# Read Freshness in an Externally Updated Singleton Directory

Leena Varik-Mendes
Department of Computer Science, Ravelin Institute of Technology
Tartu, Estonia
l.varik-mendes@ravelin-institute.example

Chidi S. Nwafor-Dalen
Distributed Infrastructures Group, Okpara Technical College
Enugu, Nigeria

Júlia Serra Belmonte
Department of Computer Science, Ravelin Institute of Technology
Tartu, Estonia

capstract:2611.00008v1 [cs.DC] 4 Nov 2026

## Abstract

A locator can return the newest value in its store while directing a client to the wrong site. We study a single-item directory in which a relocation takes effect outside the store and reaches a distinguished ingress after an independent reporting delay. The directory has three replicas and disseminates updates on staggered schedules. We derive the probability that a completed read is behind the physical version, conditional on the latest time at which an ingress update could have reached the responding replicas. In 80 eight-hour discrete-event runs, the two-reply policy reduces the wrong-site rate from 12.95% under a home-replica read to 11.28%, with a paired difference of −1.67 percentage points (95% interval −1.97 to −1.38). Removing replica batching reduces the two-reply rate by 0.52 points. Removing reporting delay reduces it by 10.72 points. A comparison with the ingress register alone misses most of the errors that would send a client to the wrong site. We give version and location scores separately since a site may become correct again before its record catches up.

## CCS Concepts

• Computer systems organization → Distributed architectures; • Information systems → Data replication tools.

## Keywords

replication, bounded staleness, directory services, discrete-event replay, age of information

**Reference Format:**
Leena Varik-Mendes, Chidi S. Nwafor-Dalen, and Júlia Serra Belmonte. 2026. Read Freshness in an Externally Updated Singleton Directory. Preprint, November 4, 2026.

## 1 Introduction

A read can be current relative to a write history and stale relative to the thing described by that history. The distinction is usually harmless for data created by the store. It matters for a location directory. A client may act on a location as soon as a read completes, while a move can take effect before anybody submits an update. In that interval all serving replicas can agree and the returned location can still be unusable. Counting inconsistent replies between replicas overlooks the interval completely.

We study the distinction for a singleton with a finite set of sites and one reporting channel. A completed move changes the physical location immediately. Its report takes some time to reach the directory and may arrive after the report of a later move. After ingress, two follower replicas copy the current record on staggered schedules. A read can consult its home replica, wait for two of three replicas, or contact ingress directly. Each option returns an ordinary positive response with a site number or fails at the deadline. We score both the response and what a client could do with it.

Each read consults the same key and is scored against both the accepted and the physical version at completion (Section 4.2). Policies share the movement, reporting and request histories. A move that takes place during a read can change the target version for a slower read of the same request.

We use a discrete-event model because the observation layer and directory layer must be varied independently. A study confined to the date stamps in the directory could estimate the dissemination term but would have no events for reports that had yet to arrive. We retain the physical event history in the evaluator and keep it out of the replica state. The main comparison measures how much reading a second copy changes the fraction of successful responses that would send somebody to an empty site. A zero-report-delay run and a zero-copy-delay run use the same ledgers and provide the two counterfactuals.

## 2 Related Work

### 2.1 Replicated registers

Logical ordering distinguishes causally related events from concurrent events [12]. We can order our updates with a single increasing counter because the physical object gives a serial order and the observer hands reports a counter in that order. Arrival order at the database can differ. The distinction matters at a return to an earlier site. The new event has a new identifier, even though the directory may eventually carry a previously used site number.

Linearizability supplies a point between invocation and response at which each operation can be considered to take effect

[10]. Atomic register algorithms [2] and intersecting read and write quorums [7] relate a read to completed writes through particular update and read protocols. A read that waits for two copies in our experiment has no such intersection guarantee. A report is installed at ingress before either follower needs to see it. Even an atomic register at ingress would order operations on reports, with the same delay before a move becomes a report.

Dynamo uses configurable read and write replica counts [5]. Probabilistically bounded staleness relates the number of versions or time missed to those counts and to dissemination delays [3]. Continuous consistency measures divergence between copies against application requirements [19]. We keep version divergence as a diagnostic and add a second endpoint with the physical ledger as reference. A monotonic read guarantee [17] could protect a reader from going back to a previously discarded version. It would still permit a reader to receive a location that has been superseded by an unreported move. We leave per-client session state out of the comparison so that a recording error on one read cannot affect the routing of the next.

