CRDTs in the Wild

Jakob Meyer-Hilberg

10. Februar 2023

Seminar ATVS WS22/23

Conflict-free Replicated Data Type (CRDT)

Was sind CRDTs?

Dokument,
an dem verschiedene Teilnehmer arbeiten

1. CRDTS garantieren, dass jeder Teilnehmer zu **jeder Zeit** das Dokument **verändern** kann.
2. Es gibt einen Algorithmus, der Dokumente von verschiedenen Teilnehmern **mergen** (zusammenführen) kann.
3. Die Dokumentänderungen aller Teilnehmer **konvergieren** zu einem einheitlichen Dokument.

Auswahl üblicher CRDT-Datentypen

G-Counter: (Grow-only Counter) Zähler, der nur hochzählt.

SequenzCRDT: Datentyp, der in kollaborativen Texteditoren verwendet wird (geordnete Liste).

PN-Counter: (Positive-Negative Counter) Besteht aus zwei G-Countern, mit denen hoch- und runtergezählt werden kann. Exemplarischer Anwendungsfall: Zählung im Parkhaus mit mehreren Zählern.

G-Set: (Grow-only Set) Beispielsweise die Besucherliste einer Website.

2p-Set: (Two-Phase Set) gültige und ungültige öffentliche Schlüssel (public keys)

LWW-Element-Set: (Last-Write-Wins Element-Set) Zeitlich letzte Änderung überschreibt alle vorhergehenden Änderungen.

Konsistenzmodelle im Verteilten System

Vier Burger werden gleichzeitig belegt.

eventual consistency (englisch: eventual = abschließend)

Reihenfolge unwichtig

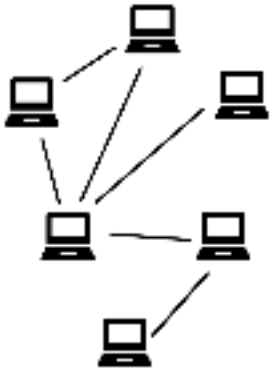
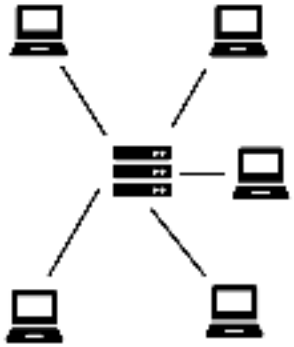
strong eventual consistency (SEC)

Reihenfolge wichtig

strong consistency

Zugriff blockiert,
wenn geschrieben wird

CRDTs versus OTs (Operation Transformation)

CRDTs	OTs
Strong Eventual Consistency (SEC)	
+ beliebige Netzwerktopologie 	Zentraler Server 
Operation-based oder State-based (Deltas)	nur Operation
- Tombstones	Gelöschtes nicht im Dokument

Zum Programmieren verwendbare Bibliotheken

Automerge (javascript)

```
const doc1 = Automerge.init("1234")
doc1 = Automerge.change(doc1,  doc => {
    doc.cards[0].done = true
})
doc3 = Automerge.merge(doc2, doc1)
```

Yjs (javascript)

