Evaluating CRDTs for Real time Document Editing

Team SCORE

ACM *DocEng2011* : Evaluating CRDTs for Real-time Document Editing

Contents

- Motivation
- Structure of *logs*
- Framework
- Methodology
- Algorithms evaluated
- Experimental evaluation
- Conclusion and future work

Motivation

- Evaluated and compare different algorithms in terms of performance on a collaborative text editor requires real traces.

- Experience with real traces reflects the reality

Modify TeamEdit

- To retrieve the traces, we used TeamEdit
- TeamEdit is collaborative text editor used in our experiment
- Uses a central server only for allow communication between replica
- Distingue different replica by unique number
- Modify TeamEdit to :
  - Save all operations in *logs* (with log4j).
  - Add the functionality of Undo/Redo

Logs

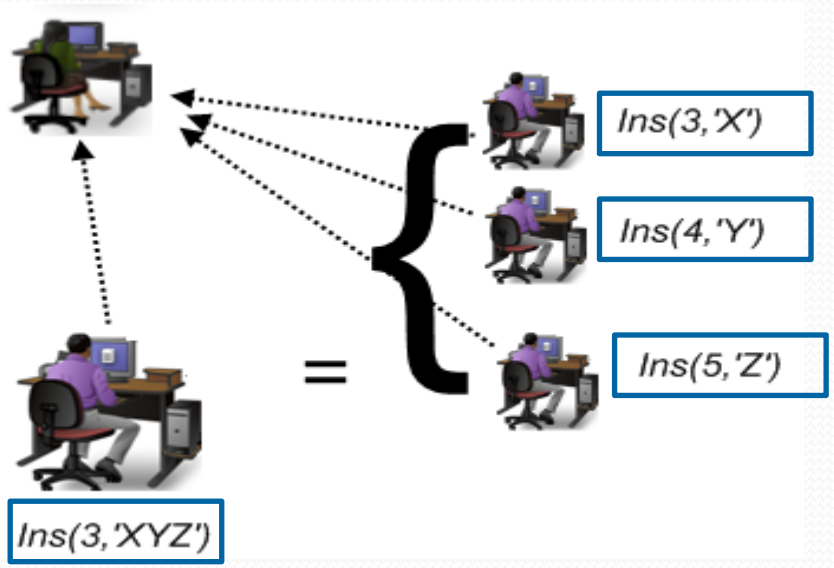
- Structure

```
2011-02-14 09:32:49,078 INFO  recu - Ins("Notre projet",464,[0-91,1-44,2-219,3-66,4-93],1297672362718,1,4)
2011-02-14 09:32:49,125 INFO  recu - Del(363,30,[0-91,1-44,2-220,3-66,4-93],1297672362784,1,2)
2011-02-14 09:32:49,078 INFO  recu - Ins("l",300,[0-91,1-44,2-220,3-66,4-94],1297672362718,1,4)
2011-02-14 09:32:49,125 INFO  recu - Del(300,1,[0-91,1-44,2-220,3-66,4-95],1297672362784,1,4)
```

