EpTO: Implementation of a Large-Scale Epidemic Total Order Algorithm

Master Thesis

Jocelyn Thode[1] Hugues Mercier[2] Miguel Matos[3]

Neuchâtel, Switzerland, February 8, 2017

[1] Department of Informatics, University of Fribourg, Switzerland
jocelyn.thode@unifr.ch

[2] Institute of Computer Science, University of Neuchâtel, Switzerland
hugues.mercier@unine.ch

[3] HASLab - INESC TEC, University of Minho, Portugal
miguelmatos@di.uminho.pt

Outline

- Motivation
- EpTO explained
- LSDSuite
- Evaluation
- Conclusion

Introduction

Motivation

EPTO was only evaluated using a **simulation**. We need an evaluation with real peers to:

- Expose possible **limitations**
- Confirm simulation **results**

Motivation

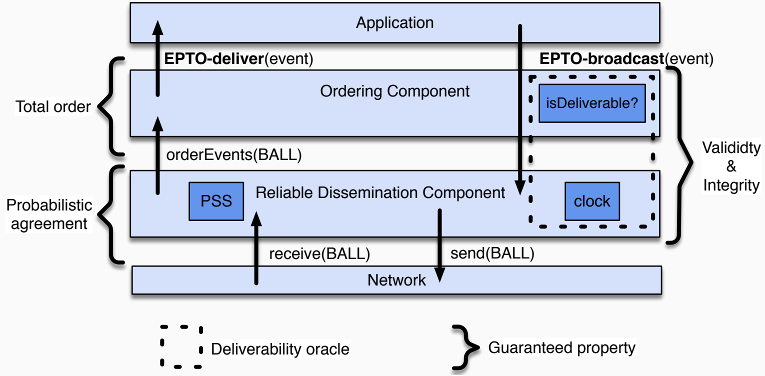
- Comparing EPTO meant testing it against other algorithms
- No framework to easily benchmark algorithms without having to rewrite them

What is EPTO?

Epidemic Total Order Algorithm:

- Probabilistic dissemination algorithm
    - Using balls-and-bins
- Provides deterministic total order, integrity and validity
- Scales well with the number of peers
    - Parameters increase logarithmically
- Churn resistant

EPTO architecture

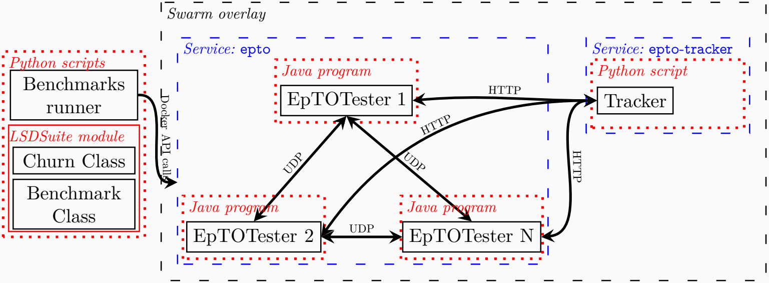


M. Matos, H. Mercier, P. Felber, R. Oliveira and J. Pereira, “EpTO: An epidemic total order algorithm for Large-scale distributed systems”, in Proceedings of the 16th Annual Middleware Conference, ACM, 2015, pp. 100–111.

LSDSuite features

- Compatible with any distributed algorithm provided it runs on Docker
- support for a user-provided tracker
- Automated benchmarking execution
- Containers allow for more than 1 peer per physical node
- Can simulate churn or follow real traces

