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**

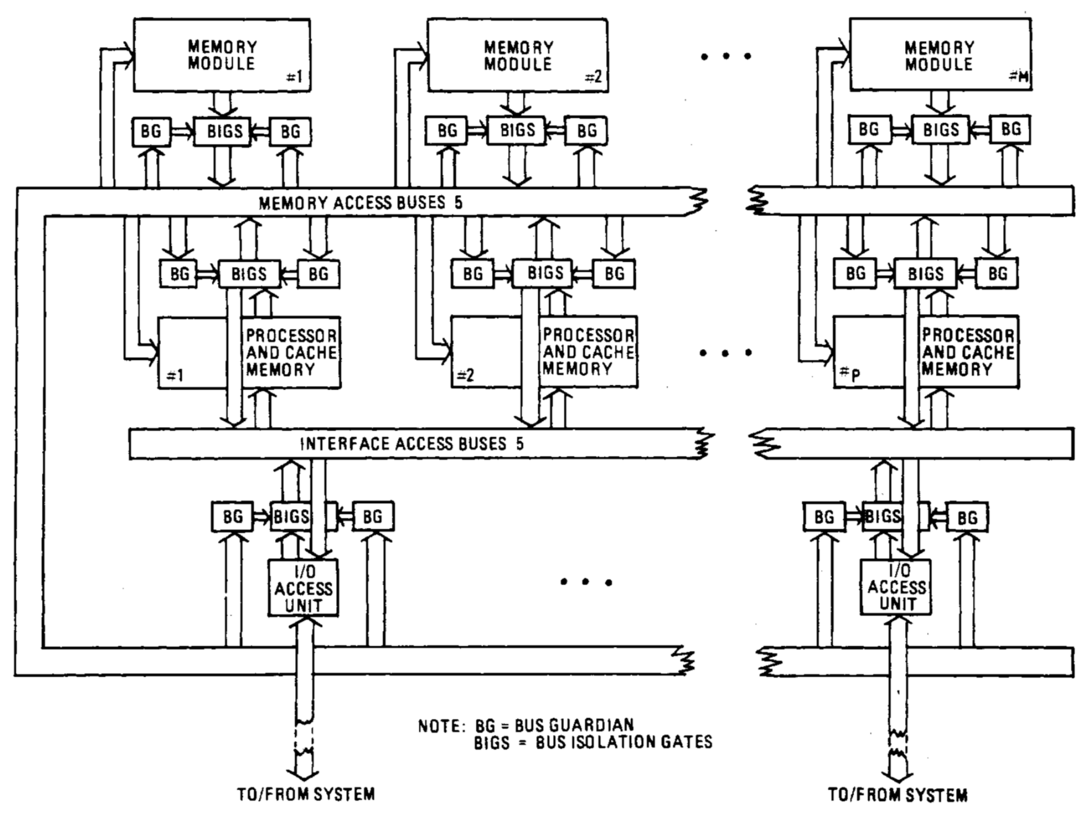
Example: FTMP

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?

