Appendix

Contents

Abstract 1
1 Introduction 1
2 Overview 3
3 Well-Organized Replicated Data Types 5
3.1 Replicated Data Types 5
3.2 Semantics 7
3.3 Guarantees 10
4 AR Replication 11
4.1 AR Apps 11
4.2 Replication Protocol 12
5 Evaluation 18
5.1 Implementation 18
5.2 Experimental Results 19
5.3 Latency, Location Staleness, and Throughput 20
5.4 Impact of Request Load 20
5.5 Scalability 21
5.6 Impact of Board Topology 22
5.7 Impact of Network Latency 22
5.8 Fault Tolerance 23
6 Related Work 24
7 Conclusions 25
8 Data-Availability Statement 26
References 26
9 Revision 31
9.1 Presentation Comments 32
Contents 34
10 Proofs 35
10.1 Proof of Well-Organization Properties 35
10.2 Proof of Correctness of the Protocol 42
11 Implementation details 60
12 Additional Results 63
12.1 Impact of Request Load 63
12.2 Scalability 63
12.3 Impact of Board Topology 63
12.4 Impact of Network Latency - Homogeneous 63
12.5 Impact of Network Latency - Heterogeneous 63
12.6 Fault Tolerance 63

10 Proofs

10.1 Proof of Well-Organization Properties

Application of a call c to a state σ, $c(\sigma)$ is naturally lifted to application $x(\sigma)$ of an execution history x to a state σ.

DEFINITION 10.1 (PERMISSIBLE EXECUTION). *A replicated execution xs is permissible, written as $\mathcal{P}(xs)$, iff for every process p, every call $c \in xs(p)$ is permissible in the state resulting from the sub-history of $xs(p)$ before c,* i.e., $\mathcal{P}(\mathsf{pre}(xs(p), c)(\sigma_0), c)$.

DEFINITION 10.2 (STATE-CONFLICT-SYNCHRONIZING). *A replicated execution xs is state-conflict-synchronizing, written as* $\mathsf{SConfSync}(xs)$, *iff for every pair of processes p and p' and pair of calls c and c' such that $c \bowtie_S c'$,*

1. $c \in xs(p) \land c' \in xs(p') \rightarrow$
$c \in xs(p') \lor c' \in xs(p)$
2. $c' \prec_{xs(p)} c \land c \in xs(p') \rightarrow c' \prec_{xs(p')} c$

LEMMA 10.1 (INVARIANT).
For all W and τ, if $W_0 \xrightarrow{\tau}{}^* W$, then
let $\langle ss, xs \rangle := W$ *in*
let $\overline{[p_i \mapsto \sigma_i]}_{i \in \{1..|P|\}} := ss$ *in*
let $\overline{[p_i \mapsto x_i]}_{i \in \{1..|P|\}} := xs$ *in*
(A_0) *For all $i, j \in \{1..|P|\}$, u, v, and r,*
$u(v)^r_{p_i} \in x_j \rightarrow (p_i, u(v)^r_{p_i} r) \in \tau \land$
$u(v)^r_{p_i} \in x_j \leftrightarrow (p_j, u(v)^r_{p_i} r) \in \tau$
(A_1) *For all $i \in \{1..|P|\}$,*
$\sigma_i = x_i(\sigma_0)$
(A_2) $\mathcal{P}(xs)$
(A_3) $\mathsf{SConfSync}(xs)$
(A_4) $\forall c = u(v)^r_p \in \mathsf{Pending}(xs)$.
$\mathsf{PRCommAll}(xs, p, c)$
(A_5) $\forall u, v, p, r, c, p'$.
$c = u(v)^r_p \in xs(p') \rightarrow c \in xs(p)$

PROOF. The proof is by induction on the steps.
Case analysis on the step:

Case CALL:

A_0:
The call is on the label and is added to $xs(p)$.

A_1:
By the induction hypothesis and the premise $\sigma' = u(v)(\sigma)$ of the CALL rule.

A_2:
Immediate from the premise $\mathcal{P}(\sigma, c)$ of the CALL rule.

A_3:
The condition 1 of SConfSync for the new call c:
We have
$c \in xs(p) \wedge c' \in xs(p')$
We should show
$c \in xs(p') \vee c' \in xs(p)$
By the contra-positive of the premise CallSConfSync, we have
$c' \in xs(p') \wedge c \bowtie_S c' \longrightarrow c' \in xs(p)$
Thus, we have
$c' \in xs(p)$.

The condition 2 of SConfSync for the new call c:
Since the identifier r of the call c is unique, from the contra-positive of A_0, we have that for all p',
$c \notin xs(p')$
Therefore, the implication trivially holds.

A_4:
Immediate from the two premises of the Call rule:
CallIComm(xs', c) and
$xs' = xs[p \mapsto (xs(p) ::: c)]$

A_5:
The call $c = u(v)^r_p$ is added to $xs(p)$ itself.

Case Prop:

A_0:
$c = u(v)^r_{p'}$,
By the premise of the Prop rule, we have
$c \in xs(p')$
By the induction hypothesis of A_0,
$(p', u(v)^r) \in \tau$

A_1:
By the induction hypothesis of A_1 and the premise $\sigma' = u(v)(\sigma)$.

A_2:
Immediate from the premise $\mathcal{P}(\sigma, c)$ of the Call rule.

A_3:
The condition 1 of SConfSync for the new call c:
We have
(1) $c \in xs(p) ::: c$
(2) $c' \in xs(p')$
We should show that
$c \in xs(p') \vee c' \in xs(p) ::: c$

Let p_c be the origin of c. We have that
(3) $c \in xs(p_c)$
By induction hypothesis for A_3.1 on [2] and [3], we have
$c \in xs(p') \vee c' \in xs(p_c)$
We consider each conjunct in turn:
Case 1: $c' \in xs(p)$
Thus,
$c' \in xs(p) :: c$
That is the first disjunct of the goal.
Case 2: (4) $c' \in xs(p_c)$
From [3] and [4], we consider two cases:
Case 2.1: (5) $c \prec_{xs(p_c)} c'$
By induction hypothesis for A_3.2 on [5] and [2], we have
$c \prec_{xs(p')} c'$
that is
$c \in xs(p')$
Case 2.2: (6) $c' \prec_{xs(p_c)} c$
By the contra-positive of the premise PropSConfSync, we have
$c' \prec_{xs(p_c)} c \wedge c \bowtie_S c' \longrightarrow c' \in xs(p)$
Therefore
$c' \in xs(p)$
Thus,
$c' \in xs(p) :: c$

The condition 2 of SConfSync for the new call c:
$c = u(v)^r_{p'}$
We have that
(1) $c' \prec_{xs(p'')} c$
(2) $c \in xs(p) ::: c$
We show that
$c' \prec_{xs(p):::c} c$
By A_5, we have
(3) $c \in xs(p')$
By induction hypothesis for A_3.2 on [1] and [3], we have
(4) $c' \prec_{xs(p')} c$
By the contra-positive of the premise PropSConfSync, we have that
(5) $c' \prec_{xs(p')} c \wedge c \bowtie_S c' \longrightarrow c' \in xs(p)$
By [5] on [4], we have
$c' \in xs(p)$
Therefore,
$c' \prec_{xs(p):::c} c$

A_4:
By delivering c, the set of Pending calls can only shrink but not grow.
Further, for the process p that delivered c, in the sequence $(xs(p) \setminus x) \circ c'$, c is added to the end of $(xs(p) \setminus x)$ and is removed form c', resulting in the same post-state.

A_5:
The call $c = u(v)^r_{p'}$ that is added to $xs(p)$ is taken from $xs(p')$ itself.

Case Query:

A_0:
The histories and the states stay the same.

A_1:
The histories and the states stay the same.

A_2:
The histories stay the same.

A_3:
The histories stay the same.

A_4:
The histories and the set of Pending calls stay the same.

A_5:
The histories stay the same.

□

Lemma 10.2 (Convergence). *For all ss, xs, p and p′, if* $W_0 \longrightarrow^* \langle ss, xs\rangle$ *and* $xs(p) \sim xs(p')$ *then* $ss(p) = ss(p')$.

Proof. By $xs(p) \sim xs(p')$, the two histories $xs(p)$ and $xs(p')$ have the same set of calls. By the invariant $A_3.2$ of Lemma 10.1, their conflicting calls have the same orders. Therefore, one can be converted to the other without changing its post state by reordering its commutative calls (as shown Lemma 1 of [40]). Therefore, the post-states of the two histories are equal, *i.e.*, $xs(p)(\sigma_0)$ and $xs(p')(\sigma_0)$. Thus, by the invariant A_1 of Lemma 10.1, we have $ss(p) = ss(p')$. □

Lemma 10.3 (Integrity). *For all ss and p, if* $W_0 \longrightarrow^* \langle ss, \_\rangle$ *then* $\mathcal{I}(ss(p))$.

Proof. This lemma is immediate from the invariant A_2 of Lemma 10.1, and the facts that (1) the initial state has integrity, *i.e.*, $\mathcal{I}(\sigma_0)$, and (2) the post-state of a call is the pre-state of the next call. □

Lemma 10.4 (Eventual Delivery). *For all* W, τ, p, p' *and* c, *if* $W_0 \xrightarrow{\tau}{}^* W$, $(p, c) \notin \tau$, *and* $(p', c) \in \tau$, *then there exists* τ' *and* W' *such that* $W \xrightarrow{\tau'\cdot(p,\ c)}{}^* W'$.

Proof.
We have
$W_0 \xrightarrow{\tau}{}^* W$
$(p', c) \in \tau$
$(p, c) \notin \tau$
By invariant A_o,

$\forall c, p.\ c \in xs(p) \leftrightarrow (p, c) \in \tau.$

Thus, we have that

$c \in xs(p') \wedge c \notin xs(p)$

Thus,

$c \in \mathsf{Pending}(xs)$

We prove that every call $c \in \mathsf{Pending}(xs)$ can be delivered to all processes by induction on a linear extension of their causal order in xs:

$c_1, c_2, \ldots, c_n$

The induction hypothesis is that

There exists W' such that

$\langle ss, xs\rangle \xrightarrow{\tau'}{}^{*} W'$ where $W' = \langle ss', xs'\rangle$

$\forall j < i, p.\ (p, c_j) \notin \tau \rightarrow (p,\ c_j) \in \tau'$

We show that

There exists W'' such that

$\langle ss, xs\rangle \xrightarrow{\tau' \cdot \tau''}{}^{*} W''$ where $W'' = \langle ss'', xs''\rangle$

$\forall p.\ (p, c_i) \notin \tau \rightarrow (p,\ c_i) \in \tau' \cdot \tau''$

Consider the call $c_i = u(v)^r_{p'}$ such that $(p', c_i) \in \tau$, and a process p such that $(p, c_i) \notin \tau \cdot \tau'$.

By invariant A_o,

(1) $c_i \in xs'(p') \setminus xs'(p)$

Let A be the set of calls in $xs'(p)$:

$A \coloneqq \{c \mid c \in xs'(p)\}$

Let xs^* be the sub-history before c_i in $xs'(p')$.

$xs^* \coloneqq \mathsf{pre}(xs'(p'), c_i)$

Let L be the calls in xs^*:

$L \coloneqq \{c \mid c \in xs^*\}$

Let σ be the pre-state of c_i in $xs(p')$:

(2) $\sigma = xs^*(\sigma_0)$

Consider a call $c' \in L$.

Thus, we have

$c' \prec_{xs'(p')} c_i$

Thus, c' is causally before c_i.

By the induction hypothesis

$(p,\ c') \in \tau'$

Thus, by the A_0 property

$c' \in xs'(p)$

Therefore,

$L \subseteq A$

and

$\forall c'.\ c' \prec_{xs(p')} c_i \rightarrow c' \in xs(p)$

thus, it trivially holds that

(3) $\mathsf{PropSComm}\,(xs', p', p, c_i)$

Let R be the set of calls in $xs'(p)$ but not in xs^*:

$R \coloneqq A \setminus L$

By the contra-positive of invariant $A_3.2$, we have

$\forall c_1, c_2.$

$c_2 \prec_{xs'(p)} c_1 \wedge c_1 \in xs'(p') \wedge c_2 \not\prec_{xs'(p')} c_1 \ \rightarrow$

$c_1 \not\bowtie_S c_2$

Thus,
For all $c_1 \in L, c_2 \in R$, such that $c_2 \prec_{xs'(p)} c_1$, we have $c_1 \Phi_{\mathcal{S}} c_2$.
Therefore,
by commuting these calls with each other, the history $xs'(p)$ can be converted to the history $xs'(p)|_L \cdot xs'(p)|_R$ with the same post-state:
$xs'(p)(\sigma_0) = (xs'(p)|_L \cdot xs'(p)|_R)(\sigma_0)$
(For a history x and a set of calls C, let the projected sub-history $x|_C$ be the subsequence of the calls C in x.)
By simple rewriting, we have
$(xs'(p)|_L \cdot xs'(p)|_R)(\sigma_0) =$
$(xs'(p)|_R)((xs'(p)|_L)(\sigma_0))$
Thus
(4) $xs'(p)(\sigma_0) = (xs'(p)|_R)((xs'(p)|_L)(\sigma_0))$
By invariant A_3.2, we have
Any conflicting pair of calls in L have the same order in both xs^* and $xs'(p)$.
Therefore, similar to Lemma 10.2, commuting calls in one can convert it to the other without changing the post-state.
Thus,
(5) $(xs'(p)|_L)(\sigma_0) = xs^*(\sigma_0)$
From [2] and [5],
(6) $(xs'(p)|_L)(\sigma_0) = \sigma$
From [4] and [6], we have
(7) $xs'(p)(\sigma_0) = (xs'(p)|_R)(\sigma)$
That is the post-state of $xs'(p)$ is equal to applying the R calls to the pre-state of c_i in $xs(p')$.
By invariant A_4 (PRCommAll) with $C = \emptyset$, we have
$c_i \triangleright^{\sigma}_{\mathcal{I}} xs'(p)|_R$
Thus,
c_i is permissible in the state $(xs'(p)|_R)(\sigma)$
Thus, from [7], we have
c_i is permissible in the state $xs'(p)(\sigma_0)$
By invariant A_1,
$ss'(p) = xs'(p)(\sigma_0)$
Thus,
c_i is permissible in the state $ss'(p)$:
(8) $\mathcal{P}(c_i, ss'(p))$
From [1], [8] and [3], we have that
The rule Prop is enabled for p and c_i at the state W' and can be executed:
$W' \xrightarrow{p,\ c_i} W^*$
Similarly c_i can be delivered to other processes in steps τ'', and
by concatenating these steps to the steps given by the induction hypothesis, we get
$\langle ss, xs\rangle \xrightarrow{\tau'\cdot\tau''}{}^{*} W''$. where $W'' = \langle ss'', xs''\rangle$
$\forall p.\ (p, c_i) \notin \tau \longrightarrow (p,\ c_i) \in \tau \cdot \tau''$

□

A rule is enabled in a state if that state satisfies its pre-conditions. An infinite execution from a state W is an infinite sequence of steps starting from W that we write as $W \xrightarrow{\tau'}{}^{*}$. An infinite execution is fair if whenever a rule is enabled, either it eventually executes or it becomes permanently disabled.

Lemma 10.5 (Eventual Delivery). *For all W, τ, p, p' and c, if $W_0 \xrightarrow{\tau}{}^{*} W$, $(p, c) \notin \tau$, and $(p', c) \in \tau$, then for every fair infinite execution $W \xrightarrow{\tau'}{}^{*}$, we have $(p,\ c) \in \tau'$.*

Proof. The reasoning follows similar to Lemma 10.4. Consider a linear extension of the causal order of the Pending calls: $c_1, c_2, ..., c_n$. As shown in the proof of Lemma 10.4, by induction, the delivery of c_1 to c_{i-1} in p' makes c_i enabled at p'. Since the execution is fair, the rule Prop is eventually executed for c_i at p'.