LSDSuite architecture



LSDSuite Configuration and Logging

The protocol, churn and framework configuration is done through YAML files

The protocols logs must be written in a file to be extracted to the host

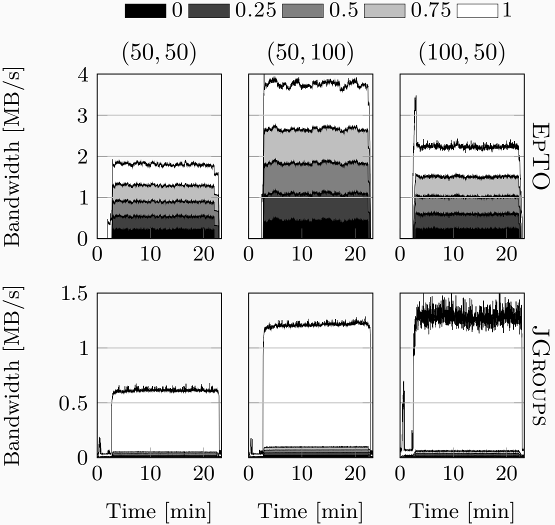
Evaluation

Evaluation

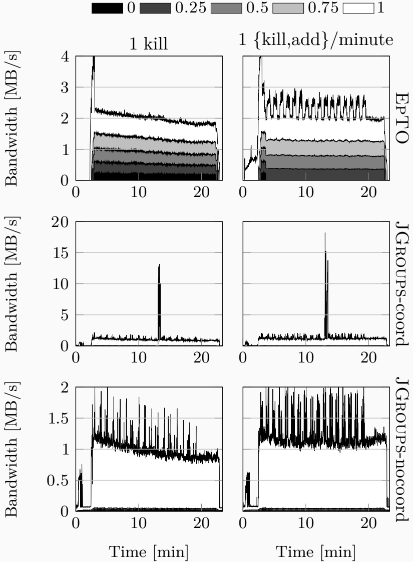
We evaluate EpTO against JGroups SEQUENCER, scaling peers and global event throughput per second.

We write (n, e) where n is the number of peers and e is the global event throughput per second.

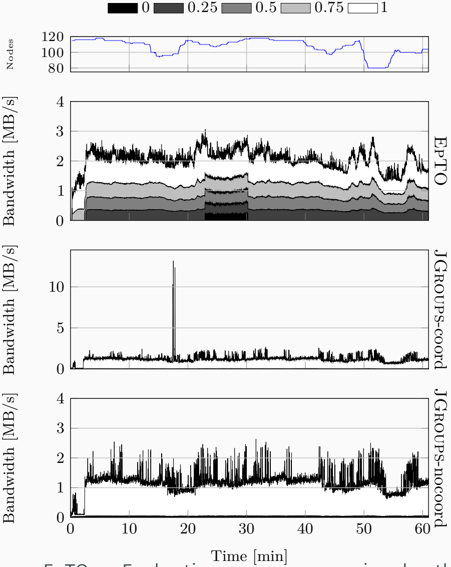
Bandwidth


0
0.25
0.5
0.75
1
(50, 50)
(50, 100)
(100, 50)
Bandwidth [MB/s]
0
1
2
3
4
EPTO
Bandwidth [MB/s]
0
0.5
1
1.5
JGROUPS
0
10
20
Time [min]
Time [min]
Time [min]