- Save it in format XML

- Each algorithm transforms operations logs into one or more specific operations

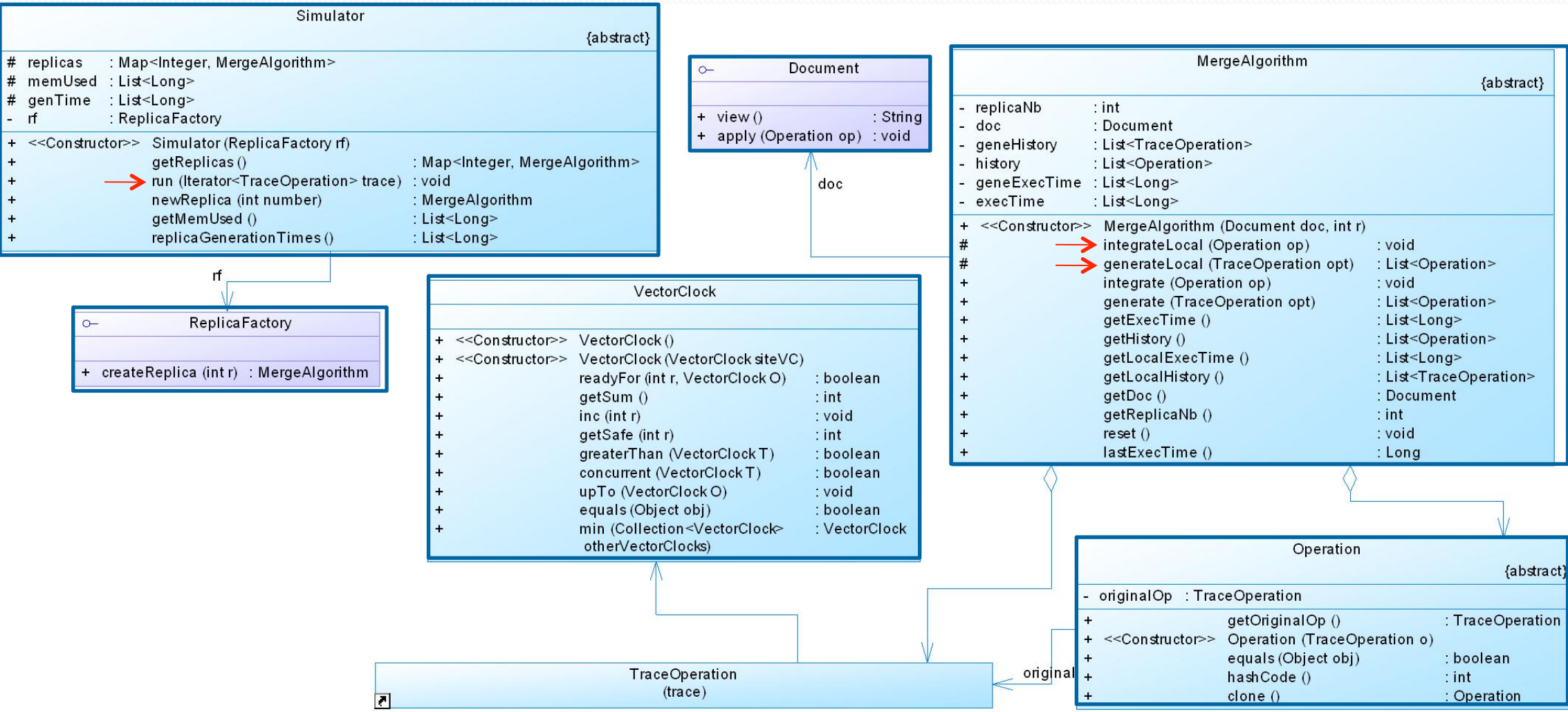
- XYZ → TraceOperation or User Operation
- X
- Y → Character Operation or Operation
- Z



SOCT2/TTF

Framework

package Core



Framework

package Core

- MergeAlgorithm class
  - Method generate

➢ Makes calls to the function *generateLocal* implemented by each algorithms

```
1   function List<Operation> generate(TraceOperaion)
2   Begin
3       let genHistory.add(TraceOperaion);
4           StartTime := System.nanoTime();
5           List<Operation> l = generateLocal(TraceOperaion);
6           Time = System.nanoTime() - startTime;
7           oh = Time/size(l);
8           i = size(history);
9           geneExecTime.add(t);
10      for (Operation op in l)
11          execTime.add(oh);
12          History.add(op);
13          i++;
14      done;
15
16      return l;
17  end;
```

➢ Calculate time of generation

Framework

package Core

- MergeAlgorithm class
  - Method integrate

➢ Integrate of remote Operations

➢ Makes calls to the function *integrateLocal* implemented by each algorithms

➢ Calculate time of integration

```
1  function integrate(Operation)
2  Begin
3      let StartTime = System.nanoTime();
4          History.add(op);
5          integrateLocal(op);
6          execTime.add(System.nanoTime() - startTime);
7  end
```

