Achal: Building Highly Reliable Networked Control Systems

Arpan Gujarati, Malte Appel, and Björn B. Brandenburg

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

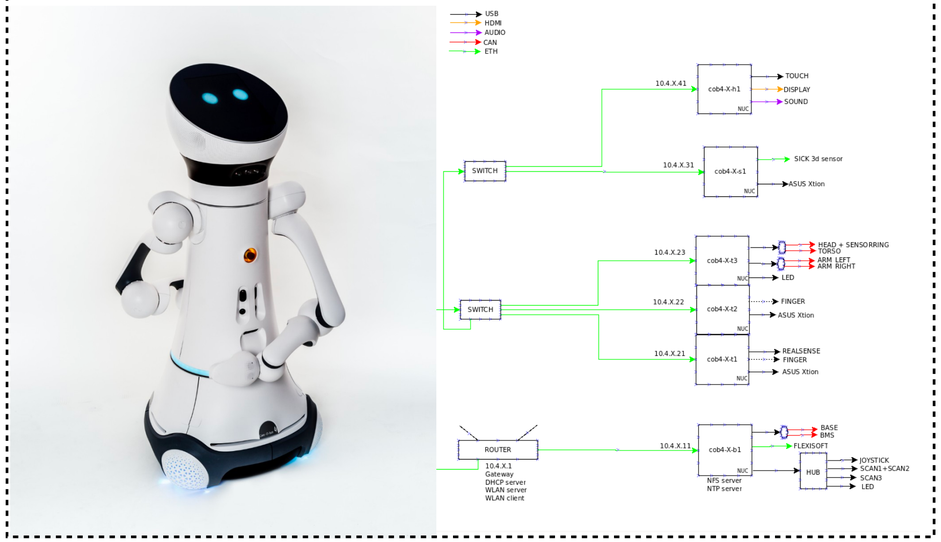
In a nutshell

Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet

In a nutshell

Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet

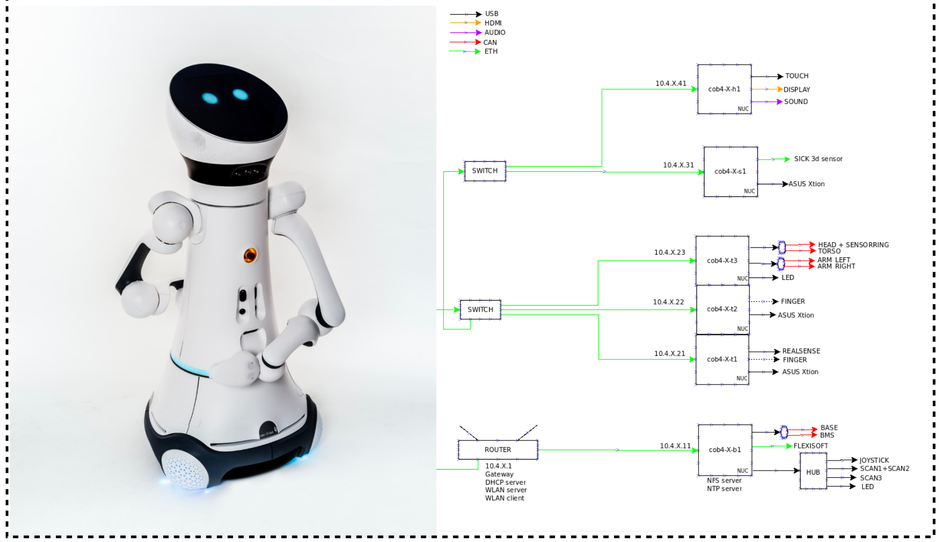
NCS at the core of next-generation CPS



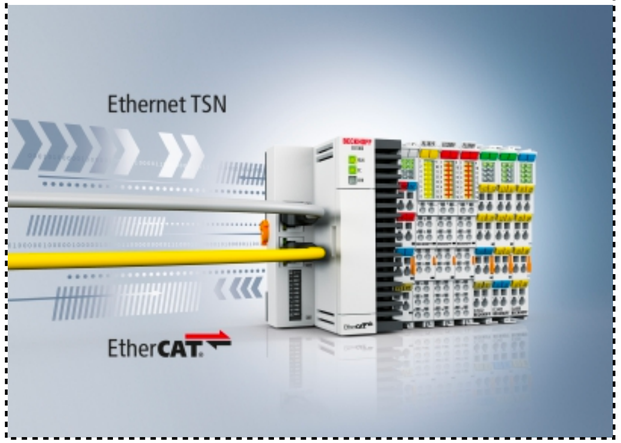
In a nutshell

Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet

NCS at the core of next-generation CPS