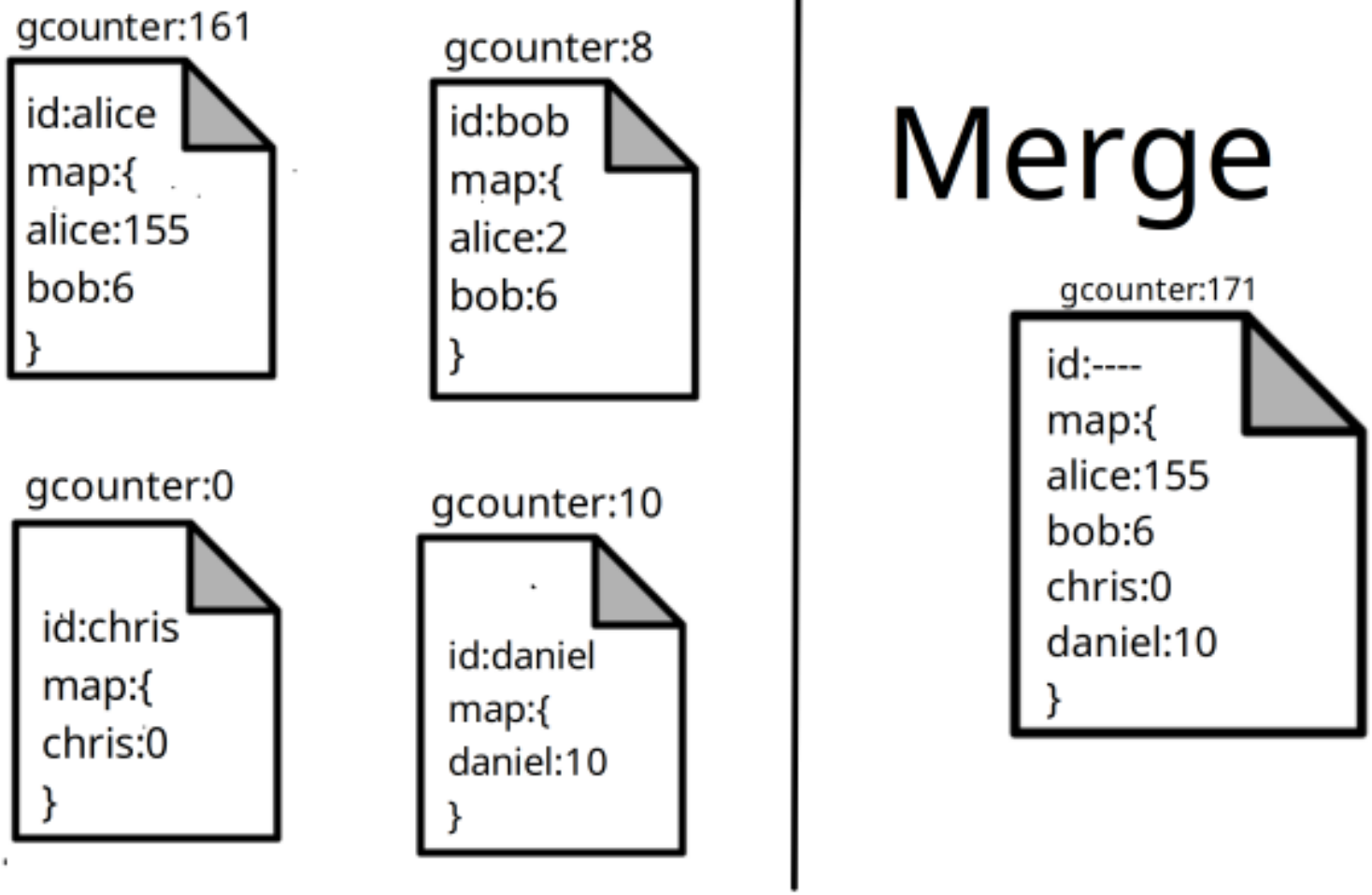
```
const ydoc = new Y.Doc()
ydoc.getArray('my-array').insert(0, ['val'])
```

xi-rope-engine (rust)

```
let mut doc = Engine::empty();
doc.set_session_id((1234 as u64, 0));
doc.merge(&doc2);
```