Framework

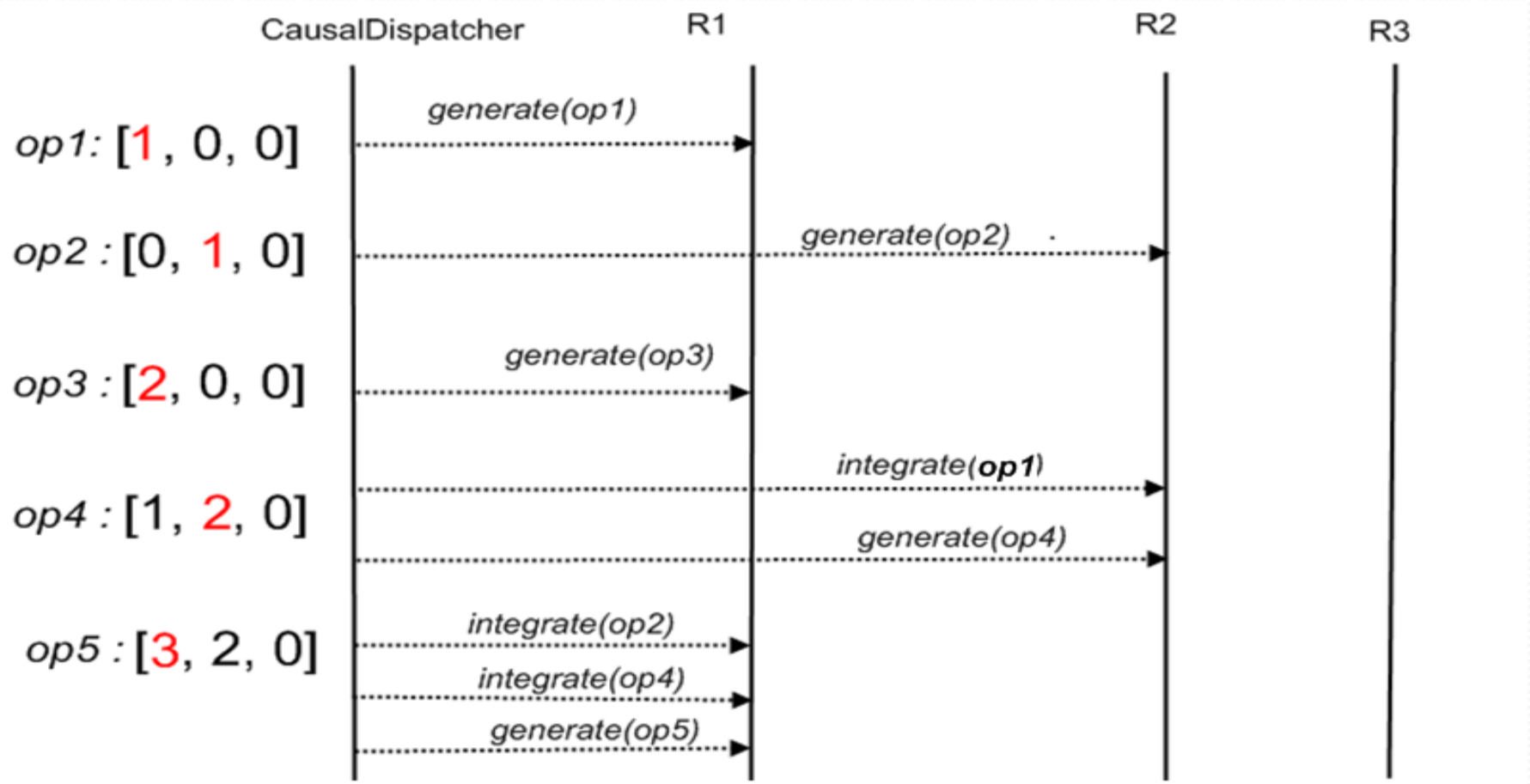
Package sim

- CausalDispatcher class
  - Derived from the class *Simulator*
  - Implement method *run*
  - Allows each replica to receive generated character operations in the same order as that in *logs*
  - ensures diffusion of operations in the causal order

```
1   function insertCausalOrder(List<TraceOperation> concurrentOps, TraceOperation opt)
2   Begin
3       let ListIterator<TraceOperation> it := concurrentOps.listIterator();
4           boolean cont := true;
5       while (it.hasNext() && cont)
6           TraceOperation t := it.next();
7           if (t.getVC().greaterThan(opt.getVC()))
8               cont := false;
9               it.previous();
10          endif
11      endwhile
12      it.add(opt);
13  end
```

