Achal: Building Highly Reliable Networked Control Systems

Arpan Gujarati, Malte Appel, and Björn B. Brandenburg
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.
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
Achal is a **Byzantine Fault Tolerant (BFT) middleware** for actively replicating networked control systems over Ethernet.

Non-malicious Byzantine errors (inconsistent broadcasts) 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.

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

Achal is designed to mask these errors.

Use of latency-aware Ethernet standards on the rise.
Motivation
Motivation

Inadequate resources (developer hours, computing power, component costs)

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

Reliability

Expected operations in the future

Commercial, space, and military aircraft
Autonomous vehicles
Surgical robots
Other robots and drones

# failure events will rise proportionally

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

Controller task replica $R_1$
- **Local Achal Instance $A_1$**
  - **Frontend**
  - **Backend**
    - Local datastore
    - Fuse semantics
    - BFT coordination

- **Networking Layer**
  - Clock synchronisation, e.g., using PTP
  - Ethernet TSN

- **Controller task replica $R_2$**
- **Controller task replica $R_3$**
- **Controller task replica $R_4$**

- **Frontend** $A_2$
- **Frontend** $A_3$
- **Frontend** $A_4$

- **Read** ($k$, $t$)
- **Write** ($k$, $v$, $t$)
Key contribution: **CPS-friendly** BFT middleware

The datastore is aware of the temporal properties of the applications, e.g., the data age and freshness requirements of the control loops.
Key contribution: **CPS-friendly** BFT middleware

Controller task replica \( R_1 \)

- local Achal instance \( A_1 \)
  - read(\( k, t \))
  - write(\( k, v, t \))

Frontend

- local datastore

Backend

- fuse semantics
  - BFT coordination

Networking Layer

- clock synchronisation, e.g., using PTP
- Ethernet TSN

A classic leaderless BFT protocol for state coordination
Key contribution: **CPS-friendly** BFT middleware

Controller task replica $R_1$

```
read(key k, timestamp t)
write(key k, value v, timestamp t)
```

Local Achal Instance $A_1$

Frontend

```
Local datastore
```

Fuse semantics

BFT coordination

Networking Layer

```
Clock synchronisation, e.g., using PTP
Ethernet TSN
```

To filter noise in CPS applications
Key contribution: **CPS-friendly** BFT middleware

Controller task replica $R_1$

- read(key $k$, timestamp $t$)
- write(key $k$, value $v$, timestamp $t$)

Local Achal Instance $A_1$
- Frontend
- Local datastore
- Backend
- Fuse semantics
- BFT coordination

Networking Layer
- Clock synchronisation, e.g., using PTP
- Ethernet TSN

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

- Controller task replica $R_1$
  - read(key $k$, timestamp $t$)
  - write(key $k$, value $v$, timestamp $t$)

- Local Achal Instance $A_1$
  - Frontend
  - Local datastore
  - Backend
    - Fuse semantics
    - BFT coordination

- Networking Layer
  - Clock synchronisation, e.g., using PTP
  - Ethernet TSN

- Predictable implementation
  - Schedulability analysis
  - Reliability analysis

Arpan Gujarati (MPI-SWS)
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?
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?
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?