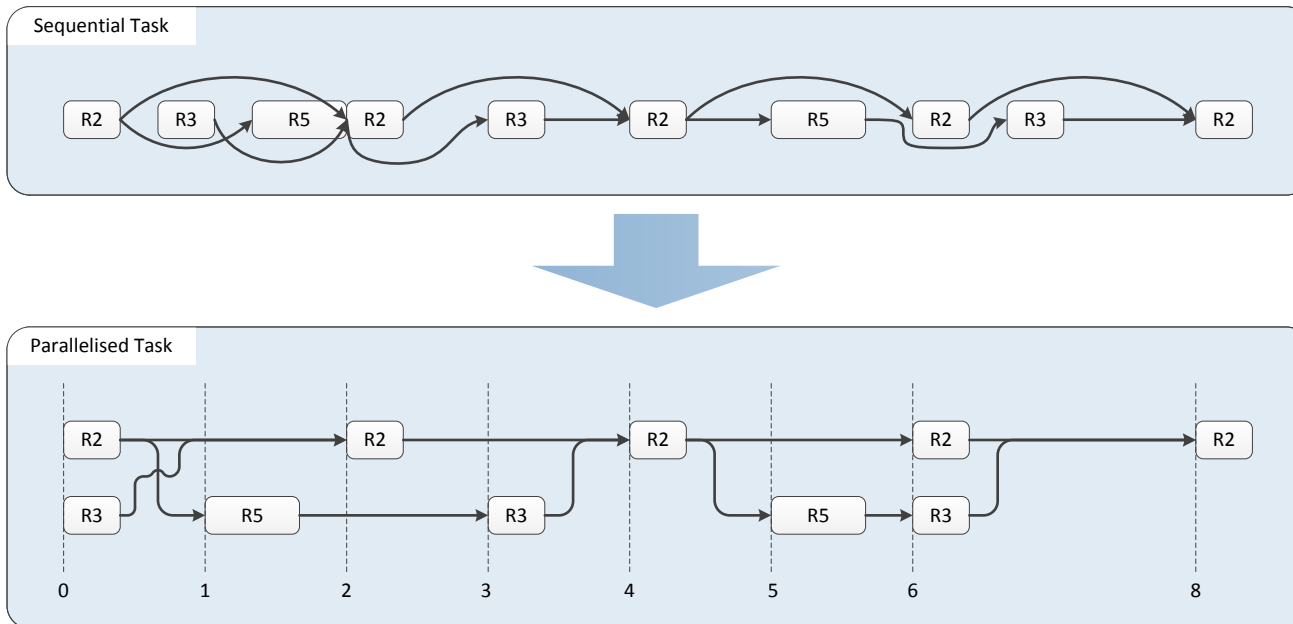
- Based on BIOS API of original ECU vendor
 - Single-core TriCore system (ESX-3XL from Sensorteknik Wiedemann)
 - Basic functions for thread scheduling, I/O, timing
- Modifications:
 - Thread-safe re-design and implementation (sources not available)
 - Simplified thread management: One thread per core
- Supported I/O functionalities
 - CAN
 - Generic I/O for PWM, digital and analogue input
 - Interrupt and DMA system



- Properties:
 - Evaluation platform for parallelised automotive applications executed in parallel
 - Provide minimum subset of AUTOSAR software architecture
 - Comprise basic functionalities required on any core
 - Provide basic communication and synchronisation features for multi-/many-core processors
- Implementations:
 - Multicore Implementation according to AUTOSAR 4.x
 - Implementation with optimisations for parallelised applications