## 2.2 Status age and directories

The age-of-information literature distinguishes the delivery time of a status update from its generation time [11, 18]. Our scoring function differs from age. In an idle period a record can be old in seconds and still name the right site. A fresh record can name the wrong site shortly after the next relocation. With frequent revisits, a record can also have the right value for the wrong version. Plaza-Verran and Makonnen-Tate [15] describes this last effect in an occupancy directory where a desk goes vacant and becomes occupied between polls. Our recurrence calculation in Section 4 uses the actual number of moves skipped and a six-state transition kernel.

Site registers are sometimes created as a by-product of a checkout ledger. Aoki-Pellan and Groenveld [1] reports discrepancies between checkout and placement clocks when collection and return are recorded at different stations. Sulari and Diarra-Preston [16] reviews inventory mechanisms in rooms where equipment is shared without a person assigned to issue it. We take the location service apart at the reporting boundary and model the time after physical placement. That makes the effect of bypassing a delayed report measurable without changing how often the object moves.

# 3 System and Workload

## 3.1 Sites and movement

The model represents an office floor with six numbered surfaces that can hold its shared full-strip stapler. The site register stores one pair, a movement version and a destination. A move ends at the first placement on the destination surface. The model assigns zero transit time to moves between named sites. The subsequent stapling and return are separate operations, with a return modeled as a move in its own right. Removing the item from the floor would require an explicit additional state. There is no such state in the present workload. The registry carries no information about staples remaining in the magazine.

Site 0 is the known initial position and appears at all three replicas at the start. Movers draw a destination uniformly from the other five sites. The replica IDs L0, L1 and L2 are independent of the surface numbering.

Each request comes from a client who has already decided to consult the directory. The client has no local sighting and treats a returned site as a destination.

## 3.2 Observation and ingestion

The move generates a report tagged with its physical sequence number. We use a logical observer at placement to assign that number and schedule delivery to ingress. For the reference workload, the delivering step has a median delay of 75 seconds and a log-scale standard deviation of 0.9. This combines the time to record the placement and submit the resulting observation. We make no attempt to estimate those two components separately. We use 75 seconds as a scenario parameter in the manual reporting range considered by Sulari and Diarra-Preston [16].

Reports that describe different physical moves draw independent delays. This permits overtaking, especially when the office makes several moves close together. The rule at ingress is a max-register update on the version. A delayed report cannot undo a newer ingress value. The current physical version may remain absent though a previous report for the same site is already present. Assigning the sequence number at generation gives the directory an ordering primitive without giving it an early copy of the destination. The model needs one observer sequence for audit. Independent station clocks with uncertain order would need a different observation model.

## 3.3 Replication and read policies

Replica L0 acts as ingress. L1 and L2 pull its latest value at phases one-third and two-thirds through each 60-second period, respectively. A pull installs the returned value immediately. There is no modeled replication transfer time at the batch boundary. Each replica keeps one record between pulls. Staggering matters because a two-reply read that misses ingress may still consult the follower with the later pull. Keeping every version would yield the same answers under the max-version selection rule.

A client has one home replica, chosen uniformly per request. A probe to home has a median round-trip time of 25 ms. Any other probe has a median of 60 ms. Both are lognormal with log-scale standard deviation 0.55. The reachable server samples its register halfway through the simulated round trip. There is no queue. Sending two additional probes does not increase the service time of the first probe, so the comparison excludes load feedback. The setup isolates endpoint choice and order-statistic effects at the modeled rate. Real fan-out changes queue occupancy under load [4].

**One reply** sends to home and returns its value. **Two replies** sends to all three and returns the record with the largest version among the first two responses. A third response never changes the result, and there is no read repair. **Ingress** sends only to L0, including from clients whose home is L0. It is an operational reference for the freshest possible replica state without follower dis-

semination. A 500 ms deadline applies to every policy. One reachable follower is enough for some One reply requests and too few for Two replies. Conversely, Two replies can complete without using home. The choice of policy therefore changes the response population as well as the answer for a particular request.