Use of latency-aware Ethernet standards on the rise



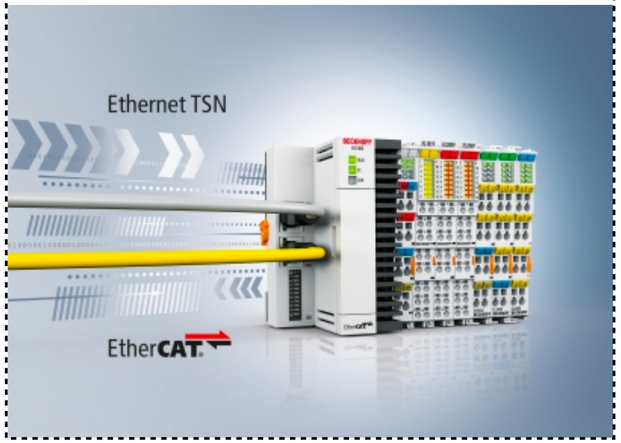
In a nutshell

Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet

NCS at the core of next-generation CPS

Non-malicious Byzantine errors (inconsistent broadacsts) due to transient faults

Use of latency-aware Ethernet standards on the rise



In a nutshell

Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet

NCS at the core of next-generation CPS

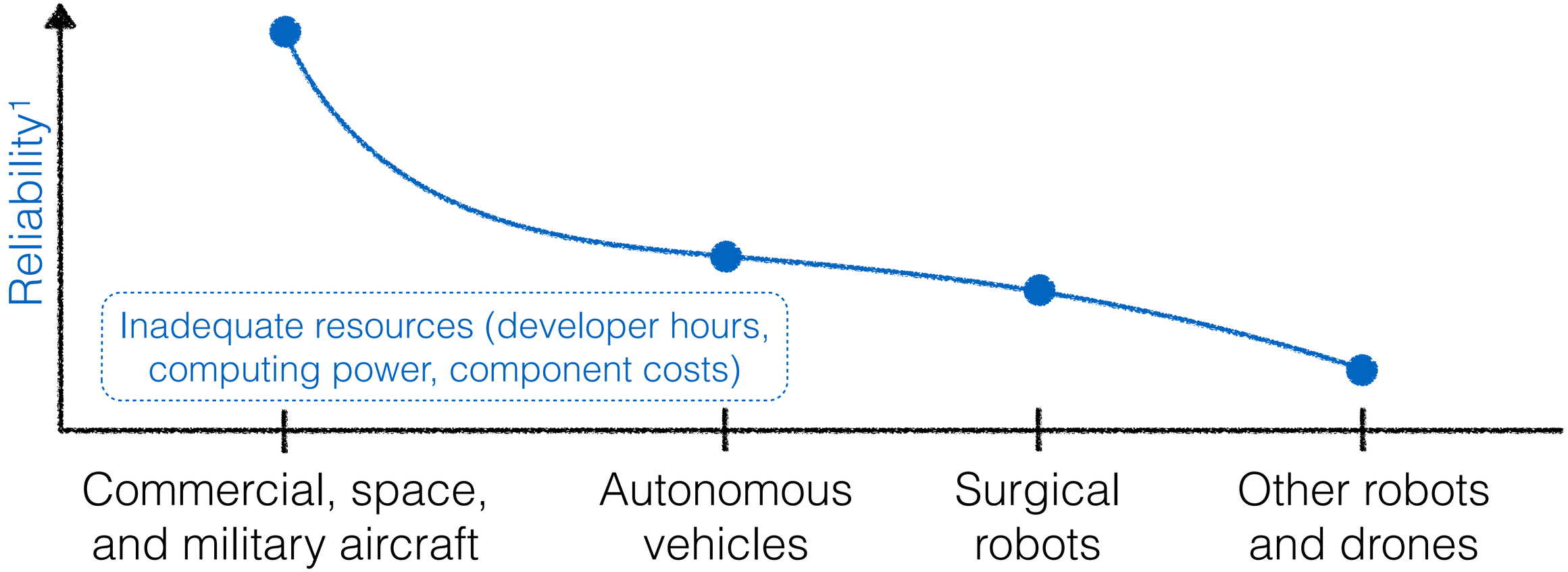
Non-malicious Byzantine errors (inconsistent broadacsts) due to transient faults

