Keynote: From groupware to large-scale trustworthy distributed collaborative systems
Claudia-Lavinia Ignat

To cite this version:

HAL Id: hal-01875534
https://hal.archives-ouvertes.fr/hal-01875534
Submitted on 17 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
From groupware to large-scale trustworthy distributed collaborative systems

Claudia-Lavinia Ignat, Inria, France
Douglas Engelbart: Augmenting Human Intellect

The Mother of all Demos, December 9, 1968

NLS: Online System
https://archive.org/details/dougengelbartarchives
Groupware, early 1990s

- «Computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.» [EGR91]

- Lotus Notes, one of the first commercial groupware allowing remote group collaboration
Groupware Time Space Matrix [J88]

**Face to face interactions**
- decision rooms, single display groupware, **shared table**, wall displays, roomware, ...

**Continuous task**
- large public display, team rooms, shift work groupware, project management, ...

**Remote interactions**
- video conferencing, instance messaging, **chats**/MUDs/virtual worlds, shared screens, **multi-user editors**, ...

**Communication + coordination**
- email, bulletin boards, blogs, asynchronous conferencing, group calendars, workflow, **version control**, wikis, ...

**Time/Space Groupware Matrix**

- **same time**
  - synchronous

- **different time**
  - asynchronous

- **same place**
  - co-located

- **different place**
  - remote
Groupware: supported solutions

- **Turn taking**: allow only one active participant at a time
  - e.g. RTCAL [SG88], SHARE [G90]

- **Locking**: concurrent editing allowed only if users lock and edit different objects
  - e.g. Colab [SFBKLS88]

- **Operational transformation**
  - e.g. GROVE [EG89]
Google Drive

March 2006
Writely
(Google Docs)

June 2006
XL2Web
(Google Sheets)

September 2007
Google Slides

2012
Google Drive
Isn’t it chaotic to all edit in the same document, even the same paragraph, at the same time?"

“Why would a group ever want to edit in the same line of text at the same time?” [EGR91]
Collaborative Systems: from users to community of users

2013: MOOC “Fundamentals of Online Education: Planning and Applications” with 40,000 participants
2016: Nuit debout, more than 70 people edit a pad
2018: online CSCW PC meeting with 120 members
Collaborative Systems: from users to community of users

Real-time Wikipedia
Limitations of Central Authority Systems

SCALABILITY

PRIVACY
Peer-to-Peer Collaborative Systems
Collaboration Modes – Concurrent Changes
Collaboration Modes – Offline Work
Collaboration Modes – Ad-hoc Collaboration
Research issues

1. How to maintain consistency of different copies in the face of concurrent modifications?

2. How to evaluate the design of collaborative systems and approaches?

3. How to secure collaboration data?
Research issues

1. How to maintain consistency of different copies in the face of concurrent modifications?

2. How to evaluate the design of collaborative systems and approaches?

3. How to secure collaboration data?
Optimistic Replication [SS05]

- Trade-off between consistency and availability
  - Optimistic replication: allows replicas to diverge

- Strong Eventual Consistency
  - Eventual delivery: An update executed at some correct replica eventually executes at all correct replicas
  - Strong convergence: Correct replicas that have executed the same updates have equivalent states
  - No consensus in background, no need to rollback

- Intention preservation
  - « Effect of each operation should be observed on all copies »
Operational transformation (OT) [EG89]

- $n$ copies of an object hosted at $n$ sites
- An object is modified by applying operations

- Each operation is
  - generated at a site (local execution), and applied immediately on the local copy
  - broadcasted to other sites
  - integrated at those sites (remote execution)

- System is correct if when it is idle all copies are identical (SEC)
Operational transformation (OT)

- General architecture with two main components:
  - An integration algorithm (diffusion, integration)
  - A set of transformation functions (conflict resolution)

- Running example for textual document = sequence of characters

Site 1

```
concurrency control
op₁ = ins(7, r)
```

Site 2

```
concurrency control
op₂ = ins(17, o)
```

```
T(ins(p₁, c₁), ins(p₂, c₂)) :-
  if (p₁ < p₂) return ins(p₁, c₁)
  else return ins(p₁ + 1, c₁)
endif
```

- Operations:
  - ins(p, c)
  - del(p)
Operational transformation
Correctness [EG89]

(TP1) \( \text{op}_1 \circ T(\text{op}_2, \text{op}_1) \equiv \text{op}_2 \circ T(\text{op}_1, \text{op}_2) \)

- Site 1
  - \( \text{op}_1 \)
  - \( \text{op}'_2 \)

- Site 2
  - \( \text{op}_2 \)
  - \( \text{op}'_1 \)