We use admission failures to study those differences without inducing stale writes. In hour $h$, numbered from zero, replica $h \bmod 3$ rejects reads initiated during seconds 600 to 720. During seconds 3,000 to 3,045, it and the next replica modulo three reject reads. Endpoints that reject a request send no response. A request admitted before a window still receives its scheduled response, subject to the deadline. Replication and ingress continue during both windows. The windows represent unavailable read endpoints during maintenance, with a separate write plane. A communication failure that isolates the whole node would affect observation and dissemination too. The rotation prevents an arbitrary selection of L0 as the first down endpoint from controlling the reported policy ordering.

**Table 1: Reference workload. Runs share these parameters and use independent seeds.**

| Parameter | Value |
|---|---|
| Independent runs | 80 |
| Workload hours per run | 8 |
| Sites | 6 |
| Replicas | 3 |
| Moves, expected per hour | 4 |
| Reads, expected per hour | 12 |
| Median reporting delay | 75 s |
| Follower copy period | 60 s |
| Deadline | 500 ms |

# 4 Model and Scoring

## 4.1 A cutoff at ingress

Let $S_v$ and $D_v$ be placement time and reporting delay of version $v$. Replica L0 can use version $v$ from time $S_v + D_v$. From a follower sample at time $s$, the latest eligible time on the ingress clock is the most recent copy tick at or before $s$. Call this time $g_j(s)$ for replica $j$. Ingress has $g_0(s) = s$. Before the first follower tick its cutoff is zero. For a response that uses a set $J$ of probes with sample times $s_j$ and completes at $C$, set

$$G = \max_{j \in J} g_j(s_j), \qquad B = C - G. \tag{1}$$

The max-version read returns the largest version whose report arrived by $G$. This follows because each replica's eligible set contains just those reports that arrived before its cutoff, together with the initial record. The union of the eligible sets equals the eligible set at the largest cutoff. The argument allows reports to be reordered on arrival. It also allows a replica to skip versions when it pulls, since the record at the next tick is the maximum over the eligible set.

The physical version is the largest $v$ with $S_v \le C$. When at least one move has occurred, the read returns that version if $S_v + D_v \le G$. For a stationary Poisson relocation process of rate $\lambda$, the backward recurrence time from $C$ to the most recent move is exponential. Network scheduling, which generates $B$, is independent of relocation and reporting. The most recent move's reporting delay $D$ is independent of its age under the mark distribution. Conditional on a reply and its cutoff lag $B = b$, version staleness is

$$p_{\mathrm{stale}}(b) = 1 - e^{-\lambda b}\, \mathbb{E}[e^{-\lambda D}]. \tag{2}$$

The $\lambda$ in Eq. (2) is in reciprocal seconds if $b$ and $D$ are in seconds. The multiplier $e^{-\lambda b}$ identifies the part controlled by dissemination and read selection. Even with $b = 0$, the probability of staleness is positive when moves take time to report. With the run initialized to a known position, the probability at fixed $C$ and $B = b$ replaces $\lambda(D + b)$ in the exponential by $\lambda \min(D + b, C)$, excluding the mass at the initial version. That finite-start expression applies before the eight-hour boundary. The calculation in Section 6 uses the event history itself, including the boundary convention in Appendix A.

In the primary workload $\lambda = 1/900\ \mathrm{s}^{-1}$ and $\mathbb{E}D = 75 e^{0.9^2/2} \approx 112.45$ s. A typical one-way sample delay of tens of milliseconds is small against that mean. Even a 60-second pull interval contributes at most 60 seconds to a follower's sampling lag. Substituting the mean delay inside the exponential gives an upper bound on the probability of version staleness, by Jensen's inequality. With immediate dissemination and negligible response latency the bound is 11.7%. It is loose because some of the reporting delays are quite long. A change in the mean alone need not have the same effect as shifting the entire delay distribution by a constant.

## 4.2 Versions and usable locations

Let $V$ be the physical version at completion, $I$ be the highest version ingested by completion, and $R$ the returned version. Successful requests obey $R \le I \le V$. We count $R = I < V$ as an observation-only miss in version space, $R < I = V$ as a distribution-only miss, and $R < I < V$ as a miss in both layers. The distribution case includes a report received during the reply's return leg. These are descriptive categories for a read history. For a miss at both layers the case label supplies no causal split of that individual outcome between the layers. We use the paired intervention runs to measure changes in rates.