The induction hypothesis is that
For all τ' such that
$\langle ss, xs\rangle \xrightarrow{\tau'}{}^{*}$
$\forall j < i, p.\ (p, c_j) \notin \tau \rightarrow (p,\ c_j) \in \tau'$
Thus,
There exists W' such that
$\langle ss, xs\rangle \xrightarrow{\tau'_1}{}^{*} W' \xrightarrow{\tau'_1}{}^{*}$ where
$W' = \langle ss', xs'\rangle$
$\tau' = \tau'_1 \cdot \tau'_2$
$\forall j < i, p.\ (p, c_j) \notin \tau \rightarrow (p,\ c_j) \in \tau'_1$
This is the same as the induction hypothesis of the proof of Lemma 10.4.
Thus, it can be similarly shown that
The rule Prop is enabled for p and c_i at the state W'.
Thus, by the fairness of the infinite execution, it will be eventually executed
Thus, $(p,\ c_i) \in \tau'$ □

10.2 Proof of Correctness of the Protocol

For brevity in this section, since the only method of our object is $move(a)$, we elide $move$, and abbreviate a call $c = move(a)$ as a. We use the bar notation $\overline{e}$ for a set of elements e, and $\overline{e_i}$ for a set of elements e indexed by i. When needed, we make the range of indices i explicit as a superscript for the bar: $\overline{e_i}^i$ We use the notation $\circ\,\overline{a}$ as the composition $a_1 \circ a_2 \circ .. \circ a_n$ of $\overline{a}$. For a map M from processes p, we write M_p as the value for process p. Similar to § 3.2, we lift set operators to histories; for example, in $\Pi = xs \setminus xs_p$, Π is the set of actions of the execution history of all processes except p. We write the length of the dimension d as $length(d)$. For an action $a = \langle m, d\rangle$, the opposite action $-a = \langle m, -d\rangle$ is the action with the same magnitude m in the opposite direction $-d$. For every vector l, we write the element in the direction d as l_d. Similarly, for a multi-set of actions A, we write $A|_d$ for the subset of actions in A in the direction d. We lift the addition $A_1 + A_2$ and subtractions $A_1 - A_2$ operations on multi-sets of actions A_1 and A_2 to point-wise addition and subtraction on directions. We note that opposite directions such as right X^+ and left X^- are considered separately. The result is a set of actions, one per direction; thus, it can be seen as a $2 \times |X|$ vector where X is the number of axes. For a single multi-set of actions A, we write $+A$ as the result of adding the actions in A. Similarly, two sets of actions can be compared by point-wise comparison on directions. A multi-set of actions A_1 is larger than another A_2, written as $A_1 > A_2$, if for every action $\langle m_1, d\rangle \in +A_1$, there exists m_2, such that $\langle m_2, d\rangle \in +A_2$ and $m_2 > m_1$.

Fig. 19 presents the transition system of the protocol in Alg. 2. The state of the transition system mirrors the state the protocol and is defined in Fig. 18.

The rule P-Call captures the *Move* request handler (at L. 13). A pre-processing step (at L. 15) shrinks the action to make it permissible if it's not already. Therefore, the handler continues with only permissible calls. This condition is captured as $\mathcal{P}(loc, a)$ in the rule P-Call. The rule further checks that the process owns enough credits in holding, and moves them to kept, updates the location, and further sends update messages to each other process. It records a trivial acknowledgment from itself for this action. Further, the available credits in the system in the opposite direction is increased. The steps (starting at L. 17) that obtain enough credit are modeled as the rules P-Deb, P-Cred, and P-Dep. The rule P-Deb captures requesting credit (at L. 54). It sends debit messages to other processes. The pool of debit messages D acts as a priority queue. The rule P-Cred captures sending credit to requesting processes (at L. 63). It deducts as much credit as it can from its holding for the request with the highest priority, and sends a credit message to the requesting process. The rule P-Dep captures receiving credit (at L. 69). The process receives a credit message, and adds the credit to its holding. The rule P-Prop captures receiving and applying an action (at L. 26). The pool of pending actions Π acts as the waiting queue. If an action sent to the process is permissible, it is

$$
\begin{array}{rcll}
\theta & := & \langle loc, holding, kept, bound\rangle & \text{Local State} \\
\Theta & := & \overline{p \mapsto \theta} & \text{Local States} \\
\Pi & := & \overline{p \mapsto \{\overline{a}\}} & \text{Pending messages} \\
A & := & \overline{p \mapsto \{\overline{a}\}} & \text{Ack messages} \\
D & := & \overline{p \mapsto \{\overline{\langle p, a\rangle}\}} & \text{Debit Requests} \\
C & := & \overline{p \mapsto \{\overline{a}\}} & \text{Credit Requests} \\
\Omega & := & \langle \Theta, \Pi, A, D, C\rangle & \text{Global state} \\
\Omega_o & := & \overline{\langle p \mapsto \langle l_0, bound_o/n, \emptyset, bound_0\rangle}^p, \emptyset, \emptyset, \emptyset, \emptyset\rangle & \text{Global state} \\
 & & bound_o = \overline{length(d)}^d - l_0 &
\end{array}
$$

Fig. 18. Replicated State

P-Call

$$\frac{\mathcal{P}(loc,a) \qquad as = bound - conflict\text{-}actions(loc,a) + 1 \qquad holding \geq as \qquad holding' = holding - (a + as) \qquad kept' = kept[r \mapsto as] \qquad loc' = move(a, loc) \qquad \Pi' = \Pi[\overline{p \mapsto \Pi_p \cup \{a^r_{p^*}\}}^{p \neq p^*}] \qquad A' = A[p^* \mapsto A(p^*) \cup \{a^r_{p^*}\}] \qquad bound' = bound + (-a)}{\langle \Theta[p^* \mapsto \langle loc, holding, kept, bound\rangle], \Pi, A, \_, \_\rangle \xRightarrow{\textsc{Call}(p^*,\, a)} \langle \Theta[p^* \mapsto \langle loc', holding', kept', bound'\rangle], \Pi', A', \_, \_\rangle}$$

P-Prop

$$\frac{\mathcal{P}(loc,a) \qquad loc' = move(a, loc) \qquad A' = A[p^* \mapsto A(p^*) \cup \{a^r_p\}] \qquad bound' = bound + (-a)}{\langle \Theta[p^* \mapsto \langle loc, \_, \_, bound\rangle], \Pi[p^* \mapsto \pi \cup \{a^r_p\}], A, \_, \_]\rangle \xRightarrow{\textsc{Prop}(p^*,\, a)} \langle \Theta[p^* \mapsto \langle loc', \_, \_, bound'\rangle], \Pi[p^* \mapsto \pi], A', \_, \_\rangle}$$

P-Rel

$$\frac{holding' = holding + kept(r) + (-a) \qquad kept = kept \setminus \{r\}}{\langle \Theta[p^* \mapsto \langle \_, holding, kept, \_\rangle], \_, A[\overline{p \mapsto S_p \cup \{a^r_{p^*}\}}^{p}], \_, \_\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', kept', \_\rangle], \_, A[\overline{p \mapsto S_p}^{p}], \_, \_\rangle}$$

P-Deb

$$\frac{\overline{as|_d = bound(d) - conflict\text{-}actions(loc,a)|_d + 1} \qquad \neg(holding \geq as) \qquad D' = D[p \mapsto D(p) \cup \overline{\langle p^*, as|_d - fraction(\max(holding|_d, 0))\rangle}^{d}{}^{p \neq p^*}]}{\langle \Theta[p^* \mapsto \langle loc, holding, \_, \_\rangle], \_, \_, D, \_\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle loc, holding, \_, \_\rangle], \_, \_, D', \_\rangle}$$

P-Cred

$$\frac{\langle p, a\rangle = max\text{-}priority(D) \qquad D' = D \setminus \{\langle p, a\rangle\} \qquad a' = \min(a, holding) \qquad holding' = holding - a' \qquad C' = C[p \mapsto C(p) \cup \{a'\}]}{\langle \Theta[p^* \mapsto \langle \_, holding, \_, \_\rangle], \_, \_, D, C\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', \_, \_\rangle], \_, \_, D', C'\rangle}$$

P-Dep

$$\frac{holding' = holding + a}{\langle \Theta[p^* \mapsto \langle \_, holding, \_, \_\rangle], \_, \_, \_, C[p^* \mapsto S \cup \{a\}]\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', \_, \_\rangle], \_, \_, \_, C[p^* \mapsto S]\rangle}$$

Fig. 19. Protocol Transition System

applied to the current location, and an acknowledgment is sent to the sender. Further, the available credits in the system in the opposite direction is increased. The rule P-Rel captures releasing the kept credit for an action (at L. 41). If an acknowledgment for an action is received from all processes, the kept credit for the action, and further credit in the opposite of the action is returned to the holding.

Theorem 10.1 (Trace Inclusion). *For all* Ω *and* τ, *if* $\Omega_0 \overset{\tau}{\Rightarrow}^* \Omega$, *then there exists* W *such that* $W_0 \overset{\tau}{\rightarrow}^* W$.

Proof. Immediate by induction on the steps using Lemma 10.6, and the trivial fact that the refinement relation initially holds: $\mathsf{Refinement}(\Omega_o, W_0)$. □

Definition 10.3 (Refinement relation).
$\mathsf{Refinement}(\langle \Theta, \Pi, A, \_, C\rangle, \langle ss, xs\rangle) :=$
$(R_1)\ \forall p.\ loc(\Theta_p) = ss_p \wedge$
$(R_2)\ \forall p.\ \Pi_p = xs \setminus xs_p \wedge$
$(R_3)\ \forall a_p^r \in \Pi.\ a_p^r \in xs_p$
$(R_4)\ \forall p.\ bound(\Theta_p) = bound_0 + (-xs_p)$
$(R_5)\ \forall a_p^r \in \mathsf{Pending}(xs).\ \exists \Theta^*, x.$
$kept(\Theta_p) \geq bound(\Theta_p^*) - conflict\text{-}actions(loc(\Theta_p^*), a) + 1 \wedge$
$\mathsf{let}\ x := \mathsf{pre}(xs_p, a),\ \mathsf{in}$
$loc(\Theta_p^*) = x(l_0)\ \wedge$
$bound(\Theta_p^*) = bound_0 + (-x)$
$(R_6)\ A_p \subseteq xs_p$
$(R_7)\ \{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A_p}^p)$

Lemma 10.6 (Refinement). *For all* Ω_1, W, Ω' *and* ℓ, *if*
$\Omega_0 \Rightarrow^* \Omega$,
$W_0 \rightarrow^* W$,
$\mathsf{Refinement}(\Omega, W)$ *and*
$\Omega \overset{\ell}{\Rightarrow} \Omega'$,
then there exists W' *such that*
$W \overset{\ell}{\rightarrow} W'$ *and*
$\mathsf{Refinement}(\Omega', W')$.

Proof.
We have
(A0) $\Omega_0 \Rightarrow^* \Omega$, $W_0 \rightarrow^* W$
(A1) $\langle \Theta, \Pi, A, D, C\rangle := \Omega$
(A2) $\langle \Theta', \Pi', A', D', C'\rangle := \Omega'$
(A3) $\langle ss, xs\rangle := W$
(A4) $\mathsf{Refinement}(\Omega, W)$
(A5) $\Omega \overset{\ell}{\Rightarrow} \Omega'$
By Definition 10.3 on [A4], [A1], and [A3],
(A6) $(R_1)\ \forall p.\ loc(\Theta_p) = ss_p$
(A7) $(R_2)\ \forall p.\ \Pi_p = xs \setminus xs_p$
(A8) $(R_3)\ \forall a_p^r \in \Pi.\ a_p^r \in xs_p$
(A9) $(R_4)\ \forall p.\ bound(\Theta_p) = bound_0 + (-xs_p)$
(A10) $(R_5)\ \forall a_p^r \in \mathsf{Pending}(xs).\ \exists \Theta^*, x.$

$kept(\Theta_p) \geq bound(\Theta^*_p) - conflict\text{-}actions(loc(\Theta^*_p), a) + 1 \wedge$
let $x \coloneqq \text{pre}(xs_p, a)$, in
$loc(\Theta^*_p) = x(l_0) \wedge$
$bound(\Theta^*_p) = bound_0 + (-x)$

(A11) $(R_6)\ A_p \subseteq xs_p$

(A12) $(R_7)\ \{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (-\cap \overline{xs_p}^p) - (-\cap \overline{A_p}^p)$

The proof is by case analysis on the step [A5].

Case rule P-CALL:

(1) $\mathcal{P}(l, a)$
(2) $as = bound(\Theta_p) - conflict\text{-}actions(l, a) + 1$
(3) $holding \geq as$
(4) $holding' = holding - (a + as)$
(5) $kept' = kept[r \mapsto as]$
(6) $l' = move(a, l)$
(7) $\Pi' = \Pi[\overline{p' \mapsto \Pi_{p'} \cup \{a^r_p\}}^{p' \neq p}]$
(8) $\Theta = \Theta^*[p \mapsto \langle l, holding, kept, bound\rangle]$
(9) $\ell = \text{CALL}(p,\ a)$
(10) $\Theta' = \Theta^*[p \mapsto \langle l', holding', kept', bound + (-a)\rangle]$
(11) $A' = A[p \mapsto A(p) \cup \{a^r_p\}]$

By Lemma 4.1,
$\forall a, a'.\ a \Leftrightarrow_{\mathcal{S}} a'$
Thus,
(12) CallSComm(xs, p, a)

By [8]
(13) $loc(\Theta_p) = l$
By [A0], [A1],
(14.1) $\Omega_0 \Rightarrow^* \langle \Theta, \Pi, \_, \_, \_\rangle$,
By A6, A0, Lemma 10.1.A_1,
(14.2) $loc(\Theta_p) = xs_p(l_0)$,
By [2], [3], [13],
(14.3) $holding(\Theta_p) + kept(\Theta_p) \geq bound(\Theta_p) - conflict\text{-}actions(loc(\Theta_p), a) + 1$
By [A9]
(14.4) $bound(\Theta_p) = bound_0 + (-xs_p)$
By Lemma 10.7 on (14.1)-(14.4)
(14) $\forall A \subseteq \Pi_p.\ a' \in \text{compositions}(A).\ a' \triangleright^l_{\mathcal{I}}\ a.$
By [A7],
(15) $\Pi_p = xs \setminus xs_p$
By [A6],
(16) $loc(\Theta_p) = ss_p$
Lemma 10.1.A_1
(17) $ss_p = xs_p(l_0)$
By [13], [16] and [17],

(18) $l = xs_p(l_0)$

By [14], [15], and [18],

(19) let $x \coloneqq xs_p$, $l \coloneqq x(l_0)$ in
$\forall A \subseteq xs \setminus x.$
$\forall a' \in \mathsf{compositions}(A).$
$a \triangleright_{\mathcal{I}}^{l} a'$

Let

(20) $xs' = xs\,[p \mapsto (xs(p) ::: a)]$, $a \notin xs$

By [19] and [20],

let $x \coloneqq \mathsf{pre}(xs'_p, a)$, $l \coloneqq x(l_0)$ in
$\forall p'.\ a \notin xs'_{p'} \longrightarrow$
$\forall C \subseteq xs' \setminus (xs'_{p'} \cup x ::: a).\ \forall a' \in \mathsf{compositions}(C).$
$a \triangleright_{\mathcal{I}}^{\sigma} (xs_{p'} \setminus x) \circ a'$

Thus,

(20.1) $\mathsf{PRCommAll}(xs', p, a)$

Next, we show that

$\forall a_{1_{p_1}} \in \mathsf{Pending}(xs') \setminus \{a_p\}.\ \mathsf{PRCommAll}(xs', p_1, a_1)$

Let

(21) $a_{1_{p_1}} \in \mathsf{Pending}(xs') \setminus \{a_p\}$
(22) $C \subseteq xs' \setminus (xs'_{p'} \cup x ::: a).$
(23) $a' \in \mathsf{compositions}(C).$

From [20], [A10] on [21], there exists Θ^* and x,

(24) $kept(\Theta_p) \geq bound(\Theta_p^*) - conflict\text{-}actions(loc(\Theta_p^*), a) + 1$
(25) $x = \mathsf{pre}(xs'_p, a_1) \ \wedge\ loc(\Theta_p^*) = x(l_0)$
(26) $bound(\Theta_p^*) = bound_0 + (-x)$

By Lemma 10.8 on [A12] (after dropping $-(- \cap \overline{A_p}^{p})$), [24], [26], [22], [23],

(27) $a_1 \triangleright_{\mathcal{I}}^{loc(\Theta_p^*)} (xs'_{p_1} \setminus x) \circ a'.$

By [27] and [25]

let $x \coloneqq \mathsf{pre}(xs'_p, c)$, $l \coloneqq x(l_0)$ in
$a_1 \triangleright_{\mathcal{I}}^{l} (xs'_{p_1} \setminus x) \circ a'.$

Thus,

(28) $\forall a_{1_{p_1}} \in \mathsf{Pending}(xs') \setminus \{a_p\}.\ \mathsf{PRCommAll}(xs', p_1, a_1)$

From [20.1] and [28],

(29) $\forall a_p \in \mathsf{Pending}(xs').\ \mathsf{PRCommAll}(xs', p, a)$

Therefore,

(30) $\mathsf{CallIComm}(xs', a)$

By rule Call on [1], [12], [20], [30], [6],

(31) $\langle ss[p \mapsto l], xs\rangle \xrightarrow{\textsc{Call}(p,\ a)} \langle ss[p \mapsto l'], xs'\rangle$

By [A6], [8],

$ss_p = l$

Thus,

(32) $ss[p \mapsto l] = ss$

Let

(33) $ss' = ss[p \mapsto l']$

(34) $W' = \langle ss', xs' \rangle$

By [31], [32], [A3], [33], [34], [9],

(35) $W \xrightarrow{\ell} W'$

By [A6], [8], [10], [33]

(36) $\forall q \neq p.\ loc(\Theta'_q) = ss'_q$

By [10] and [33],

(37) $loc(\Theta'_p) = ss'_p$

By [36] and [37]

(38) $\forall p.\ loc(\Theta'_p) = ss'_p$

By [7]

(39) $\Pi'_p = \Pi_p \ \wedge\ \forall p' \neq p.\ \Pi'_{p'} = \Pi_{p'} \cup \{a^r_p\}$

By [A7], [39], and [30],

(40) $\forall p.\ \Pi'_p = xs' \setminus xs'_p$

By [A8], [7],

(41) $\forall a^* \in \Pi' \setminus \{a^r_p\}.\ a^* \in xs_p$

By [20],

(42) $a^r_p \in xs_p$

By [41] and [42],

(43) $\forall a^* \in \Pi'.\ a^* \in xs_p$

By [A9], [10], [20],

(44) $\forall p.\ bound(\Theta'_p) = bound_0 + (-xs'_p)$

From [20],

(45) $\mathsf{Pending}(xs') = \mathsf{Pending}(xs) \cup \{a\}$

From [5], [8], [10],

(46) $kept(\Theta'_p) \geq kept(\Theta_p)$

From [46], [A10]

(47) $(R_5)\ \forall a^r_p \in \mathsf{Pending}(xs).\ \exists \Theta^*, x.$
$kept(\Theta'_p) \geq bound(\Theta^*_p) - conflict\text{-}actions(loc(\Theta^*_p), a) + 1 \wedge$
$\mathsf{let}\ x := \mathsf{pre}(xs_p, a),\ \mathsf{in}$
$loc(\Theta^*_p) = x(l_0) \wedge$
$bound(\Theta^*_p) = bound_0 + (-x)$

The new pending call is a. We show that for a, there exists Θ_p and xs_p such that the following properties [48]-[51] hold.

