Technische Universität Braunschweig

RAILCHAIN

mFUND Das Startkapital für die Mobilität 4.0

ZugChain: Blockchain-Based Juridical Data Recording in Railway Systems

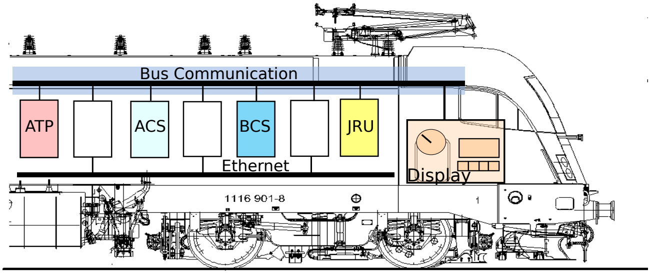
Conference on Dependable Systems and Networks 2022

Signe Rüsch[1], Kai Bleeke[1], Ines Messadi[1], Stefan Schmidt[1], Andreas Krampf[2], Katharina Olze[2], Susanne Stahnke[3], Robert Schmid[4], Lukas Pirl[4], Roland Kittel[5], Andreas Polze[4], Marquart Franz[3], Matthias Müller[2], Leander Jehl[1,6], and Rüdiger Kapitza[1]

[1]TU Braunschweig, Germany `{ruesch, bleeke}@ibr.cs.tu-bs.de`
[2]Siemens Mobility [3]Siemens [4]HPI, University of Potsdam
[5]DB Systel [6]University of Stavanger

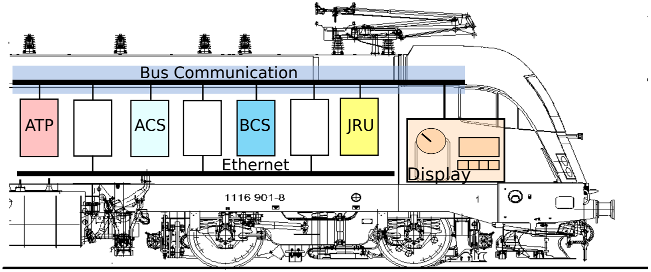
Current Juridical Data Logging

- Juridical Recording Unit **(JRU)**:
  - Train's "black box", records all juridically relevant data
  - E.g. speed, brake activity, door activity, timestamps, ...
  - Data to be logged received via bus communication



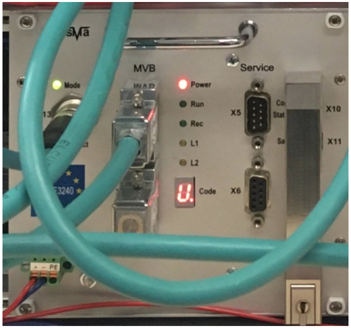
Current Juridical Data Logging

- Juridical Recording Unit **(JRU)**:
  - Train’s “black box”, records all juridically relevant data
  - E.g. speed, brake activity, door activity, timestamps, …
  - Data to be logged received via bus communication
- Detect malfunctions, accident root cause analysis
- Built to withstand **physical damage**



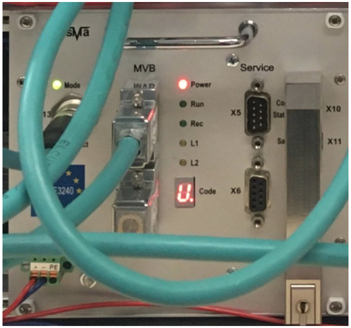
Juridical Data Logging: Problems

- **Single point of failure** [Hartong et al., 2008]
  - Potential data loss
  - Destruction in crash
  - Data manipulation during extraction



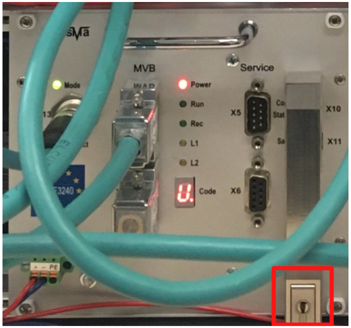
Juridical Data Logging: Problems

- **Single point of failure** [Hartong et al., 2008]
  - Potential data loss
  - Destruction in crash
  - Data manipulation during extraction
- Complex **data extraction**
  - Access via physical key
  - No remote extraction available



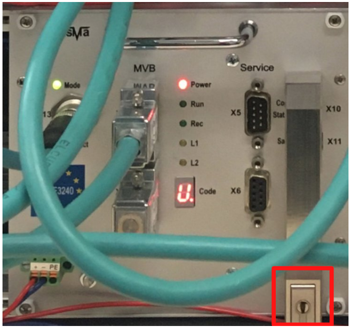
Juridical Data Logging: Problems

- **Single point of failure** [Hartong et al., 2008]
  - Potential data loss
  - Destruction in crash
  - Data manipulation during extraction
- Complex **data extraction**
  - Access via physical key
  - No remote extraction available



