The 2026 Central-Outage Case Study on heain-core

When Its Own Change Stops the Whole System

Four Scenes from the Microsoft 365 Outage of 12–13 Nov 2026: a Wrong Change, Cascading Failure, Overwhelmed Recovery and Sites without a Centre

Jakkrit Boonma (จักรกฤษ บุญมา)

Independent Researcher · ORCID: 0009-0000-7165-3787 · jaggid.boonma@yahoo.co.th

Date: 14 November 2026

Implementation: heain-core (commit ae90f0f, core equal to v1.3.0-t7), HP Z840 Workstation

Repository: github.com/JakkritB/heain-core (casestudies/outage/)

Abstract

On 12–13 November 2026 Microsoft 365 went down worldwide for at least about 18 hours. Five Thai banks sent customers to other channels; and Taiwanese brokers confirmed orders by fax [1]–[6]. Microsoft attributed it to "an issue during an earlier network capacity upgrade", not an attack. Lost servers pushed work onto the rest until they could not cope, and recovered servers failed again once they took requests [1][4].

On an HP Z840 the author built a 5-machine lab serving one organization's synthetic mail (a Master, three mail-sending Workers, a user machine) with no internet, and tested four scenes: M1 an honest admin's wrong change; M2 Workers dropping out one by one; M3 the Master returning to 300 and 1,000 clients at once; M4 no Master and no internet for 30 minutes. Core's code was checked first, and every scene prints what happened in the real incident (asis) beside heain's result. This paper does not simulate Microsoft's systems.

On the Z840 (commit ae90f0f) all 38 checks passed with eight findings (O1–O8). What held: Workers never took more work than they asked for, no mail was sent twice even when a machine was stopped mid-job, the Master did not fall over when 1,000 clients reconnected at once, and sites worked on for 30 minutes without a centre. What did not: one setting at the Master stopped the whole service within about 6 seconds with no canary or automatic rollback; the Master accepted work without bound; and after the storm it dispatched more slowly as its queue grew, from 3.6 jobs per second at 9,300 queued to 0 at 56,046, closest to the real "recovered but overwhelmed". Every finding needs a core change and awaits the author's decision.

Keywords: *HEAIN Framework, heain-core, Outage, Cascading Failure, Configuration Rollout, Canary, Backpressure, Retry Storm, Thundering Herd, Load Shedding, Offline Operation, Microsoft 365*

1. Introduction

1.1 Background

The first cyber case study [11] asked how far damage spreads when one valid credential falls into an attacker's hands. This case asks something different: if there is no attacker at all, and the one in error is an honest admin following procedure, does the centre everyone relies on take the whole system down with it? And when the centre returns, is the system really back to normal?

The question came from the global Microsoft 365 outage of 12–13 November 2026. The author had it made a new case study separate from the cyber case, with all four scenes M1–M4, core's code checked before design, and no waiting for Microsoft's full root-cause report [10].

1.2 The Incident Behind the Case



Figure 1 — Timeline of the Microsoft 365 outage of 12–13 Nov 2026 as Microsoft and the press reported it [1]–[6]

Topic	What was reported	Source
Time	Began about 16:00 UTC on 12 Nov (about 23:00 Thai time); Microsoft announced broad recovery at 17:18 GMT on 13 Nov; at least about 18 hours in all	[1][4]
Services	Exchange Online and everything relying on it: Outlook, calendars and Teams meetings, and mail retrieval in Copilot; desktop Outlook could still open mail already downloaded	[1][3]
Scale	Reports from 74 countries within about 1 hour; more than 130,000 Downdetector reports	[1][5]
Mechanism	Some servers became unavailable, so work flowed to the rest until congested; the secure-connection process "becomes overwhelmed"	[1][4]

Topic	What was reported	Source
Cause	"introduced by an issue during an earlier network capacity upgrade"; "not related to cyber or AI"; full report not yet released	[4]
Recovery	Network configuration change reverted, traffic rerouted, reboots and patches; recovered servers failed again on taking requests, so connections were limited and load raised gradually	[1][3]
Thailand	5 banks directed customers to apps, branches or call centres; hospital appointments by email delayed; the SET affected only for incoming email	[3][6]
Abroad	Taiwanese brokers confirmed orders by fax; several companies delayed results announcements; government agencies in several countries affected	[2][5]
Same year	22 Jan 2026 M365 North America down nearly 10 hours; 23 Jul 2026 an automated maintenance system removed too many IP routes, taking West US down about 5 hours	[7][8]