From [2], [5], [10], [13],

(48) $kept(\Theta'_p) \geq bound(\Theta_p) - conflict\text{-}actions(loc(\Theta_p), a) + 1$

From [20]

(49) $\mathsf{pre}(xs'_p, a) = xs_p$

From [20], [14.2],

(50) $\mathsf{let}\ x := \mathsf{pre}(xs'_p, a)\ \mathsf{in}\ loc(\Theta_p) = x(l_0)$

From [A9],

(51) $bound(\Theta_p) = bound_0 + (-xs_p)$

Thus,

(52) $\forall a^r_p \in \mathsf{Pending}(xs) \cup \{a\}.\ \exists \Theta^*, x.$
$kept(\Theta'_p) \geq bound(\Theta^*_p) - conflict\text{-}actions(loc(\Theta^*_p), a) + 1 \wedge$
$\mathsf{let}\ x := \mathsf{pre}(xs_p, a),\ \mathsf{in}$
$loc(\Theta^*_p) = x(l_0) \wedge$
$bound(\Theta^*_p) = bound_0 + (-x)$

(53) The relation R_6 is preserved by [A11], and further, since by [11] and [20], a is added to both A_p and xs_p.

By [20], [4], [5]

(54) $[\{a \mid a \in xs'\} + \Sigma_p(holding(\Theta'_p) + kept(\Theta'_p) + C'_p) \leq bound_0 + (-\cap \overline{xs'_p}^p) - (-\cap \overline{A'_p}^p)] =$
$[\{a \mid a \in xs\} + a + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) - (a + as) + as \leq bound_0 +$
$(-\cap \overline{xs_p}^p) - (-\cap \overline{A_p}^p)] =$
$\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (-\cap \overline{xs_p}^p) - (-\cap \overline{A_p}^p)$

(55) The relation R_6 is preserved by [54] and [A12].

By [A2], [34], [38], [40], [43], [44], [52], [53], [55]

(56) $\mathsf{Refinement}(\Omega', W')$.

The conclusion is [35] and [56]

Case rule P-Prop:

(1) $\mathcal{P}(l, a)$
(2) $l' = move(a, l)$
(3) $A' = A[p \mapsto A(p) \cup \{a^r_p\}]$
(4) $\Theta = \Theta^*[p^* \mapsto \langle l, \_, \_, bound\rangle]$
(5) $\Pi = \Pi^*[p^* \mapsto \pi \cup \{a^r_p\}]$
(6) $\ell = \textsc{Prop}(p^*,\ a)$
(7) $\Theta' = \Theta^*[p^* \mapsto \langle l', \_, \_, bound + (-a)\rangle]$
(8) $\Pi' = \Pi^*[p^* \mapsto \pi]$

By [5]

(9) $a^r_p \in \Pi_{p^*}$

By [A7], [9]

(10) $a^r_p \notin xs_{p^*}$

By [A8], [9]

(11) $a^r_p \in xs_p$

By [11], [10]

(12) $a^r_p \in xs_p \setminus xs_{p^*}$

By Lemma 4.1,
$\forall a, a'.\ a \Phi_S a'$
Thus,
(13) PropSComm(xs, p, p^*, a)

Let
(14) $xs' = xs\,[p^* \mapsto (xs(p^*) ::: a)]$

By [11], [1], [13], [14], and [2]
(15) $\langle ss[p^* \mapsto l], xs\rangle \xrightarrow{\text{PROP}(p,\ a)} \langle ss[p^* \mapsto l'], xs'\rangle$

By [A6], [4],
$ss_{p^*} = l$
Thus,
(16) $ss[p \mapsto l] = ss$
Let
(17) $ss' = ss[p^* \mapsto l']$
(18) $W' = \langle ss', xs'\rangle$

By [15], [16], [A3], [17], [18], [6],
(19) $W \xrightarrow{\ell} W'$

By [A6], [4], [7], [17]
(20) $\forall q \neq p^*.\ loc(\Theta'_q) = ss'_q$
By [8] and [17],
(21) $loc(\Theta'_{p^*}) = ss'_{p^*}$
By [20] and [21]
(22) $(R_1)\ \forall p.\ loc(\Theta'_p) = ss'_p$

By [5], [8]
(23) $\forall p \neq p^*.\ \Pi'_p = \Pi_p$
(24) $\Pi'_{p^*} = \Pi_{p^*} \setminus \{a^r_p\}$
By [A7], [23], [24], and [14],
(25) $(R_2)\$\forall p.\ \Pi'_p = xs' \setminus xs'_p$

By [5], [8],
(26) $\Pi' \subseteq \Pi$
By [A8], [26], [14]
(27) $(R_3)\ \forall a \in \Pi'.\ a \in xs_p$

By [A9], [7], [14],
(28) $(R_4)\ \forall p.\ bound(\Theta'_p) = bound_0 + (-xs'_p)$

(29) The relation R_5 is preserved since the set of pending calls can only shrink, *kept* is unchanged, and xs is only extended.

(30) The relation R_6 is preserved by [A11], and further, since by [3] and [14], a is added to both A_{p^*} and xs_{p^*}.

If $a \in \cap_p A_p$, by [A11], $a \in \cap_p xs_p$, then by [14], [11], [4], [7],

(31) $[\{a \mid a \in xs'\} + \Sigma_p(holding(\Theta'_p) + kept(\Theta'_p) + C'_p) \leq bound_0 + (- \cap \overline{xs'_p}^p) - (- \cap \overline{A_p}^p)] =$
$\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (- \cap \overline{xs_p}^p - a) - (- \cap \overline{A_p}^p - a) =$
$\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A_p}^p)$

(32) The relation R_6 is preserved by [31] and [A12].

If $a \notin \cap_p A_p$, by [14], [11], [4], [7]

(33) $\{a \mid a \in xs'\} + \Sigma_p(holding(\Theta'_p) + kept(\Theta'_p) + C'_p) =$
$\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq$
$bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A_p}^p) =$
$bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A'_p}^p) \leq$
$bound_0 + (- \cap \overline{xs'_p}^p) - (- \cap \overline{A_p}^p)$

(34) The relation R_6 is preserved in either case.

By [A2], [18], [22], [25], [27], [28], [29], [30], [34],

(35) Refinement(Ω', W').

The conclusion is [19] and [35].

Case rule P-Rel:

The abstract semantics takes an ϵ step: $W \longrightarrow W$.
The relations R_1, R_2, R_3, R_4 is simply preserved since *loc*, Π and *bound* stay unchanged in P-Rel.

The relation R_5 is preserved since the set *kept* is changed only for $a^r_{p^*}$, and we show that $a^r_{p^*} \notin \text{Pending}(xs)$.
For all p, $a^r_{p^*}$ is in A_p, and by [A11], for all p, $a^r_{p^*} \in xs_p$; therefore, $a^r_{p^*} \in \cap_p xs_p$. Thus, $a^r_{p^*} \notin \text{Pending}(xs)$.

The relation R_6 is preserved since A is only shrinking.

The relation R_7 is preserved:
$[\{a \mid a \in xs'\} + \Sigma_p(holding(\Theta'_p) + kept(\Theta'_p) + C'_p) \leq bound_0 + (- \cap \overline{xs'_p}^p) - (- \cap \overline{A'_p}^p)] =$
$[\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) + kept(r) + (-a) + -kept(r) \leq bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A_p}^p) - (-a)] =$
$\{a \mid a \in xs\} + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p) \leq bound_0 + (- \cap \overline{xs_p}^p) - (- \cap \overline{A_p}^p)$

Case rule P-Deb:

The abstract semantics takes an ϵ step: $W \longrightarrow W$.

The relations $R_1, R_2, R_3, R_4, R_5, R_6, R_7$ is simply preserved since *loc*, Π, *bound*, *kept*, *xs*, A *holding* and C stay unchanged in P-DEB and the ϵ transition.

Case rule P-CRED:
The abstract semantics takes an ϵ step: $W \longrightarrow W$.
The relations $R_1, R_2, R_3, R_4, R_5, R_6$ is simply preserved since *loc*, Π, *bound*, *kept*, *xs* and A stay unchanged in P-CRED and the ϵ transition.

The relation R_7 is preserved since the decrease from *holding* and the increase in C cancel each other.

Case rule P-DEP:
The abstract semantics takes an ϵ step: $W \longrightarrow W$.
The relations $R_1, R_2, R_3, R_4, R_5, R_6$ is simply preserved since *loc*, Π, *bound*, *kept*, *xs* and A stay unchanged in P-DEP and the ϵ transition.

The relation R_7 is preserved since the increase from *holding* and the decrease in C cancel each other. □

LEMMA 10.7. *For all* Θ, Π, p, a *and* x, *if*
$\Omega_0 \Rightarrow^* \langle \Theta, \Pi, \_, \_, \_ \rangle$,
$loc(\Theta_p) = x(l_0)$,
$holding(\Theta_p) + kept(\Theta_p) \geq bound(\Theta_p) - conflict\text{-}actions(loc(\Theta_p), a) + 1$, *and*
$bound(\Theta_p) = bound_0 + (-x)$
then
$\forall A \subseteq \Pi_p.\ a' \in \mathsf{compositions}(A).\ a' \triangleright_I^l a$.

PROOF.
We assume
(1) $\Omega_0 \Rightarrow^* \langle \Theta, \Pi, \_, \_, \_ \rangle$
(2) $loc(\Theta_p) = x(l_0)$
(3) $holding(\Theta_p) + kept(\Theta_p) \geq bound(\Theta_p) - conflict\text{-}actions(loc(\Theta_p), a) + 1$
(4) $bound(\Theta_p) = bound_0 + (-x)$
By Lemma 10.9 on [1],
(5) $\Pi_p|_d = length(d) - loc(\Theta_p)_d - \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p)|_d - (-A_p)|_d$
By [5] and [2],
(6) $\Pi_p|_d = length(d) - (l_0 + x)|_d - \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p)|_d - (-A_p)|_d$
that is
(7) $\Pi_p|_d = length(d) - (l_0(d) + x|_d - x|_{-d}) - \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p)|_d - (-A_p)|_d$
thus,
(8) $\Pi_p|_d \leq length(d) - (l_0(d) - x|_{-d}) - (holding(\Theta_p) + kept(\Theta_p))|_d$
By aggregating [8] over all d
(9) $\Pi_p \leq \overline{length(d)}^d - l_0 + (-x) - (holding(\Theta_p) + kept(\Theta_p))$
that is
(10) $\Pi_p \leq bound_0 + (-x) - (holding(\Theta_p) + kept(\Theta_p))$
By [10] and [4]
(11) $\Pi_p \leq bound(\Theta_p) - (holding(\Theta_p) + kept(\Theta_p))$
By [11] and [3],

(12) $\Pi_p < \mathit{conflict\text{-}actions}(\mathit{loc}(\Theta_p), a)$

By Property 10.1 on [12]

(13) $\forall \overline{a'} \subseteq \Pi_p.\ a \triangleright_{\mathcal{I}}^{l} \circ \overline{a'}$

By [13],

$\forall A \subseteq \Pi_p.\ a' \in \mathsf{compositions}(A).\ a \triangleright_{\mathcal{I}}^{l} a'$ □

Lemma 10.8. *For all* Θ, C, xs, p', a, a' *and* x, *if*

$\{a \mid a \in xs\} + \Sigma_p(\mathit{holding}(\Theta_p) + \mathit{kept}(\Theta_p) + C_p) \leq \mathit{bound}_0 + (- \cap \overline{xs_p}^p)$

$\mathit{kept}(\Theta_p) \geq \mathit{bound}(\Theta_p^*) - \mathit{conflict\text{-}actions}(\mathit{loc}(\Theta_p^*), a) + 1$

$\mathit{bound}(\Theta_p^*) = \mathit{bound}_0 + (-x)$

$C \subseteq xs \setminus (xs(p') \cup x ::: a)$

$a' \in \mathsf{compositions}(C)$

then

$a \triangleright_{\mathcal{I}}^{\mathit{loc}(\Theta_p^*)} (xs(p') \setminus x) \circ a'.$

Proof.

We assume

(A1) $\{a \mid a \in xs\} + \Sigma_p(\mathit{holding}(\Theta_p) + \mathit{kept}(\Theta_p) + C_p) \leq \mathit{bound}_0 + (- \cap \overline{xs_p}^p)$

(A2) $\mathit{kept}(\Theta_p) \geq \mathit{bound}(\Theta_p^*) - \mathit{conflict\text{-}actions}(\mathit{loc}(\Theta_p^*), a) + 1$

(A3) $\mathit{bound}(\Theta_p^*) = \mathit{bound}_0 + (-x)$

(A4) $C \subseteq xs \setminus (xs(p') \cup x ::: a)$

(A5) $a' \in \mathsf{compositions}(C)$

From [A1]

(1) $\{a \mid a \in xs\} + \Sigma_p(\mathit{holding}(\Theta_p) + \mathit{kept}(\Theta_p) + C_p) \leq \mathit{bound}_0 + (- \cap \overline{xs_p}^p)$

thus,

(2) $\{a \mid a \in xs\} \leq \mathit{bound}_0 + (- \cap \overline{xs_p}^p) - \Sigma_p(\mathit{holding}(\Theta_p) + \mathit{kept}(\Theta_p) + C_p)$

Let

(3) $I_x = -[(\cap \overline{xs_p}^p) \cap x]$

(4) $I_{x'} = -[(\cap \overline{xs_p}^p) \setminus x]$

From [3], and [4],

(5) $(- \cap \overline{xs_p}^p) = I_x + I_{x'}$

From [2] and [5],

(6) $\{a \mid a \in xs\} \leq \mathit{bound}_0 + I_x + I_{x'} - \Sigma_p(\mathit{holding}(\Theta_p) + \mathit{kept}(\Theta_p) + C_p)$

From [3],

(7) $I_x \leq (-x)$

From [6] and [7], and simplification

(8) $\{a \mid a \in xs\} \leq \mathit{bound}_0 + (-x) + I_{x'} - \mathit{kept}(\Theta_p)$

From [8], [A3]

(9) $\{a \mid a \in xs\} \leq \mathit{bound}(\Theta_p^*) + I_{x'} - \mathit{kept}(\Theta_p)$

thus,

(10) $\{a \mid a \in xs\} - I_{x'} \leq \mathit{bound}(\Theta_p^*) - \mathit{kept}(\Theta_p)$

From [10], [A2],

(11) $\{a \mid a \in xs\} - I_{x'} < \mathit{conflict\text{-}actions}(\mathit{loc}(\Theta_p^*), a)$

By Property 10.1 on [11]

(12) $\forall \overline{a^*} \leq \{a \mid a \in xs\} - I_{x'}.\ a \triangleright_{\mathcal{I}}^{\mathit{loc}(\Theta_p^*)} \circ \overline{a^*}$

Let us now consider the sequence $(xs(p') \setminus x) \circ a'$ where [A4] and [A5].