Juridical Data Logging: Problems

- **Single point of failure** [Hartong et al., 2008]
  - Potential data loss
  - Destruction in crash
  - Data manipulation during extraction
- Complex **data extraction**
  - Access via physical key
  - No remote extraction available
- Expensive and **proprietary** device
  - One company has logging authority
  - No distribution of trust



Juridical Data Logging: Goals

- Distributed:
  - Replace single device with **distributed, replicated system**
  - ↳ High reliability and availability
  - ↳ Ensure data integrity

Juridical Data Logging: Goals

- Distributed:
  - Replace single device with **distributed, replicated system**
  - ➥ High reliability and availability
  - ➥ Ensure data integrity
- Cost factor:
  - JRU: expensive, dedicated device
  - ➥ Use **on-train commodity hardware**
  - ➥ No changes to safety-critical infrastructure

Juridical Data Logging: Goals

- Distributed:
  - Replace single device with **distributed, replicated system**
  - ➥ High reliability and availability
  - ➥ Ensure data integrity
- Cost factor:
  - JRU: expensive, dedicated device
  - ➥ Use **on-train commodity hardware**
  - ➥ No changes to safety-critical infrastructure
- Allow **multiple companies** to contribute to juridical recording

Juridical Data Logging: Goals

- Distributed:
  - Replace single device with **distributed, replicated system**
  - ➥ High reliability and availability
  - ➥ Ensure data integrity
- Cost factor:
  - JRU: expensive, dedicated device
  - ➥ Use **on-train commodity hardware**
  - ➥ No changes to safety-critical infrastructure
- Allow **multiple companies** to contribute to juridical recording
- Facilitate **easy data export** for predictive maintenance

Distributed Juridical Data Logging

- **Mathematical analysis** of distributed JRUs [Braband et al., 2020]
  - Approach: distributed JRU functionality on commodity hardware

Distributed Juridical Data Logging

- **Mathematical analysis** of distributed JRUs [Braband et al., 2020]
  - Approach: distributed JRU functionality on commodity hardware
  - Examined train crash data provided by the UIC and ERA
  - Analysed probabilities for different scenarios (e.g. fire) and their effects on the recording hardware

Distributed Juridical Data Logging

- **Mathematical analysis** of distributed JRUs [Braband et al., 2020]
  - Approach: distributed JRU functionality on commodity hardware
  - Examined train crash data provided by the UIC and ERA
  - Analysed probabilities for different scenarios (e.g. fire) and their effects on the recording hardware
- Findings:
  - Commodity hardware easier to destroy
  - Crashes rarely affect whole train
  - Distributed JRU reaches **comparable reliability and availability**

Distributed Juridical Data Logging

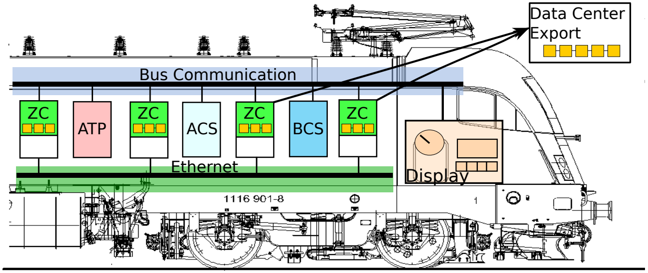
- **Mathematical analysis** of distributed JRUs [Braband et al., 2020]
  - Approach: distributed JRU functionality on commodity hardware
  - Examined train crash data provided by the UIC and ERA
  - Analysed probabilities for different scenarios (e.g. fire) and their effects on the recording hardware

- Findings:
  - Commodity hardware easier to destroy
  - Crashes rarely affect whole train
  - Distributed JRU reaches **comparable reliability and availability**

➥ How can we ensure that all nodes **consistently** log the same data?