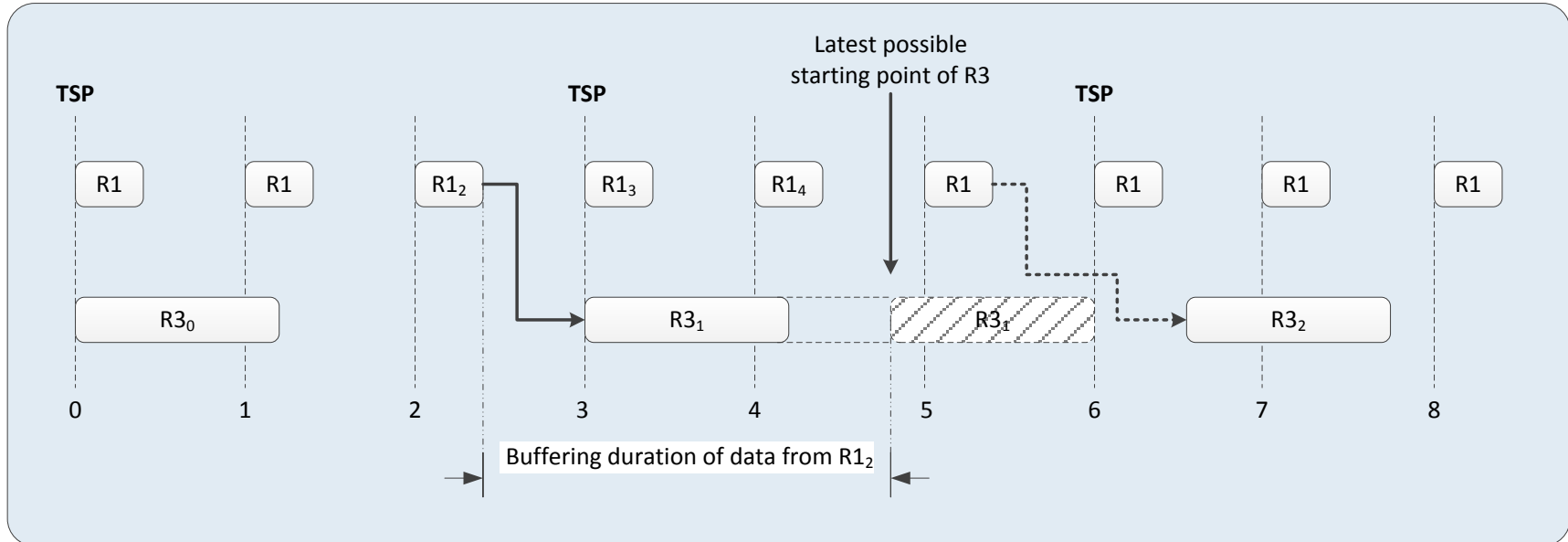
- Support for execution of parallelised tasks and runnables
 - Alternative scheduling: time synchronous execution model
 - Synchronisation buffers
 - Extended to support parallelised interrupt handler
- Communication
 - Comparison of implementation approaches of cross core service calls

- Time-triggered execution of tasks/runnables
- Exchange of data only at time synchronisation points
→ no need for explicit synchronisation!
- For regular tasks: defined in PharOS

