



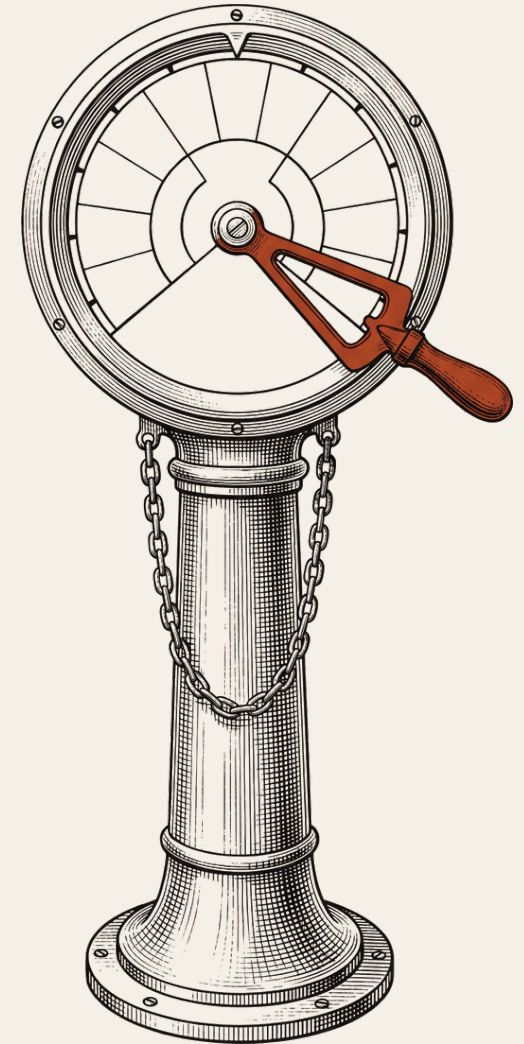
MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS



Agents Propose, Mediators Act

Safe Actions for Operations Agents

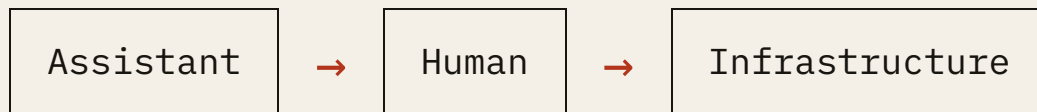
Laurent Bindschaedler · Max Planck Institute for Software Systems
Knowledge Bases and Agents for CERN · 19–23 October 2026



CONTEXT

From advice to action

UNTIL RECENTLY · CHAT ASSISTANTS SUGGEST COMMANDS



NOW · OPERATIONS AGENTS RUN THEM



THE HUMAN USED TO

- Understand what the command does
 - Check that it is right, here and now
- ⇒ **Now the agent runs the command, and both steps are gone**

? Who checks an action before it runs?

THE PROBLEM

Why an agent's actions go wrong

ILLUSTRATIVE INCIDENT		AGENT WITH SHELL ACCESS
t+0 s	Latency alert on a web service	
t+20 s	Agent restarts the database it suspects	
t+25 s	Dependent services start failing	
t+40 s	Alert persists, agent restarts it again	
<hr/>		
The outage spreads with every retry		

- Partial view of dependencies
- Damage hard to undo
- Retry loops repeat the mistake
- It improvises: a retry can run something different, for better or worse



The agent holds the credentials, so every mistake reaches the system

THE PROBLEM

Why current safeguards fall short

Prompts and instructions

Tell the agent
what not to do

Advice, not enforcement

Command allowlists

Limit which commands
it may issue

Not whether it is safe now

Approve every step

A human confirms
each command

A human on every step

Check afterwards

Validate or
revert what it did

Acts after the damage



Missing: a trusted check at the moment the command runs

THE PROBLEM

Allowed is not the same as correct

THE AGENT ASKS TO RESTART THE DATABASE SERVICE

`Restart svc/db`

Is it allowed?

Checked by access roles, allowlists, sandboxes

Uses the agent's identity and a fixed policy

Yes: the agent has the restart role

Is it right, here and now?