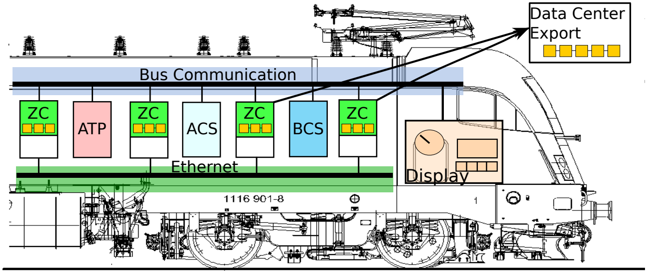
ZugChain Design

- ZugChain nodes distributed across the train
- Read safety-critical signals from **bus**



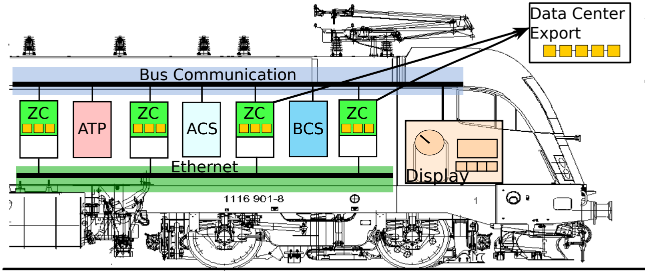
ZugChain Design

- ZugChain nodes distributed across the train
- Read safety-critical signals from **bus**
- Nodes **agree** on order and log received signals in **blockchain**



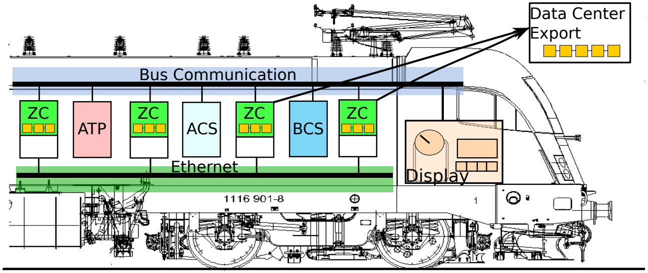
ZugChain Design

- ZugChain nodes distributed across the train
- Read safety-critical signals from **bus**
- Nodes **agree** on order and log received signals in **blockchain**
- Use **existing, non-critical links** for consensus



ZugChain Design

- ZugChain nodes distributed across the train
- Read safety-critical signals from **bus**
- Nodes **agree** on order and log received signals in **blockchain**
- Use **existing, non-critical links** for consensus
- Blockchain structure ensures **data integrity** and enables **export**



Alternative Designs

- Multiple independent nodes
  - Logs are not synchronized
  - Can lead to **data loss** in case of a crash

Alternative Designs

- Multiple independent nodes
  - Logs are not synchronized
  - Can lead to **data loss** in case of a crash
- Simple leader/follower architecture
  - Only one node needs to be read in case of a crash
  - However, data can still be **compromised**
  - Bus system can also lead to **corrupted transmissions**

Alternative Designs

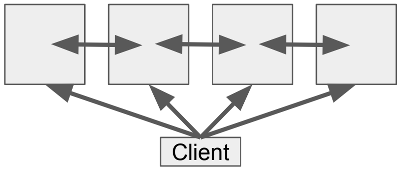
- Multiple independent nodes
  - Logs are not synchronized
  - Can lead to **data loss** in case of a crash
- Simple leader/follower architecture
  - Only one node needs to be read in case of a crash
  - However, data can still be **compromised**
  - Bus system can also lead to **corrupted transmissions**
- Agreement
  - ZugChain nodes may be **co-located** on observed machines
  - Arbitrary faults on machine can influence recording functionality

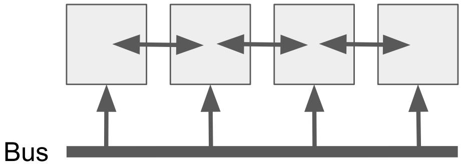
Alternative Designs

- Multiple independent nodes
  - Logs are not synchronized
  - Can lead to **data loss** in case of a crash
- Simple leader/follower architecture
  - Only one node needs to be read in case of a crash
  - However, data can still be **compromised**
  - Bus system can also lead to **corrupted transmissions**
- Agreement
  - ZugChain nodes may be **co-located** on observed machines
  - Arbitrary faults on machine can influence recording functionality

➥ Byzantine agreement is necessary for ZugChain

Bus Communication Instead of Clients

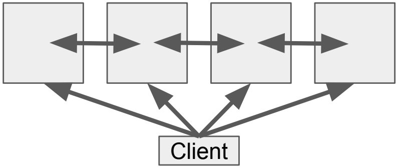


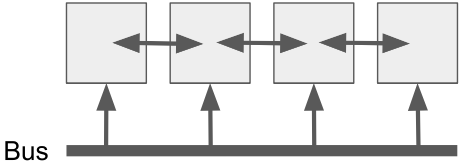


- Traditional clients
  - Authenticated requests
  - Replicas can forward requests
  - Client can re-transmit requests

- Bus communication
  - Unauthenticated broadcast
  - Synchronous communication
  - Not interested in replies