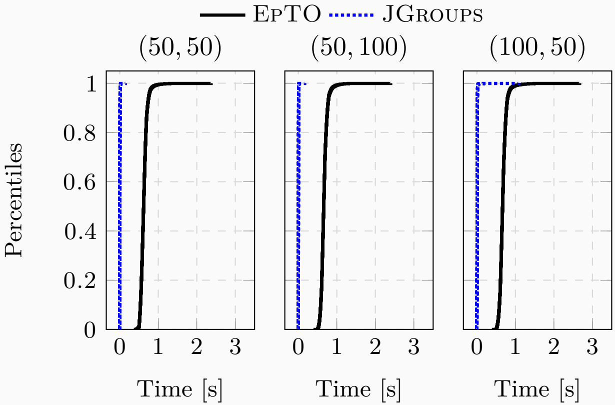
Bandwidth Synthetic Churn $(100, 50)$



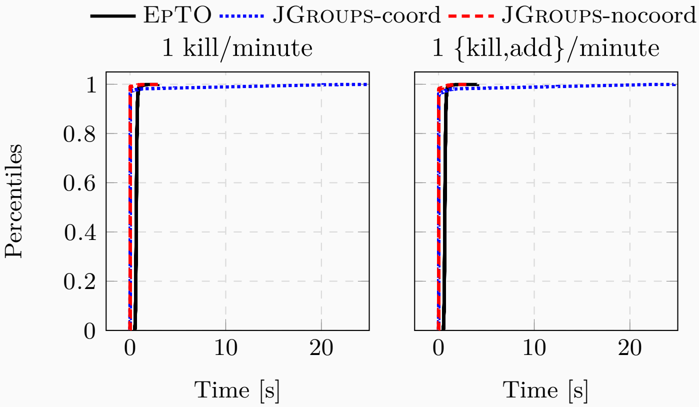
Bandwidth Real Churn



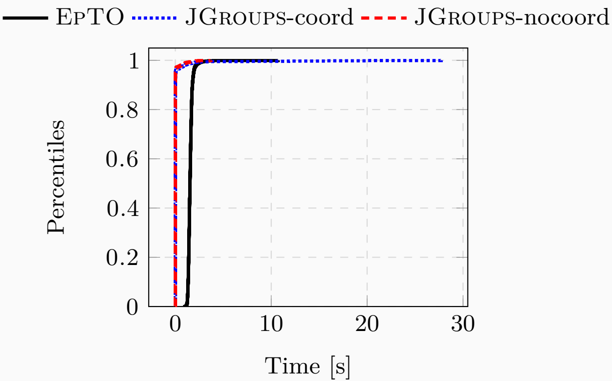
Local Dissemination Stretch



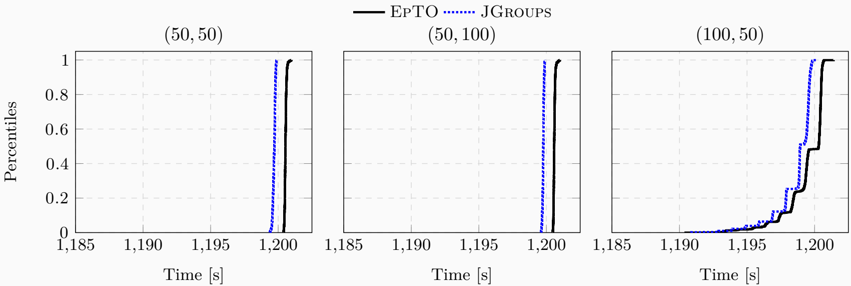
Local Dissemination Stretch Synthetic Churn $(100, 50)$



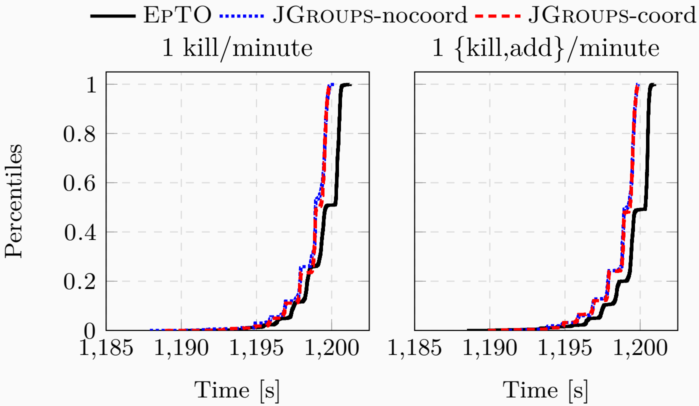
Local Dissemination Stretch Real Churn



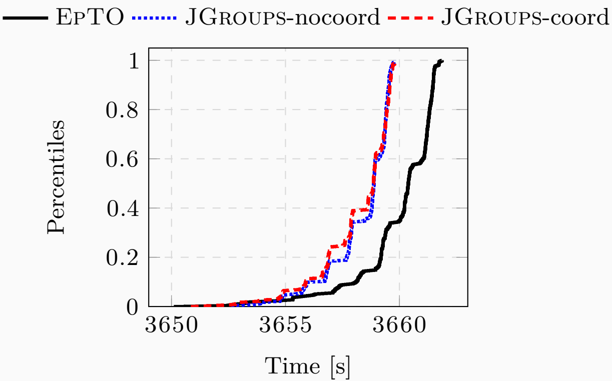
Local Times


EpTO
JGroups
(50, 50)
(50, 100)
(100, 50)
Percentiles
1
0.8
0.6
0.4
0.2
0
1,185
1,190
1,195
1,200
Time [s]

Local Times Synthetic Churn $(100, 50)$



Local Times Real Churn



Total GB sent/received

Protocol		Cluster parameters		
		(50, 50)	(50, 100)	(100, 50)
EpTO	Receive	10.84 ± 0.16	22.31 ± 0.39	26.01 ± 0.27
	Sending	10.84 ± 0.16	22.31 ± 0.39	26.01 ± 0.27
JGroups	Receive	0.78 ± 0.03	1.45 ± 0.01	1.88 ± 0.01
	Sending	0.77 ± 0.03	1.44 ± 0.01	1.84 ± 0.01

Total events sent in a stable environment

	Cluster parameters		
Protocol	$(50, 50)$	$(50, 100)$	$(100, 50)$
EpTO	$59\,993.8 \pm 3.3$	$119\,898.2 \pm 9.7$	$59\,913.0 \pm 164.3$
JGroups	$59\,961.9 \pm 10.9$	$119\,885.7 \pm 5.0$	$60\,023.1 \pm 287.1$