Denote the location written in report $v$ by $L_v$. The application endpoint is $L_R \neq L_V$. The score further down the path substitutes $L_{V(C+\tau)}$ for $L_V$, where $\tau$ is a fixed walking interval. It asks whether the indicated surface is unoccupied by this particular object at a given arrival time. It makes no claim about how long participants actually walk, or what they do once they find an empty surface. Each site change in the model chooses uniformly among the other five sites. Independence of routing from the version gap gives, for $k = V - R$,

$$\Pr(L_R \neq L_V \mid k) = \frac{5}{6}\left(1 - \left(-\frac{1}{5}\right)^{k}\right). \tag{3}$$

After one missed move the location is wrong with probability one. After two it is wrong with probability 0.8, since one of five allowed moves returns to the previous site. The exact identity in Eq. (3) follows from the eigenvalues of the transition matrix. The uniform stationary component has eigenvalue one and every direction orthogonal to it has eigenvalue $-1/5$. A real office may direct returns to a preferred home site, making the state transition matrix nonuniform. The resulting relation can be computed from powers of a transition matrix if destinations and delays are still independent. A return ritual that changes reporting delays would also change the conditioning.

## 5 Method

We generate four moves and 12 requests per hour in expectation from independent Poisson processes. The rates are low enough that task contention is irrelevant to the modeled directory, although the six locations used by one item change often enough to observe the periods before its new report reaches ingress. A high-load systems benchmark would estimate a different quantity. Our replay has 80 independent eight-hour runs. We use a separate random stream for move times, destinations, reporting variation, read times, and the read-side random variables. All policies see the same streams. The response time of any of the three probes is drawn even in policy arms that do not send it. Each variation in one parameter replays the same uniforms and draws, as specified in Appendix A.

The primary endpoints are deadline response and physical location error at completion. We report the unconditional probability of getting a positive response by a time budget and then score wrong site among positive responses only. The denominator for physical error is policy-specific. We check the paired difference on requests that both policies answer in Appendix B. In the principal table we also give the number of version misses and the result for a hypothetical 25-second walk. With no walk, returning an earlier record for the correct current site is a success on the physical endpoint and a miss on the version endpoint.

We form confidence intervals by resampling entire runs with replacement and recomputing each ratio of counts across the resample [6]. Runs are the independent units. This preserves correlations between request outcomes within a run and pairs observations across two policies for differences. We use 2,000 resamples with a fixed seed and give percentile intervals. These intervals quantify Monte Carlo variation for the specified model. There is no sampling frame of real offices. The load sweep is descriptive, and we do not test each cell against the reference condition.

## 6 Evaluation

### 6.1 Deadline and location

The run generated 2,525 moves and 7,621 requests. One reply answered 7,457 requests and Two replies answered 7,500. Another 7,456 requests were answered under Ingress. Thus Two replies provided an answer on 98.41% of requests, roughly 0.56 percentage points more than One reply, and had a higher median response latency. In the single-endpoint window, it can still use the other two replicas. During the two-endpoint window it times out. Ingress sometimes avoids that timeout and sometimes loses a read that two followers can answer.

**Table 2: Outcomes and response latencies. Site errors at completion and after a 25-second walk are separate scores of the same successful replies. Quantiles are conditional on a reply.**

| Policy | Replies | Timeouts | Old version | Wrong site | Wrong after walk | Median (ms) | p95 (ms) | p99 (ms) |
|---|---|---|---|---|---|---|---|---|
| One reply | 7,457 | 164 | 974 | 966 | 1,134 | 25.2 | 62.2 | 90.1 |
| Two replies | 7,500 | 121 | 854 | 846 | 1,019 | 47.8 | 95.7 | 128.1 |
| Ingress | 7,456 | 165 | 815 | 808 | 982 | 46.4 | 132.3 | 204.7 |