Bus Communication Instead of Clients



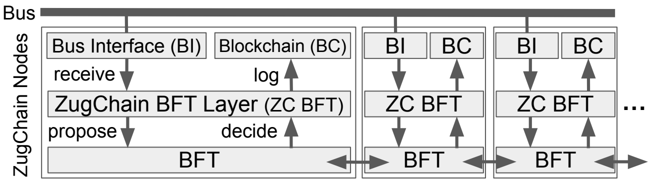


- Traditional clients
  - Authenticated requests
  - Replicas can forward requests
  - Client can re-transmit requests
- Bus communication
  - Unauthenticated broadcast
  - Synchronous communication
  - Not interested in replies

Bus clients have to be treated differently

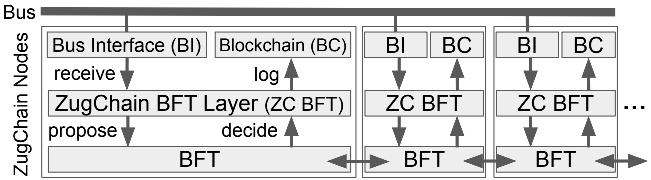
ZugChain Design: Communication Layer

- Underlying Byzantine agreement protocol, e.g. **PBFT**
- Nodes can read **identical** or **diverging** data from bus
- Communication layer ensures **no payload duplication**
  - Log all juridically relevant data, but **filter** duplicates
  - Prevent **omission** of data



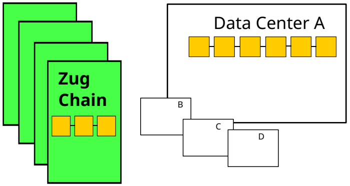
ZugChain Design: Communication Layer

- Nodes assume others read identical data
- **Primary** proposes request, **backups** don't propose
- But they set a **"soft" timer** for each request:
  - If it expires before matching pre-prepare, **broadcasting** to all
  - Start **"hard" timer**, before view change



Export Protocol: Goals

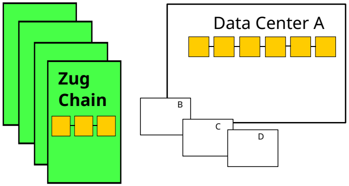
- Storage on the node is limited
  - Recent data is of highest priority
  - Block headers necessary for blockchain
- ➥ We regularly export data to avoid deletion



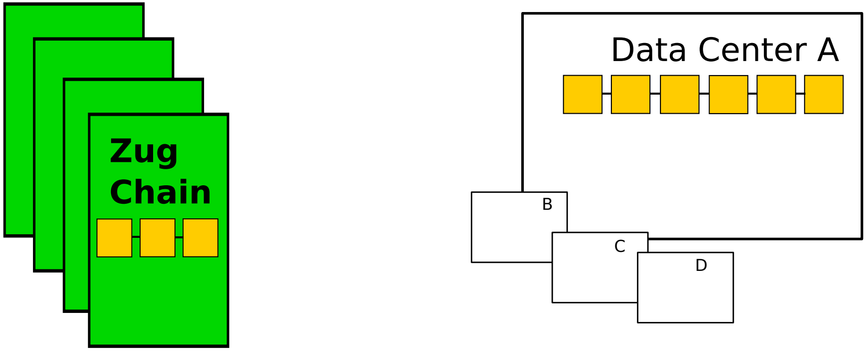
Export Protocol: Goals

- Storage on the node is limited
  - Recent data is of highest priority
  - Block headers necessary for blockchain

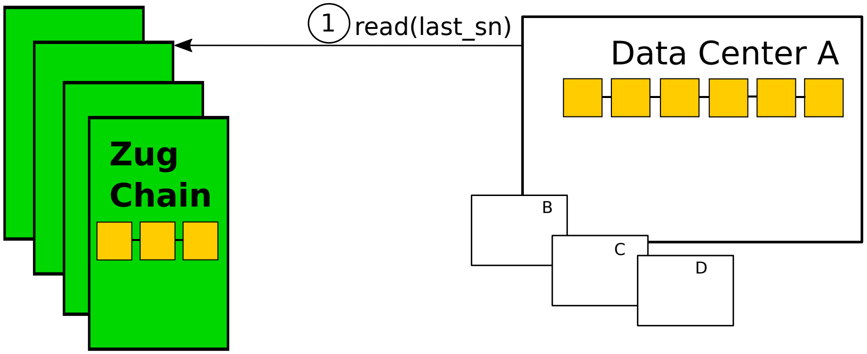
  ⤷ We regularly export data to avoid deletion
- Export to data centers of railway companies
  - Multiple parties sign off on deletion
- Blockchain facilitates export
  - Verify integrity from a single response