Use of latency-aware Ethernet standards on the rise

Achal is designed to mask these errors

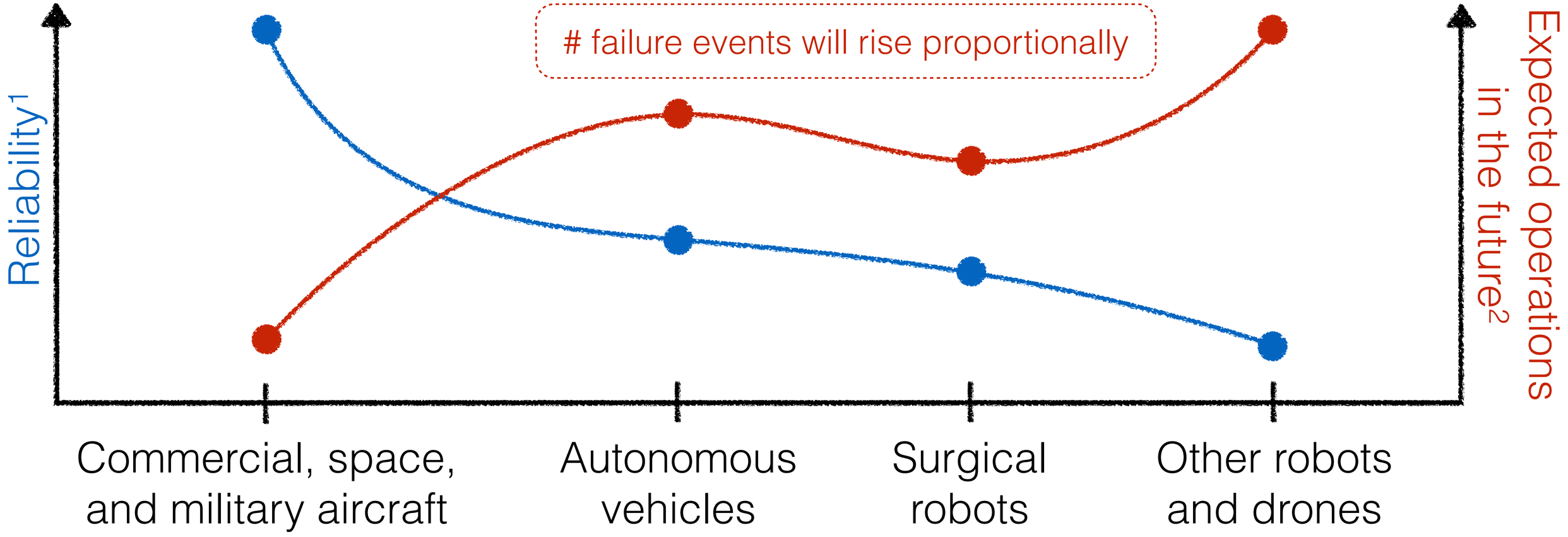
Motivation

Motivation



[1] Banerjee et al. “Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on over a Million Miles of Field Data.” DSN (2018)

Motivation



[1] Banerjee et al. “Hands Off the Wheel in Autonomous Vehicles?: A Systems Perspective on over a Million Miles of Field Data.” DSN (2018)

[2] SESAR Joint Undertaking. “European Drones Outlook Study-Unlocking the value for Europe.” SESAR, Brussels (2016)

Problem

Today's commercial aircraft systems are highly reliable!

It will become imperative to also make the **next generation of fully-autonomous CPS**, such as autonomous vehicles, drones, and robots, **highly reliable?**

Problem

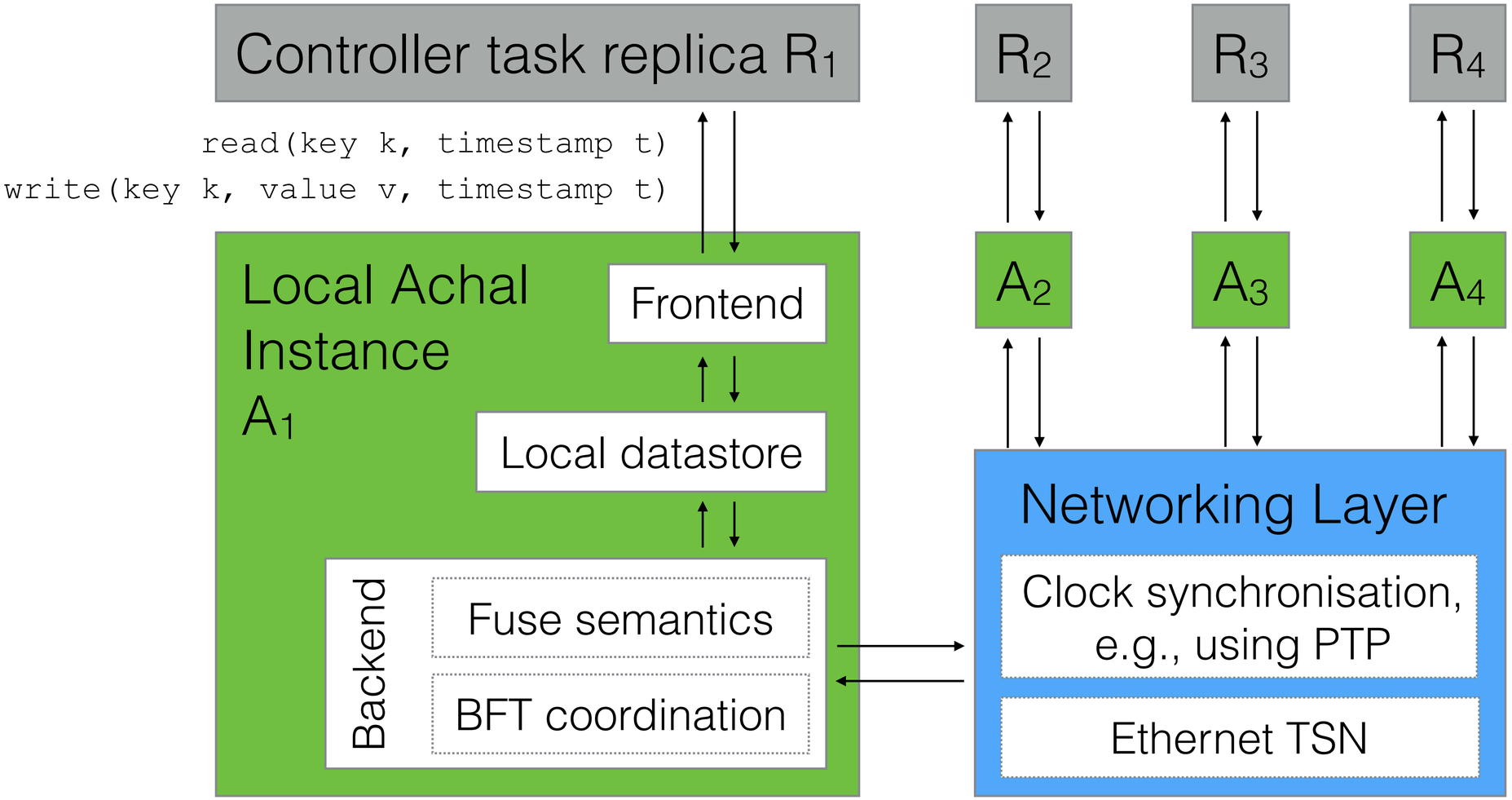
Today's commercial aircraft systems are highly reliable!

