XFT: Practical Fault Tolerance beyond Crashes

Shengyun Liu; Paolo Viotti; Christian Cachin; Vivien Quéma; Marko Vukolić
Synchrony vs Asynchrony

CFT

BFT

New Perspective
Crash Fault Tolerance

- Synchronous CFT: $f + 1$ replicas
- Asynchronous CFT (i.e. Paxos): $2f + 1$ replicas
Byzantine Fault Tolerance

- Synchronous BFT (i.e. Authenticated): $f + 1$ replicas
- Asynchronous BFT (i.e. PBFT): $3f + 1$ replicas
Synchronous Replica

Synchronous Group: a group of replicas, in which every pair of replicas can communicate among each other within delay $\Delta$.

Partitioned replica: a replica that is not in the largest synchronous group.

Synchronous replica: a replica that is not partitioned.
Get Rid of Asynchronous System

Regard partitioned replicas as faulty!
XFT: Practical Fault Tolerance beyond Crashes

Shengyun Liu; Paolo Viotti; Christian Cachin; Vivien Quéma; Marko Vukolić
Different Conditions

CFT:
- Synchrony/Asynchrony
- Crash Faults

BFT:
- Synchrony/Asynchrony
- Byzantine Faults
Different Conditions

Crash Faults
Non-Crash Faults
Partition Faults
Observations

Accidental non-crash faults usually do not lead to network partitions.

Malicious faults rarely cause the whole network to break down.

Assumptions in fault-tolerance model is strong, but not practical.
Observations

Extremely bad cases are rare, it is not worth paying for asynchronous BTF premium.
"Extremely Bad"
"Extremely Bad"

Anarchy: The system is in anarchy at a given moment $s$ iff $t_{nc}(s) > 0$ and $t_c(s) + t_{nc}(s) + t_p(s) > t$. 
"Extremely Bad"

Anarchy: The system is in anarchy at a given moment s iff $t_{nc}(s) > 0$ and $t_c(s) + t_{nc}(s) + t_p(s) > t$.

Change the SMR problem from "tolerating $f$ faults" to "tolerating non-anarchy faults".
New Model by New Assumption

XFT Protocol:

Protocol P is an XFT-Protocol if P satisfies safety in all executions in which the system in anarchy.
Possible Usage

Tolerating "accidental" non-crash faults

Wide-area networks and geo-replicated systems

Blockchain
XFT: Practical Fault Tolerance beyond Crashes

Shengyun Liu; Paolo Viotti; Christian Cachin; Vivien Quémé; Marko Vukolić
XPaxos

A group of $t + 1$ replicas must have at least 1 correct replica, outside anarchy.

Correct replicas are synchronous to each other.

Views as groups of $t + 1$ replicas ensure consistency.
Common Case

Leader-based, acting similar to Paxos.

Only active replicas \((t + 1)\) replicas in each view) participate.

When \(t = 1\), the algorithm is specially optimized.

Clients confirm commit only when receiving \(t + 1\) replies.
Active Replicas

In each view, exactly $t + 1$ replicas are active. (Lazy replication)

The common case runs if active replicas are synchronous.
General case: $t \geq 2$
General case: $t \geq 2$

Client sends request to primary
General case: $t \geq 2$

Client sends request to primary

Primary sends $(\text{req}, \text{prep})$ to its $t$ followers, and log it into its PrepareLog
General case: $t \geq 2$

Client sends request to primary

Primary sends $\langle req, prep \rangle$ to its $t$ followers, and log it into its PrepareLog

Followers log PrepareLog and send COMMIT to all active replicas
General case: $t \geq 2$

Client sends request to primary

Primary sends $<\text{req, prep}>$ to its $t$ followers, and log it into its PrepareLog

Followers log PrepareLog and send COMMIT to all active replicas

If $t$ COMMIT messages are received, log it into CommitLog, execute it and reply to clients when possible
General case: $t \geq 2$

Client sends request to primary

Primary sends $<\text{req, prep}>$ to its $t$ followers, and log it into its PrepareLog

Followers log PrepareLog and send COMMIT to all active replicas

If $t$ COMMIT messages are received, log it into CommitLog, execute it and reply to clients when possible

Client commits when receiving $t + 1$ matching replies
Special case: $t = 1$
Special case: $t = 1$