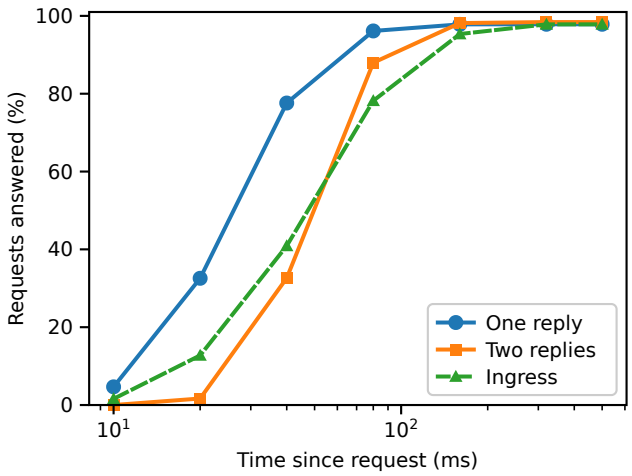



**Figure 1: Response probability by elapsed time, out of all 7,621 requests in each policy. The final point is the deadline. Points are evaluated at the shown times and the connecting lines guide the eye.**

In Figure 1, Ingress serves fewer responses during the first 40 ms than One reply, and more than Two replies. Later, the ordering between Ingress and Two replies reverses. This is possible because Two replies finishes on the second order statistic among three probes. Ingress can be faster when L0 is the home replica, but when L0 is remote, a long round trip must finish. The reference cost of the second reply is 22.6 ms difference in medians. That subtraction compares population medians and is not the median individual delay penalty.

Wrong-site fractions were 12.95% for One reply (95% interval 12.02% to 13.97%), 11.28% for Two replies (10.40% to 12.18%) and 10.84% for Ingress (10.00% to 11.73%). The paired difference of Two replies minus One reply was −1.67 percentage points (−1.97 to −1.38). To include timeouts, we also counted correct replies out of all requests. These were 6,491, 6,654 and 6,648 respectively, or 85.17%, 87.31% and 87.23% of all requests. Ingress had the lowest wrong-site rate among replies, while Two replies had the highest count of correct replies per request. These policies have different failure sets. In particular the conditional comparison admits requests for which only one policy could direct the client anywhere.

**Table 3: Version misses by layer at completion. The columns are mutually exclusive among replies. Counts sum to the old-version column of Table 2.**

| Policy | Observation only | Distribution only | Both |
|---|---|---|---|
| One reply | 812 | 155 | 7 |
| Two replies | 811 | 41 | 2 |
| Ingress | 815 | 0 | 0 |

For the Two replies policy, 811 of 854 version misses occur when the data path serves exactly the ingress version. Although the other 43 version misses include a delay attributable to distribution, only 42 of those produce a wrong destination. For each policy, seven or eight old records name the correct site. The small number is consistent with one missing version accounting for most misses. Eq. (3) rules out a coincident site when a single move goes unreported. We saw no distribution miss at Ingress in the reference draws. The timing convention allows one if a report arrives after L0 samples the record and before the client's response arrives.

The walk simulation changes the absolute failure level. For Two replies, 1,019 of 7,500 positive responses point to an empty surface at 25 seconds, against 846 at response time. The additional 173 is a net count. Some destinations change from wrong to right in the walking interval. Longer response times also let a new move take place during the read. For the fixed 25-second transit analysis, a client already at the destination is treated the same as one across the office. We use the interval to distinguish the lookup deadline from the time at which the answer is tested.

## 6.2 Reporting and dissemination

Figure 2 compares the three policies under a sweep of the median report delay. Changing the scale from 75 to zero seconds changes the two-reply wrong-site count from 846 to 42, a reduction of 10.72 percentage points among the same 7,500 replies. Batch copying and the response leg still contribute errors. Setting the copy period to zero while retaining the 75-second median leaves 807 errors, or 10.76%, for a reduction of 0.52 points. This run gives followers a copy of the ingress state instantaneously. At 15- and 30-second copy periods the wrong-site rates were 10.92% and 11.00%. The two zero-delay experiments change different parts of the same history and their effects need not add.