It will become imperative to also make the **next generation of fully-autonomous CPS**, such as autonomous vehicles, drones, and robots, **highly reliable?**

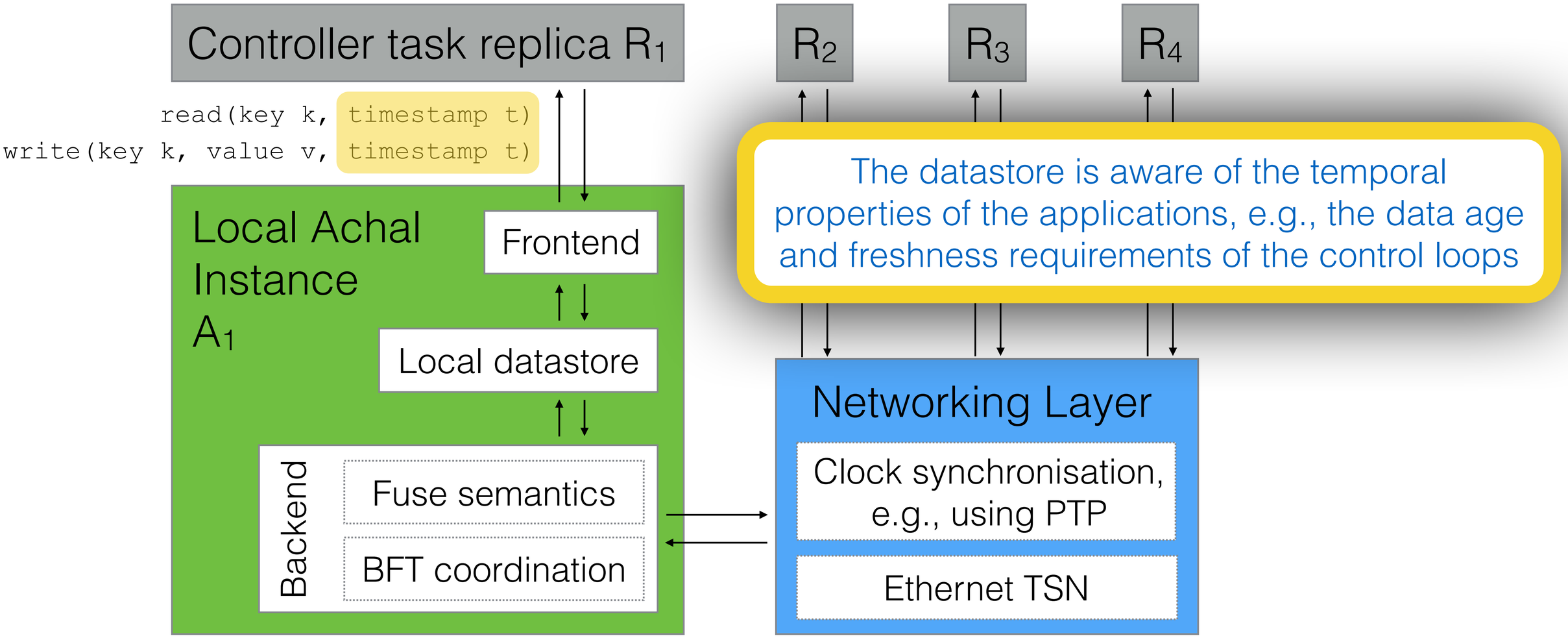
Objective Building highly reliable CPS using cheap, fast, but unreliable **COTS components** at **economical costs**