Table 1 — What was reported about the incident (sources checked 14 Nov 2026)

1.3 Status of the Incident Information

This section relates the incident as Microsoft stated it and as the press reported it. There is no full root-cause report yet; Microsoft's explanation changed during the incident and may change again when the report appears. This paper does not test or simulate Microsoft's products and does not claim that heain would have prevented that incident. The lab uses a synthetic mail service, where "mail" is just the name of a service the organization relies on; no real email is sent.

1.4 Research Questions

1. **RQ1** How fast does an honest admin's wrong change spread, and does anything in the system stop or roll it back by itself? (M1)
2. **RQ2** When working machines drop out one by one, are the remaining ones pushed past capacity until they fail in turn? (M2)
3. **RQ3** When the centre returns and everyone reconnects at once, does it survive, and how fast is it back to normal? (M3)
4. **RQ4** With no centre and no internet, what can sites still do, and what stops with a stated reason? (M4)
5. **RQ5** What must change in core?

1.5 Scope

- **No attacker:** every change goes through the correct procedure; the one in error is an honest admin.
- **Microsoft not simulated:** the lab is one fictional organization using heain for its own service.

- **Load on one machine:** every container is on the HP Z840; load figures are for comparing before and after fixes, not the capacity of real machines.
- **Core untouched:** this round builds only the lab, tools and scenes; core changes await the author's decision on the findings.
- **This deployment is the author's hypothetical design,** not any organization's system, and not reviewed by any agency.

2. Simulation Principles and Scene Mapping

Every scene has the same frame as the cyber case: real incident behind it → what is simulated → asis → heain → pass criteria → what is measured. Pass criteria are checked automatically (PASS/FAIL), measurements are printed as MEASURE, and things that are not the desired behavior but are not script errors are recorded as FINDINGs for the author to decide. Table 2 shows the pattern in the real incident that each scene addresses.

#	Pattern in the real incident	Question for heain	Scene
P1	One change took effect in many regions at once	Does a configuration change reach every node at once? Is there a canary or automatic rollback?	M1
P2	Some servers lost; work flowed to the rest until they failed in turn	Is there a ceiling on work moved to the remaining machines?	M2
P3	Recovered servers failed again when pending requests arrived together	Can a Master returning after a long outage take every client at once?	M3
P4	Data already on the machine stayed usable (desktop Outlook)	How far can a Worker carry on with what it already has?	M4
P5	Organizations need channels that do not depend on one provider	Which work truly needs the centre or the internet?	M4

Table 2 — Patterns in the real incident, questions for heain, and scenes

3. Method

3.1 Code Checked before Design

The author had core's code (9808fa7) checked on three unknowns before the scenes were designed. Table 3 gives the result, which set the expectation for each scene.

Topic	What was found	Expectation
How config reaches every node	Each node pulls its parent's values every config.sync_interval (10 s); no canary, no health criteria; commit confirmed rolls back only when the admin does not confirm	M1 finds a gap
Request ceilings on reconnection	A per-zone journal reconcile ceiling (429), WorkGate on low disk (507) and 503 during Master start-up (C20); no ceiling found for ordinary requests	M3 must measure
How work flows to the rest	Dispatch (P2) chooses by real capacity and uses leases; a lost machine's work is redispatched when its lease expires	M2 likely passes

