Realtime Collaboration

with Firebase

About This Talk

This talk condenses lessons learned while implementing a realtime rich text editing system at Celtx.

It was presented live to the members of the NDev software development community in 2019 as part of NDev meetup #33.

About me

Mike Burton

Senior dev at Celtx

Primary for

realtime collaboration

Terrible

Cartoonist

Celtx

Realtime-collaborative rich text editors for multiple media industries

Realtime Collaboration

Two or more users changing the same data at the same time

The Two Generals Problem

How do we know our collaborators have received all of our messages?

- We send a message
- They acknowledge
- We acknowledge their ACK
- They ACK our ACK-ACK
- ...

What do we know for sure? This is the

Two Generals Problem

The Two Generals Problem was the first computer communication problem to be proved to be unsolvable.

1.
Data Models

Operational Transformations,
Conflict-Free Replicated Data Types,
And Beyond

Operational Transformations

Focuses on ***operations***

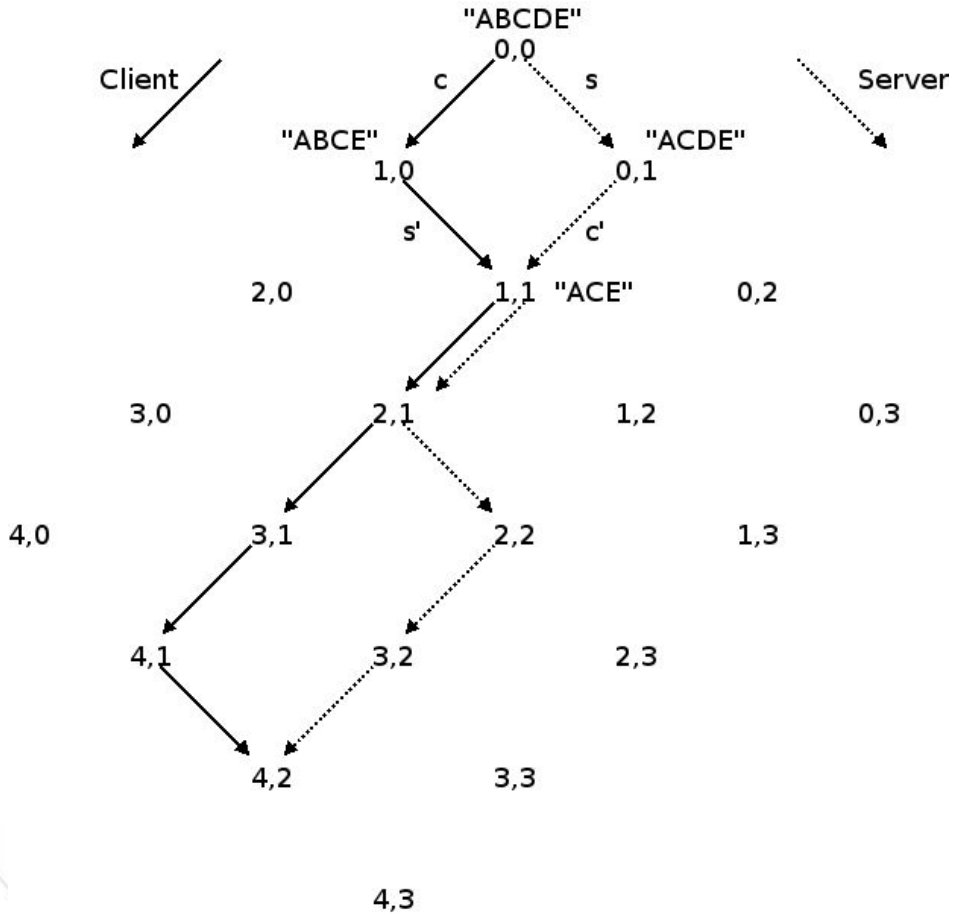
- ***Causality Preservation*** and ***Convergence***
- Operations are context-dependent
- ***Inclusive*** or ***Exclusive***
- Many implementations and extensions, not all of which guarantee convergence!

Example: Google/Apache Wave

Participants in the wave may take different paths through the state space.

When composed, however, the operations **converge**.

Here **c + s'** and **s + c'**, for example, lead to the same state: **(1,1) "ACE"**



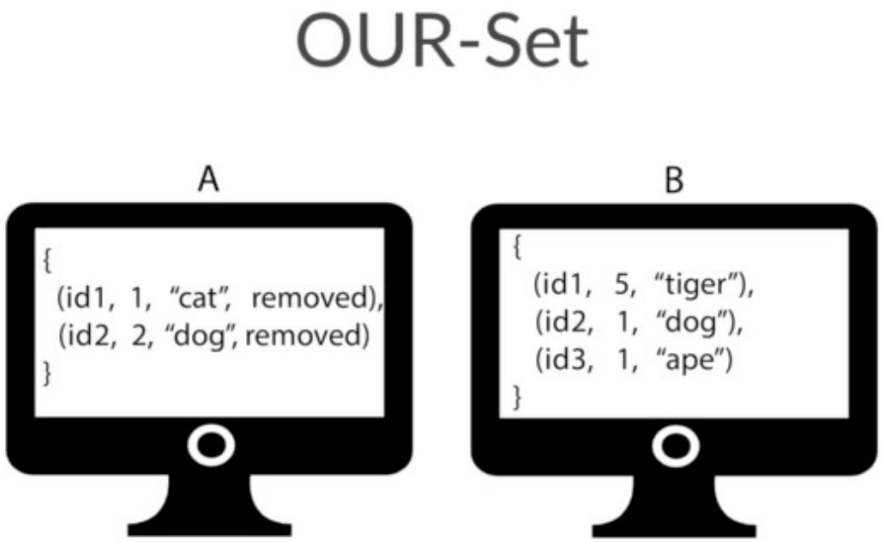
Source: **Google Wave Operational Transformation** white paper
https://svn.apache.org/repos/asf/incubator/wave/whitepapers/operational-transform/operational-transform.html