\( T(\text{op}_2: \text{operation}, \text{op}_1: \text{operation}) = \text{op}'_2 \)
- \( \text{op}_1 \) and \( \text{op}_2 \) concurrent, defined on a state \( S \)
- \( \text{op}'_2 \) same effects as \( \text{op}_2 \), defined on \( S.\text{op}_1 \)
Operational transformation
Correctness [RNG96]

(TP2) \( T(op_3, op_1 \circ T(op_2, op_1)) = T(op_3, op_2 \circ T(op_1, op_2)) \)
Operational transformation (OT)  
Existing approaches

• Two main families:
  • Transformation functions satisfying both TP1 and TP2: SOCT2 [SCF97] + TTF [OUMI06]
  • Control algorithms avoiding (needs of) TP2: SOCT4 [VCFS00], Jupiter [NCDL95]
Operational transformation (OT)

Summary

- Transforms non communting operations to make them commute
- Genericity
- Time complexity
  - Average: $O(H \cdot c)$ \quad H: #ops
  - Worst case: $O(H^2)$ \quad c: avg. #conc. ops
- Difficult to write correct transformation functions
- State vectors used for detecting concurrency $\Rightarrow$ scalability limitations
- Not very suitable for large scale peer-to-peer collaboration
Conflict-free Replicated Data Types (CRDT) [SPBZ11]

- Design operations to be commutative by construction
- Abstract data types
  - Designed to be replicated at multiple sites
  - Any replica can be modified without coordination
  - State convergence is guaranteed
- State-based and operation-based approaches
Conflict-free Replicated Data Types (CRDT)
State-based Replication

- **Algorithm**
  - Periodically, replica at $p_i$ sends its current state to $p_j$
  - Replica $p_j$ merges received state into its local state by executing $m$
  - After receiving all updates (irrespective of order), each replica will have same state
Conflict-free Replicated Data Types (CRDT) 
State-based Replication

• Merge operator:
  • **Commutative:** \( x \cdot y = y \cdot x \)
  • **Associative:** \( (x \cdot y) \cdot z = x \cdot (y \cdot z) \)
  • **Idempotent:** \( x \cdot x = x \)

• A semi-lattice is a Partial order \( \leq \) set \( S \) with a least upper bound (LUB), denoted \( \sqcup \)
  • \( m = x \sqcup y \) is a LUB of \{ \( x, y \) \} under \( \leq \) if and only if
    \( \forall m', x \leq m' \land y \leq m' \Rightarrow x \leq m \land y \leq m \land m \leq m' \)
  • It follows that \( \sqcup \) is commutative, associative and idempotent
Conflict-free Replicated Data Types (CRDT)
Convergent Replicated Data Type (CvRDT)

• Example
Conflict-free Replicated Data Types (CRDT)  
Operation-based Replication

- An update split into \((t,u)\): \(t\) is a side-effect-free prepare-update method and \(u\) is an effect-update method

- Algorithm
  - Updates delivered to all replicas
  - Causally-ordered broadcast, every message delivered to every node exactly once w.r.t. happen-before order

- Commutativity holds for concurrent updates
Conflict-free Replicated Data Types (CRDT)
Commutative Replicated Data Type (CmRDT)

• Example

\[
\begin{align*}
\{5\} & \quad \{5\} \cup \{3\} = \{3, 5\} & \{3, 5\} \cup \{7\} = \{3, 5, 7\} \\
\{5\} & \quad \{5\} \cup \{3\} = \{3, 5\} \\
\{5\} & \quad \{3, 5\} \cup \{7\} = \{3, 5, 7\} \\
\{5\} & \quad \{5\} \cup \{7\} = \{5, 7\} & \{5, 7\} \cup \{3\} = \{3, 5, 7\}
\end{align*}
\]
Conflict-free Replicated Data Types (CRDT) 
CvRDT vs. CmRDT

• Both approaches are equivalent
  • A state-based object can emulate an operation-based object, and vice-versa

• Operation-based:
  • More efficient since you only ship small updates
  • But require exactly once causally-ordered broadcast

• State-based:
  • Only require reliable broadcast
  • Communication overhead of shipping the whole state

• Delta State-based [ASB18]:
  • Small messages
  • Dissemination over unreliable communication channels
Consistency Maintenance
Conflict-free Replicated Data Types (CRDT)

- Register
  - Last-Writer Wins
  - Multi-Value
- Set
  - Grow-Only
  - 2-Phase
  - Observed-Remove
  - Observed-Update-Remove

- Map
- Counter
- Graph
  - Directed
  - Monotonic DAG
  - Edit graph
- **Sequence**

SwiftCloud
edge + cloud
Availability + consistency: DC switch
Causal + transactional
3000+ client replicas

Antidote
SyncFree
EU project
High performance, sharded, transactional, causal
Aims to scale to 100s of DCs
- Very modular
- Partial replication
- Small but safe metadata (vector clock)
In DC: strong consistency, physical clocks (Clock-SI)
Industrial apps: Virtual Wallet, SocialApp,
configuration management, FMK