Table 3 — Code check before design

3.2 The Lab

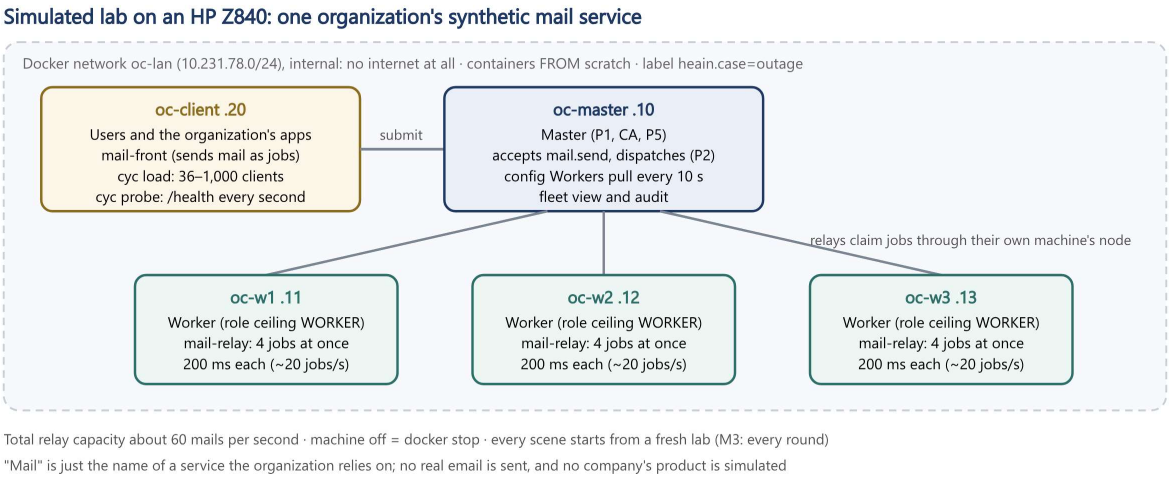


Figure 2 — Simulated lab on an HP Z840: one organization's synthetic mail service

The lab has 5 containers on an internal docker network with no route to the internet from the start (Figure 2). The Master and three Workers are set up with heain-setup per the lab manual; Workers have role ceiling WORKER by default. The mail-relay app on each Worker claims jobs through its own machine's node and handles 4 jobs at once at 200 ms each, about 60 mails per second across the three. The mail-front app on the user machine submits jobs to the Master.

3.3 Tools

Command	What it does
cyc relay	The mail-relay app: registers (capability mail.send, execution job), heartbeats, claims jobs by free slots; counts jobs worked, mails delivered, unique tickets, duplicate deliveries, expired leases and peak concurrency, kept across restarts
cyc load -mode submit	Submits jobs at a set rate, each with its own Idempotency-Key; logs every request as CSV
cyc load -mode mix	N clients cycling heartbeat, mode, submit and journal status; either without backoff (retry in 100 ms) or honouring Retry-After with jittered backoff [12][13]
cyc probe	GET /health every second (2 s timeout), logging status and response time
cyc call -H	Sends extra headers (e.g. Idempotency-Key) from test commands

Table 4 — Tools added to cyc (casestudies/cyber/tools/outage.go)

3.4 Scenes

Scene	What is done	Pass criteria
M1a	Load of 20 mails per second; the admin sets admission.disk_min_free_mb to 100,000,000 at the Master (a 2a value, not through P5), then sets it back after about 26 seconds	Every refusal states a reason; accepted mail delivered in full, no duplicates
M1b	Jobs take 1.5 s; the admin lowers dispatch.lease_default from 30 to 1 s to "release stuck jobs faster"	No job lost silently
M1c	The M1a change as commit confirmed for 1 minute: (i) not confirmed (ii) confirmed by the admin	(i) rolls back by itself, down to the Workers
M2	Load of 36 mails per second (60% of three machines); stop w1, then w2, until w3 takes 180% of its capacity, then restart; 30 seconds per phase	No machine holds more than its capacity; no duplicate mail; all delivered eventually; Master answers /health within 1 s
M3	Clients call the Master once a second; Master stopped 300 s, then started; rounds a) 300 without backoff b) 300 honouring Retry-After c) 1,000 without backoff; a fresh lab each round	Master does not crash or restart; refusals are retryable; lab complete after the storm
M4	Master stopped 1,800 s on a network with no internet, then the organization's essential work is walked through	Site work continues; what cannot be done states why; catches up when the Master returns

Table 5 — Scenes M1–M4

3.5 Rounds

Round	Where	Result
Dev	Development machine (short times)	M1–M4 pass; findings match the Z840
oc1	HP Z840 (full times)	Cyber L1–L5 all pass with the new cyc; M1, M2, M4 pass; M3 stopped at round c because the admin API took more than 8 s after the storm
oc1b	HP Z840 (full times)	M3 uses a fresh lab each round and measures draining instead of the failing check; M1–M4 pass 38 checks (M1 9, M2 5, M3 10, M4 14), no failures; commit ae90f0f verified with ls-remote

Table 6 — Rounds of the outage case

All figures in section 4 come from round oc1b on the HP Z840, run with full times (M3 stops the Master for 300 s, M4 for 1,800 s). The failure in oc1 is the origin of finding O6, which the author reproduced in the development machine before changing the script. The cyber case still passes in full with the new tools (L1 14, L2 15, L3 17, L4 34, L5 13).