Let
(13) $R' = R \circ a'$ where $R = xs(p') \setminus x$.
If $R \cap I_{x'} = \emptyset$ then, by [18], we immediately have
$a \triangleright_{\mathcal{I}}^{loc(\Theta^*_p)} (xs(p') \setminus x) \circ a'$
Now consider an action $-a \in R \cap I_{x'}$.
Since $-a \in I_{x'}$, by [3],
(14) $a \notin x$
The opposite credit $-a$ is issued only after a is executed at every process. Thus,
(15) $a \in xs_{p'}$
From [14] and [15],
(16) $a \in (xs(p') \setminus x)$
Thus, in the sequence R, the two opposite actions a and then $-a$ can be removed without affecting the post-state. This process can be repeated for every such action $-a \in I_{x'}$.
Let the resulting sequence be R^*. It has no action in $I_{x'}$. Therefore, by [18]
$a \triangleright_{\mathcal{I}}^{loc(\Theta^*_p)} (xs(p') \setminus x) \circ a'$.
We also note that even if the opposite action uses a part of the opposite credit $-a$, then part of the preceding a action can be canceled, to similarly result in the same post-state. The size of R is smaller than $\{a \mid a \in xs\}$, and [18] still applies. □

By construction, for any location l and action a_1, any sequence of actions $\overline{a_2}$ that sum to less than $conflict\text{-}actions(l, a_1)$ invariant-commute with a_1. We capture this property as follows:

Property 10.1 (Property of *conflict-actions*). *For all location l, actions a_1 and $\overline{a_2}$, if $+\overline{a_2} < conflict\text{-}actions(l, a_1)$. then $a_1 \triangleright_{\mathcal{P}}^{l} \circ \overline{a_2}$.*

Lemma 10.9 (Preservation of credits). *For all Θ, Π, A, D, and C, if $\Omega_0 \Rightarrow^* \langle \Theta, \Pi, A, D, C \rangle$, then*
for all p^* and d, $loc(\Theta_{p^*})_d + \Pi_{p^*}|_d + (-A_p)|_d + \Sigma_p(holding(\Theta_p) + kept(\Theta_p) + C_p)|_d = length(d)$.

Proof.
Proof is by induction on the steps.

Base case: Ω_0
$l_0(d) + \Sigma_p(length(d) - l_0(d))/n = length(d)$

Induction Hypothesis:
$\langle \Theta, \Pi, A, D, C \rangle \Rightarrow \langle \Theta', \Pi', A', D', C' \rangle$
(IH) For all p and d,
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d = length(d)$.
We show that
For all p and d,
$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d = length(d)$.

Case rule P-Call:
If d is orthogonal to a, the conclusion directly reduces to IH.
We consider two cases (where p^* is the stepping process):
Case $p = p^*$
$loc(\Theta'_p)_d + \Pi'_p|_d + (-A_q)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$
$(loc(\Theta_p) + a)_d + \Pi_p|_d + (-(A_{p^*} + a))|_d +$

$\Sigma_{q \neq p^*}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$
$(holding(\Theta_{p^*}) - (a + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a)) +$
$kept(\Theta_{p^*}) + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a) + C_{p^*}))|_d =$
We consider two cases:
Case: a is in the direction d:
$(loc(\Theta_p) + a)_d + \Pi_p|_d + (-A_{p^*})|_d +$
$\Sigma_{q \neq p^*}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$
$(holding(\Theta_{p^*}) - (a + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a)) +$
$kept(\Theta_{p^*}) + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a) + C_{p^*})|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$
Case: a is in the opposite direction of d:
$(loc(\Theta_p) + a)_d + \Pi_p|_d + (-(A_{p^*} + a))|_d +$
$\Sigma_{q \neq p^*}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$
$(holding(\Theta_{p^*}) - conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a) +$
$kept(\Theta_{p^*}) + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a) + C_{p^*})|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$

Case $p \neq p^*$
$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$
$(loc(\Theta_p)_d + (\Pi_p + a)|_d + (-A'_p)|_d +$
$\Sigma_{q \neq p^*}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$
$(holding(\Theta_{p^*}) - (a + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a)) +$
$kept(\Theta_{p^*}) + conflict\text{-}sync\text{-}credit(loc(\Theta_{p^*}), a) + C_{p^*})|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$

Case rule P-Prop:
We consider two cases (where p^* is the stepping process):
Case $p = p^*$
$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$
$(loc(\Theta_p) + a)_d + (\Pi_p - a)|_d + (-(A_p + a))|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
We consider two cases:
Case: a is in the direction d:
$(loc(\Theta_p) + a)_d + (\Pi_p - a)|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$
Case: a is in the opposite direction of d:
$(loc(\Theta_p) + a)_d + \Pi_p|_d + (-(A_p + a))|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$

Case $p \neq p^*$
$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$
$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$
$length(d)$

Case rule P-Rel:

$$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$$
$$loc(\Theta'_p)_d + \Pi'_p|_d + (-(A_p - a))|_d +$$
$$\Sigma_{q \neq p^*}(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d +$$
$$(holding(\Theta_{p^*}) + kept(r) + (-a) + kept(\Theta_{p^*}) - kept(r) + C'_q)|_d =$$
$$loc(\Theta'_p)_d + \Pi'_p|_d + (-A_p)|_d - (-a)|_d +$$
$$\Sigma_{q \neq p^*}(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d +$$
$$(holding(\Theta_{p^*}) + kept(r) + (-a) + kept(\Theta_{p^*}) - kept(r) + C'_q)|_d =$$
$$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$$
$$length(d)$$

Case rule P-Deb:

The elements $loc\ \Pi$, A, $holding$, $kept$ and C all stay the same.

Case rule P-Cred:

$$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$$
$$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_{q \notin \{p^*, p\}}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$$
$$holding(\Theta_{p^*}) - a' + kept(\Theta_{p^*}) + C_{p^*})|_d +$$
$$holding(\Theta_p) + kept(\Theta_p) + C_p + a')|_d =$$
$$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$$
$$length(d)$$

Case rule P-Dep:

$$loc(\Theta'_p)_d + \Pi'_p|_d + (-A'_p)|_d + \Sigma_q(holding(\Theta'_q) + kept(\Theta'_q) + C'_q)|_d =$$
$$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_{q \neq p^*\}}(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d +$$
$$holding(\Theta_{p^*}) + a + kept(\Theta_{p^*}) + C_{p^*} - a)|_d =$$
$$loc(\Theta_p)_d + \Pi_p|_d + (-A_p)|_d + \Sigma_q(holding(\Theta_q) + kept(\Theta_q) + C_q)|_d =$$
$$length(d)$$

□

P-Call

$$\frac{\mathcal{P}(loc,a) \quad as = bound - conflict\text{-}actions(loc,a) + 1 \quad holding \geq as \quad holding' = holding - (a + as) \quad kept' = kept[r \mapsto as] \quad loc' = move(a, loc) \quad \Pi' = \Pi[\overline{p \mapsto \Pi_p \cup \{a^r_{p^*}\}}^{p \neq p^*}] \quad A' = A[p^* \mapsto A(p^*) \cup \{a^r_{p^*}\}] \quad bound' = bound + (-a)}{\langle \Theta[p^* \mapsto \langle loc, holding, kept, bound, \_, \_, \_, \_\rangle], \Pi, A, \_, \_\rangle \xRightarrow{\textsc{Call}(p^*,\, a)} \langle \Theta[p^* \mapsto \langle loc', holding', kept', bound, \_, \_, \_, \_\rangle], \Pi', A', \_, \_\rangle}$$

P-Prop

$$\frac{\mathcal{P}(loc,a) \quad loc' = move(a, loc) \quad A' = A[p^* \mapsto A(p^*) \cup \{a^r_p\}] \quad bound' = bound + (-a) \quad moved = moved(p) \cup \{a\} \quad opposite = opposite(p) \cup \{-a\}}{\langle \Theta[p^* \mapsto \langle loc, \_, \_, bound, moved, opposite, \_, \_\rangle], \Pi[p^* \mapsto \pi \cup \{a^r_p\}], A, \_, \_]\rangle \xRightarrow{\textsc{Prop}(p^*,\, a)} \langle \Theta[p^* \mapsto \langle loc', \_, \_, bound', moved', opposite', \_, \_\rangle], \Pi[p^* \mapsto \pi], A', \_, \_\rangle}$$

P-Rel

$$\frac{holding' = holding + kept(r) + (-a) \quad kept = kept \setminus \{r\}}{\langle \Theta[p^* \mapsto \langle \_, holding, kept, \_, \_, \_, \_, \_\rangle], \_, A[\overline{p \mapsto S_p \cup \{a^r_{p^*}\}}^{p}], \_, \_\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', kept', \_, \_, \_, \_, \_\rangle], \_, A[\overline{p \mapsto S_p}^{p}], \_, \_\rangle}$$

P-Deb

$$\frac{\overline{as|_d = bound(d) - conflict\text{-}actions(loc,a)|_d + 1} \quad \neg(holding \geq as) \quad D' = D[p \mapsto D(p) \cup \overline{\langle p^*, as|_d - fraction(\max(holding|_d, 0))\rangle}^{d}{}^{p \neq p^*}]}{\langle \Theta[p^* \mapsto \langle loc, holding, \_, \_, \_, \_, \_, \_\rangle], \_, \_, D, \_\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle loc, holding, \_, \_, \_, \_, \_, \_\rangle], \_, \_, D', \_\rangle}$$

P-Cred

$$\frac{\langle p, a\rangle = max\text{-}priority(D) \quad D' = D \setminus \{\langle p, a\rangle\} \quad a' = \min(a, holding) \quad holding' = holding - a' \quad C' = C[p, p^* \mapsto C(p, p^*) \cup \{a'\}] \quad sent = sent[p \mapsto sent(p) \cup \{a'\}]}{\langle \Theta[p^* \mapsto \langle \_, holding, \_, \_, \_, \_, sent, \_\rangle], \_, \_, D, C\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', \_, \_, \_, \_, sent', \_\rangle], \_, \_, D', C'\rangle}$$

P-Dep

$$\frac{holding' = holding + a \quad received = received[p \mapsto received(p) \cup \{a\}]}{\langle \Theta[p^* \mapsto \langle \_, holding, \_, \_, \_, \_, \_, received\rangle], \_, \_, \_, C[p^*, p \mapsto S \cup \{a\}]\rangle \Rightarrow \langle \Theta[p^* \mapsto \langle \_, holding', \_, \_, \_, \_, \_, received'\rangle], \_, \_, \_, C[p^*, p \mapsto S]\rangle}$$

Fig. 20. Protocol Transition System

Similar to the incremental presentation in the main body, we now extend the transition system with fault-tolerance in Fig. 20. We write the additional state variables that track credits in blue. In the following theorems, let $exchange\Theta_{p'})(p) = sent(\Theta_{p'})(p) - received(\Theta_{p'})(p)$. Further, let $\Pi_{p_1}|_{p_2} = \{a^r_{p'} \in \Pi_{p_1} \mid p' = p_2\}$. Similarly, let $A_{p_1}|_{p_2} = \{a^r_{p'} \in A_{p_1} \mid p' = p_2\}$.

Lemma 10.10. *For all Θ, Π, A, D, C and p' if $\Omega_0 \Rightarrow^* \langle \Theta, \Pi, A, D, C\rangle$ is an execution where the processes $\mathcal{F}$ have failed, and time Δ is past since the last one failed, then the total credit that $\mathcal{F}$ held is $\Sigma_{p\in\mathcal{F}}\ (init - moved(\Theta_{p'}(p)) + opposite(\Theta_{p'}(p)) + \Sigma_{p'\in P\setminus\mathcal{F}}\ exchange(\Theta_{p'})(p))$.*

Proof.
Let the sum of the credits in failed processes be
(1) $S = \Sigma_{p\in\mathcal{F}}\ (holding(\Theta_p) + kept(\Theta_p) + C(p))$
By Lemma 10.11.(I1)
(2) $S = \Sigma_{p\in\mathcal{F}}\ (init - \Pi_{p'}|_p - moved(\Theta_{p'})(p) + (opposite(\Theta_{p'}(p)) \setminus (- \cup_{p'} A_{p'})|_p)) + \Sigma_{p'}\ sent(\Theta_{p'})(p) - \Sigma_{p'}\ sent(\Theta_p)(p')) =$
Since processes communicate with reliable broadcast, and time Δ is past since the last one failed, all the messages from failed processes $p \in \mathcal{F}$ are delivered to correct processes $p' \in P \setminus \mathcal{F}$. Thus,
(3) $\Pi_{p'}|_p = \emptyset$
(4) $A_{p'}|_p = \emptyset$
(5) $C(p')(p) = \emptyset$
By [2], [3], and [4],
(6) $S = \Sigma_{p\in\mathcal{F}}\ (init - moved(\Theta_{p'})(p) + opposite(\Theta_{p'})(p) + \Sigma_{p'}\ sent(\Theta_{p'})(p) - \Sigma_{p'}\ sent(\Theta_p)(p'))$
By Lemma 10.11.(I2) and [5],
(7) $sent(\Theta_p)(p') = received(\Theta_{p'})(p)$
By [6] and [7]
(8) $S = \Sigma_{p\in\mathcal{F}}\ (init - moved(\Theta_{p'})(p) + opposite(\Theta_{p'})(p) + \Sigma_{p'}\ sent(\Theta_{p'})(p) - \Sigma_{p'}\ received(\Theta_{p'})(p))$
By [8] and [7] (canceling sent and received for failed processes $\mathcal{F}$
(9) $S = \Sigma_{p\in\mathcal{F}}\ (init - moved(\Theta_{p'})(p) + opposite(\Theta_{p'})(p) + \Sigma_{p'\in P\setminus\mathcal{F}}\ sent(\Theta_{p'})(p) - \Sigma_{p'\in P\setminus\mathcal{F}}\ received(\Theta_{p'})(p)$
By [9] and definition of *exchange*
$\Sigma_{p\in\mathcal{F}}\ (init - moved(\Theta_{p'}(p)) + opposite(\Theta_{p'}(p)) + \Sigma_{p'\in P\setminus\mathcal{F}}\ exchange(\Theta_{p'})(p))$ □

Lemma 10.11. *For all Θ, Π, A, D, and C if $\Omega_0 \Rightarrow^* \langle \Theta, \Pi, A, D, C\rangle$, then*
(I1) $holding(\Theta_p) + kept(\Theta_p) + C(p) =$
$init - \Pi_{p'}|_p - moved(\Theta_{p'})(p) +$
$(opposite(\Theta_{p'})(p) \setminus (- \cup_{p'} A_{p'}|_p)) +$
$\Sigma_{p'}\ sent(\Theta_{p'})(p) - \Sigma_{p'}\ sent(\Theta_p)(p')$
(I2) $sent(\Theta_p)(p') - received(\Theta_{p'})(p) = C(p')(p)$.

Proof.
Proof is by induction on the steps.

First invariant (1):
Base case: Ω_0
$holding(\Theta_p) = init$

Case rule P-Call:
Actions a and as are removed from *holding*, and action as is added to *kept* on the left.
Action $-a$ is added to $-\Pi_{p'}$ on the right.

Case rule P-Prop:
Action a is removed from $\Pi_{p'}$, and added to $moved(\Theta_{p'})$ on the right.
Action $-a$ is added to *opposite*, and added to $(- \cup_{p'} A(p'))$ if not already in it on the right.

Case rule P-Rel:
Actions $kept(r)$ and $-a$ are added to *holding*, and action $kept(r)$ is removed from *kept* on the left.
Action $-a$ is removed from $\backslash(- \cup_{p'} A(p'))$ on the right.

Case rule P-Deb:
No change

Case rule P-Cred:
For p^*:
Action a' is removed from $holding(\Theta_{p^*})$ on the left.
Action a' is added to $-sent(\Theta_{p^*})(p)$ on the right.
For p
Action a' is added to $C(p)$ on the left.
Action a' is added to $sent(\Theta(p^*)(p)$ on the right.

Case rule P-Dep:
For p^*:
Action a is added to $holding(\Theta_{p^*}$ on the left.
Action a is removed from $C(p^*)$ on the left.
For p
No change

First invariant (2):
$sent(\Theta_p)(p') - received(\Theta_{p'})(p) = C(p')(p)$.
Base case: Ω_0
$\emptyset - \emptyset = \emptyset$

Case rule P-Call:
No Change

Case rule P-Prop:
No Change

Case rule P-Rel:
No Change

Case rule P-Deb:
No Change

Case rule P-Cred:

Action a' is added to $sent(\Theta_{p^*})(p)$ on the left.
Action a' is added to $C(p)(p^*)$ on the right.

Case rule P-Dep:
Action a is added to $received(\Theta_{p^*})(p)$ on the left.
Action a is removed from $C(p)(p^*)$ on the right.

□

11 Implementation details

In this section, we detail the implementation of our protocol presented in § 4. The implementation consists of two main parts: (1) pre-computing the conflicting actions and storing them in a table; and (2) executing the runtime protocol (Alg. 1-Alg. 2), including querying the table.

