Exploiting Commutativity For Practical Fast Replication

> Seo Jin Park and John Ousterhout, *Stanford University*
>
> 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19)
>
> https://www.usenix.org/conference/nsdi19/presentation/park

1. What is the main **research contribution**? In other words, what have the authors tried to prove or demonstrate in this paper?

```
For non-computative replication request, authors reduce the waiting time from
2 RTTs to 1 RTT by using  Consistent Unordered Replication Protocol(CURP).
```

2. What are the three most closely-related **prior research works**, and what do the authors do differently than these prior works?

```
Generalized Paxos and Fast Paxos allow clients to send requests directly to
replicas and reduce latency from 2 RTTs to 1.5 RTT, while CURP use witness to
reduce RTT for computative requests.

Speculative Paxos and Network-Ordered Paxos reduce latency almost to 1 RTT by
seri- alizing client requests within network. But both of them depend on some
specific network environment but CURP can be deployed on any kind of
environment with little change to original architecture with witness.
```

3. Which article is the top search result (the article on this topic with the most citations)?

```
Since I am new to this topic, so I am not sure about my keyword,
If we refer "consensus algorithm" as the keyword, then I think the top search
result would be "In Search of an Understandable Consensus Algorithm".

However, this article is kind of made some improvement on "consensus
algorithm" by reducing RTTs and I really don't know which keyword I should
use, since some papers may try to solve this problem but their titles does
not reveal this kind of information. I may say "Fast paxos" is the top search
result.
```

4. What were the limitations of the experiment(s) described in the paper?

```
Lacking scenario that master, backup or witness's crashes, since they mention
that master and witness would stop servicing clients requests during
recovery. This will largely affection system's overall performance.
```

5. What questions remain unanswered by this paper?

```
1. CURP cannot handle byzantine faults(Beginning of Sec 3.1)
2. How to set the batch size of syncs(Last paragraph of Sec 4.4)
   We cannot set batch size too large since it will bring too much witness
rejections, while we also cannot set it too small since it will affect CURP's
performance. The size need to be fine-tuned based on specific scenario.
3. How to determine whether two operations are commutative for non key-value
stores, especially for DBMS with triggers and views.(Sec 3.2.2)
```

6. Are the research results reproducible? Did the authors release their data and source code? If so, where?

```
No, they did not provide code.
```

7. Are you convinced by the authors' arguments? Why or why not?

```
No. It's kind of true that with witness RTT will be 1. However, CURP brings f
witnesses if we have f backups. Clients will send (f + 1) requests and wait
for (f + 1) responses. So the actual wait time would be the maximum of those
(f + 1) RTTs. Any one of these witnesses' crash will largely delay system's
performance.
```

8. What would you like to see added to this paper?

```
Experiments on master, witness and backup's crashes.
```

9. Is the paper written for an academic or an industry audience? Which industry engineers (at which companies) would benefit from reading this paper?

```
I think this paper is for industry audiences.
Any engineers who want to reduce clients' response time for requesting
replication data can benefit from this paper.
```