Key contribution: **CPS-friendly** BFT middleware

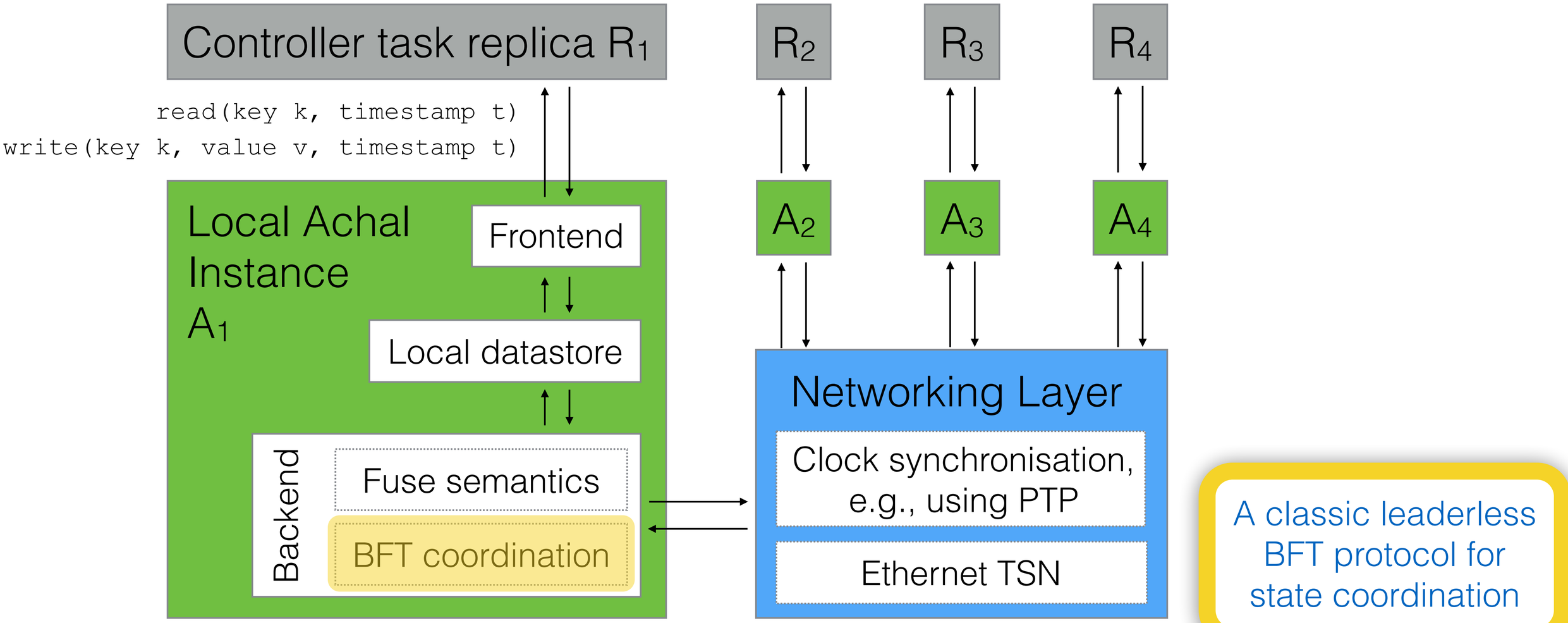
Key contribution: **CPS-friendly** BFT middleware