(4) NMSI: strong, parallel
T1
T2
x
y
x?
Wait-Free
T2
x?
T3
y?
Forward
Wait-Free
Read from causal snapshot
Scalability properties:
- Wait-Free Queries
- Forward Freshness
- Mini. Commitment Synchronisation
- Genuine Partial Replication
Conflict-free Replicated Data Types (CRDT) (Text) Sequence [PMSL09] [WUM09]

• Document = linear sequence of elements
  • Each element has a unique identifier
  • Identifier constant for the lifetime of the document
  • Dense total order of identifiers consistent with element order:
    • $\forall id_x, id_y: id_x < id_y \Rightarrow \exists id_z: id_x < id_z < id_y$

• Different approaches for generating identifiers:
  • TreeDoc, Logoot, LogootSplit, …
Conflict-free Replicated Data Types (CRDT)

Logoot [WUM09]

- Logoot identifiers: \( <p_1, s_1, h_1>, <p_2, s_2, h_2>, \ldots, <p_k, s_k, h_k> \)
  - \( p_i \): integer
  - \( s_i \): site identifier
  - \( h_i \): logical clock at site \( s_i \)

<table>
<thead>
<tr>
<th>(&lt;1,2,1&gt;)</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;1,2,2&gt;)</td>
<td>o</td>
</tr>
<tr>
<td>(&lt;2,1,2&gt;)</td>
<td>n</td>
</tr>
<tr>
<td>(&lt;3,1,3&gt;)</td>
<td>c</td>
</tr>
<tr>
<td>(&lt;3,1,3&gt;&lt;8,4,5&gt;)</td>
<td>u</td>
</tr>
<tr>
<td>(&lt;3,2,5&gt;)</td>
<td>r</td>
</tr>
<tr>
<td>(&lt;4,1,7&gt;)</td>
<td>e</td>
</tr>
<tr>
<td>(&lt;4,1,7&gt;&lt;9,2,6&gt;)</td>
<td>n</td>
</tr>
<tr>
<td>(&lt;7,2,8&gt;)</td>
<td>c</td>
</tr>
<tr>
<td>(&lt;9,1,7&gt;)</td>
<td>y</td>
</tr>
<tr>
<td>(&lt;10,2,8&gt;)</td>
<td></td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;)</td>
<td>c</td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;&lt;6,5,1&gt;)</td>
<td>o</td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;&lt;7,8,2&gt;)</td>
<td>n</td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;&lt;7,8,2&gt;&lt;12,3,5&gt;)</td>
<td>t</td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;&lt;7,8,2&gt;&lt;13,3,6&gt;)</td>
<td>r</td>
</tr>
<tr>
<td>(&lt;12,3,1&gt;&lt;7,8,2&gt;&lt;14,3,7&gt;)</td>
<td>l</td>
</tr>
</tbody>
</table>

- Time complexity
  - Average: \( O(k \log(n)) \)
  - Worst case: \( O(H \log(H)) \)
  - \( H \): \#ops
  - \( n \): doc. size (non deleted chars.)
  - \( k \): avg. size of Logoot identifier

- No need for concurrency detection
- Identifiers storage cost
- New design for each data type
- Suitable for large-scale collaboration
Conflict-free Replicated Data Types (CRDT)

LogootSplit identifiers

<table>
<thead>
<tr>
<th>site_id</th>
<th>clock</th>
<th>begin</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁</td>
<td>…</td>
<td>pₙ</td>
<td></td>
</tr>
</tbody>
</table>

Base

Interval

1,1,[0,16] concurrency contr

1,1,[0,5] concur

1,15,2,1,[0,0] r

1,1,[6,16] ency contr

Insert r between “concur” and “ency contr”

Insert o between “ency contr” and “l”
OT vs. operation-based CRDT

- CRDT: more formalised approach
- OT: more generic and guided
  - Generic concurrency control algorithm
  - Operation transformations specific to application domain
- CRDT: different solutions for concurrency handling for different data types
- CRDT: Metadata overhead
Delays in MUTE [NEOIC17] https://coedit.re/
Delays in GoogleDocs [D116]
Research issues

1. How to maintain consistency of different copies in the face of concurrent modifications?

2. How to evaluate the design of collaborative systems and approaches?

3. How to secure collaboration data?
User Study: The effect of delay on users

- Delays in seeing modifications of other users
  - Network delay
  - Time complexity of consistency maintenance algorithms
  - Types of architecture

- How does delay influence group performance?