4. Results

4.1 M1a One Wrong Capacity Value

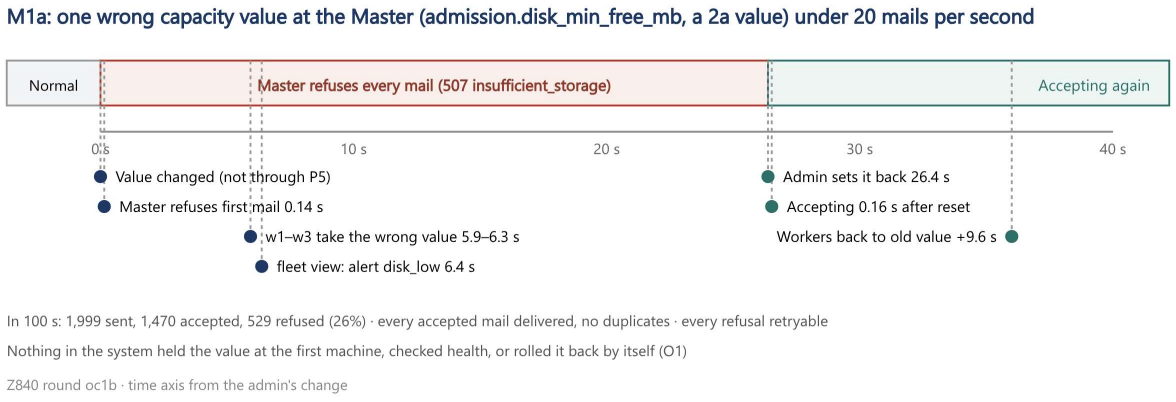


Figure 3 — M1a: timeline after the admin sets a wrong capacity value at the Master

The admin changed admission.disk_min_free_mb at the Master alone, as a 2a value that takes effect at once without approval (Figure 3). The Master refused its first mail after 0.14 seconds, all three Workers took the wrong value within 5.9–6.3 seconds, and the fleet view raised alert disk_low after 6.4 seconds. When the admin set it back at 26.4 seconds, the Master accepted mail again within 0.16 seconds and the Workers all had the old value within 9.6 seconds.

In 100 seconds users sent 1,999 mails, of which 529 (26%) were refused, each with reason 507 insufficient_storage, which is retryable; the 1,470 accepted were all delivered with no duplicates. The

system thus lost no data, but the whole service stopped because of one value, and nothing in the system held that value at the first machine, checked health, or rolled it back by itself (O1).

4.2 M1b A Lease Shorter than the Work

With jobs taking 1.5 seconds and the admin lowering the lease to 1 second, every job's lease expired before it finished, so it was redispatched and done again. 29 mails were worked 87 times (3.0 times each) and none succeeded. Every mail that ran out of retries went to an approver (p3_retries_exhausted, 29 entries), so no job was lost silently. But the change was not checked against real job times (O3), and when a relay returned a job after it had run out of retries, it got 500 internal "job key not found", marked retryable, 65 times, instead of 409 lease_expired (O2).

4.3 M1c Commit Confirmed

With the M1a change made as commit confirmed for 1 minute and no one confirming, the Master rolled back by itself after 61.6 seconds and the Workers all had the old value within 68.6 seconds. But when the admin, still able to reach the admin API, saw nothing wrong and confirmed, the wrong value stayed while every mail was refused. Commit confirmed therefore guards only against changes that lock the admin out, not changes that stop the service (O4).