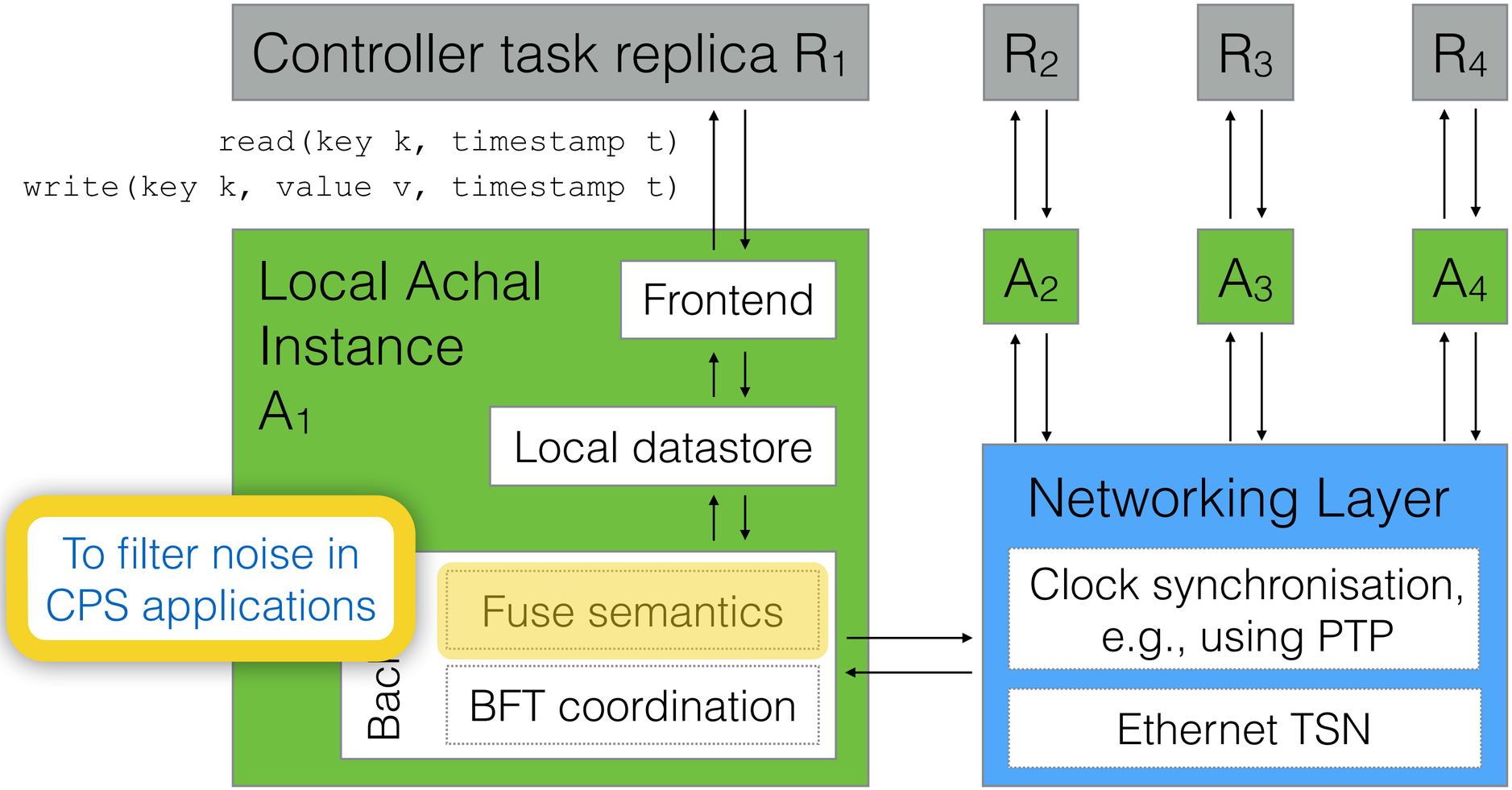
Key contribution: **CPS-friendly** BFT middleware



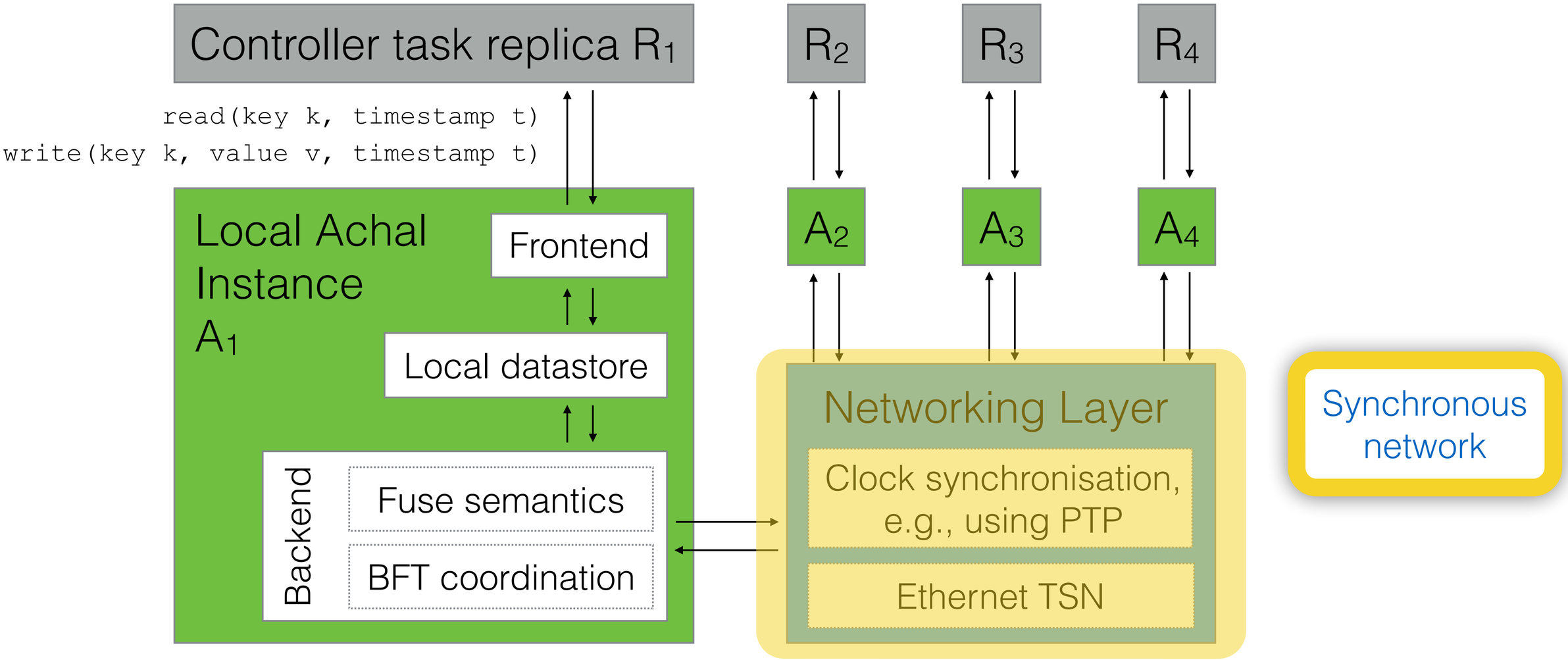
Key contribution: **CPS-friendly** BFT middleware