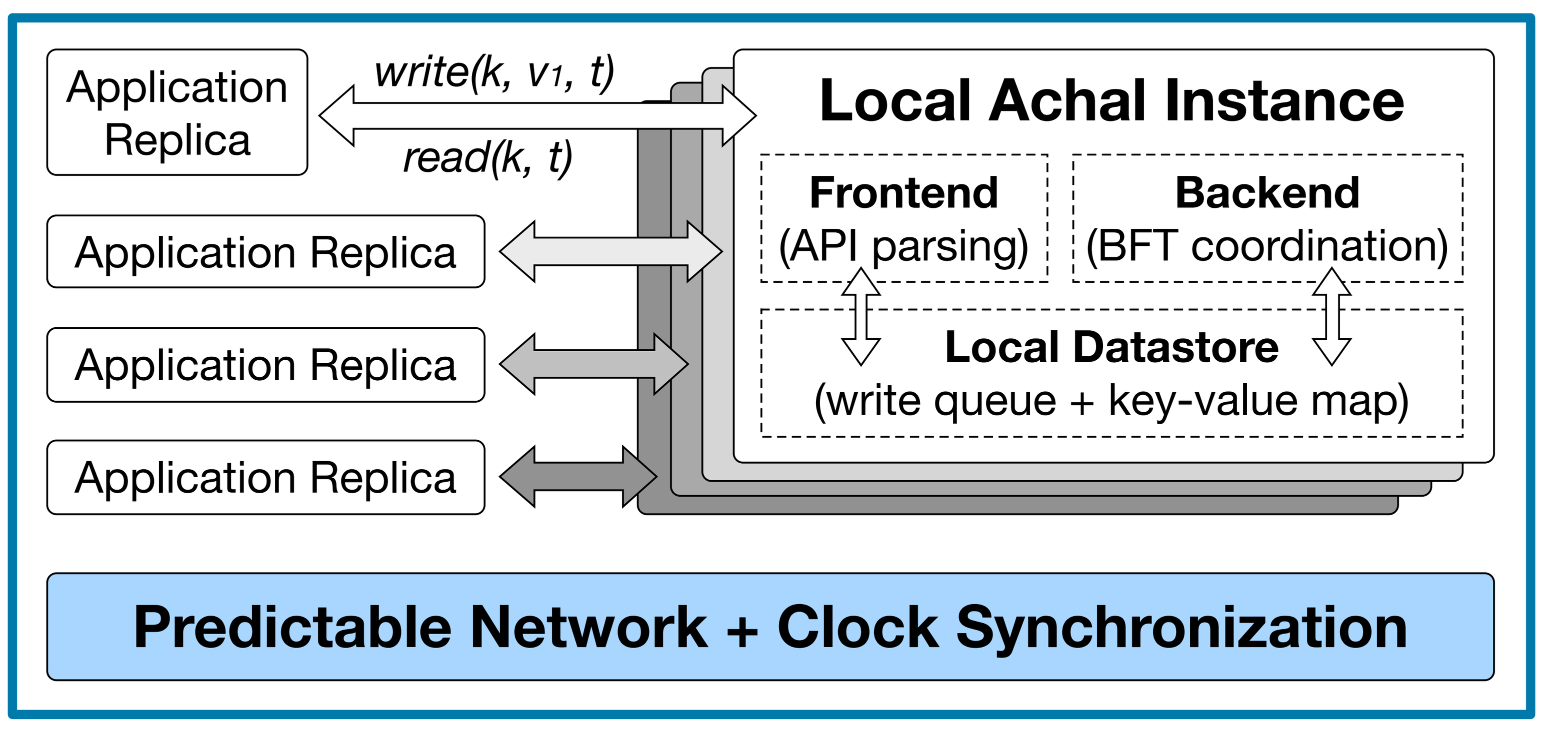
Our approach

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

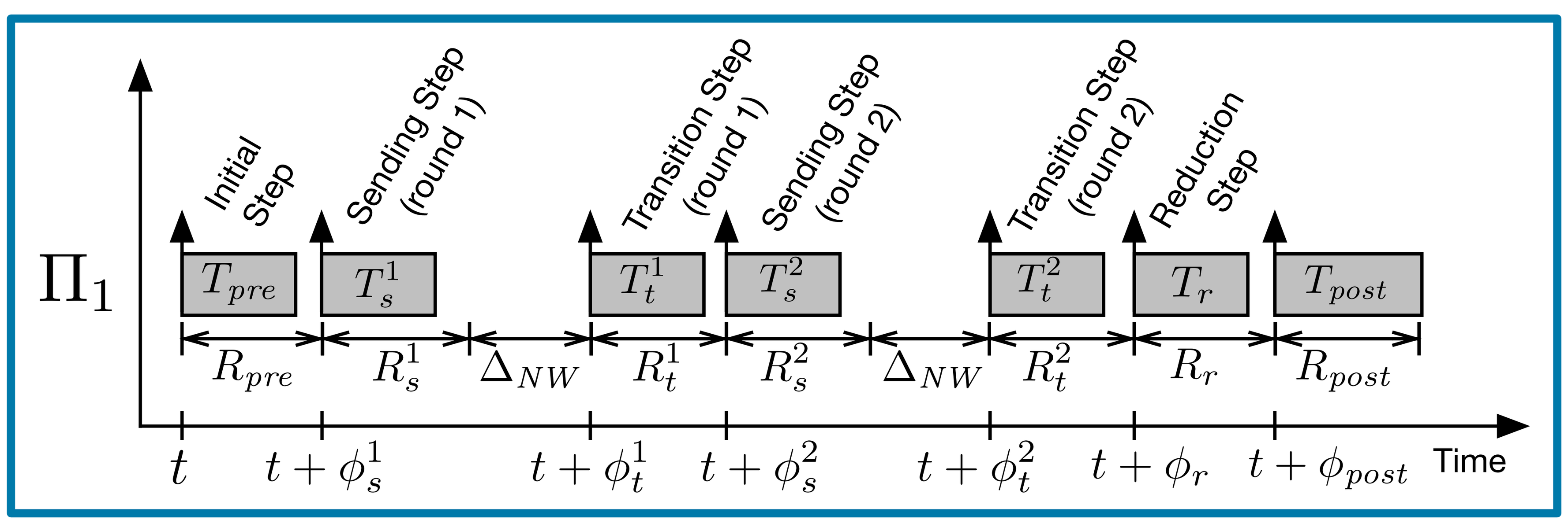
Algorithm 4.2 Periodic task of a PID controller for balancing an inverted pendulum, programmed over Achal.

```
procedure PeriodicTaskActivation
    time ← timeOfLastActivation()
    current ← getSensorData()
    error ← target − current
    integral ← Achal.read("integralKey", time) + error
    derivative ← error − Achal.read("errorKey", time)
    force ← kp ∗ error + ki ∗ integral + kd ∗ derivative
    time ← timeOfNextActivation()
    Achal.write("errorKey", error, time)
    Achal.write("integralKey", integral, time)
    actuate(force)
```