Client sends request to primary
Special case: $t = 1$

Client sends request to primary

Primary sends <req, commit> to its $t$ followers, and log it into its PrepareLog
Special case: $t = 1$

Client sends request to primary

Primary sends $<\text{req, commit}>$ to its $t$ followers, and log it into its PrepareLog

Followers log CommitLog and send COMMIT to primary
Special case: $t = 1$

Client sends request to primary

Primary sends <req, commit> to its $t$ followers, and log it into its PrepareLog

Followers log CommitLog and send COMMIT to primary

Primary commits and replies to the client
Special case: $t = 1$

Client sends request to primary

Primary sends $\langle\text{req, commit}\rangle$ to its $t$ followers, and log it into its PrepareLog

Followers log CommitLog and send COMMIT to primary

Primary commits and replies to the client

Client commits when receiving primary’s matching replies
Strong Definition in XPaxos

A correct replica is fault-free and not partitioned.
View-Change

ALL replicas participates in view-change.

Outside anarchy, there are at least $t + 1$ replicas.

Every group of $t + 1$ replicas has at least 1 correct replica.

Communication between 2 correct replicas is reliable.
View-Change Initiation

Only an active replica in a certain view can initiate a view change.

Triggered upon receiving invalid messages or valid SUSPECT.

Stops participating in the current view and sends SUSPECT message to ALL replicas.
View Change (view i to view i + 1)
View Change (view $i$ to view $i + 1$)

ALL replica send VIEW-CHANGE to active replicas in view $i + 1$. 
View Change (view i to view i + 1)

ALL replica send VIEW-CHANGE to active replicas in view i + 1.

New active replicas collect $n - t$ VIEW-CHANGE and send VC-FINAL messages to other active replicas.
View Change (view i to view i + 1)

All replica send VIEW-CHANGE to active replicas in view i + 1.

New active replicas collect n - t VIEW-CHANGE and send VC-FINAL messages to other active replicas.

All active replicas receive ALL VC-FINAL, and select the highest-view requests.
View Change (view i to view i + 1)

ALL replica send VIEW-CHANGE to active replicas in view i + 1.

New active replicas collect n - t VIEW-CHANGE and send VC-FINAL messages to other active replicas.

All active replicas receive ALL VC-FINAL, and select the highest-view requests.

New primary sends NEW-VIEW to other active replicas.
In Each Message

VIEW-CHANGE contains sender’s CommitLog.

VC-FINAL contains all VIEW-CHANGE received by the replica.

NEW-VIEW contains PrepareLog, which is the set of highest requests from VC-FINAL.
View Change (view 0 to view 1)

\[ \text{sg}_0: \{s_0, s_1\} \]
\[ \text{sg}_1: \{s_0, s_2\} \]
Possible Problem
Possible Problem

Consistency holds in this case as it is outside anarchy, but can we do better?
Possible Problem

Consistency holds in this case as it is outside anarchy, but can we do better?
Possible Problem
Possible Problem

Byzantine fault is hard to detect, so we don't know if the system is still safe (outside anarchy)!
Fault Detection

To detect outside anarchy those non-crash faults that may cause inconsistency in anarchy.

Particularly, we focus on message-loss, which cannot be detected by digital signature.
Fault Detection - How?

Replicas provide both PrepareLog and CommitLog in view change.

A message loss should happen in both logs; however, it then will conflict with at least one correct replica's log.
Fault Detection - Scenario
Correctness

Consistency:

Availability:
Correctness

Consistency:

There is always at least 1 correct common replica in 2 different synchronous groups.

Availability:
Correctness

Consistency:

There is always at least 1 correct common replica in 2 different synchronous groups.

Availability:

There is at least 1 correct synchronous group outside anarchy.
Performance

![Performance Graph]

- XPaxos
- Paxos
- PBFT
- Zyzzyva
- Zab
Reliability - Consistency

Assumption: Fault states of replicas are i.i.d.

Measure reliability using nines of reliability.

Consistency of XPaxos is higher than CFT, but lower than BFT.
XFT: Practical Fault Tolerance beyond Crashes

Shengyun Liu; Paolo Viotti; Christian Cachin; Vivien Quéma; Marko Vukolić

Synchrony vs Asynchrony

XFT

XPaxos