(1) Pre-computing conflicts. Given the AR board and its restricted zones, we use the Z3 SMT solver [25] to determine the conflicting actions for the current location l of the virtual object and the action a. We illustrate this through the following 2D example. In Fig. 21, a replica wants to perform action a shown in orange. Two examples from the space of conflicting sequences of actions in the X^+ and Y^+ directions are shown in blue and green. Blue is a sequence of 12.5 units of X^+, and 87.5 units of Y^+. Green is 75 units of X^+, and 12.5 units of Y^+. Both are conflicting sequences: when combined with a, the result is in a restricted zone. If we calculate the minimum magnitude in the X^+ direction, we get 12.5. The minimum conflicting action is $\langle X^+, 12.5\rangle$. To prevent conflicts, we need to prevent actions of at least 12.5 units in the X^+ direction. Alternatively, if we calculate the minimum magnitude in the Y^+ direction, we get 12.5, and the minimum conflicting action is $\langle Y^+, 12.5\rangle$. To prevent conflicts, we need to prevent actions of at least 12.5 units in the Y^+ direction. Both solutions are correct. We pre-compute the minimum conflicting actions for each location l and action a using the Python Z3 API, and generate a JSON table of conflicts.

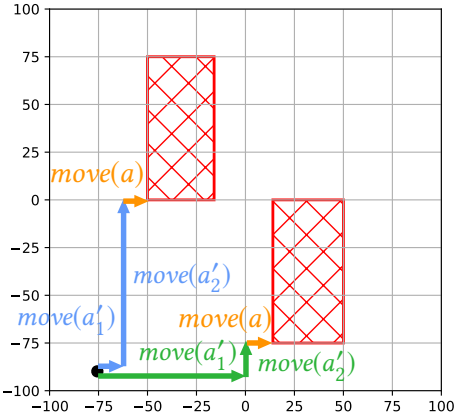


Fig. 21. An example of conflicting actions when performing action a. There are two sets (blue and green) of actions that can cause conflict. We slightly displace the a'_1 actions so that they do not fully overlap.

However, it is time-consuming to generate all conflicting actions using Z3 SMT solver. To accelerate the time of generating table of conflicts, we learned the rule of the solver when generating conflicting actions. We find that the conflicting actions generated by Z3 SMT solver can be generalized into computation when only a single restricted zone is presented in the AR board. Although different set of conflicting actions are generated by Z3 SMT solver, they provide the same hint that given the current location l and action a, the conflicting actions are the minimum magnitude of aggregated action that brings the action a into the restricted zone.

We illustrate this rule in details, Fig. 22 presents three cases with different locations l and action a. In Fig. 22a, the conflicting actions a'_1 and a'_2 can be computed by finding the closest point inside the restricted zone that action a can enter potentially which is the coordinate of (-50, 0). We then take this point and subtracts the action a (right, 12,5) starting from location l (-75, -25) to get the conflicting actions a'_1 (right, 12.5) and a'_2 (up, 25). The same rule can be applied to example Fig. 22b, where the closest point inside the restricted zone is (-12.5, 25). Another case in example Fig. 22c yields the closest point of (-50, 0) because the action a itself has a direction of right, and to enter the restricted zone, it must also enter from the right.

Therefore, given any topology, we can synthesize the conflicting tables by combining the computation results from the single restricted zone topology and only utilize Z3 SMT solver when in complex regions where more than two restricted zones are nearby. With this optimization, the pre-computation time can be reduced up to 97% for the Corner topology (from 14.8 hours to 0.4 hours).

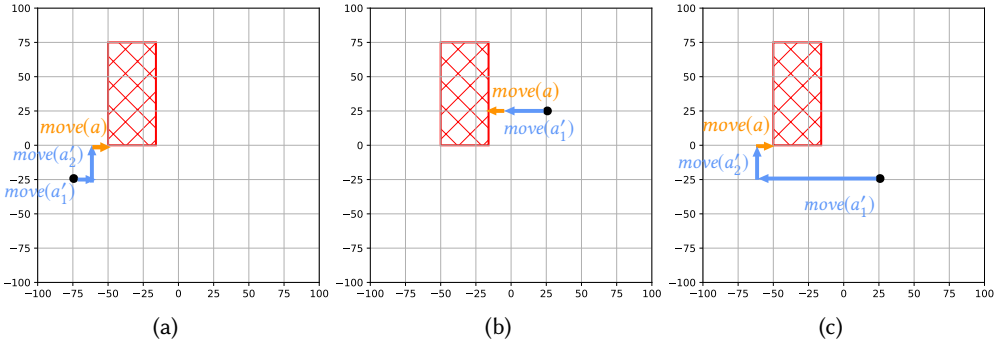


Fig. 22. Example of conflicts actions (blue) when only one restricted zone presents. These can be generalized given current location l and action a (orange), the conflicting actions (blue) lead to the closest point in the restricted zone.

Besides table generating time, we also make several optimizations to accelerate the credit-sharing process. We discretize the board, action magnitude, and action direction (left, right, up, down).

We only check likely compositions of other replicas' actions. One could naively generate the table by checking integrity violation constraints for all possible numbers of and compositions of other actions exhaustively. However, in practice, we find that this is unnecessary. First, for a given board, location, and action, certain orders of other replicas' actions cannot cause conflicts. Second, the actions of a potentially conflicting sequence might be infeasible. Considering the request frequency in AR applications, the number of concurrent actions can be bounded. Further, the number of users, and the magnitude of each action that users can take can be bounded. Therefore, their aggregated action may not be large enough to cause a conflict. In our example in Fig. 21, while the orange action a is performed, the actions of blue or green might be too large and infeasible. No credit is acquired for an infeasible conflicting sequence.

(2) Executing the protocol. We implemented the protocols presented in § 4 in Java for both the Android and desktop simulations. When a user seeks to take an action (in Alg. 2 at L. 15), the table from the previous step is queried with the current location of the virtual object, and the action to find the conflicting actions of other users (at L. 16). If needed, credits are then acquired (at L. 17).

Efficiently acquiring credits. Efficiently acquiring these credits is challenging because a replica does not know who to request credits from. We examined two configurable parameters: the *fraction* of credits to request from each replica, and the number of replicas (users) to acquire credits from, as a refinement of the *broadcast* that requests credits in Alg. 2 (at L. 58). Requesting too many credits from too many replicas results in credits being unfairly distributed most of the time, preventing replicas from performing conflict-free actions. On the other hand, requesting too few credits will lead to multiple rounds of communication, and hence increased latency. We systematically test the parameters to see the impact on the average latency of each action. We performed a grid search of *fraction* (50%, 60%, .., 100%) and the number of replicas to request from (1, 2, 3, 4) in a simulation with 7 replicas. In Fig. 23, we present the results, showing that the higher fraction and more users per request, the lower the average latency. In the rest of the experiments, we request from 4 replicas

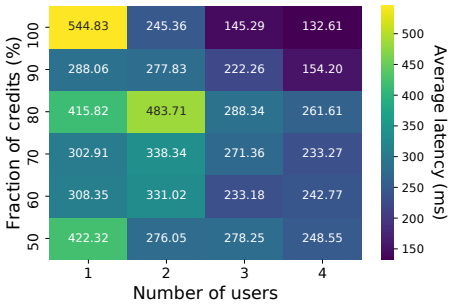


Fig. 23. Grid search result for configuring parameters [number of users, fraction of credits] with 4 users and 100% having the lowest average latency in a 7-user system.

with 100% *fraction* of the total credit needed, as it yields the lowest average latency of 132.61 ms in the simulation.

Further, to provide additional support, we implemented a complementary mechanism of selecting users that are likely to have credits. Rather than opting for a random user when requesting credits, each replica maintains a list of other replicas who have recently borrowed credits from it, and tends to choose those replicas to request from. Furthermore, when receiving credits, each replica keeps a record of which other replicas replied with fewer than requested credits, as an indication that they have insufficient credits. With these techniques, replicas can make smarter decisions when requesting credits from other replicas. Finally, to prevent a few replicas from holding all credits in the system, each replica randomly distributes their held credits when finishing actions and sensing that they hold more than 80% of the total system credits.

Waiting set. Next, we detail the implementation of our waiting set w in Alg. 2 (at L. 29). When receiving an action from another replica, it is possible that this action is not permissible at the current location (at L. 28) although it is rarely the case. Instead of recording individual pending actions in the waiting set, we summarize the pending actions in the waiting set into one aggregate action. This is more efficient at the slight risk of a potential delay, due to waiting for the aggregate action rather than the small sub-actions to become permissible.

12 Additional Results

In this section, we present the Full-sized plot of our experimental results in § 5 and also a corresponding 2D topology (Middle in Fig. 32b) results. Each subsection describes the location staleness and system throughput results for the corresponding subsections.

12.1 Impact of Request Load

Under different request loads in Fig. 26, all four methods are less responsive and have a higher location staleness with Hambazi having a median staleness percentage of 1.5%, compared to 1.41%, 1.5%, and 15.0% for Netcode, Hamsaz, and Firestore respectively.

12.2 Scalability

In terms of system throughput (Fig. 28c), Hambazi can finish almost all calls even with 7 devices issuing nearly simultaneous calls, while Netcode and Hamsaz both have unfinished calls 0.4% for all number of devices. For the Firestore baseline, although it is capable of completing calls with fewer devices (unfinished rate grows from 0%to 10.7%), note that the median latency grows significantly after more than 3 devices, with a median latency of 304.0 ms for 3 devices and 1012.0 ms for 7 devices. This indicates that while the throughput is still high, the server-based approach can only handle the contention at a slow pace resulting in a high location staleness of 7.5% with 7 devices.

12.3 Impact of Board Topology

As shown in Fig. 30b, the Corner3D and Middle3D have an average location staleness of 0.69% and 0.67% respectively, and the easiest Triangle3D has 0.33%. For the Dynamic3D topology, it has 0.39%. Fig. 31 shows the throughput results.

12.4 Impact of Network Latency - Homogeneous

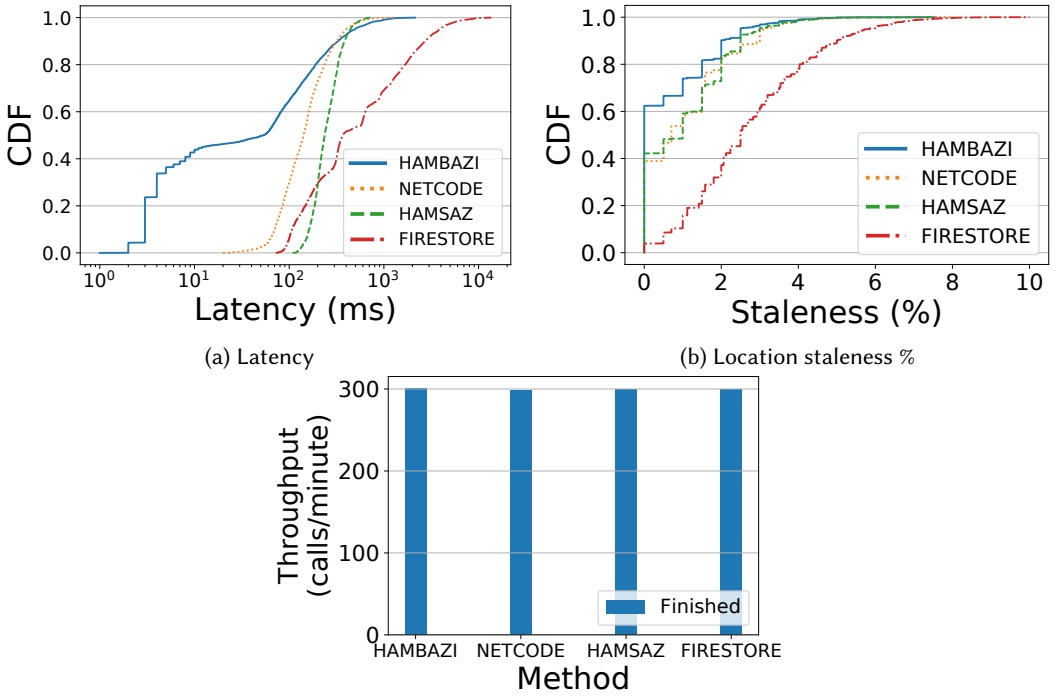
In terms of the location staleness (Fig. 35b), baseline methods experience increased staleness as the RTT increases. However, Hambazi still maintains zero staleness even with 5x RTT.

12.5 Impact of Network Latency - Heterogeneous

The location staleness of Hambazi (Fig. 37b) also remains zero staleness on the slower Device 0.Similarly, for Hamsaz, location staleness is higher for Device 0 with poor network conditions with a median staleness of 1.0%. Netcode, generally have poor staleness among all devices with a median latency of up to 1.2%. Note that Device 2 is the leader/host device in Hamsaz/Netcode and hence has zero staleness.

12.6 Fault Tolerance

The location staleness of Hambazi (Fig. 39b) sometimes decreases when failures are present because there are fewer devices in the system that contribute action calls. Similar results are shown in different topologies in Fig. 40-Fig. 47.

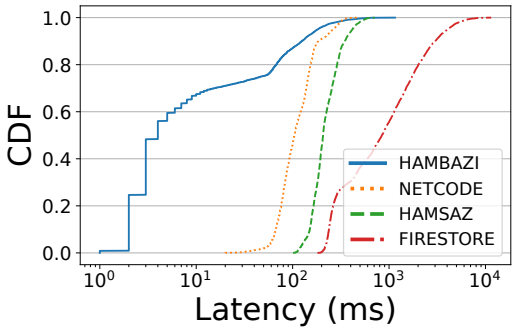


(a) Latency

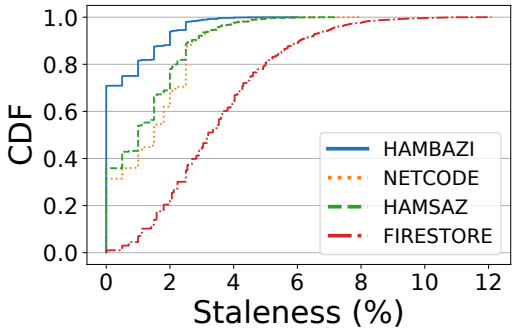
(b) Location staleness %

(c) System throughput

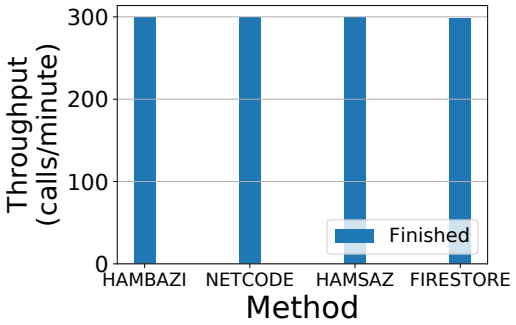
Fig. 24. Compared to NETCODE, HAMSAZ, and FIRESTORE, HAMBAZI has lower latency 90% of the time, the lowest location staleness, and similar throughput.



(a) Latency

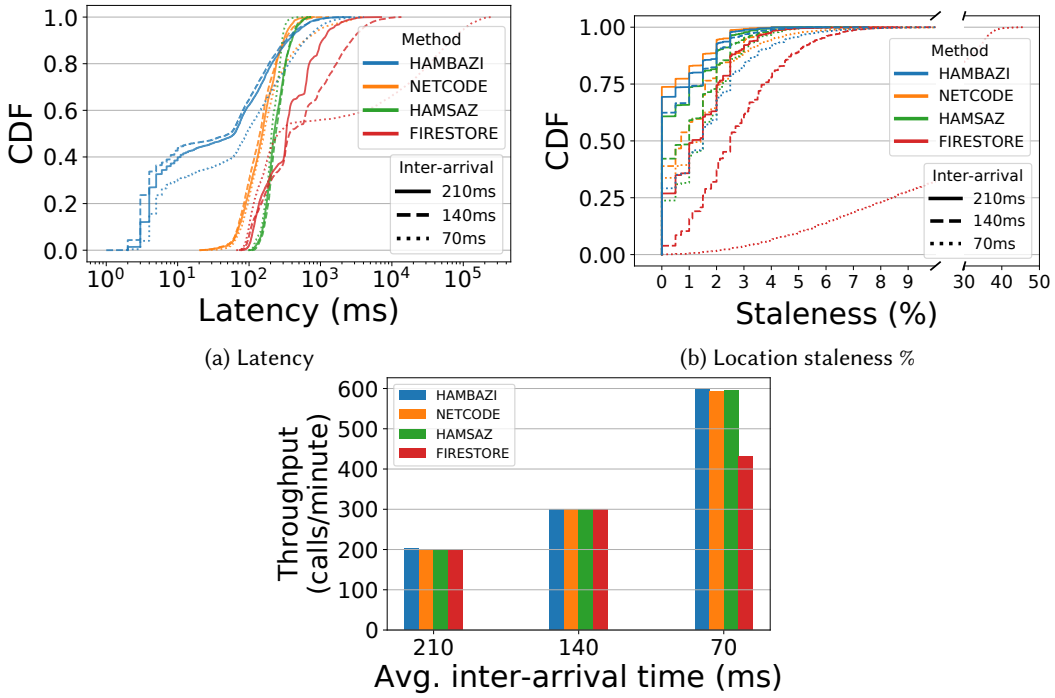


(b) Location staleness %



(c) System throughput

Fig. 25. In 2D case, compared to NETCODE, HAMSAZ, and FIRESTORE, HAMBAZI has lower latency 99% of the time, the lowest location staleness, and similar throughput.