Framework

Package sim



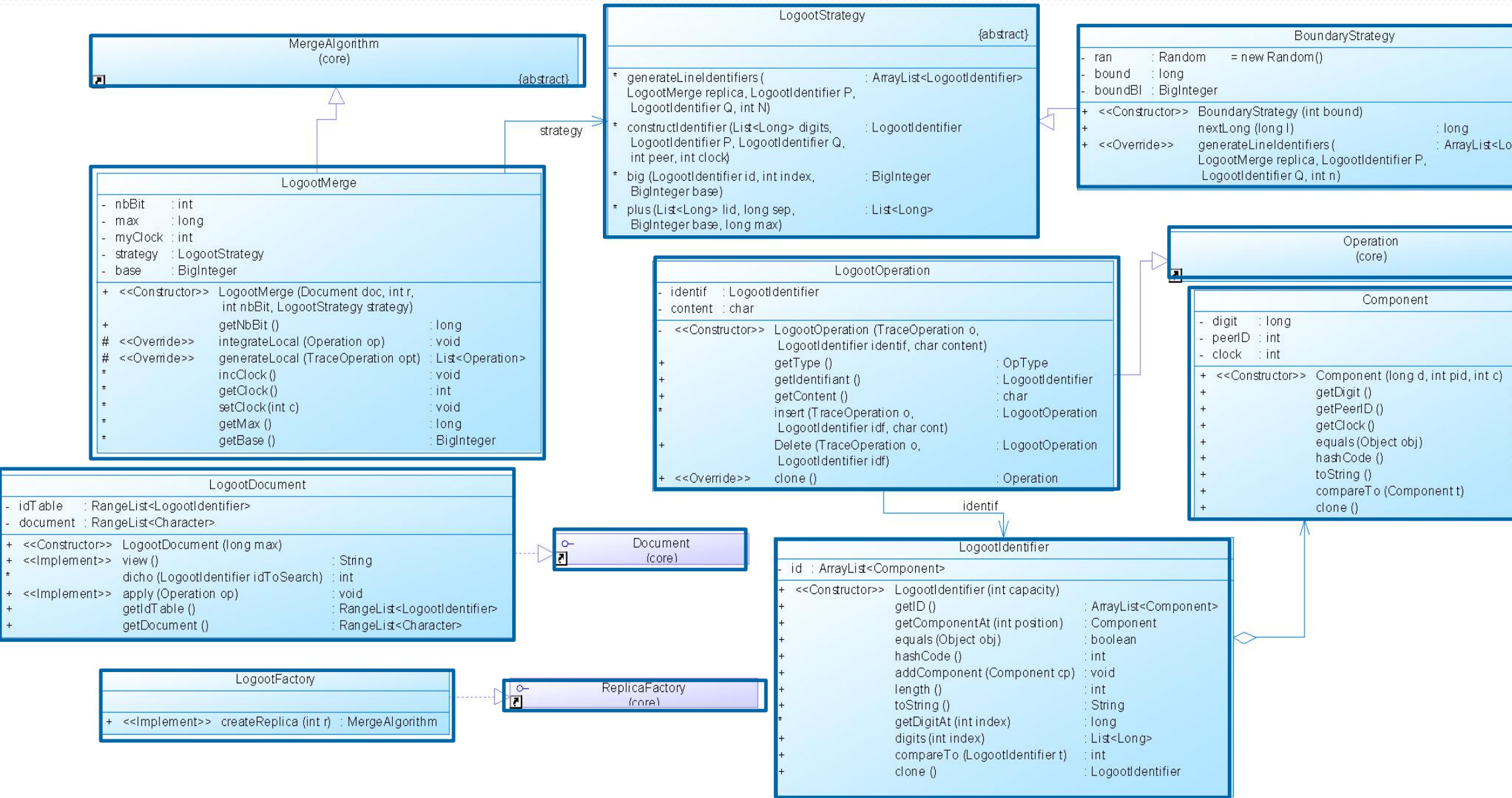
Framework

Add new algorithms

- To integrate new algorithm in our framework :
- Implement different class of package core
  - MergeAlgorithm
  - Operation
  - Document
  - ReplicaFactory

Framework

Package Logoot



Framework

Package Logoot

- LogootOperation
- Derived from the class *Operation(core)*
- Contains methods for creating operations insert and delete

```
1  function LogootOperation insert(TraceOperaion,Identifiant,content)
2  Begin
3      return LogootOpration(TraceOperaion, Identifiant, content);
4  end;
```

```
1  function LogootOperation Delete(TraceOperaion,Identifiant)
2  Begin
3      return LogootOpration(TraceOperaion, Identifiant, null);
4  end;
```

Framework

Package Logoot

- LogootDocument class
  - Implements interface *Document(core)*
    - Implements method view to display content of document
    - Implements method apply to integrate operation in document
    - Contains the binary search method: called when inserting or deleting Logoot identifiers.

```
function apply(Operation op)
Begin
    let logOp := (LogootOperation) op;
    idToSearch = getIdentifiant(logOp);
    pos = ReseachDicho(idToSearch);

    switch(getType(logOp))
    case(Insert)
        idTable.add(pos, idToSearch);
        document.add(pos, getContent(logOp));
    case(Delete)
        if(idTable(pos) = idToSearch)
            idTable.remove(pos);
            document.remove(pos);
    end;
end;
```

Framework

Package Logoot

- LogootMerge class
  - Derived from the class *MergeAlgorithm(core)*
  - Implement the two abstract method of class *MergeAlgorithm*

1- IntegrateLocal

```
1  function integrateLocal(Operation op)
2  Begin
3          getDoc().apply(op);
4  end;
```

