

# Verlässliche Systemsoftware

Übungen zur Vorlesung

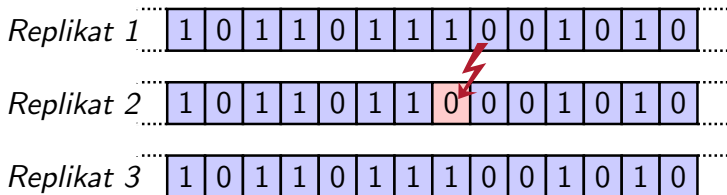
Aufgabe 3: TMR

**Phillip Raffeck, Simon Schuster, Peter Ulbrich**

Technische Universität Dortmund  
Lehrstuhl für Informatik 12 (Arbeitsgruppe Systemsoftware)  
<https://sys.cs.tu-dortmund.de>

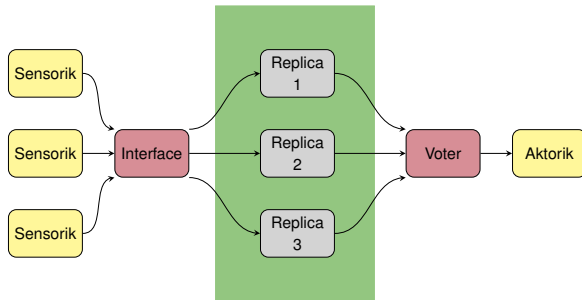
Wintersemester 2020





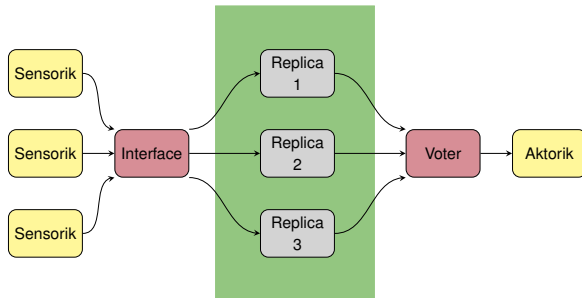
- **Wie viele Replikate** benötigt man zur Fehlermaskierung?
- Arten des Fehlverhaltens (von  $n$  Replikaten sind  $f$  fehlerhaft)
  - 1 fail-silent  $\rightarrow$  Anzahl Replikate:  $n = f + 1$
  - 2 fail-consistent  $\rightarrow$  Anzahl Replikate:  $n = 2f + 1$
  - 3 malicious  $\rightarrow$  Anzahl Replikate:  $n = 3f + 1$ , bösartige verteilte Systeme

# Triple Modular Redundancy



- Schnittstelle sammelt Eingangsdaten (Replikdeterminismus)
- Insbesondere: Mögl. Mehrheitsentscheid für redundante Sensordaten
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis
- Ergebnis wird an Aktuator versendet

# Triple Modular Redundancy



## Redundanzbereich

Ausschließlich Replikatausführung

- Mehrheitsentscheid über Berechnungsergebnisse
- Erweiterung der Ausgangsseite mit Informationsredundanz



## Replikat 1

```
void repl_1(void *p){  
    ticks_t time =  
        ezs_get_time();  
  
    ...  
}
```

## Replikat 2

```
void repl_2(void *p){  
    ticks_t time =  
        ezs_get_time();  
  
    ...  
}
```

## Replikat 3

```
void repl_3(void *p){  
    ticks_t time =  
        ezs_get_time();  
  
    ...  
}
```

## Sicherstellung Replikdeterminismus

- Globale diskrete Zeitbasis
- Einigung über Eingabewerte
- Statische Kontrollstruktur der Replikate
- Deterministische Algorithmen

## Sicherstellung Systemverhalten

☞ Replikate müssen *innerhalb bestimmter Zeitspanne* terminieren



# Process-Level Redundancy