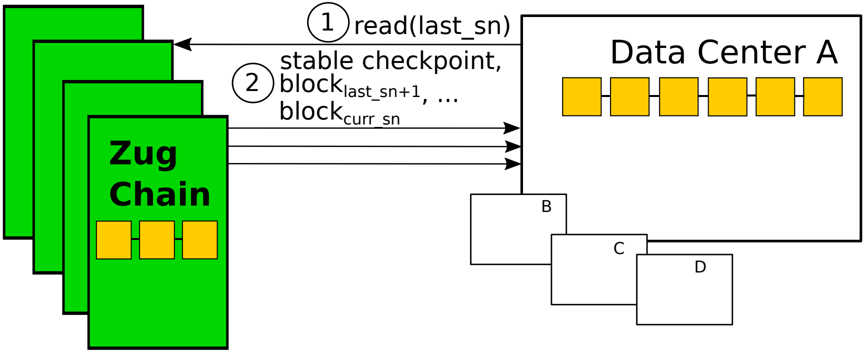
ZugChain Export Protocol: Steps



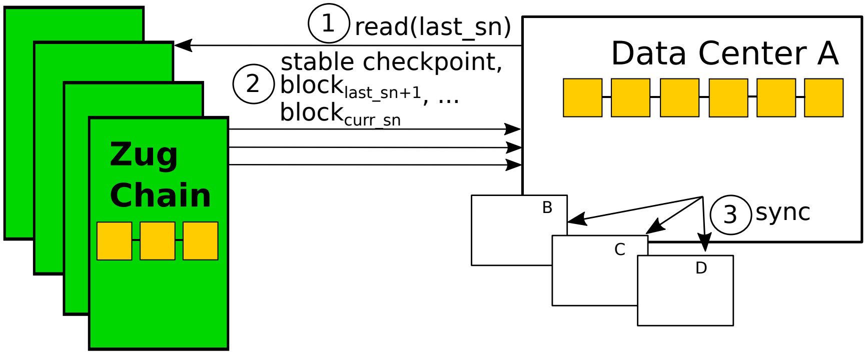
ZugChain Export Protocol: Steps



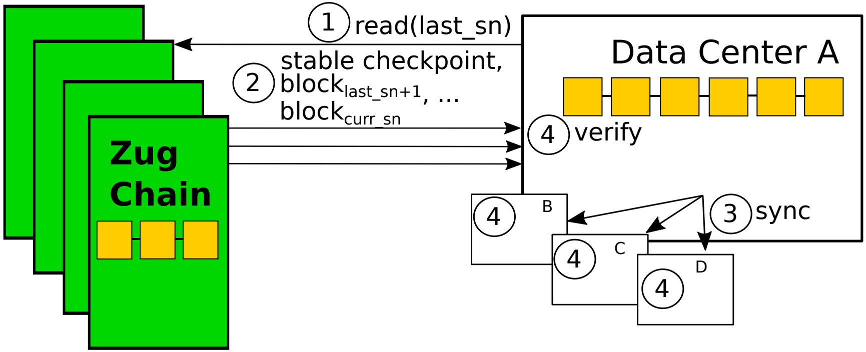
ZugChain Export Protocol: Steps



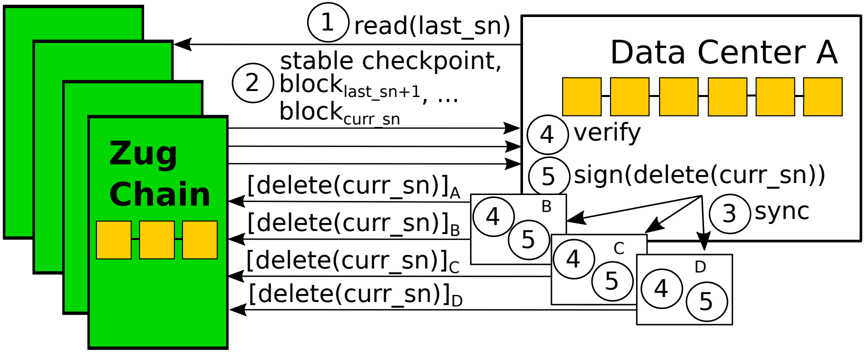
ZugChain Export Protocol: Steps


1 read(last_sn)
2 stable checkpoint, block$_{last\_sn+1}$, ... block$_{curr\_sn}$
Data Center A
Zug Chain
3 sync
B
C
D