Framework

Package Logoot

2- generateLocal

```
function List<Operation> generateLocal(TraceOperaion)
Begin
    let List<Operation> listeOp := {};
        position := Position(TraceOperaion);
        ListeChar = {};
        lg := LogootDocument();
    switch(type(TraceOperaion))
    case(Insert)
        let N := getContent(opt).length();
            content := getContent(opt);
            patch := generateLineIdentifiers(LogootMerge, lg(position), lg(position+1), N);
        for (cmpt := 0; cmpt < patch.size(); cmpt++)
            c := content.charAt(cmpt);
            LogootOp log := insert(opt, patch(cmpt), c);
            listeOp.add(log);
            ListeChar.add(c);
        done
        lg.IdTable().addAll(position + 1, patch);
        lg.Document().addAll(position + 1, ListeChar);
    case(Delete)
        let offset := getOffset(opt);
        for (k = 1; k <= offset; k++)
            LogootOperation wop := Delete(opt, lg(position + k));
            listeOp.add(wop);
        done
        lg.getIdTable().remove(position + 1, offset);
        lg.getDocument().remove(position + 1, offset);
    end
    return listeOp;
end;
```

Methodology

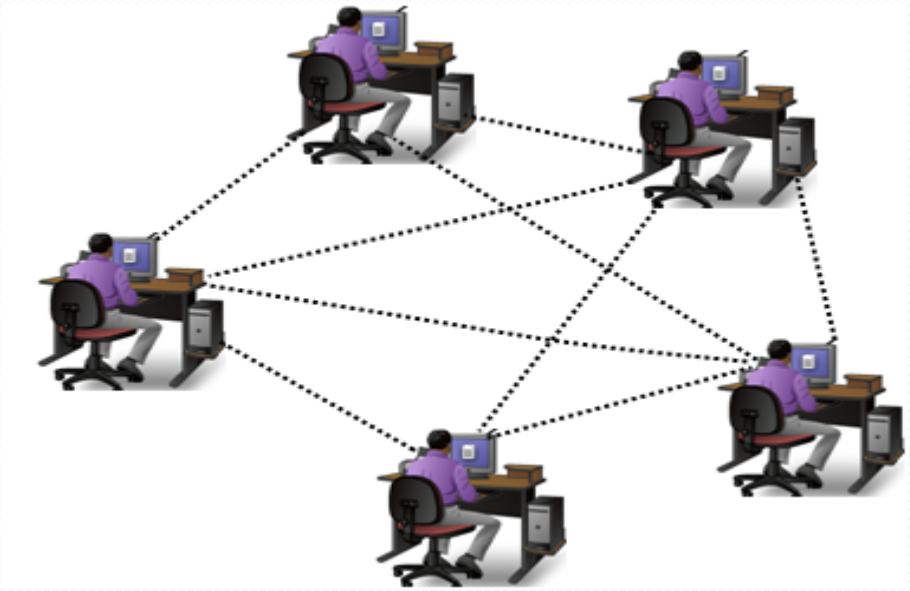
Obtaining Logs From real-time P2P collaboration

- We performed two collaborations with groups of students :

- 1) Report : collaboration with 13 students

  - Group1 : 4 students
  - Group2: 4 students
  - Group3: 5 students

- Write report for the module "Software Engineering Project".



- Student was not allowed to use any other communication tools except TeamEdit

- Duration : 1h30 hours

Methodology

Obtaining Logs From real-time P2P collaboration

- 2) Series

- With 18 students divided into nine groups of two students

- Each group was asked
  - To make a transcription of a certain hero
  - To describe the environment and actions that happened during the episode

- Duration : 1h30

Methodology

Description of collaboration Logs

Run each algorithm on each log :

- ten time
- In different JVM
- In same server

	Report			Series	
	GROUP1	GROUP2	GROUP3	DOC1	DOC2
No. User operation	11 211	11 066	13 702	9 042	9 828
No. character operation	26 956	47 992	42 443	29 882	10 268
%Del	12	12	12	9	5

Tableau I : Total number of User/Character operations

Algorithms Evaluated

- OT
  - SOCT2/TTF
- CRDT
  - LOGOOT
  - RGA
  - WOOT
  - WOOTO
  - WOOTH

Experimental evaluation

Execution times *in ns* of « **User operations**»

		REPORT			SERIES	
		GROUP1	GROUP2	GROUP3	DOC1	DOC2
Logoot	MAX	750 888	900 888	2 321 888	2 267 333	77 000
	AVG	5 618	7 272	6 523	5 046	5 358
Woot	MAX	6 377 472 222	752 285 610 000	355 445 076 111	818 192 000	2 233 888
	AVG	1 231 471	98 645 594	26 642 879	559 864	26 413
WootO	MAX	13 489 333	208 984 888	340 068 333	494 030 333	161 750
	AVG	43 237	112 380	96 057	110 092	23 203
WootH	MAX	3 622 666	40 042 222	156 406 777	164 934 111	453 222
	AVG	26 269	42 775	48 960	45 563	16 388
RGA	MAX	998 111	2 082 333	1 970 888	2 670 750	550 222
	AVG	26 518	31 876	32 469	19 717	16 646
SOCT2	MAX	5 753 333	24 741 375	15 311 777	8 911 666	146 777
	AVG	20 826	40 193	29 682	27 299	19 037