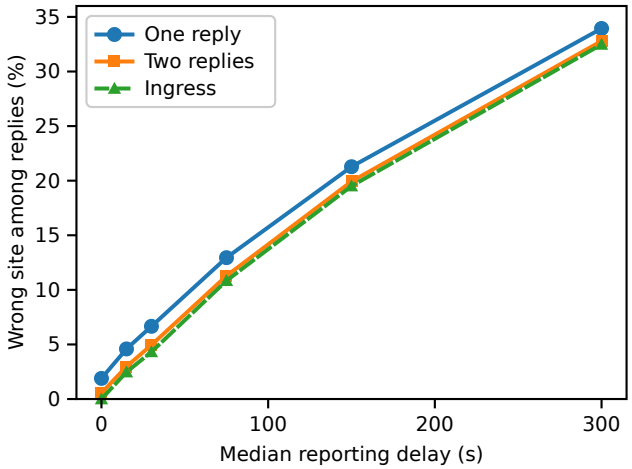



**Figure 2: Wrong-site rate among replies as the median movement reporting delay varies. A median of zero gives immediate reporting. Request and link schedules are held fixed across points.**

With reporting held at the reference median, increasing the move rate from four to eight per hour raises the two-reply wrong-site rate from 11.28% to 20.51%. At 16 moves per hour and a 300-second report median, the simulated rate is 62.41%. The version-based expression in Eq. (2) approaches one as the gap grows. The site-based endpoint saturates below one because the location chain can return to a previously reported site. Figure 3 shows how these effects change across the scenario grid. The zero-median column gives the error that remains when all moves enter the store immediately. Its values also rise as moves become more frequent. The copy schedule admits more new versions between ticks at higher relocation rates.

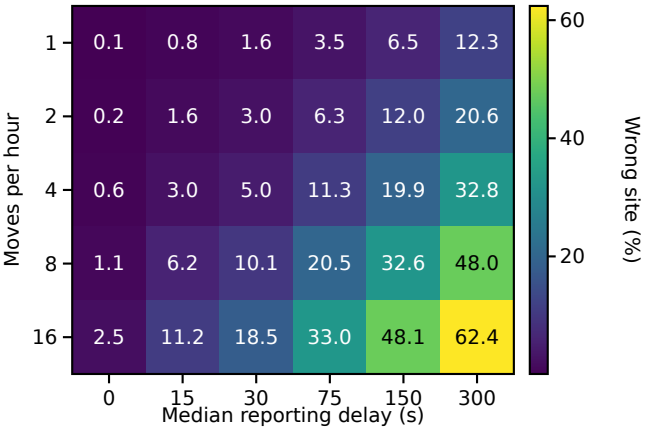



**Figure 3: Wrong-site rate (%) for Two replies across the workload grid. Rows vary the expected hourly move count. Columns vary the median reporting delay. The reference workload is the 4 moves/h, 75-second cell.**

A consistency-only service monitor would see most of the improvement from reducing the copy interval if it compared replies with the accepted ingress version. The destination monitor sees a change of less than one percentage point under immediate dissemination. Both monitors would be correctly implemented. They would be measuring different histories. The distinction is especially visible in runs with long reports because a late report gives all three replicas the same stale record for an extended interval. Selecting another copy cannot resolve that ambiguity.

## 7 Discussion

### 7.1 What can be certified before a read

A freshness certificate for this application must identify the stage at which somebody actually saw the stapler. A lease can constrain writers and provide cache consistency for data updates [9]. A lease on this directory cannot constrain a move that has already happened outside it. For a putative bound on physical freshness, a lease holder would need authority over release of the item or ongoing direct observations of its site. Either introduces an additional operational protocol. Waiting for the lease to expire without an observation only changes the site’s status from known-at-a-past-time to unknown.

A placement timestamp supports a prediction under a specified move model. A replication timestamp answers a different question. A follower may have copied its record ten seconds ago when the last placement happened yesterday.

Stronger replication schemes change which in-store guarantee is available. A Raft-based version history [14] can order accepted moves and elect a new leader, given its required connectivity and a suitably implemented read path. We selected a simple fixed ingress because the loss of write authority was outside the failure schedule. The trade between an availability guarantee and an atomic service during arbitrary partitions [8] concerns the accepted operations. It imposes no limit on observation lag. A leader election may improve access to reports already in the replicated log. It cannot recover a placement nobody has reported.

### 7.2 External validity and next measurements

The main workload assumes stationary independent moves, an exogenous read stream, and lossless reporting with finite delays. Busy periods may cause arrivals to bunch while suppressing reports. A requester may decide to use the directory only after checking the nearest surface, which conditions the workload on the absence of the stapler there. Such correlations could raise or lower the error for any single policy. A marked process fitted to observations would need to account for who was already looking for the object and what happened immediately before a move. A placement log alone is not a sample of those choices.