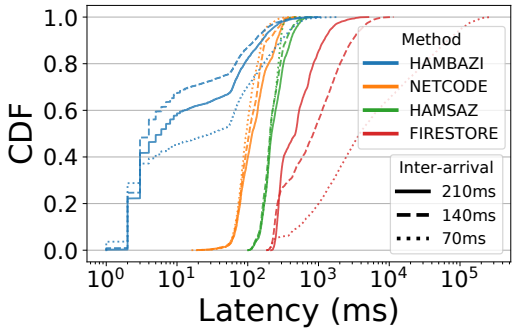


(a) Latency

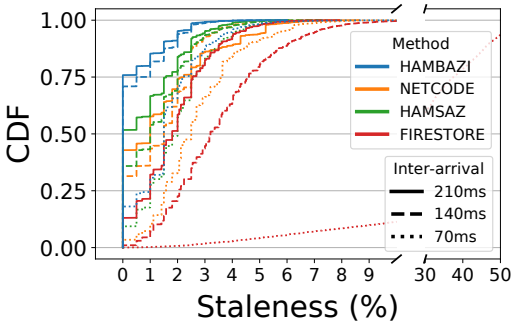
(b) Location staleness %

(c) System throughput

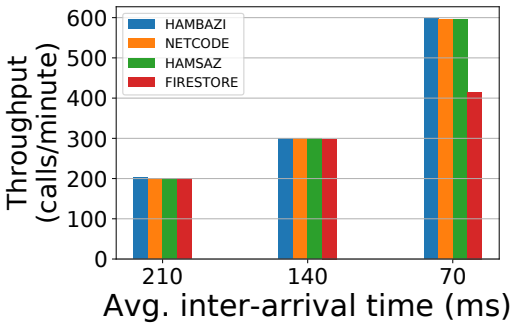
Fig. 26. Varying the mean inter-arrival times from light load (210 ms) to intense load (70 ms). With the intense load, HAMBAZI yields the lowest latency 70% of the time and maintains reasonably low location staleness compared to the baselines. In terms of system throughput, HAMBAZI is capable of finishing all calls while NETCODE, HAMSAZ and FIRESTORE have 0.2%, 0.2%, and 27.8% unfinished calls per minute respectively.



(a) Latency

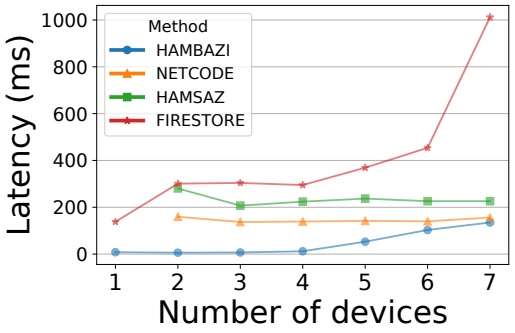


(b) Location staleness %

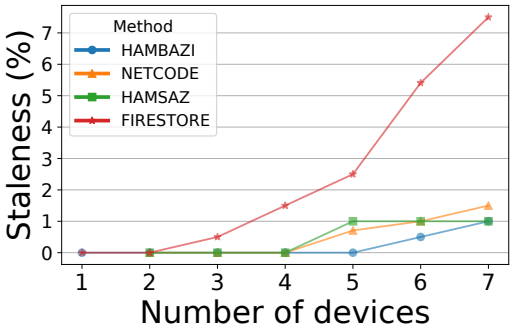


(c) System throughput

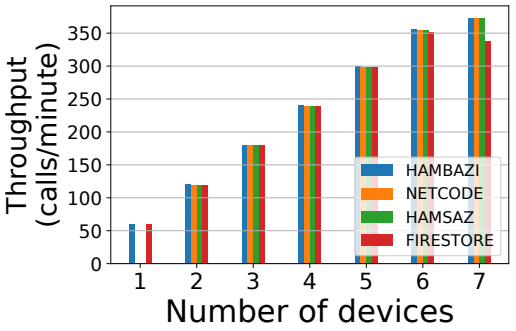
Fig. 27. In 2D case, with the intense load, HAMBAZI yields the lowest latency 75% of the time and maintains reasonably low location staleness compared to the baselines. In terms of system throughput, HAMBAZI is capable of finishing all calls while NETCODE, HAMSAZ and FIRESTORE have 0.2%, 0.2%, and 30.5% unfinished calls per minute respectively.



(a) Latency

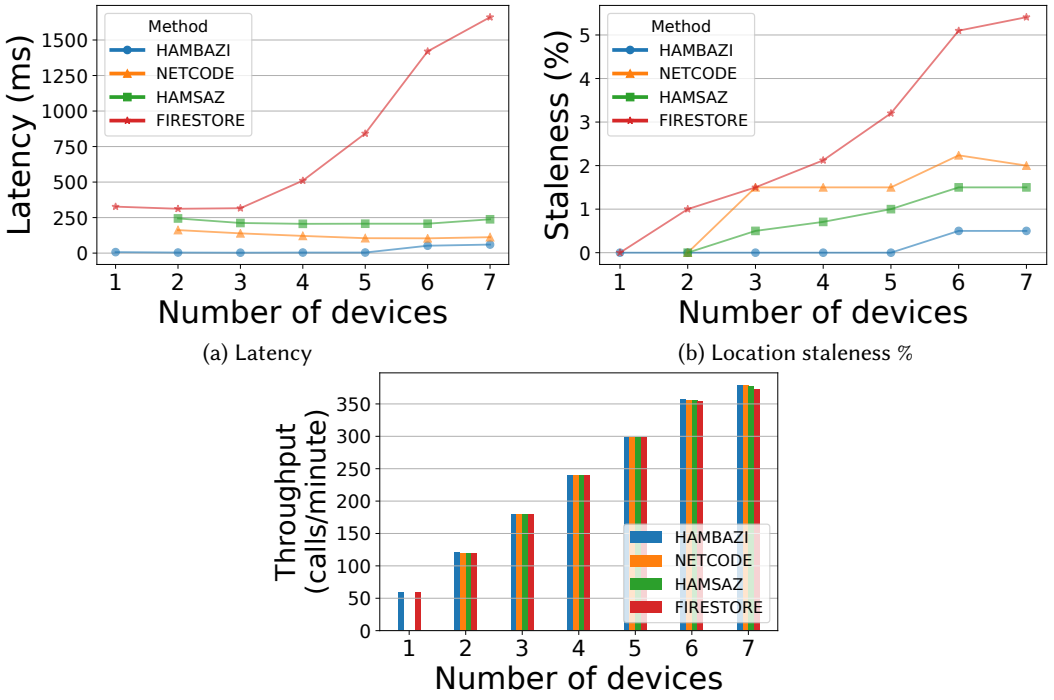


(b) Location staleness %



(c) System throughput

Fig. 28. With an increasing number of devices issuing requests, HAMBAZI still benefits from conflict-free actions and results in the lowest latency and staleness compared to baselines.

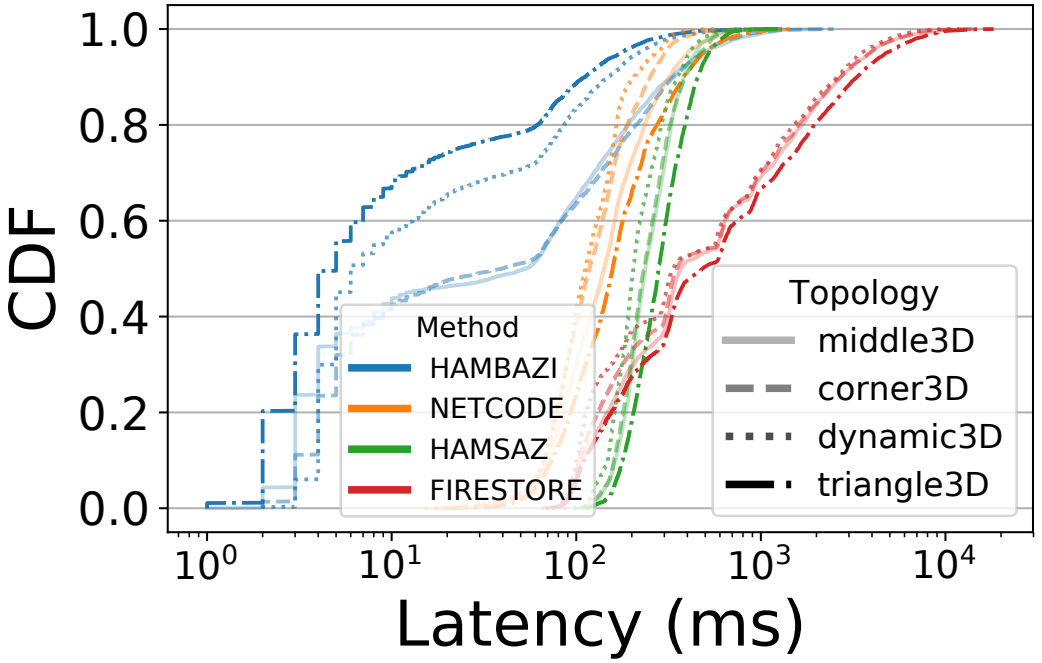


(a) Latency

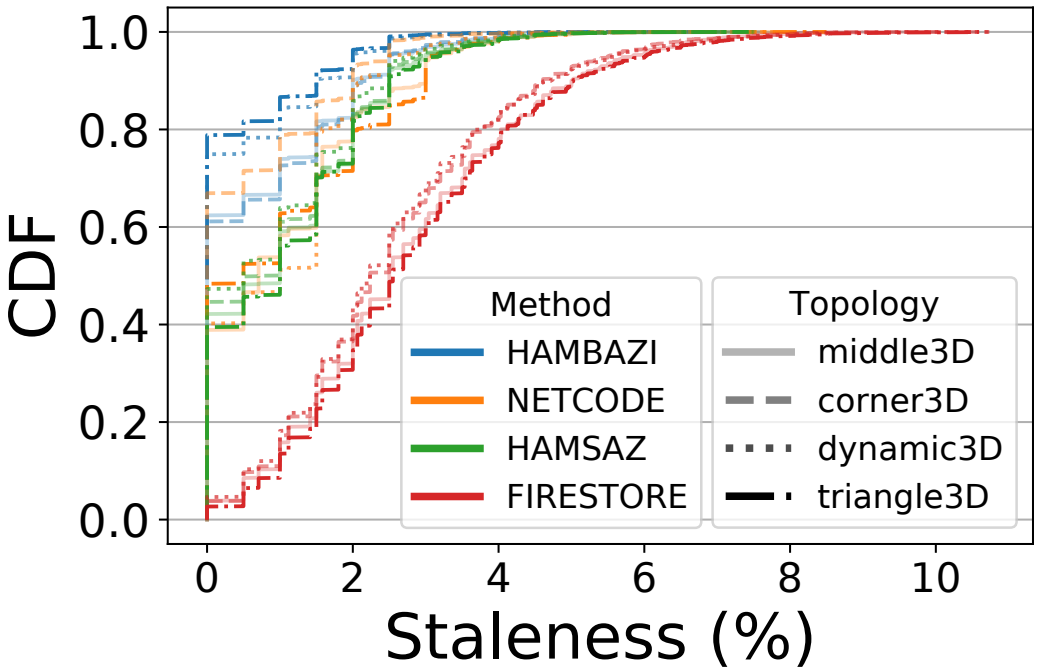
(b) Location staleness %

(c) System throughput

Fig. 29. In 2D case, HAMBAZI still benefits from conflict-free actions and results in the lowest latency and staleness.



(a) Latency



(b) Location staleness %

Fig. 30. HAMBAZI outperforms the baselines in terms of latency and location staleness.

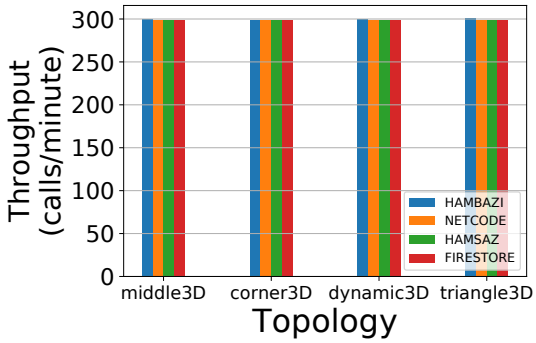
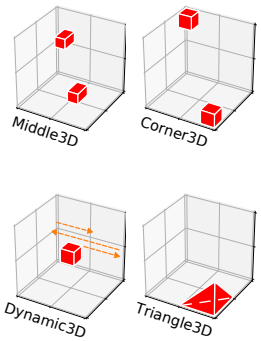
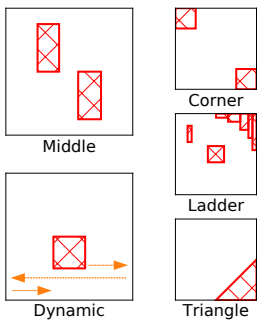


Fig. 31. Different topologies impact the amount of conflict-free actions, but the system can still handle all action calls.

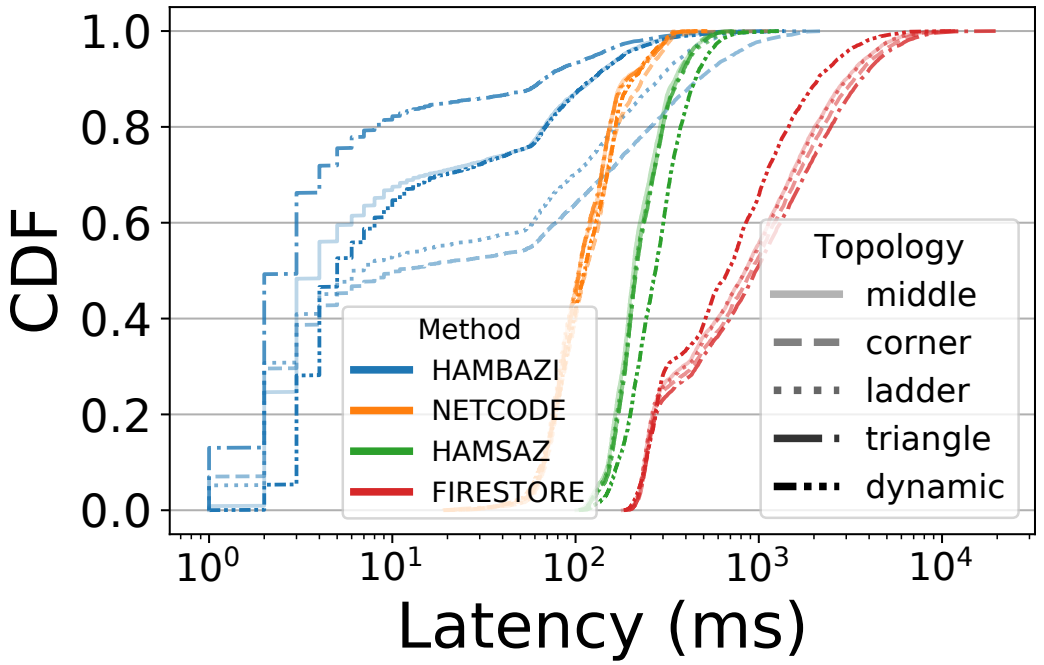


(a) Our default Middle3D topology and other 3D topologies.

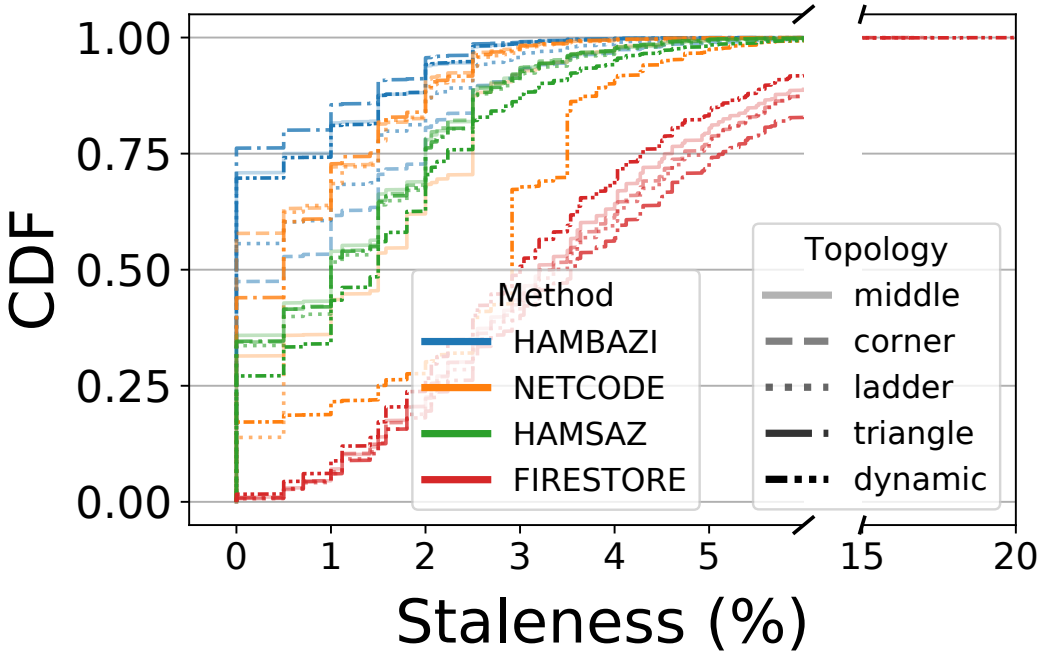


(b) Additional 2D topologies.