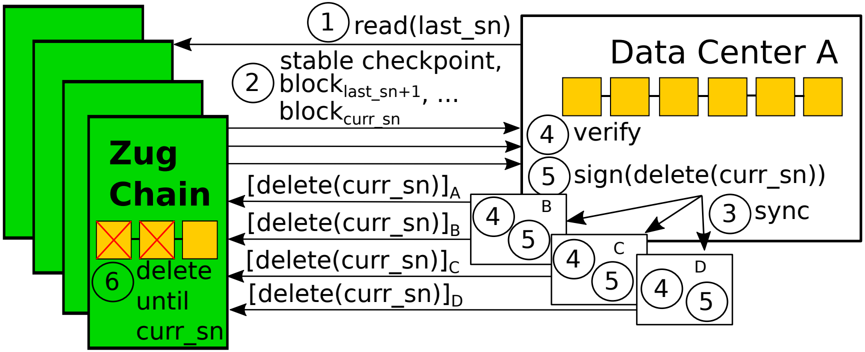
ZugChain Export Protocol: Steps


1 read(last_sn)
2 stable checkpoint, block$_{last\_sn+1}$, ... block$_{curr\_sn}$
Data Center A
4 verify
3 sync
Zug Chain
4 B
4 C
4 D

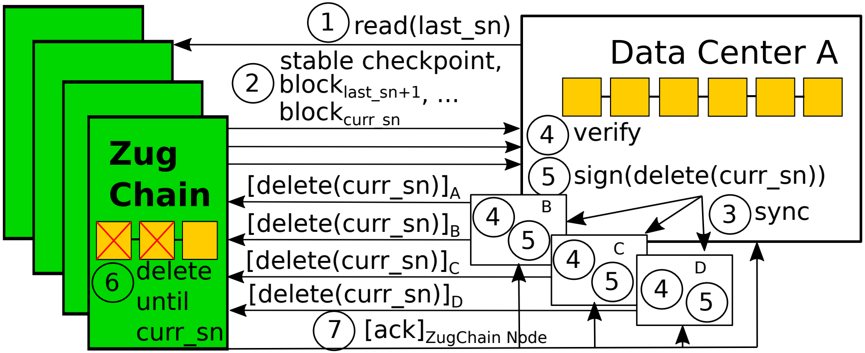
ZugChain Export Protocol: Steps



ZugChain Export Protocol: Steps

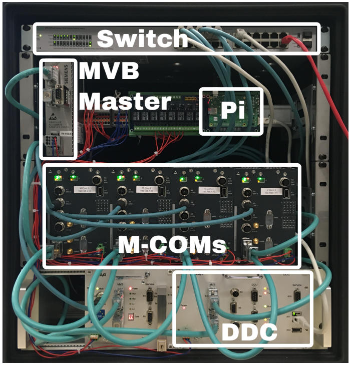


ZugChain Export Protocol: Steps



ZugChain Evaluation Setting

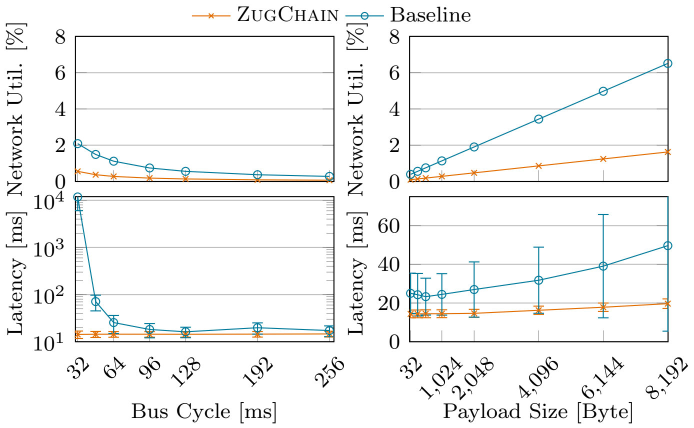
- 4 M-COMs:
  - Quad-Core CPU
  - 2GB RAM
  - 100Mbit/s Ethernet and MVB connection
  - Yocto Linux, kernel v3.10.17
- Communication layer vs. PBFT:
  - Naïve baseline
  - Order **all** data, i.e. requests logged 4×
- Vary bus cycle time and payload size



Background: Multifunction Vehicle Bus (MVB)

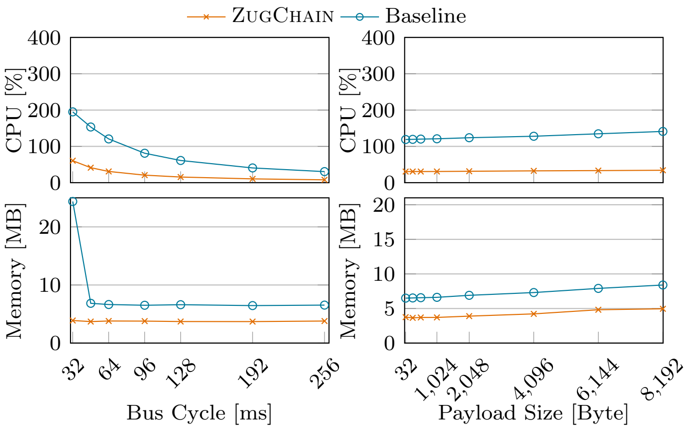
- Fieldbus supported by Siemens and ABB
- Synchronized leader/follower communication
  - Bus master sets the **cycle time** with each follower
- Three types of data:
  - Process Data
  - Message Data
  - Supervisory Data
- Focus on **process data** for juridical recording
- Communication errors can still occur