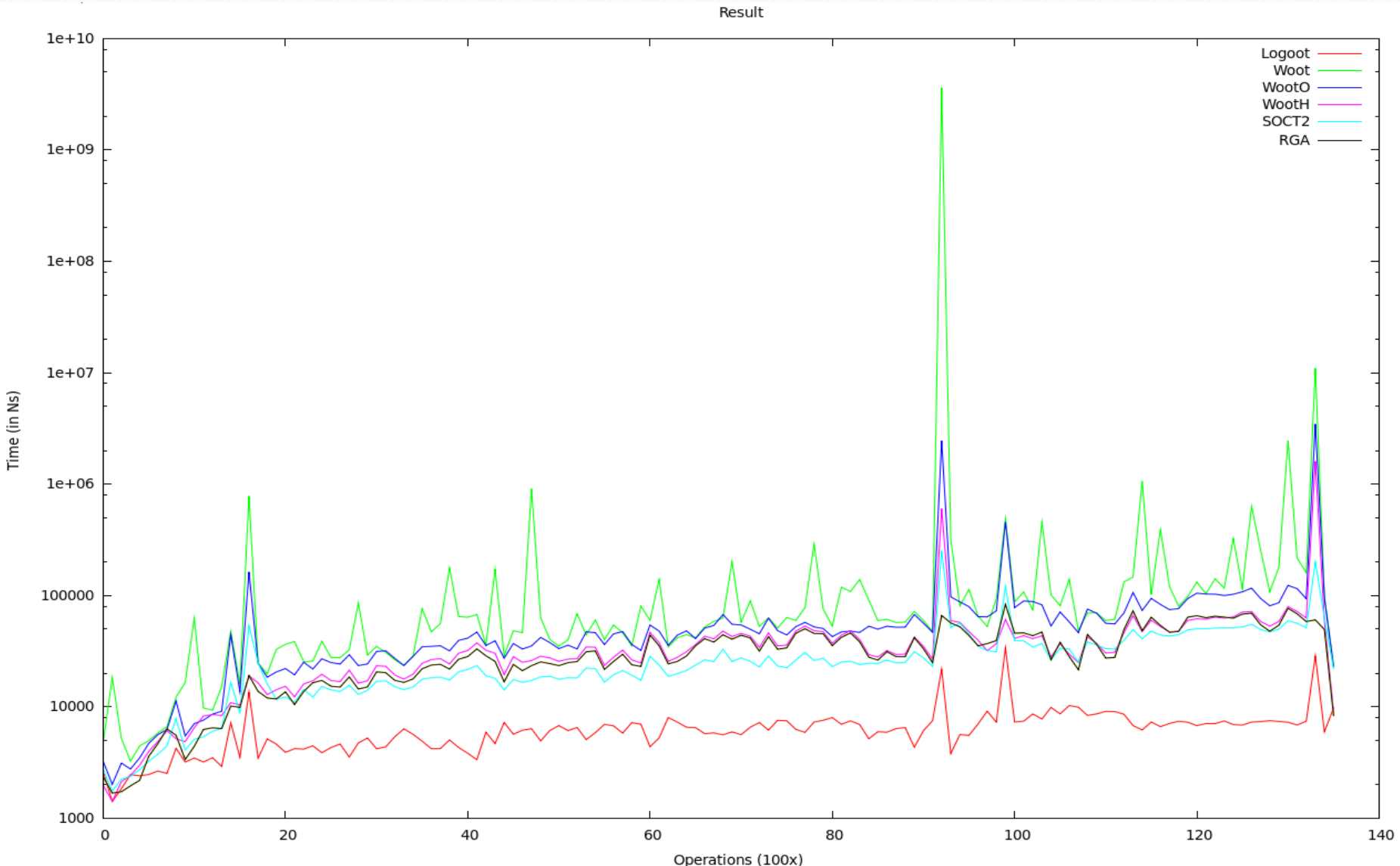
Experimental evaluation

Execution times *in ns* of « **Character operations**»