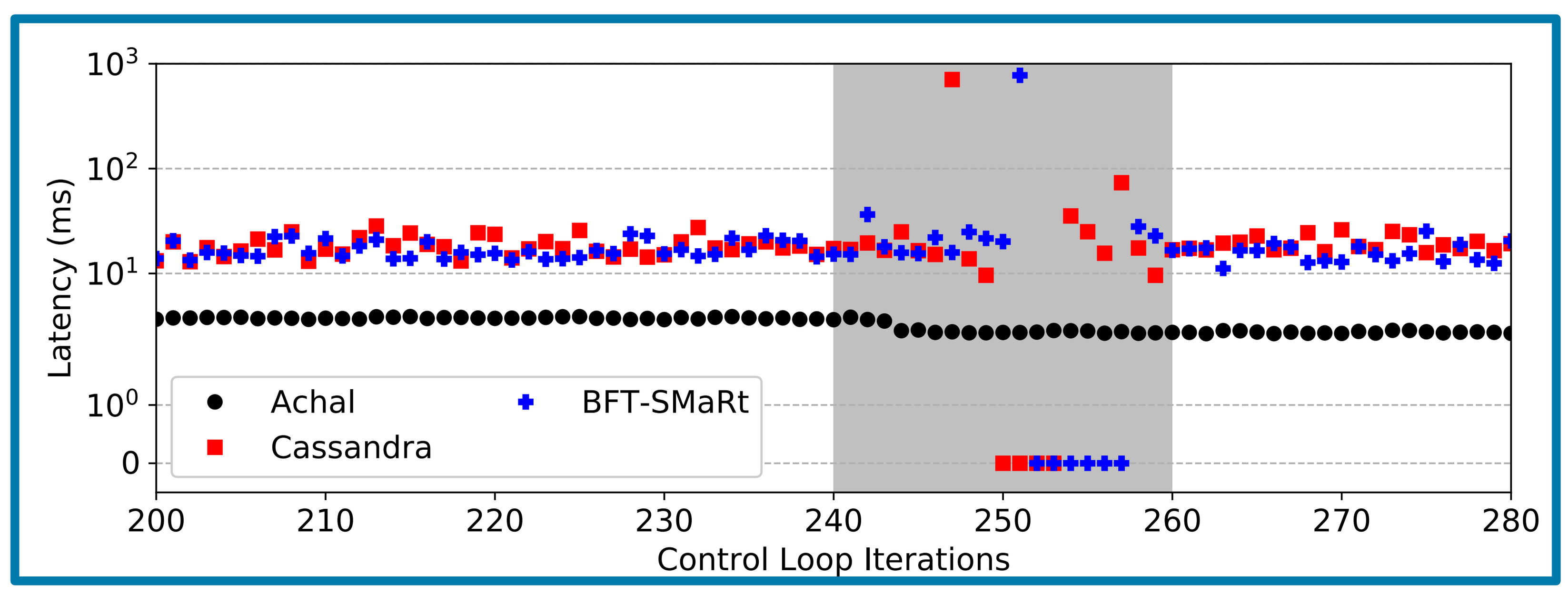
- Fault-induced errors
- Execution time jitter
- Task re-executions
- Message reorderings

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