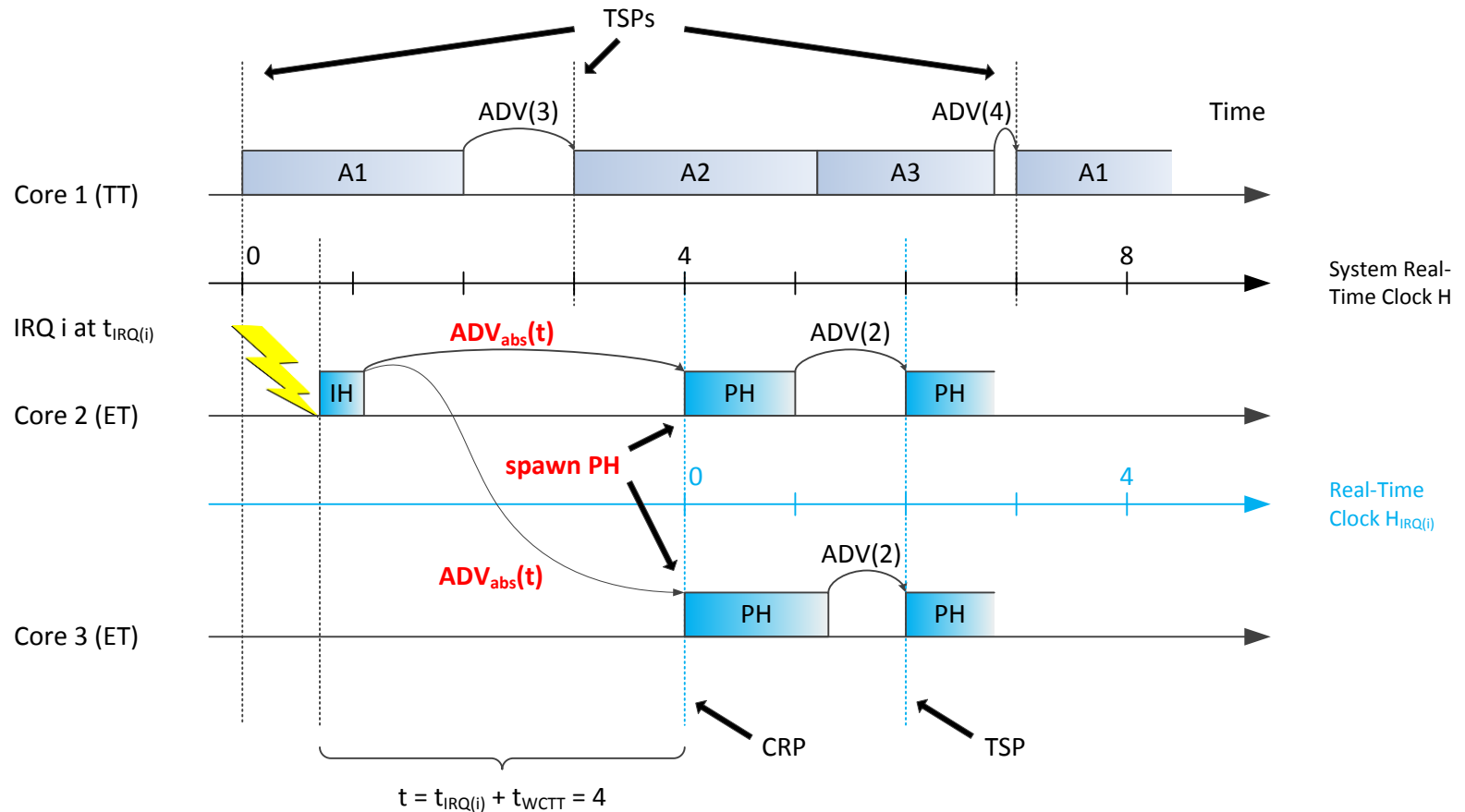


- Aim: each runnable execution has time-consistent view of input data

→ buffer data, access relative to TSP



→ Exploit time synchronous execution model also for ISRs!



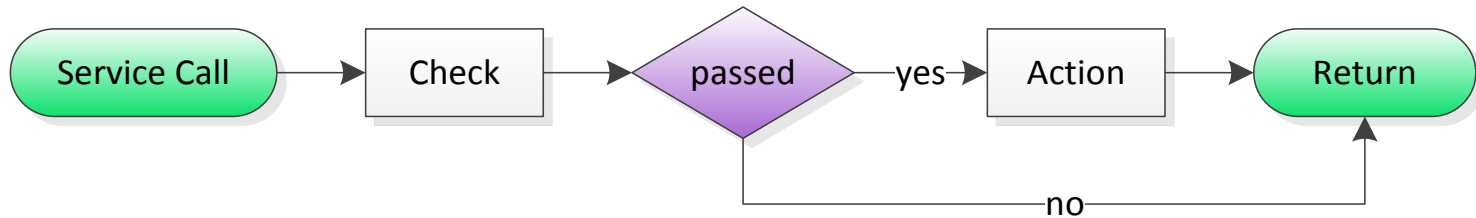
A_x Agent x (TT domain)
IH Initial Handler (ET domain)
PH Processing Handler (ET domain)

CRP Clock Reference Point – time base for future ADV instructions on corresponding cores
ADV_{abs}(t) Advance Absolute Instruction – advances to an absolute point of system real-time clock

t_{IRQ(i)} Interrupt event i at time t
t_{WCTT} Worst Case Traversal Time

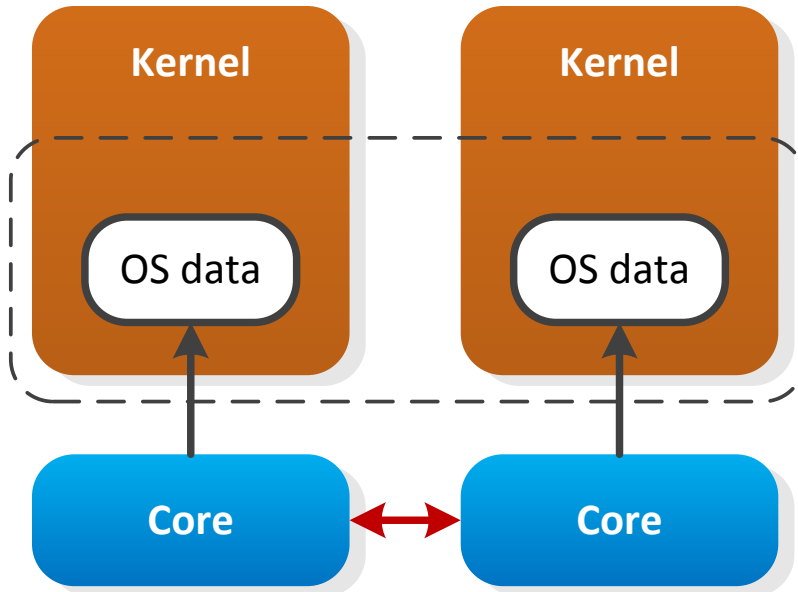
- [illegible]