Conflict-free Replicated Data Types

Focuses on **states**

- **Commutative** vs **Convergent** RDTs
- Many (many, many) possible approaches
  - G-Set
  - OR-Set
  - LWW-Set
- Many implementations focus on two operations:
  - Merge - setwise union
  - Lookup - “meaningful” reduction

Example: NavCloud



Merge

{ (id1, 5, "tiger"), (id2, 2, "dog", removed), (id3, 1, "ape") }

Lookup

{ "tiger", "ape" }

Source: Practical Demystification of CRDTs
https://speakerdeck.com/ajantis/practical-demystification-of-crdts-lambdadays-2016

2.
Firebase

Realtime tools

Two Solutions?

Realtime Database

- 2013 launch product
- Joined Google 2014

Firestore

- Developed at Google
- “Best of RTDB and Google Cloud Storage”
- Beta ended Jan 31, 2019

Realtime Database

JSON-like data tree

Limits: 32 level max, 100,000 connections

Restricted querying capability

Queries are always deep*

Strong design recommendation for shallow data

Overall: Recommended only situationally

*: Exception: REST queries can be shallow

Firestore

Collection-of-documents tree structure

Data held in document fields

Limits: 1MB documents, <1MB fields

Rich query syntax

Queries are shallow by default

Overall: Recommended by default

3.

Simplifying Realtime Collaboration

Using a centralized authority

The Power of Ordering

Can we work around convergence?

- SQL: “pessimistic” and “optimistic” locking
- VSS/CVS: “locked” editable content
- Central Authority + pseudo-OT

Pseudo-OT

What changes under a central authority?

- Authoritative ordering
- No need to guarantee operations applied in a different order yield same result
- Invert local operations before applying remote ones

Example: ProseMirror

```
export function receiveTransaction(state, steps, clientIDs, options) {
  let collabState = collabKey.getState(state)
  let version = collabState.version + steps.length
  let ourID = collabKey.get(state).spec.config.clientID
  let ours = 0
  while (ours < clientIDs.length && clientIDs[ours] == ourID) ++ours
  let unconfirmed = collabState.unconfirmed.slice(ours)
  steps = ours ? steps.slice(ours) : steps
  if (!steps.length)
    return state.tr.setMeta(collabKey, new CollabState(version, unconfirmed))

  let nUnconfirmed = unconfirmed.length
  let tr = state.tr
  if (nUnconfirmed) {
    unconfirmed = rebaseSteps(unconfirmed, steps, tr)
  } else {
    for (let i = 0; i < steps.length; i++) tr.step(steps[i])
    unconfirmed = []
  }
  let newCollabState = new CollabState(version, unconfirmed)
  return tr.setMeta("rebased", nUnconfirmed)
           .setMeta("addToHistory", false).setMeta(collabKey, newCollabState)
}
```

ProseMirror: State

```
let collabState = collabKey.getState(state)
let version = collabState.version + steps.length
let ourID = collabKey.get(state).spec.config.clientID
```

ProseMirror: Steps

```
let ours = 0
while (ours < clientIDs.length && clientIDs[ours] == ourID)
    ++ours
let unconfirmed = collabState.unconfirmed.slice(ours)
steps = ours ? steps.slice(ours) : steps
if (!steps.length)
  return state.tr.setMeta(collabKey,
                          new CollabState(version, unconfirmed))
```

ProseMirror: Rebase

```
let nUnconfirmed = unconfirmed.length
let tr = state.tr
if (nUnconfirmed) {
  unconfirmed = rebaseSteps(unconfirmed, steps, tr)
} else {
  for (let i = 0; i < steps.length; i++) tr.step(steps[i])
  unconfirmed = []
}
let newCollabState = new CollabState(version, unconfirmed)
```

4.

Some Problems with Pseudo-OT

Just in case you thought this would be easy

Problem 1: Long Step List

Naive Approach:

1. Start with a known base state
2. Apply all changes from the beginning

Problem:

Eventually this takes a long time, especially in a browser

Solution:

“Reduce” step list to new known state

Problem 2: Steps vs States

Naive Approach:

1. Start with a known base state
2. Apply changes
3. Periodically roll up into new base state

Problem:

Steps must start from the new base state

Solution:

Tie the step number to the base state

Problem 3: Save Consistency

Naive Approach:

1. Start with a known base state
2. Apply changes
3. Roll up a new base state with step #

Problem:

New base states could contain “local” changes

Solution:

1. Server-side saves
2. Client-side transactions

5.
Interactive Demo

Using JavaScript

Thanks!

Any questions?

You can find me at:

Twitter: @oldmanhero

mgb@perfectminutegames.com

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash & Death to the Stock Photo (license)