		REPORT			SERIES	
		GROUP1	GROUP2	GROUP3	DOC1	DOC2
Logoot	MAX	91 493	126 702	110 345	80 052	35 991
	AVG	4 762	5 769	5 383	3 396	4 508
Woot	MAX	5 628 912 689	688 562 942 877	4 804 760 142	770 106 931	1 669 358
	AVG	544 254	22 602 003	8 773 027	224 276	28 742
WootO	MAX	2 026 985	72 937 474	8 903 271	1 334 399	132 683
	AVG	54 191	97 312	98 587	84 382	25 259
WootH	MAX	694 146	44 189 710	4 329 757	566 516	200 255
	AVG	2 503	3 971	7 476	8 356	2 415
RGA	MAX	750 069	1 294 824	1 001 556	402 933	252 489
	AVG	2 283	1 591	2 723	1 425	2 413
SOCT2	MAX	5 286 073	13 278 450	5 832 092	20 727 215	2 848 290
	AVG	80 509	573 482	129 634	1 383 306	304 537

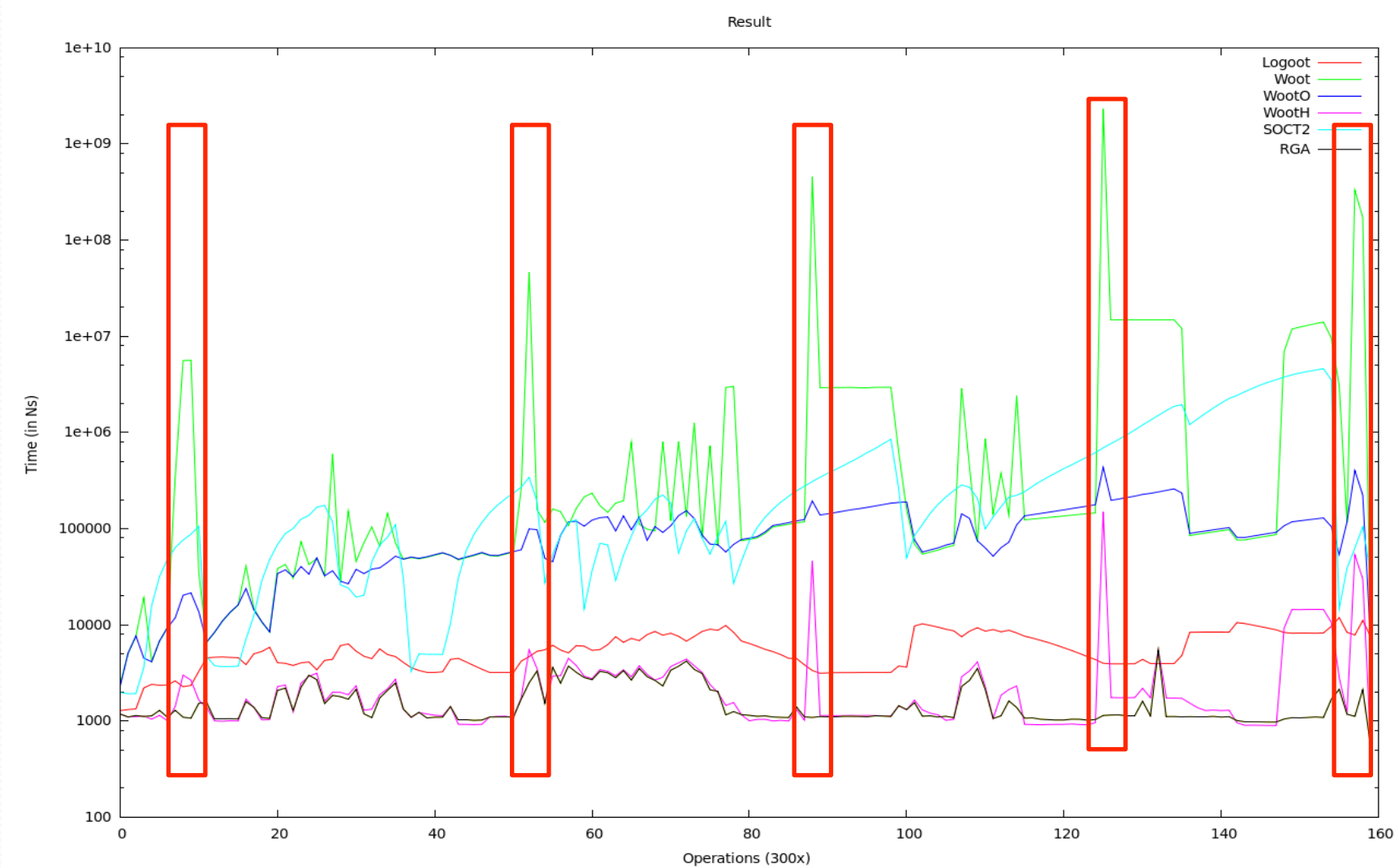
Experimental evaluation

Behaviours (User operations in Group 3)