We use no return-to-home preference and no in-hand state. Some returned locations would count as currently occupied even if the current user were still operating the item. If the client needs to act, availability for pickup can lag behind physical presence at a site. Adding that state would affect the target of the read, not just the location transition probabilities. Montiel-Falk

and Juvet-Kamara [13] observes that occupancy and release records can be separated by several minutes even after a terminal has received the location update. Queueing to use the item would need a separate study.

The read-only failure model excludes a partition between placement terminals and ingress. With correlated read and write failures, the response sets themselves could be selected by delay in reporting. The mark distribution in Eq. (2) would then have to be conditional on the path through which a request succeeded. Also, the numerical latency quantiles depend on the synthetic round-trip model and omit reaction time, rendering time and the time to get up from a desk. The 25-second transit term is an accounting case for a follow-on measurement of staff time.

In a pilot deployment the first thing to log is a passive placement time that can be collected separately from a report timestamp. Otherwise, long periods of uninterrupted replica agreement are observationally equivalent to long periods without a new report. This becomes a practical issue when the object returns to a desk between copies. A protocol audit can correctly find that no out-of-order write survived and still have no evidence that anyone was sent to the right desk.

## 8 Conclusion

The applicable write boundary depends on what a lookup authorizes the reader to do. If it authorizes consulting the last filed report, then the accepted register supplies a complete history for evaluation. If it authorizes walking to a surface, the read must be validated against the object's trajectory over that interval. An operational change as small as leaving a stapler at the desk where it was last used moves the point of truth outside the storage system. Replicating the surface number does not replicate the underlying evidence.

## A Replay Specification

The simulator uses seconds internally. For run $i = 0, \dots, 79$, initialize five instances of Python 3.14's Random with integer seeds $7241 + 100i + j$, $j = 0, \dots, 4$. Use streams in this order for movement, destination, reporting marks, read instants and the read-side variables. Each run starts at time zero with version zero, site zero. Generate movement and read instants by summing exponential draws through the first event outside $[0, 28800)$. Use expovariate with the relevant rate divided by 3,600. Exclude the first event at or beyond the end of the run. Drain admitted reads past the boundary and leave the physical site fixed for the additional walk interval.

Enumerate movement events from one. For each event draw $m$ with randrange(5). If $m$ is no smaller than the previous site ID, use $m + 1$ as the new site. Sample $z$ with gauss(0,1) on the report stream and add $M \exp(0.9z)$ seconds to the move time, for median $M$. For a follower with phase fraction $\phi$, install at $P(\lceil a/P - \phi \rceil + \phi)$, where $a$ is ingress time and $P$ the period. Treat $P = 0$ as install time $a$ for both followers. Sort each installation stream by scheduled time and take a prefix maximum of versions. A lookup at exactly an installation time includes that installation. This per-report construction gives the same max-version history as pulling the latest version at each tick, including simultaneous installs.

For each read, draw the home with randrange(3). In replica ID order draw a standard normal mark using gauss(0,1) for each round trip $d_j$, with the median specialized to the chosen home. Discard probes to failed endpoints at admission or with a round trip above 0.5 seconds. Fetch the remaining probe versions at $t + d_j/2$ where $t$ is invocation time. In the Two replies policy choose the first two by the order $(d_j, j)$ if they exist. Use the largest returned version and the larger of the two delays. Resolve a timeout at 0.5 seconds. Calculate quantiles using the observation at rank $\lceil pn \rceil$ among $n$ positive round trips. The bootstrap uses one additional Random stream with seed 1724 and randrange(80) for each draw. Sort the 2,000 resulting estimates and use one-based ranks 50 and 1,950. Draw new bootstrap indices from the same initial seed for each statistic so that the pairs use identical replicate runs.

An implementation can check the simulation at three points. The baseline gets 2,525 placements and 7,621 requests before any filtering. The zero copy period agrees across all replicas at equal sample times. An asymmetry between two sampled copies in that run must come from sampling at different instants. At zero reporting delay, absent deadline failures, an ingress probe that samples at completion has zero version misses. We use that case to check ordering at coincident times.