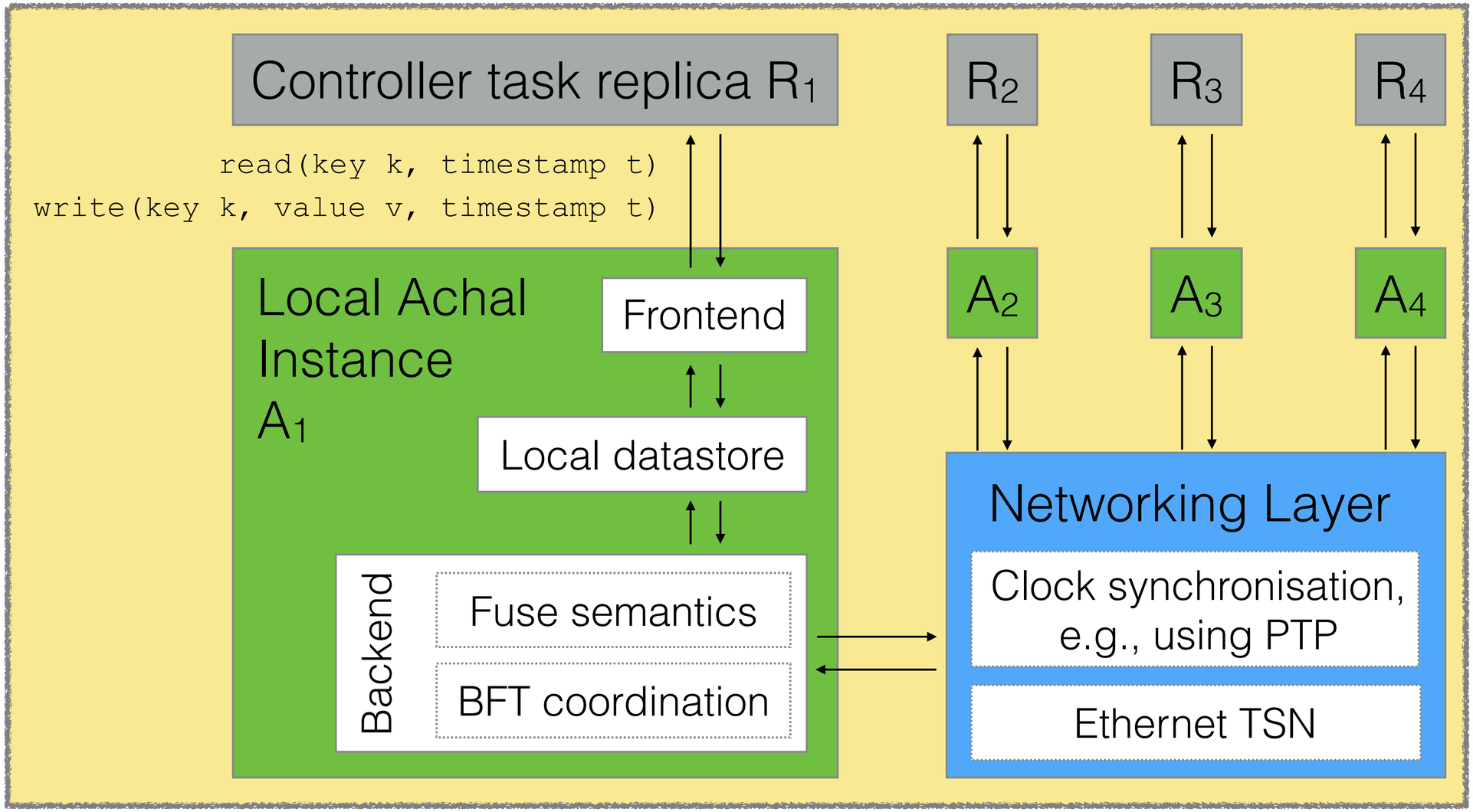
Key contribution: **CPS-friendly** BFT middleware



Key contribution: **CPS-friendly** BFT middleware



Key contribution: **CPS-friendly** BFT middleware



Predictable implementation

+

Schedulability analysis

+

Reliability analysis

For more details ...