Beispiel eines CRDT G-Counters: Merge

Grow-only Counter



G-Counter javascript

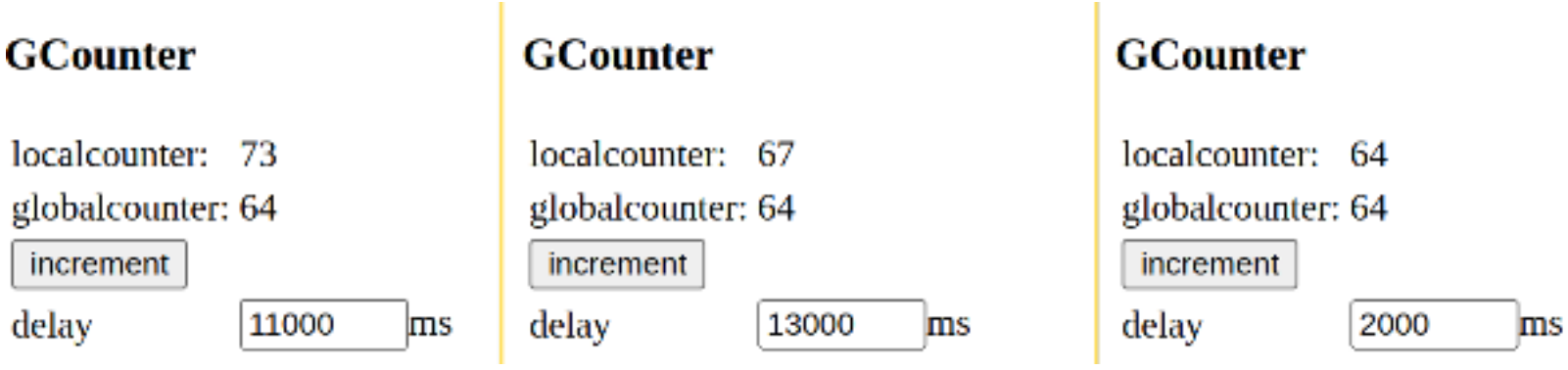
```
1  export class GCounter {
2      merge(document) {
3          for(var iter_id in document.map){
4              if(this.map[iter_id] == undefined ||
5                  document.map[iter_id] > this.map[iter_id]){
6                  this.map[iter_id] = document.map[iter_id];
7      }   }   }
8      increment() {
9          this.map[this.id] = this.map[this.id] + 1;
10     }
11     value() {
12         let sum = 0;
13         for(var iter_id in this.map){
14                 sum +=this.map[iter_id];
15         }
16         return sum;
17     }
18     // more function like constructor ...
19 }
```

G-Counter javascript demo

```
1  import {GCounter} from "./gcounter.js";
2  let doc1 = new GCounter({id:"alice"});
3  let doc2 = new GCounter({id:"bob"});
4
5  doc1.increment();
6  doc1.increment();
7  doc2.increment();
8  console.log(doc1.value()); // output: 2
9  console.log(doc2.value()); // output: 1
10
11 doc1.merge(doc2)
12 console.log(doc1.value()); // output: 3
```