Fig. 32. Four 3D and five 2D AR game boards topologies are evaluated. With one dynamic topology in both 3D and 2D.



(a) Latency



(b) Location staleness %

Fig. 33. In 2D case, even with more challenging topologies (Corner, Ladder), HAMBAZI outperforms the baselines in terms of latency and location staleness.

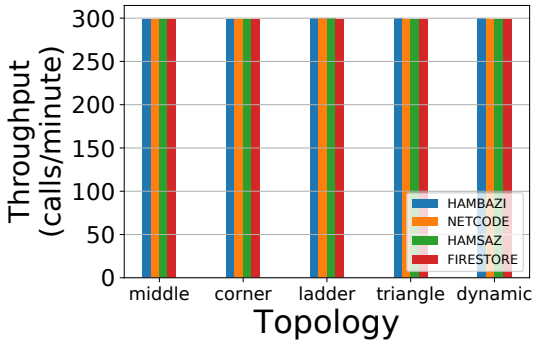
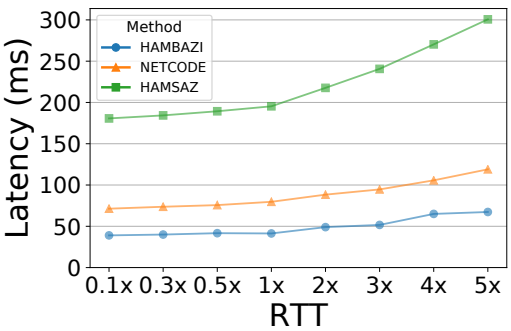
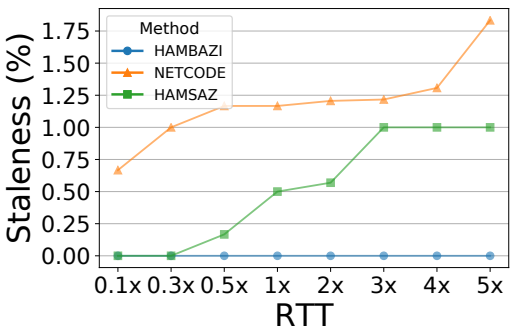


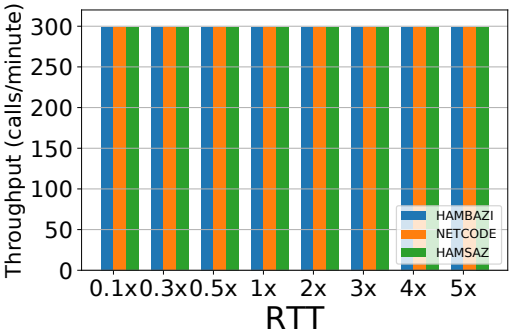
Fig. 34. In 2D case, the system can still handle all action calls.



(a) Latency

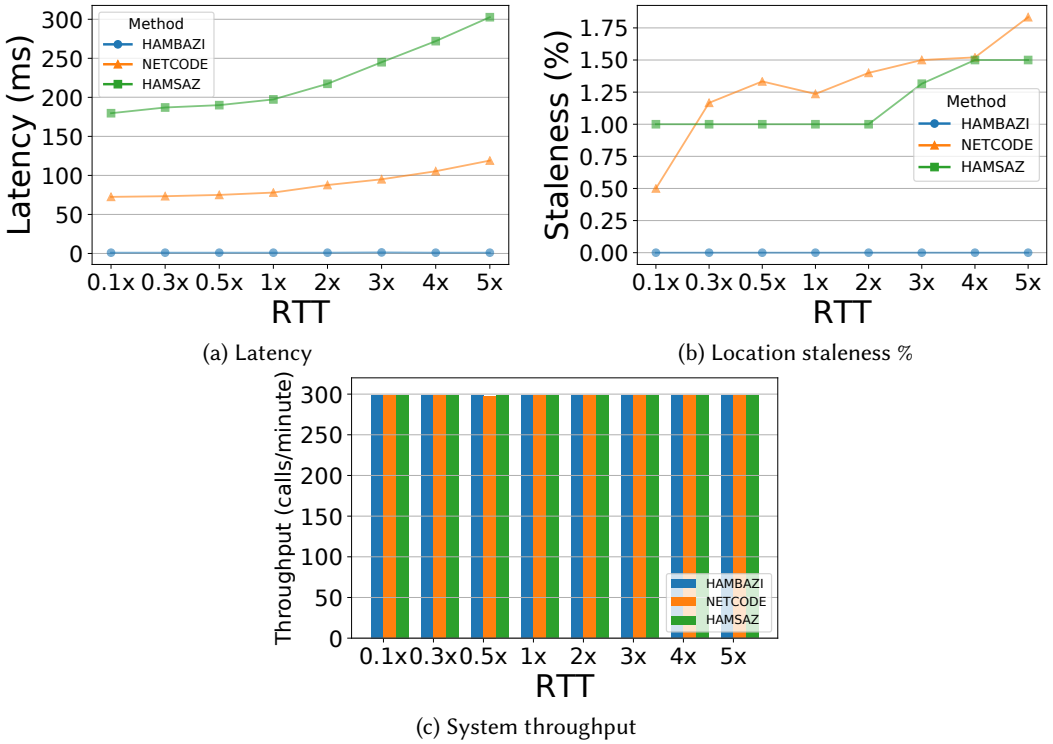


(b) Location staleness %



(c) System throughput

Fig. 35. Slower network impacts the NETCODE and HAMSAZ with increasing latency and location staleness, while HAMBAZI can still achieve a median latency of 67.3 ms and zero staleness when 5x RTT is applied.

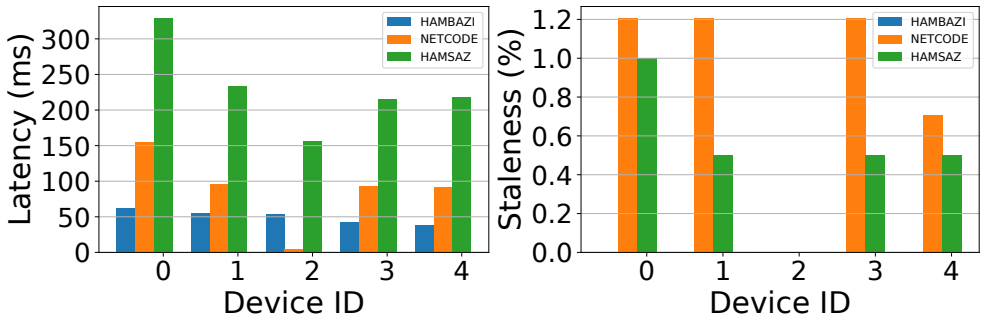


(a) Latency

(b) Location staleness %

(c) System throughput

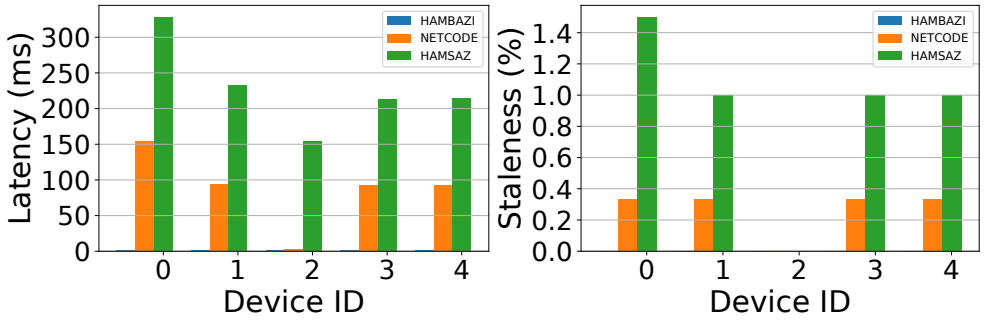
Fig. 36. In 2D case, HAMBAZI can still achieve a median latency of 1 ms and zero staleness when 5x RTT is applied.



(a) Device 0 has slower network.

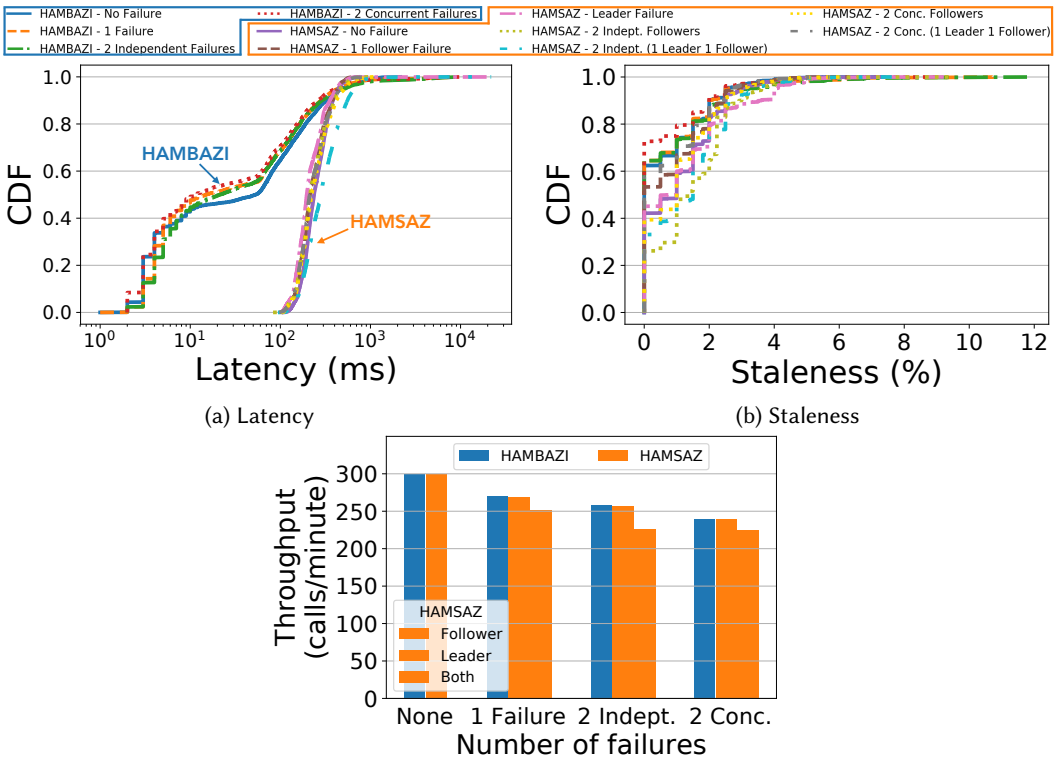
(b) Device 0 has slower network.

Fig. 37. Device 0 has slower networks. HAMBAZI is less impacted by network heterogeneity, while NETCODE and HAMSAZ have up to 1.6x and 1.5x of the median latency for Device 0 compared to other devices.



(a) Device 0 has slower network. (b) Device 0 has slower network.

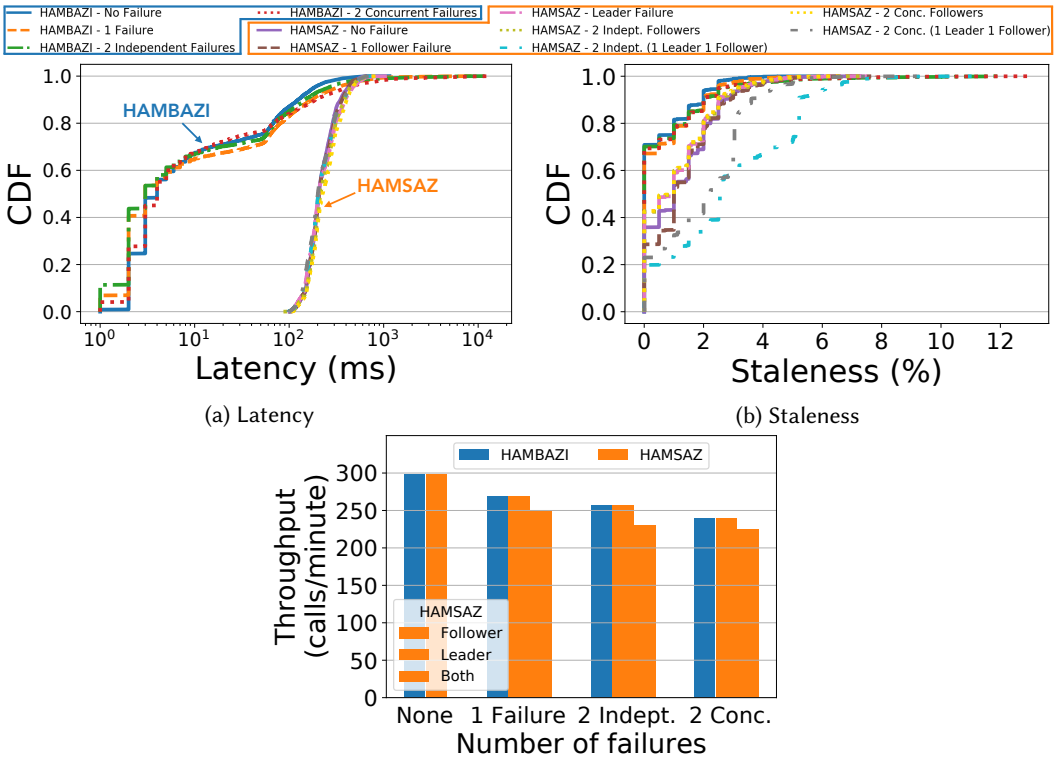
Fig. 38. In 2D case, Device 0 has slower networks. HAMBAZI is less impacted by network heterogeneity with a median latency of 1 ms to all devices, while NETCODE and HAMSAZ have up to 1.6x and 1.5x of the median latency for Device 0 compared to other devices.



(a) Latency (b) Staleness

(c) Throughput

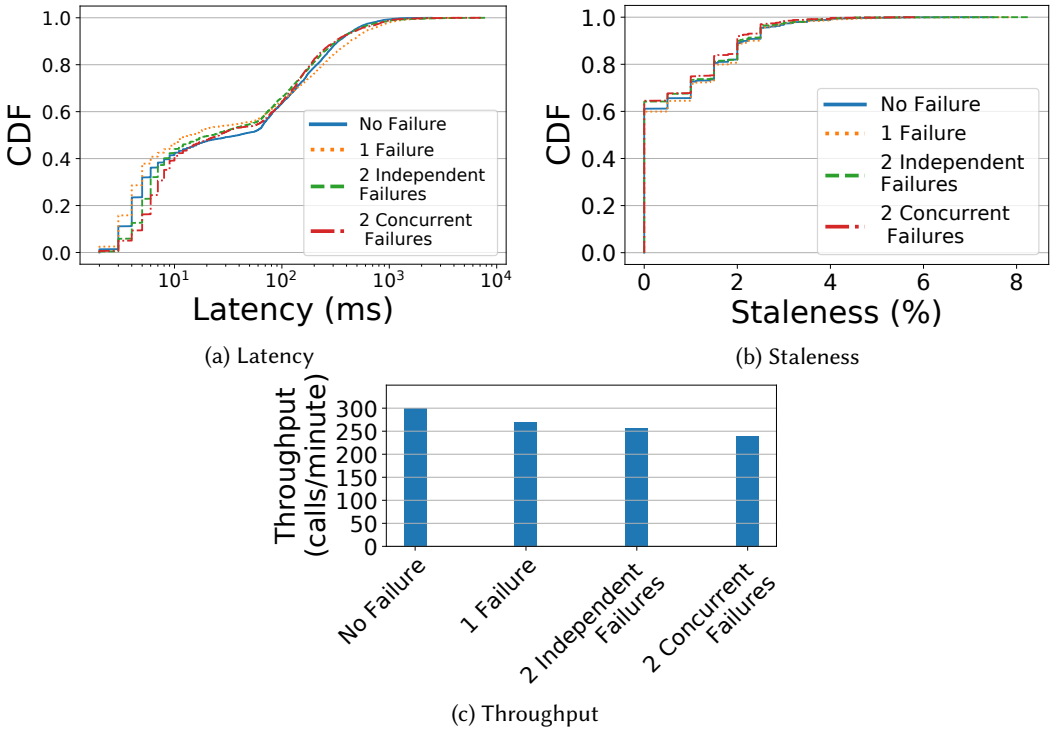
Fig. 39. Latency, staleness, and throughput comparison when failures are present in Middle3D topology. HAMBAZI can continue the operations even when failures are present. The overall latency is not affected, with slightly long tails depending on the pre-configured recovery wait time. Throughput decreases as the failed devices can not contribute calls to the system.