- General Structure of Service call:



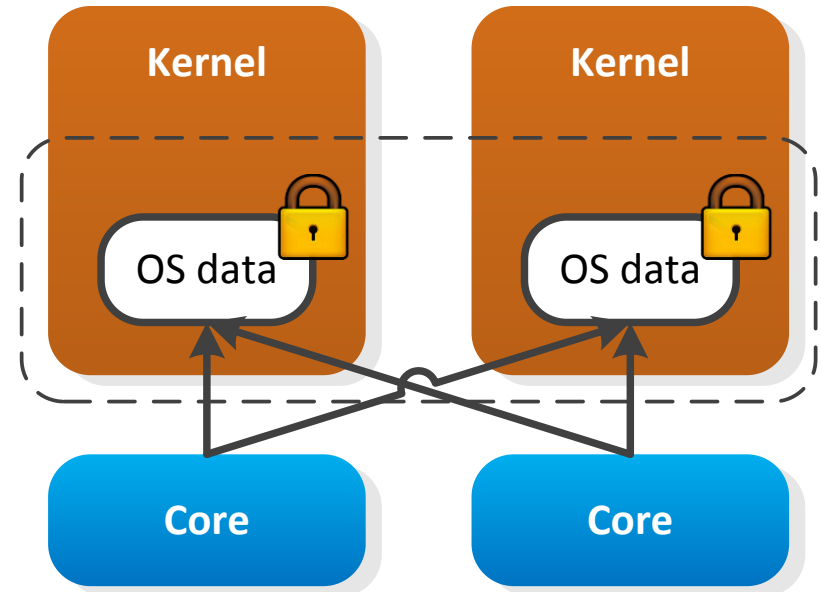
- Some service calls can affect other cores
 - e.g. ActivateTask
- Questions:
 - Where should the single parts be executed?
 - How is communication organised?

Message based OS data
access across core



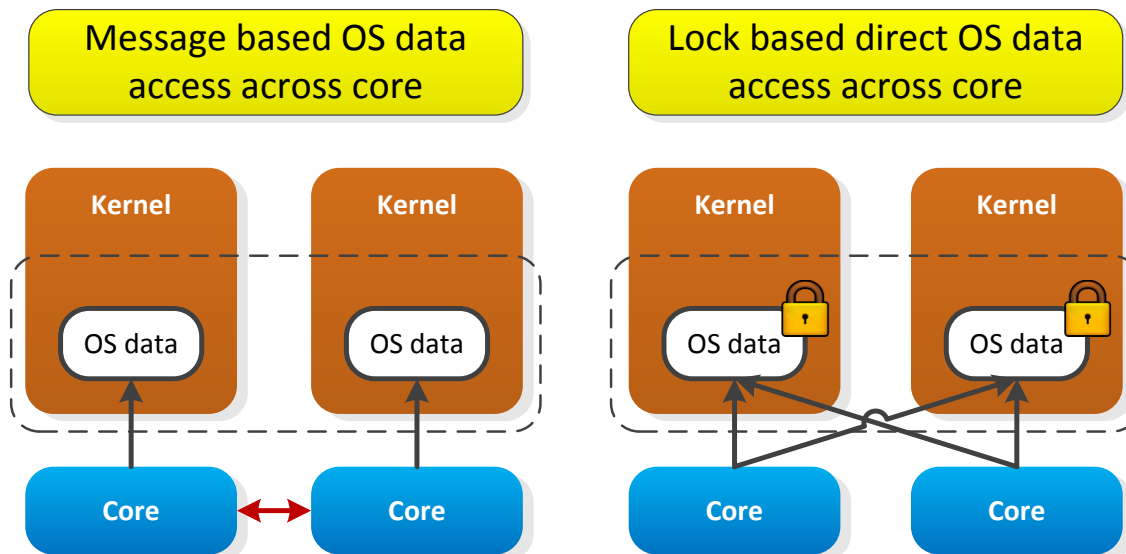
Everything on Target Core

Lock based direct OS data
access across core



Check & Action on Source core,
Only Rescheduling on Target Core

- Advantage for lock based approach
- Less remote core blocking times
- Faster execution



- RTE bases for parMERASA applications
 - Avionic: TinyIMA
 - Construction Machinery: pBIOS4CM
 - Automotive: TinyAUTOSAR
 - Adaption for application parallelisation
- Automotive RTE: TinyAUTOSAR
 - Minimum evaluation platform
- Investigations using TinyAUTOSAR:
 - Time-synchronous execution
 - Synchronisation buffers
 - Synchronous execution of parallelised interrupt handler
 - Performance of remote service calls: messages vs. locks