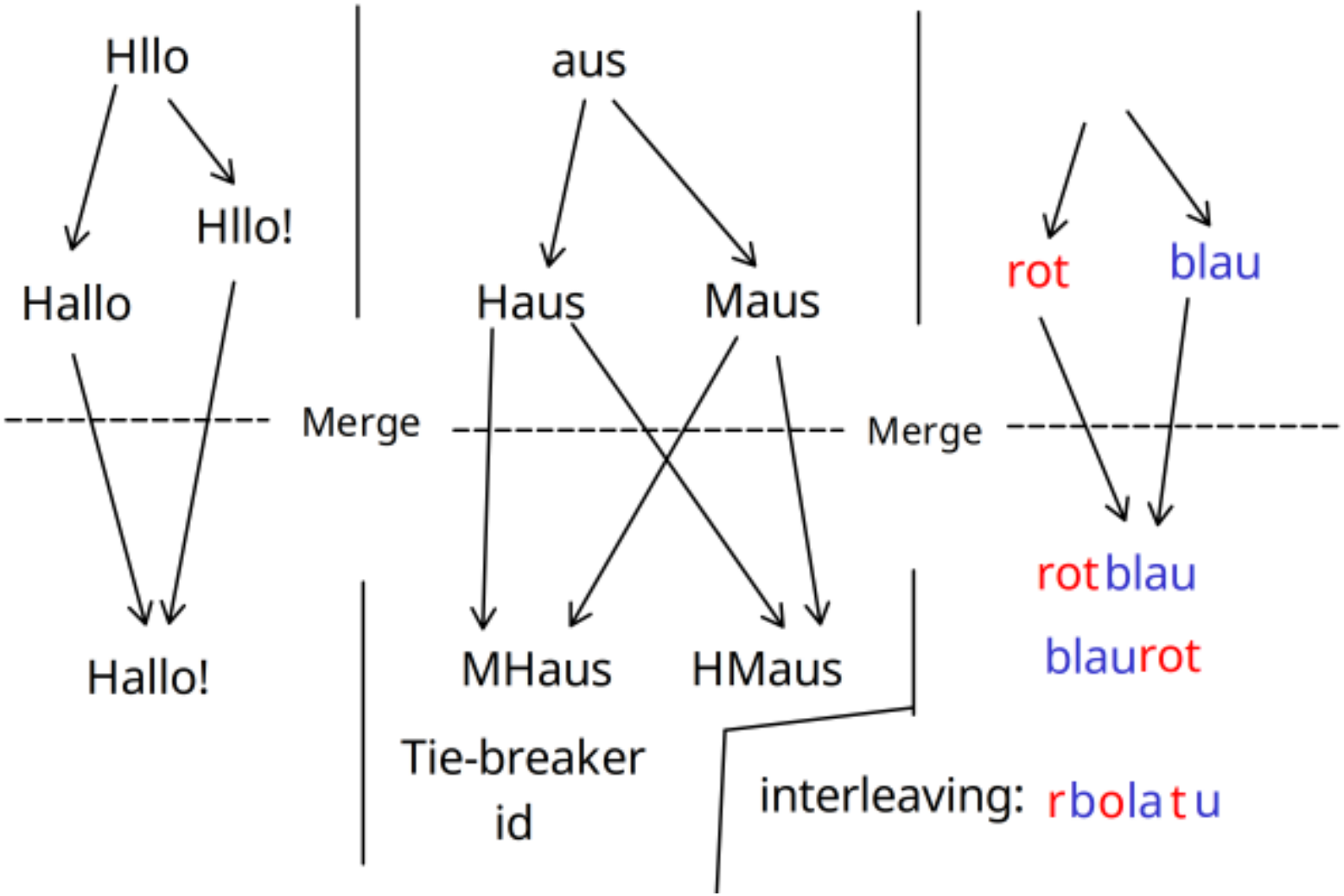
Minimalistische Fullstack-Web-Anwendung

Drei Chrome-Browser nebeneinander,
verbunden durch einen Websocket-Broadcastserver




```
1  // Simulierte Netzwerkverzögerung
2  // Client-Side
3  setTimeout(() => {
4      websocket.send(message);
5  }, delay)
```

SequenzCRDT skizziert



G-Counter mit SequenzCRDT implementieren

Die hier verwendete Klasse SequenceCRDT
wurde in 80 Zeilen implementiert.

```
1  import {SequenceCRDT} from "./sequencecrdt.js"
2  export class GCounter {
3      constructor({id}) {
4          this.sequencecrdt = new SequenceCRDT(id);
5      }
6      merge(document) {
7          this.sequencecrdt.merge(document.sequencecrdt)
8      }
9      increment() {
10         this.sequencecrdt.insert_into(0,"1")
11     }
12     value() {
13         return this.sequencecrdt.value().length
14     }
15 }
```

Vergleich von G-Counter-Implementierungen

Tabelle: Hier wird die Komplexität in Abhängigkeit von Teilnehmer t und Zähler c zusammengefasst. Diese Tabelle wurde basierend auf theoretischen Überlegungen erstellt.

	Speicher	Merge
gcounter plain javascript	$\mathcal{O}(t)$	$\mathcal{O}(t)$
gcounter automerge	$\mathcal{O}(t)$	$\mathcal{O}(t)$
xi rope sequenz als gcounter	$\mathcal{O}(c)$	$\mathcal{O}(c)$
javascript sequenz als gcounter	$\mathcal{O}(c)$	$\mathcal{O}(c^2)$

Schluss und Ausblick

Schluss:

- Verschiedene CRDT-Datentypen und ihre potenzielle Anwendung.
- Strong Eventual Consistency für asynchrone nicht-blockierende Anwendungen
- Minimalistische Fullstack-Webapp mit CRDTs
- G-Counter auf verschiedene Arten implementiert

Ausblick:

- Netzwerktopologie: P2P/WebRTC verwenden
- CRDTs spezifizieren und mathematisch verifizieren
- Teilnehmer-ID mit Kryptographie absichern