## B Cutoff Bound and Matched Reads

The probability in Eq. (2) can also be written as $p_0 + (1 - p_0)(1 - e^{-\lambda b})$ for $p_0 = 1 - \mathbb{E}[\exp(-\lambda D)]$. The first term is the version error from unobserved movement. The second is the probability that the read cutoff misses a move that would be visible at completion. In stationarity, the second cannot exceed $1 - e^{-\lambda b}$. At four moves per hour, even a cutoff two minutes behind completion would have a bound of 12.48 percentage points before multiplying by $1 - p_0$. The bound compares version errors. To map to site errors one needs the gap distribution and a return kernel, since increasing the sampled version does not monotonically decrease site error in each realized trajectory.

One reply and Two replies both answered 7,408 requests. On these requests the counts of wrong replies were 953 and 837. The difference was −1.57 percentage points (95% paired interval −1.86 to −1.28). Scoring both replies at the later completion time gives the same counts in these draws. This restricted comparison excludes requests answered only when a second server is consulted.

## Acknowledgments

We thank M. Iversen-Calvo for checking the time conventions and É. Ndiaye-Lind for the first version of the placement ledger.

## References

[1] Reina Aoki-Pellan and R. Maarten Groenveld. 2021. Clock Placement in Unattended Equipment Checkout. In *Proceedings of the Workshop on Distributed Inventory Systems*. 18–27.

[2] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. 1995. Sharing Memory Robustly in Message-Passing Systems. *Journal of the ACM* 42 , 1 (1995), 124–142.

[3] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. 2012. Probabilistically Bounded Staleness for

Practical Partial Quorums. *Proceedings of the VLDB Endowment* 5 , 8 (2012), 776–787.

[4] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Communications of the ACM* 56 , 2 (2013), 74–80.

[5] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's Highly Available Key-value Store. In *Proceedings of the Twenty-First ACM Symposium on Operating Systems Principles*. 205–220.

[6] Bradley Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics* 7 , 1 (1979), 1–26.

[7] David K. Gifford. 1979. Weighted Voting for Replicated Data. In *Proceedings of the Seventh ACM Symposium on Operating Systems Principles*. 150–162.

[8] Seth Gilbert and Nancy Lynch. 2002. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News* 33 , 2 (2002), 51–59.

[9] Cary G. Gray and David R. Cheriton. 1989. Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. In *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*. 202–210.

[10] Maurice P. Herlihy and Jeannette M. Wing. 1990. Linearizability: A Correctness Condition for Concurrent Objects. *ACM Transactions on Programming Languages and Systems* 12 , 3 (1990), 463–492.

[11] Sanjit Kaul, Roy Yates, and Marco Gruteser. 2012. Real-Time Status: How Often Should One Update? In *Proceedings of IEEE INFOCOM*. 2731–2735.

[12] Leslie Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM* 21 , 7 (1978), 558–565.

[13] Camila Montiel-Falk and Anselme Juvet-Kamara. 2025. Availability Semantics for Shared Desktop Equipment. In *Proceedings of the Workshop on Distributed Inventory Systems*. 33–41.

[14] Diego Ongaro and John Ousterhout. 2014. In Search of an Understandable Consensus Algorithm. In *USENIX Annual Technical Conference*. 305–319.

[15] Alba Plaza-Verran and Sara Makonnen-Tate. 2022. Recurrent States in Low-Frequency Occupancy Registers. *Journal of Pervasive Infrastructure* 9 , 2 (2022), 81–98.

[16] Keiko Sulari and Fadima Diarra-Preston. 2024. Reporting Intervals in Small-Office Equipment Registers. Gannet Institute for Workplace Instrumentation.

[17] Douglas B. Terry, Alan J. Demers, Karin Petersen, Mike J. Spreitzer, Marvin M. Theimer, and Brent B. Welch. 1994. Session Guarantees for Weakly Consistent Replicated Data. In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*. 140–149.

[18] Roy D. Yates, Yin Sun, D. Richard Brown, Sanjit K. Kaul, Eytan Modiano, and Sennur Ulukus. 2021. Age of Information: An Introduction and Survey. *IEEE Journal on Selected Areas in Communications* 39 , 5 (2021), 1183–1210.

[19] Haifeng Yu and Amin Vahdat. 2000. Design and Evaluation of a Continuous Consistency Model for Replicated Services. In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation.*.