4.4 M2 Workers Dropping Out One by One

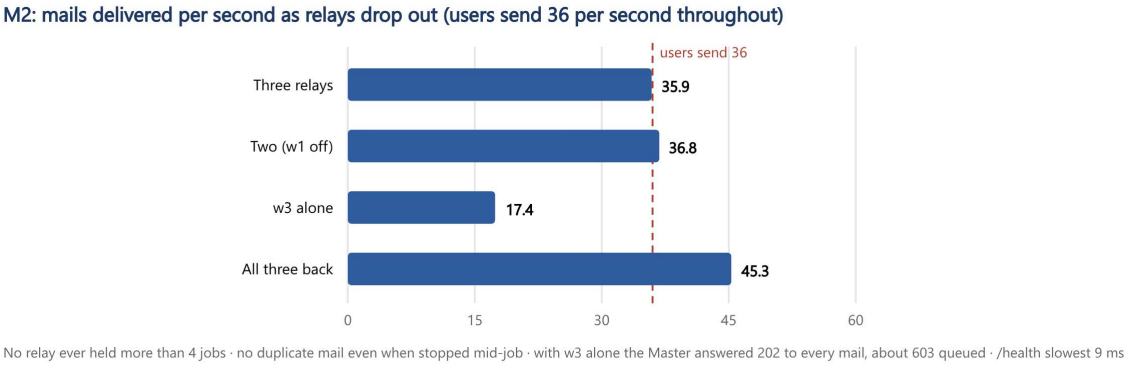


Figure 4 — M2: mails delivered per second in each phase

Users sent 36 mails per second throughout. Three relays delivered 35.9 per second; with w1 off, the remaining two still delivered 36.8; with w2 off too, w3 alone delivered 17.4, which is what one relay can do; and with all three back, they delivered 45.3 per second to drain the backlog (Figure 4).

- **No machine held more than its capacity:** a relay held at most 4 jobs at once, because work is pulled, not pushed, so the remaining machines did not fail in turn as in the real incident.
- **No duplicate mail:** w1 and w2 were stopped while holding jobs; those were redispatched when their leases expired, and 6,479 of 6,479 mails were delivered.
- **The Master was not slow:** /health answered 200 all 180 times, slowest 9 ms.

- **But no backpressure:** while one relay remained, the Master answered 202 to every mail; about 603 queued, and senders were not told mail would be slow (O5).

4.5 M3 The Master Returns to Everyone at Once

	a) 300 clients	b) 300 clients honouring Retry-After	c) 1,000 clients
Requests in the first 60 s	22,388 (373 per s)	12,458 (208 per s)	74,829 (1,247 per s)
503 in the first 60 s	4,543	11	16,107
Back to 95% answered for 30 s	2 s	2 s	2 s
Master peak memory	75 MiB	60 MiB	173 MiB
Master crashed or restarted	No	No	No
Jobs queued after the storm	13,439	9,300	56,046
Dispatched in the next 60 s	2.6 per s	3.6 per s	0.0 per s
Admin API (/v1/admin/nodes)	1.4 s	1.0 s	44.6 s
Master CPU	142–152%	129–137%	135–160%

Table 7 — M3: Master stopped 300 s, then started (first 60 s after start, and after the storm)

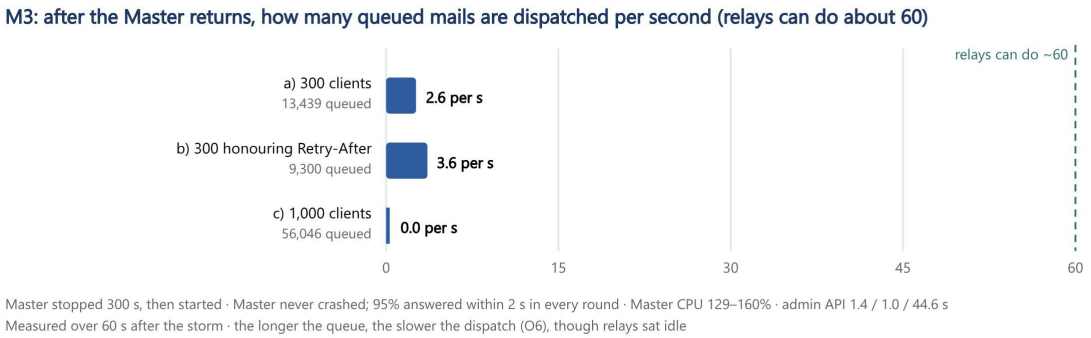


Figure 5 — M3: dispatch rate of queued jobs after the Master returns, compared with what relays can do

The first part of M3 matched expectations. The Master did not fall over even with 1,000 clients calling 1,247 times per second, was back to 95% answered within 2 seconds, and every refusal was a retryable 503, except one journal status request in round a that got 403 while the Master was just starting (O7). Round b shows that if clients honour Retry-After and back off, early load nearly halves and 503s almost vanish [12][13].