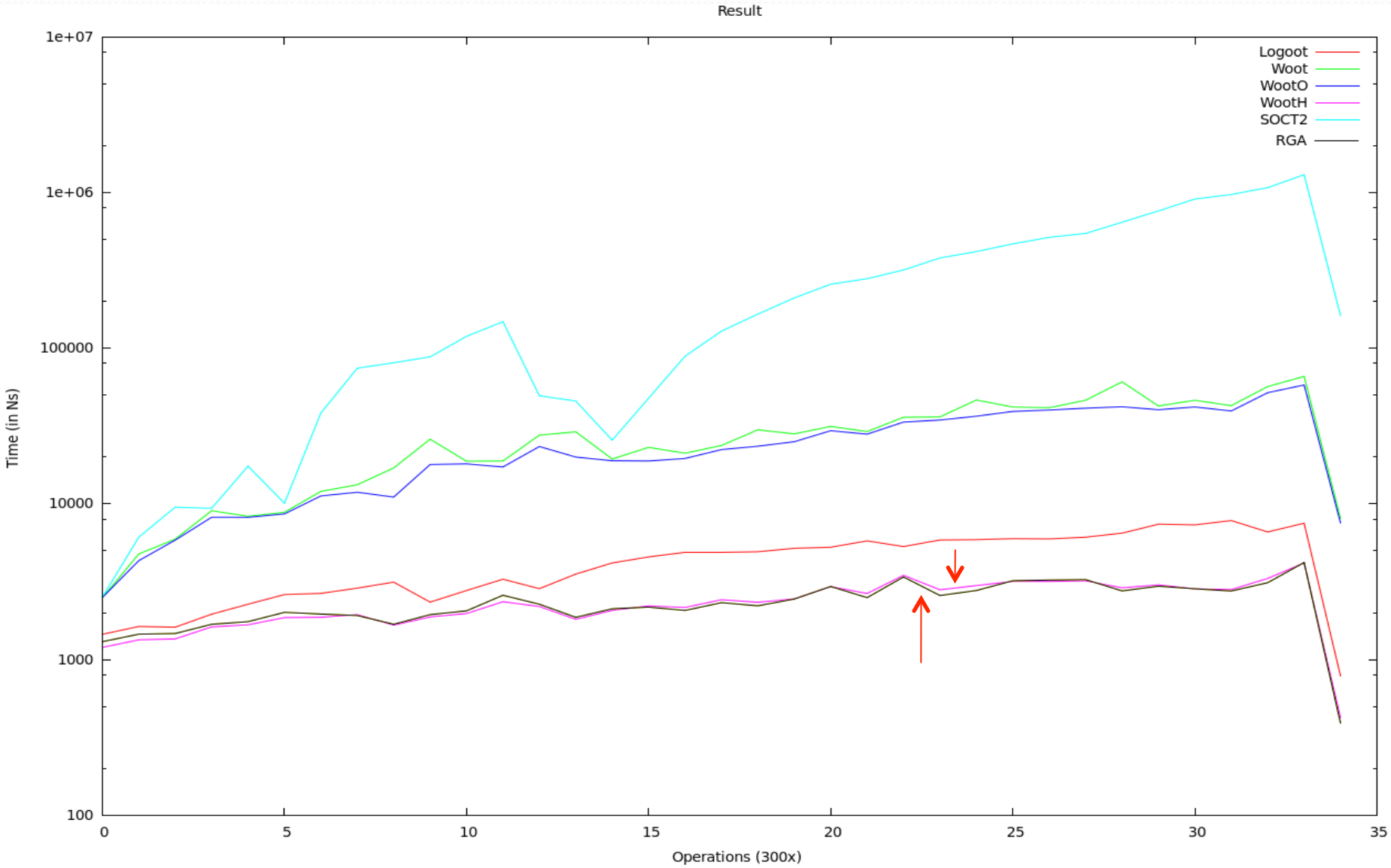
Experimental evaluation

Behaviours (character operations in group2)



Experimental evaluation

Behaviours (character operation in Series 2)



Conclusion and future work

- The first work that evaluates algorithms by using collaboration traces including concurrency.
- CRDT algorithms initially designed for peer-to-peer collaboration are suitable for real-time collaboration.
- CRDT algorithm outperform some representative operational transformation approaches
- Integrate other algorithms in our framework
- More criteria of comparison : communication complexity, memory used …
- Generate large size traces of real-time collaboration
- Extend our framework and evaluation for other types of collaboration

Thank you

for your attention