(a) Latency (b) Staleness

(c) Throughput

Fig. 40. In 2D case, Latency, staleness, and throughput comparison when failures are present in Middle topology. HAMBAZI can continue the operations even when failures are present. Throughput decreases as the failed devices can not contribute calls to the system.

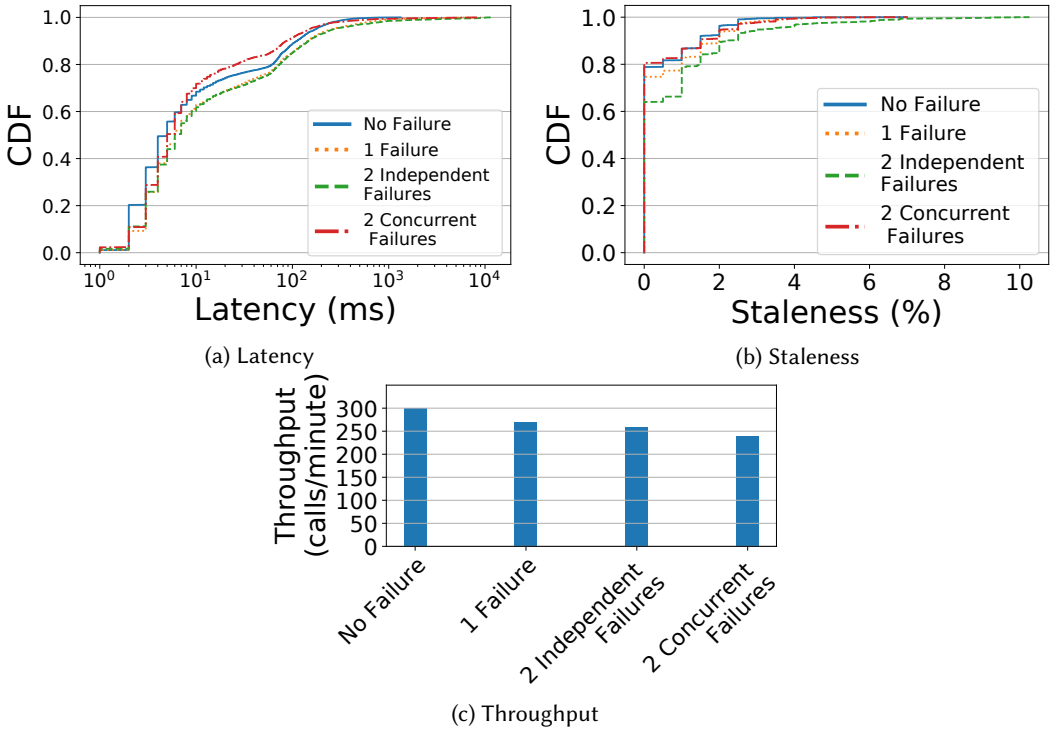


(a) Latency

(b) Staleness

(c) Throughput

Fig. 41. Latency, staleness, and throughput comparison when failures are present in the "Corner3D" topology.

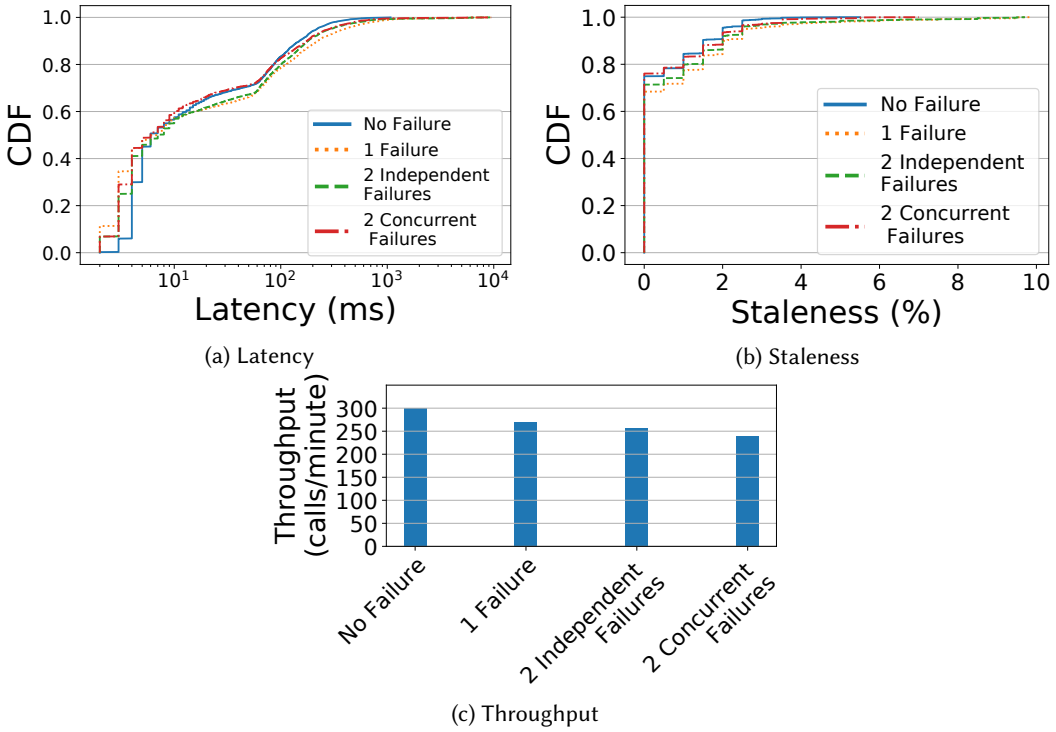


(a) Latency

(b) Staleness

(c) Throughput

Fig. 42. Latency, staleness, and throughput comparison when failures are present in the "Triangle3D" topology.

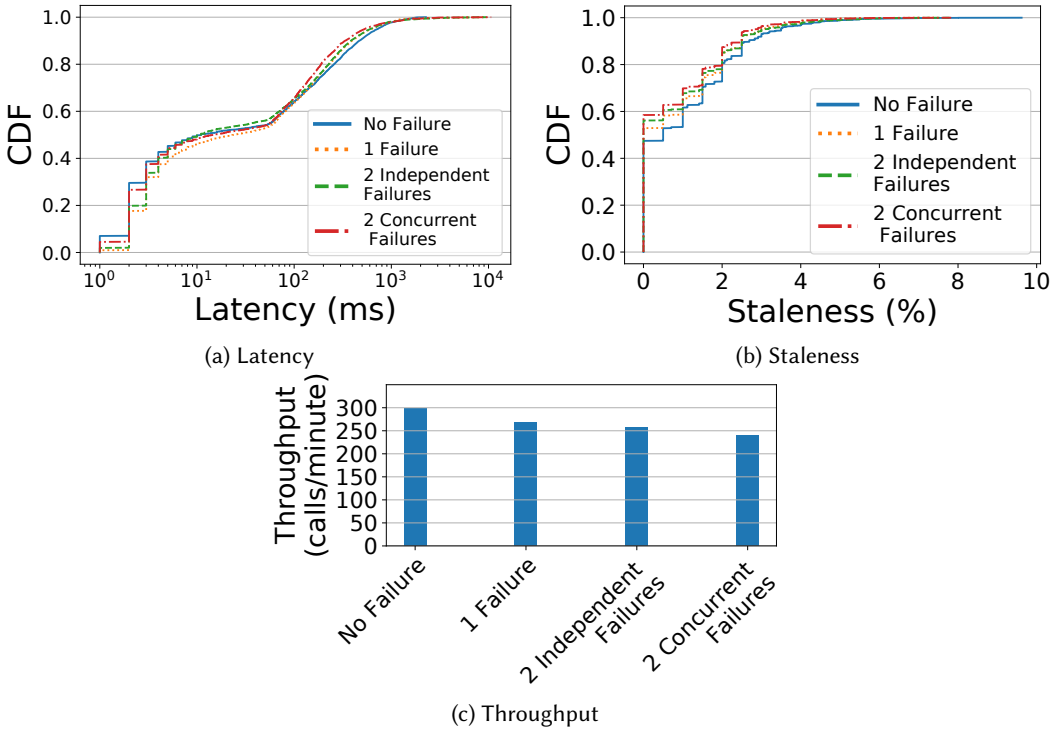


(a) Latency

(b) Staleness

(c) Throughput

Fig. 43. Latency, staleness, and throughput comparison when failures are present in the "Dynamic3D" topology.

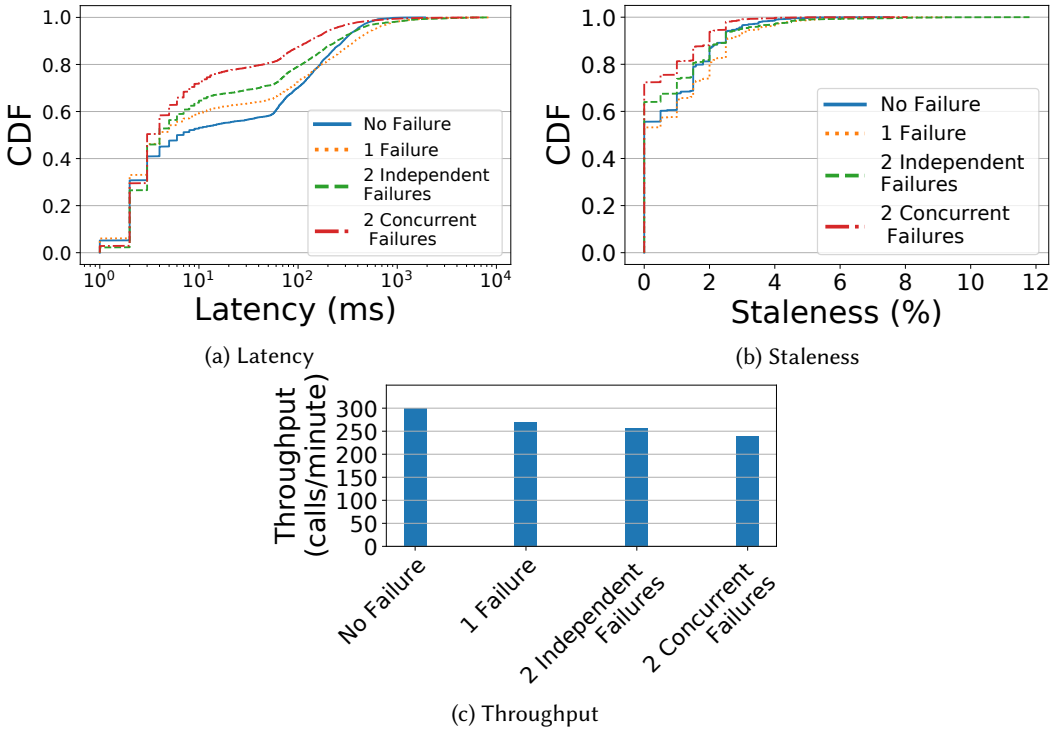


(a) Latency

(b) Staleness

(c) Throughput

Fig. 44. Latency, staleness, and throughput comparison when failures are present in the "Corner" topology.

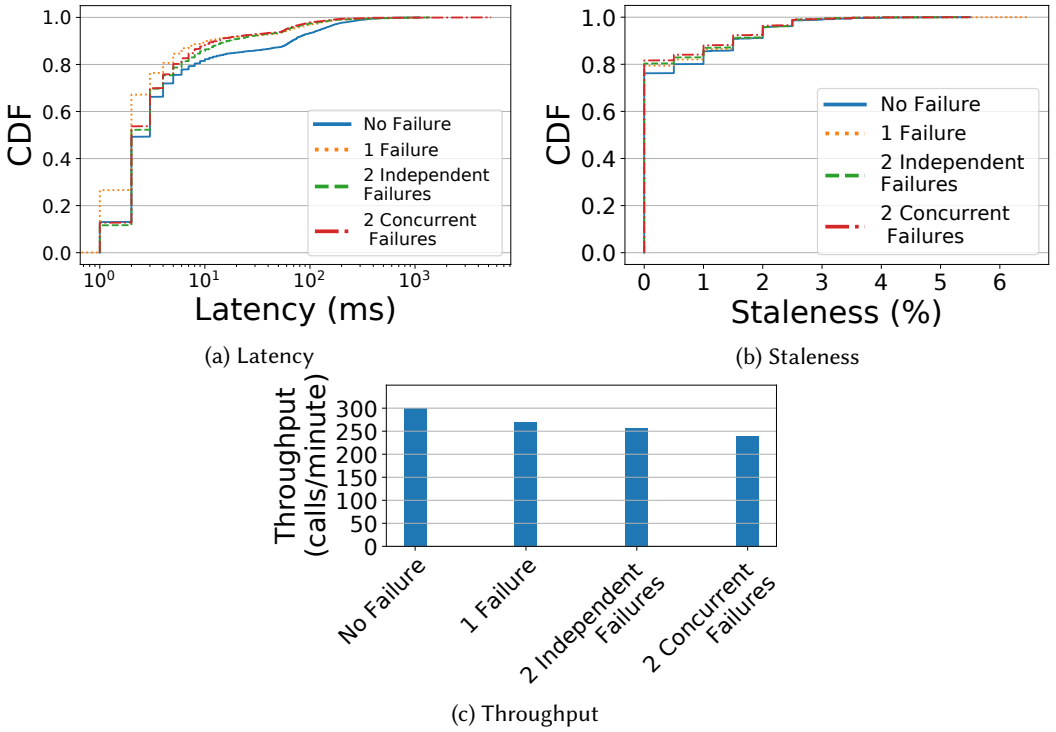


(a) Latency

(b) Staleness

(c) Throughput

Fig. 45. Latency, staleness, and throughput comparison when failures are present in the "Ladder" topology.

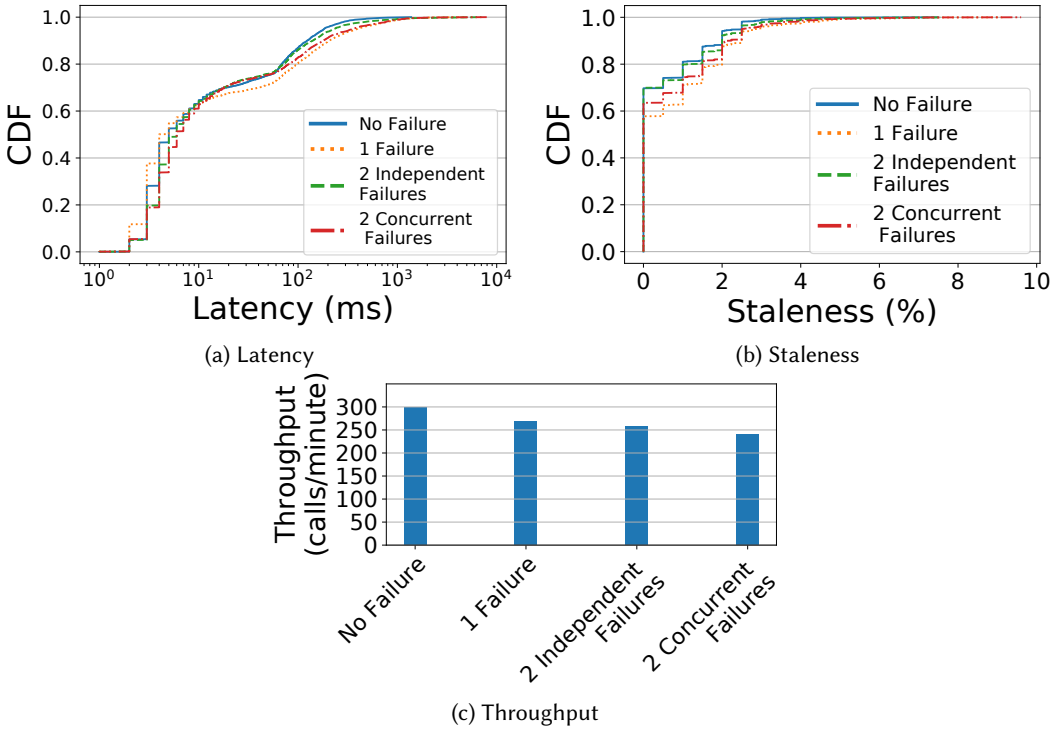


(a) Latency

(b) Staleness

(c) Throughput

Fig. 46. Latency, staleness, and throughput comparison when failures are present in the "Triangle" topology.



(a) Latency

(b) Staleness

(c) Throughput

Fig. 47. Latency, staleness, and throughput comparison when failures are present in the "Dynamic" topology.