How to **program** networked control systems over Achal's time-aware API?

For more details ...

How to **program** networked control systems over Achal's time-aware API?

How is the **timing predictability** ensured?

For more details ...

How to **program** networked control systems over Achal's time-aware API?

How is the **timing predictability** ensured?

How to quantify its **reliability** in the presence of stochastic transient faults?

For more details ...

How to **program** networked control systems over Achal's time-aware API?

How is the **timing predictability** ensured?

How to quantify its **reliability** in the presence of stochastic transient faults?

How does its **performance** compare with that of BFT-SMaRt and Cassandra?

For more details ...

How to **program** networked control systems over Achal's time-aware API?

How is the **timing predictability** ensured?

How to quantify its **reliability** in the presence of stochastic transient faults?

How does its **performance** compare with that of BFT-SMaRt and Cassandra?

Building "Ultra-Reliable" Networked Control Systems using COTS Components

Arpan Gujarati, Malte Appel & Björn B. Brandenburg

What is ultra-reliability?

Quantifiably negligible failure rates in the presence of **stochastic fault processes**

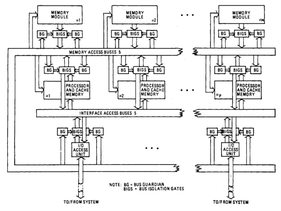
Example: FTMP

Fault-Tolerant Multiprocessor by Hopkins et al. (1978)

Custom hardware with dynamic redundancy and tight synchronization

Analysis of component-level failures using combinatorial methods

Under 10^{-10} failures/hr



Future CPS

Drone/robot/AV fleets using **cheap, fast, but unreliable** COTS hardware

*Can we achieve **FTMP-like ultra-reliability on unreliable COTS** processors and networks?*

Can Raspberry Pi's over Ethernet be ultra-reliable?

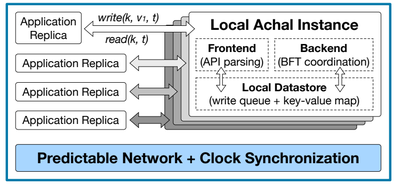
Our approach

Ultra-reliability as an **emergent property** over unreliable hardware

Existing methods for software fault tolerance, e.g., robust controllers, replication, BFT protocols → Predictable (time-sensitive) implementations → Quantitative reliability analysis to demonstrate ultra-reliability

This work — *Achal* — a predictable BFT middleware for NCS

Design overview



Read/write API based on absolute time

Avoids data races despite runtime variations

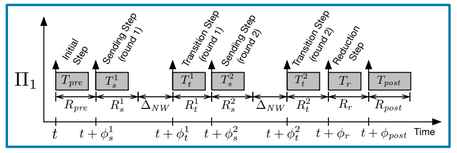
Algorithm 4.2 Periodic task of a PID controller for balancing an inverted pendulum, programmed over Achal.

```
1: procedure PeriodicTaskActivation
2:     time ← timeOfLastActivation()
3:     current ← getSensorData()
4:     error ← target − current
5:     integral ← Achal.read("integralKey", time) + error
6:     derivative ← error − Achal.read("errorKey", time)
7:     force ← kp ∗ error + ki ∗ integral + kd ∗ derivative
8:     time ← timeOfNextActivation()
9:     Achal.write("errorKey", error, time)
10:    Achal.write("integralKey", integral, time)
11:    actuate(force)
```

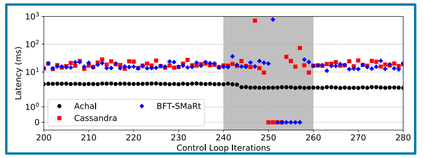
- Fault-induced errors
- Execution time jitter
- Task re-executions
- Message reorderings

Inspired from the Logical Execution Time (LET) paradigm (Henzinger et al., 2001)

Predictable hard real-time implementation ensures by design that writes are committed on time



Initial latency results (crash in gray region)



Prior, ongoing, and future work

Quantified conservative failure rates for temporally robust, actively replicated NCS on Achal-over-Ethernet and CAN
- Accounted for time and value domain errors at the message granularity (more fine-grained than in FTMP)

Currently **evaluating Achal** using different NCS applications and against related work on BFT protocols & key-value stores

Next step — ultra-reliable **clock synchronization protocols** over Ethernet; is the Precision Time Protocol ultra-reliable?

COTS = Commercial Off-The-Shelf | CPS = Cyber-Physical Systems | NCS = Networked Control Systems | BFT = Byzantine Fault-Tolerant | CAN = Controller Area Network