[Diagrams of Thin client architecture and Thick client architecture]
Experiment design

- 20 groups of 4 students
  - Perform several collaborative editing tasks
    - A proofreading task
    - A sorting task
    - A note taking task
  - Use the provided collaborative editor (Etherpad) + chat
  - Each group experienced a certain delay (0, 4, 6, 8, 10 s)
- Registration of user keyboard inputs
- Video recording of user activities on desktop
Note-taking [IOFSC15]

The participants ranged in age from 21 – 27. All participants used French in their daily activities. An electronic announcement solicited participation. One of the researchers organized interested participants into sets of 4 and scheduled the session. All participants received a 10 Euro gift certificate for their participation.

2.2 Apparatus

The experiment was conducted using four GNU/Linux desktop computers in a classroom setting. Participants were separated by partitions and could not directly observe other team members while they worked, although typing activity was audible. The server running the Etherpad application was hosted on an Amazon Elastic Compute Cloud (EC2) instance located in the US East (Northern Virginia) Region. Each desktop ran the Mozilla Firefox web browser executing the Etherpad web client application. Etherpad hosted the task stimuli and a Chat dialogue facility (see Figure 2). User operations appeared color-coded in both the text and chat. Etherpad relies on a client-server architecture where each client/user edits a copy of the shared document. When a user performed a modification it was immediately displayed on the local copy of the document and then sent to the server. The server merged the change received from the user with other user changes and then transmitted the updates to the other users. When a user edited a sequence of characters, the first change on the character was immediately sent to the server, while the other changes were sent at once only upon reception of an acknowledgement from the server. With each change sent to the server, it created a new version of the document. Gstreamer software enabled the video recording of user activity. We also instrumented Etherpad to register all user keyboard inputs on the client side and to introduce delays on the server-side. The editor window displayed 50 lines of text. Users editing above the field of view of a collaborator could cause the lines within the collaborator’s view to “jump” inexplicably. Such a property is consistent with
Delay reduces Group Performance

- Delay increases error rate and redundancy
Delay reduces Group Performance

- Delay decreases proportion of keywords
Design implications

• Reduce the delay by the choice of the architecture and synchronisation algorithms

• Make users aware of existing delays such that they can compensate for the delay by coordination strategies

• Analyse real collaboration traces to understand collaboration patterns and behavior [NI18]
Research issues

1. How to maintain consistency of different copies in the face of concurrent modifications?

2. How to evaluate the design of collaborative systems and approaches?

3. How to secure collaboration data?
Security in peer-to-peer collaboration

- How to learn and verify the other party's key?
  - Trust-based access control
Trust establishment

• How to learn and verify the other party’s key before establish a secure communication channel?
  • Out of band trust establishment
  • Trust establishment by the provider
Out of band trust establishment

- Unintuitive, error-prone

Alice, is FGY345 your public key?
Bob, what is a public key?

My key fingerprint is
Trust establishment by the provider
Centralized key server

- Clients query providers for keys of other users
- Users have to trust provider, e.g., WhatsApp

Register Alice with $PK_A$

1. Secure Message Provider
2. Alice’s key $PK'_A$

Alice
Client A

Bob
Client B
Transparent log

Alice
PK_A

Bob
PK_B

Key Server

Register

Query

Users
Certificate transparency [L14]/CONIKS [MBBFF15]

- Gossiping
  - No client incentive
  - Subject to Sybil and Eclipse attacks
Trusternity: Blockchain-based Auditing of Transparent Log Servers [NEIP18]
Trust-based access control

- Dynamic trust values among users

- How to define an access control based on trust and how to compute trust based on collaborative experience?
Trust computation

- Respect/Violation of contracts
  - Contracts in collaborative editing (share, edit)

- Reporting of fake news in Facebook
- Quality of user contributions
Validation of trust-based collaboration

- Using game theory (trust game) [BDM95]

User 1 sends 8€ to User 2
User 2 receives 3 × 8€ = 24€
New balance: 2€
User 2 sends back 17€ to User 1
Final balance: 19€
Validation of trust-based collaboration

- Proposal of a trust metric reflecting user behavior [DI16]
- User studies on various trust game variations
  - Trust can replace knowing the identity of collaborators
  - People take into account the trust value of the partner in their future collaboration
Large-scale trustworthy distributed collaborative systems

- New uses and new practices due to large scale adoption
- New challenges
  - Consistency of replicated data
  - User studies
  - Trust and Security
References

References

- **[EG89]** Concurrency Control in GroupWare Systems, C.A. Ellis and S.J. Gibbs. ACM SIGMOD Record 18(2), 1989.
References

- **[PMSL09]** A commutative replicated data type for cooperative editing, N. Preguica, M. Joan, M. Shapiro, and M. Letia. ICDCS 2009.
References

References

- [NEIP18] Blockchain-Based Auditing of Transparent Log Servers. Hoang-Long Nguyen, Jean-Philippe Eisenbarth, Claudia-Lavinia Ignat, Olivier Perrin. DBSec 2018
References

Thank you

COAST Team

http://team.inria.fr/coast/