The second part did not. During the storm, clients submitted many times more work than the relays could do, and the Master accepted all of it (O5). After the storm the Master used 130–160% CPU and dispatched more slowly the more was queued, from 3.6 jobs per second at 9,300 queued to 0 at 56,046, while the relays sat idle and could do about 60 per second. The admin API slowed to 44.6

seconds, which is why round oc1 stopped (Figure 5). The lab's "recovered but overwhelmed" therefore came not from requests arriving together but from work queued inside the Master itself (O6). The cause in code is not yet confirmed; the hypothesis is that each dispatch scans the whole queue, which must be proven in the core-change proposal.

4.6 M4 No Centre, No Internet

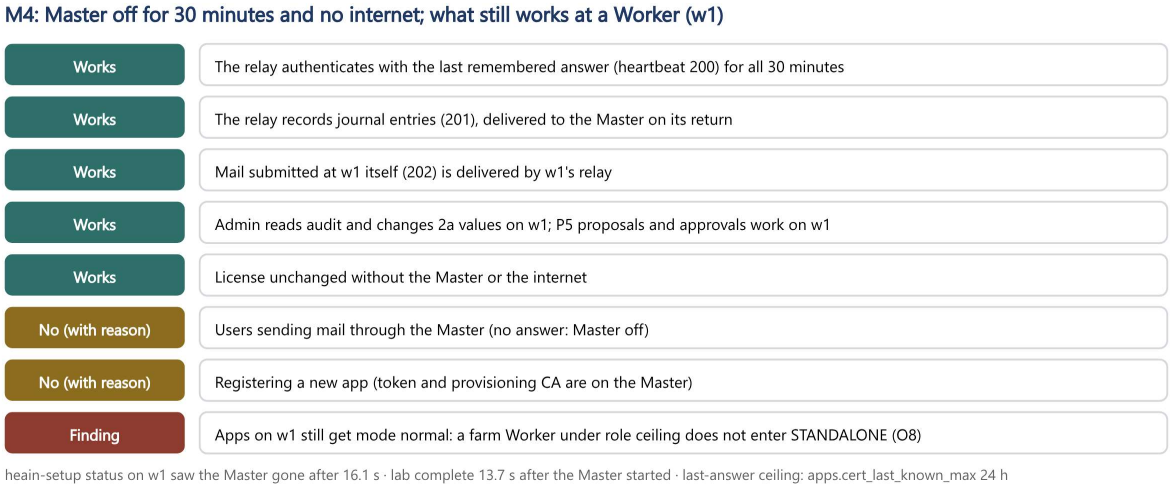


Figure 6 — M4: what works and what does not at w1 while the Master is off for 30 minutes

The Master was off for 30 minutes on a network with no internet. heain-setup status on w1 showed the Master gone after 16.1 seconds. The relay on w1 authenticated with its last remembered answer throughout the 30 minutes and recorded journal entries, and mail submitted at w1 itself was delivered by w1's relay. The admin could read audit and change 2a values, and P5 proposals and approvals worked on w1; the license did not change. Two things could not be done, both with a stated reason: users sending through the Master, and registering a new app. When the Master returned, the lab was complete within 13.7 seconds, w1's journal reached the Master, and mail flowed through the Master again (Figure 6).

What expires	Value	Effect on expiry
apps.cert_last_known_max	24 hours	The Worker stops using its last answer on app certificates
App certificate (relay)	89 days	The app cannot renew if the Master has not returned
w1 node certificate	730 days	—

Table 8 — How long the Master can be gone before anything expires (lab values)

The finding in M4 is that apps on w1 kept getting mode normal, because a Worker with a role ceiling is a farm member, not a child in the hierarchy, and so does not enter STANDALONE. The apps

therefore did not know the centre was gone, though heain-setup status could tell (O8). In the cyber case L4, pulling the LAN cable did enter STANDALONE, because that lab used -takeover.