Evaluation – Network Utilization and Latency



↪ ZugChain: less **bandwidth** and lower, more stable **latencies**!

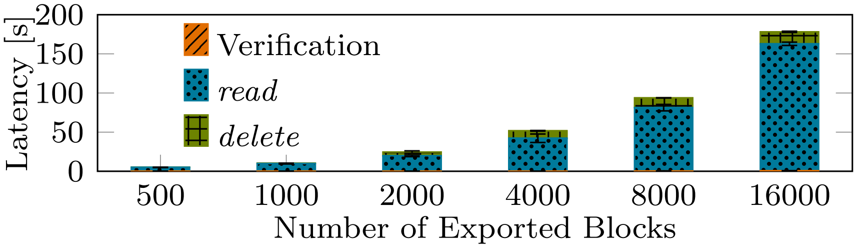
Evaluation – Resource Usage



➥ ZugChain better suited for **shared, resource-constrainend devices!**

ZugChain Export Evaluation – LTE

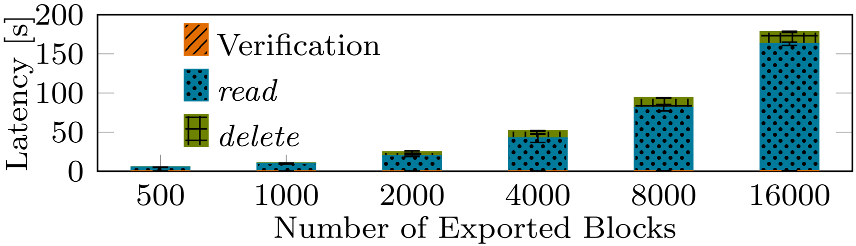
- 5 minutes (500 blocks) to 3 hours (16k blocks) of data
  - Exported to AWS VM t2.xlarge



➥ Export feasible during train stops or continuously!

ZugChain Export Evaluation – LTE

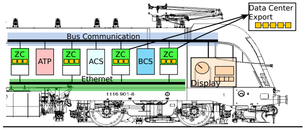
- 5 minutes (500 blocks) to 3 hours (16k blocks) of data
  - Exported to AWS VM t2.xlarge

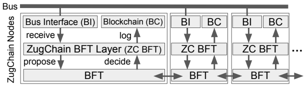


➥ Export feasible during train stops or continuously!
➥ View change and Byzantine fault measurements in paper!

Conclusion

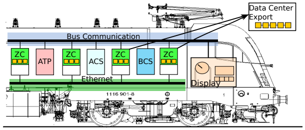
- Further **digitalization** in railway operations
- Maintain JRU's **reliability and availability**

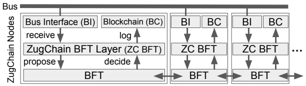




Conclusion

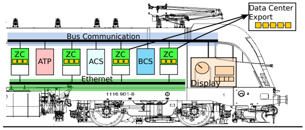
- Further **digitalization** in railway operations
- Maintain JRU's **reliability and availability**
- Allow data **integrity** verification after crashes by using blockchains
  - Even with only one remaining copy

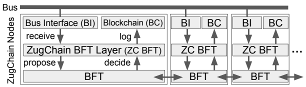




Conclusion

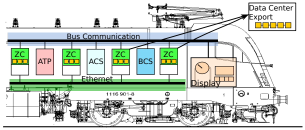
- Further **digitalization** in railway operations
- Maintain JRU's **reliability and availability**
- Allow data **integrity** verification after crashes by using blockchains
  - Even with only one remaining copy
- Easy **data export** to railway companies

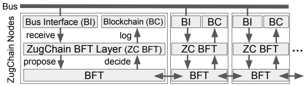




Conclusion

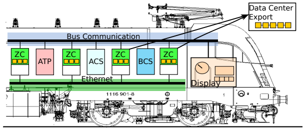
- Further **digitalization** in railway operations
- Maintain JRU's **reliability and availability**
- Allow data **integrity** verification after crashes by using blockchains
  - Even with only one remaining copy
- Easy **data export** to railway companies
- Efficient filtering in communication layer for good **performance** on real train hardware