Total events sent with a synthetic churn $(100, 50)$

	Cluster parameters	
Protocol	1 kill/minute	1{kill,add}/minute
EPTO	53 898.5 ± 133.9	59 798.6 ± 140.1
JGROUPS-coord	53 834.7 ± 175.5	59 507.9 ± 240.9
JGROUPS-nocoord	53 830.5 ± 200.3	59 450.5 ± 175.1

Total events sent during a real trace

Protocol	Events sent
EpTO	165 844.2 ± 210.2
JGroups-coord	166 183.0 ± 1368.1
JGroups-nocoord	166 585.8 ± 824.9

Conclusion

Limitations

- Difference not strong enough at 100 peers scale
- High CPU usage
- Docker problems on AWS/GCE

Future Work

- Obtain more resources to have stronger results
- Implement a Push-Pull EPTO version
- Use Kubernetes instead of Docker + Docker swarm
- Refine Framework Architecture

Questions?

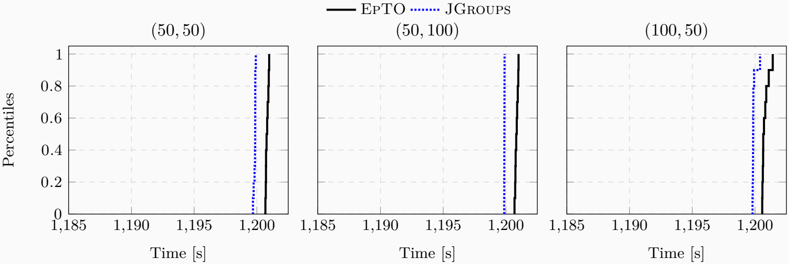
Total GB sent/received Synthetic Churn (100, 50)

Protocol		Churn parameters	
		1 kill/minute	1{kill,add}/minute
EPTO	Receive	21.00 ± 0.24	26.32 ± 0.32
	Sending	21.21 ± 0.25	26.57 ± 0.32
JGROUPS-coord	Receive	1.47 ± 0.02	1.75 ± 0.02
	Sending	1.43 ± 0.02	1.70 ± 0.02
JGROUPS-nocoord	Receive	1.45 ± 0.01	1.73 ± 0.02
	Sending	1.41 ± 0.01	1.68 ± 0.02

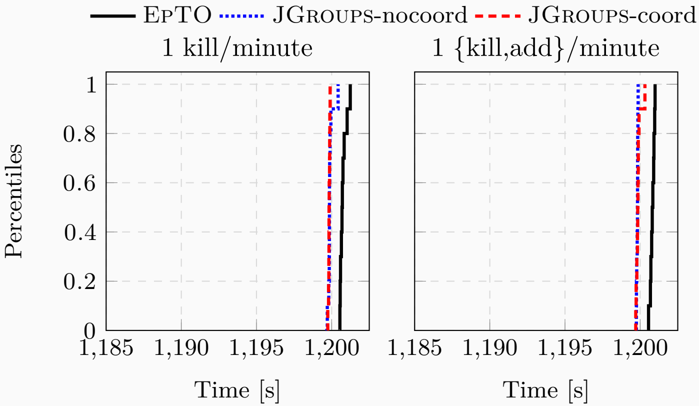
Total GB sent/received Real Churn

		Churn parameters
Protocol		Real Trace
EpTO	Receive	81.41 ± 1.08
	Sending	82.67 ± 1.08
JGroups-coord	Receive	5.56 ± 0.08
	Sending	5.40 ± 0.08
JGroups-nocoord	Receive	5.58 ± 0.05
	Sending	5.43 ± 0.05

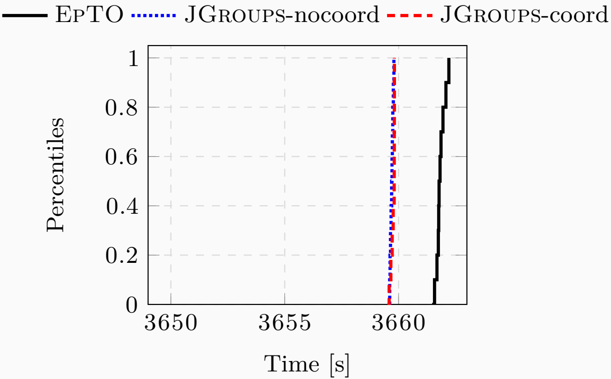
Global Times



Global Times Synthetic Churn



Global Times Real Churn


EpTO
JGroups-nocoord
JGroups-coord
Percentiles
Time [s]