4.7 The Real Incident Compared with the Lab

Aspect	asis (as reported)	heain (lab)
Wrong change	Took effect in many regions at once; hours to recover	Whole system stopped within ~6 s; accepting again 0.16 s after reset; no canary (O1)
Work flowing to the rest	The remaining machines could not cope and failed in turn	The remaining machines never held more than capacity, but the Master accepted work without limit (O5)
Recovered, then failed again	Recovered servers failed on taking requests	The Master did not crash and answered 95% within 2 s, but could not dispatch its backlog (O6)
Sites	Only data already on the machine was usable	Workers worked on for 30 minutes, including new work submitted at the Worker, but apps did not know the centre was gone (O8)

Table 9 — asis compared with heain

5. Findings

All eight findings need a core change by the project's procedure: write a proposal, have the author decide, then run the full regression (Table 10). The directions in the table are the paper author's suggestions for that decision, not yet a design.

#	Scene	Finding	Suggested direction	Priority
O6	M3	Dispatch slows with queue length, to 0 at ~56,000 jobs; the admin API slows to tens of seconds	Find the cause in code first, then make dispatch independent of queue length	1
O5	M2, M3	The Master accepts work without limit (202 to every job) and does not tell senders	A queue ceiling per capability or per app, answering 429 with Retry-After when full	1
O1	M1a	One 2a value stops the whole service within ~6 s, with no canary or automatic rollback	Roll config out in stages (canary) with health criteria, rolling back by itself on failure	2
O4	M1c	Commit confirmed does not stop a change that halts the service once the admin confirms	Tie confirmation to the same health criteria as O1	2
O3	M1b	The lease can be changed without a check against real job times	Warn or refuse when the lease is shorter than measured job times	3

#	Scene	Finding	Suggested direction	Priority
O2	M1b	Returning a job after retries are exhausted gets a retryable 500 instead of 409	Answer 409 lease_expired	3
O7	M3	App requests get 403 while the Master is just starting	Answer 503 as in C20 until the registry is ready	3
O8	M4	Apps on a Worker under role ceiling get mode normal although the Master is gone	Make mode reflect that the Master cannot be reached	3

Table 10 — Findings O1–O8 and suggested directions

O6 and O5 are directly linked. With a queue ceiling at the Master (O5), work would not pile up into the tens of thousands, and dispatch slowing with the queue (O6) would not show at this level. But fixing O5 alone hides O6 under the ceiling, so both should be fixed and measured before and after with M3 round c, the round that shows the symptom most clearly.

6. Discussion

6.1 Answers to the Research Questions

- **RQ1:** One wrong value spread through the whole system in about 6 seconds, and nothing stopped it by itself. Commit confirmed helps only when no one confirms.
- **RQ2:** No. The remaining machines were not pushed past capacity, because work is pulled by capacity, but the surplus piled up at the Master without bound and without telling senders.
- **RQ3:** The Master survived and answered quickly, but it was not back to normal in service terms, because it dispatched its backlog more slowly the longer the queue.
- **RQ4:** Sites worked on for 30 minutes, including new work submitted at the Worker itself. What could not be done stated why, but apps were not told the centre was gone.
- **RQ5:** All eight findings need core changes. The most urgent are O6 and O5, which are directly linked.

6.2 What Can Be Done Now against What Needs a Core Change

Item	Doable now	Needs a core change
Changes that stop the service	Always use commit confirmed and check results at the Workers before confirming	Canary and health criteria (O1, O4)
Clients reconnecting at once	Have apps honour Retry-After with jittered backoff (nearly halved early load in round b)	—
Unbounded backlog	Have sending apps limit their own unfinished jobs	Queue ceiling and 429 (O5); dispatch that does not slow with the queue (O6)
Apps not knowing the centre is gone	Use heain-setup status or the fleet view instead	Correct mode (O8)
Lease	Set the lease longer than the slowest job and have apps use /extend	Value checking (O3)

Table 11 — Separating what configuration and practice can do now from what needs a core change