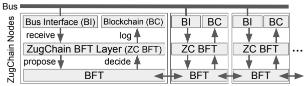




Conclusion

- Further **digitalization** in railway operations
- Maintain JRU's **reliability and availability**
- Allow data **integrity** verification after crashes by using blockchains
  - Even with only one remaining copy
- Easy **data export** to railway companies
- Efficient filtering in communication layer for good **performance** on real train hardware

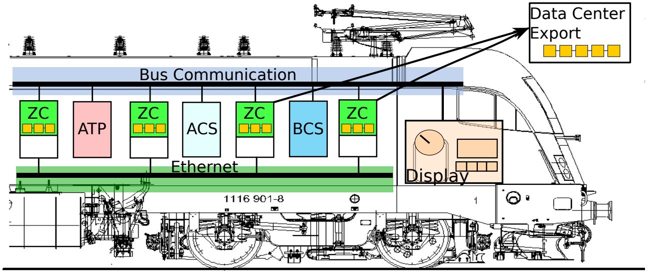




Thank you for your attention! Questions?
`{ruesch,bleeke}@ibr.cs.tu-bs.de`

ZugChain Design: Crash Scenario

- After a crash, any tampering of data should be **detected**
- Probability of multiple nodes surviving crash are high [Braband, 2020]
- If only one copy survives …
  - **Signatures** on blocks and consensus messages in blockchain
  - Verify that this is an authentic log
  - Tampered data cannot contain **2f+1 valid replica signatures**!



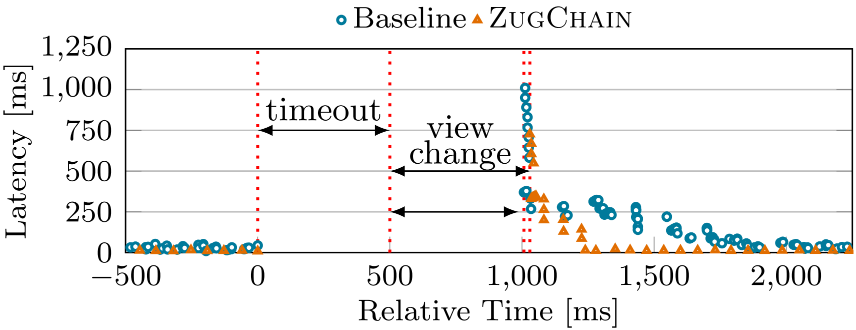
ZugChain Interface

Module	Call	Explanation
① down	Propose(r)	proposes request to consensus group
① down	Suspect(id)	suspect node to be faulty, init. view change
① up	Decide(r, sn)	totally ordered request and seq.no.
① up	NewPrimary	returns new primary after view change
② down	Receive(req)	read parsed request from bus
② up	Log(req, id, sn)	append request to totally ordered log

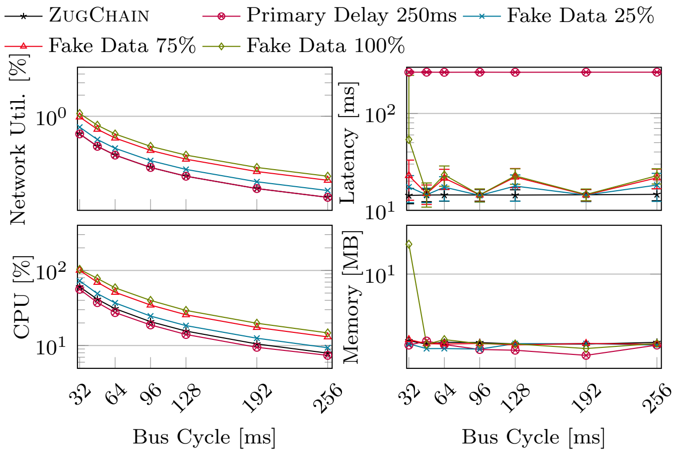
Table 1: Interfaces of BFT (①) and ZugChain (②).

ZugChain View Change Evaluation

- Complex operation: after timeout, exchange all open requests
- ↳ ZugChain has to handle fewer messages
- Duration: ZugChain 530ms, PBFT 507ms
- Stabilization time: ZugChain 210ms, PBFT 824ms



ZugChain Byzantine Behaviour Evaluation


ZugChain
Primary Delay 250ms
Fake Data 25%
Fake Data 75%
Fake Data 100%
Network Util. [%]
10^0
Latency [ms]
10^2
10^1
CPU [%]
10^2
10^1
Memory [MB]
10^1
Bus Cycle [ms]
Bus Cycle [ms]