6.3 Limitations

- **One machine:** every container is on the HP Z840; load and CPU figures are for comparing before and after fixes, not the capacity of real machines or networks.
- **Simulated clients:** the 300 and 1,000 clients are goroutines on one machine sharing one app certificate, so the TLS handshake load is lighter than many real apps.
- **Job mix:** in M3 a quarter of requests are submissions, which builds a large backlog; a real organization's mix may differ.
- **Cause of O6 not confirmed:** the paper reports measured symptoms; the cause in code must be proven in the core-change proposal.
- **Real Windows lab not included:** scenes M1–M4 have not yet run on the ASUS and HP Z8s.
- **Incident information:** Microsoft has not yet released its full report; the explanation of cause may change.

7. Conclusion and Recommendations

This case tested heain-core on what earlier cases had not: a centre that fails because of its own change, and the load waiting for it when it returns. What was designed to hold did: pulling work by capacity, never doing work twice, working on at sites, and a Master that does not fall over when everyone reconnects at once. But the simulation found two layers of gap never tested before. The first is configuration distribution, which has no canary and no automatic rollback. The second is the

Master's job queue, which accepts without limit and dispatches more slowly as it grows, so a system that is "back" still cannot serve.

1. Write the core-change proposal for O6 and O5 first, finding O6's cause in code and measuring before and after with M3 round c.
2. Write O1 and O4 as one proposal: staged config rollout with health criteria.
3. Combine O2, O3, O7 and O8 into one proposal.
4. After the fixes, rerun M1–M4 and the full regression of every case study.
5. Run scenes M1–M4 on the real Windows lab per the t7 manual once fixed.

References

[1] Al Jazeera. Microsoft hit by global outages: What happened and why? 13 November 2026. aljazeera.com/news/2026/11/13/microsoft-hit-by-global-outages-what-happened-and-why

[2] Nation TV (citing Reuters). Paralysis across days! Microsoft down more than 12 hours. November 2026. nationtv.tv/news/foreign/378986147 (in Thai).

[3] Thansettakij. Behind the global Microsoft 365 outage. November 2026. thansettakij.com/technology/673712 (in Thai).

[4] ABC7. Microsoft outage live updates. 13 November 2026. abc7ny.com/live-updates/microsoft-outage-live-updates-what-know-outlook-365-other-apps-remain-down/20036817/

[5] CBS News. Microsoft battling major outage. 13 November 2026. cbsnews.com/news/microsoft-battling-major-outage/

[6] Naewna. Microsoft outage spreads to Thai banks. November 2026. naewna.com/n/international/164437/; Thairath. 13 November 2026. thairath.co.th/lifestyle/tech/2972363 (in Thai).

[7] The Register. Microsoft 365 outage drags on for nearly 10 hours. 23 January 2026. theregister.com/2026/01/23/microsoft_365_outage/

[8] W.Media. Microsoft Azure outage: cloud services disrupted after West US network failure, now resolved. July 2026. w.media/microsoft-azure-outage-cloud-services-disrupted-after-west-us-network-failure-now-resolved/

[9] Boonma, J. casestudies/outage/ in the heain-core repository (commit ae90f0f), 2026.

[10] Boonma, J. Project documents claude/study-2026-11-14-microsoft-outage and claude/plan-2026-11-14-outage-case-design, 14 November 2026.

[11] Boonma, J. "The 2026 Cyber-Attack Case Study on heain-core." 13 November 2026 (companion paper).

[12] Beyer, B., Jones, C., Petoff, J. and Murphy, N. R. (eds.). Site Reliability Engineering, chapters 21 Handling Overload and 22 Addressing Cascading Failures. O'Reilly, 2016.

[13] Brooker, M. "Timeouts, retries, and backoff with jitter." Amazon Builders' Library. aws.amazon.com/builders-library/timeouts-retries-and-backoff-with-jitter/

Author and copyright holder: Mr. Jakkrit Boonma (Independent Researcher) | ORCID: 0009-0000-7165-3787 |

Email: jaggid.boonma@yahoo.co.th

© 2026 Jakkrit Boonma. All rights reserved. Ownership and copyright in this document belong to Mr. Jakkrit Boonma from the date of publication, under applicable copyright law.