Dependents still serve traffic

Needs the current state, at the moment it runs

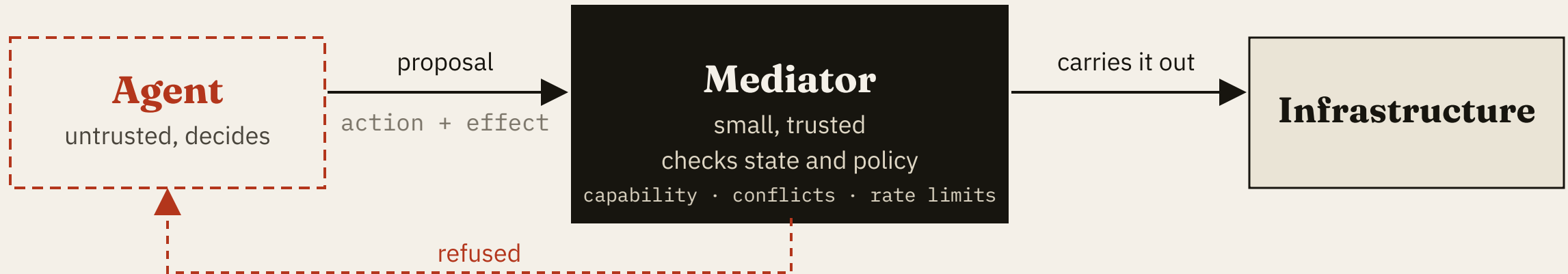
No: a restart now spreads the outage



Access control answers only the first question

WHAT WE PROPOSE

Mediation



Key idea: the agent decides what to do, but never does it

WHAT WE PROPOSE

Mediation is not just an approval step

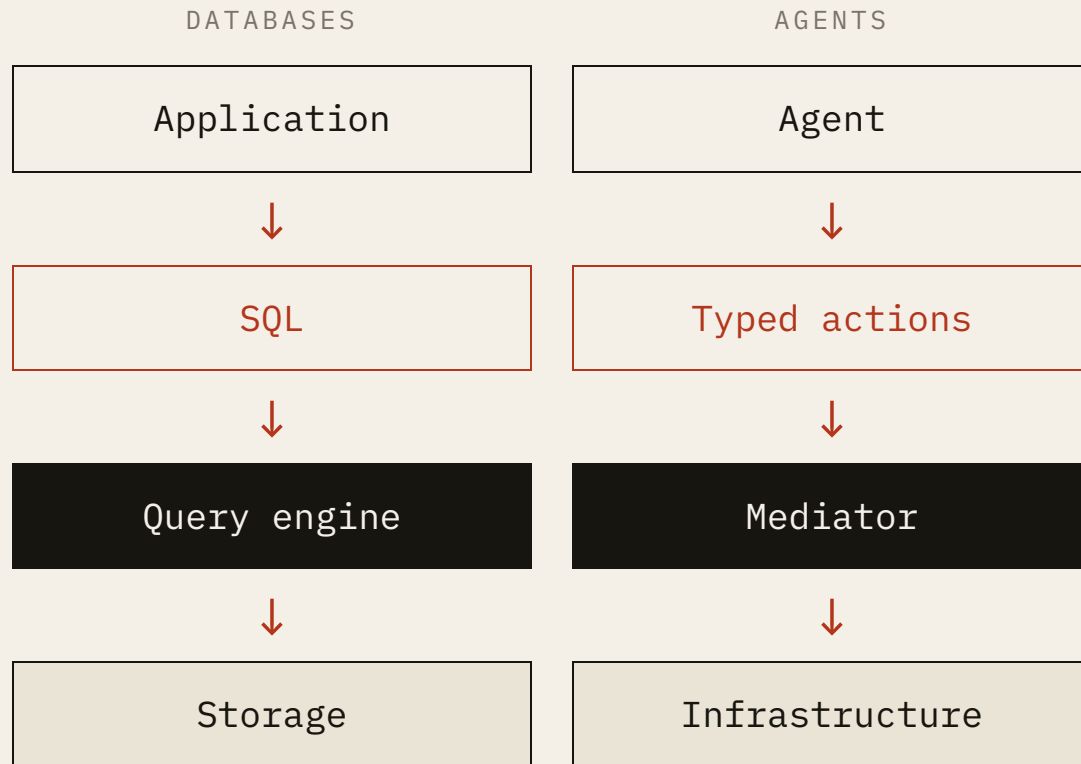
	APPROVE THE COMMAND	MEDIATE THE EFFECT
Who can act	The agent usually holds the credentials	Only the mediator holds them
What is checked	A command string, by a person who may not know what it does	The declared effect, against policy and current state
What runs	Whatever comes next. After a crash, the retry can differ from what was approved	Exactly the checked transaction. Anything different is checked again
On failure	Partial changes stay behind	The mediator runs the declared compensation



The mediator does not forward commands. It turns checked proposals into its own actions

WHAT WE PROPOSE

Interpose a language, as databases do



- No raw memory writes, only queries
 - The language bounds what can be said
 - The engine decides whether and how it runs
- ⇒ **Agents get a language for effects, not a shell**



Typed actuation: every action declares its effect, and the mediator enforces it

WHAT WE PROPOSE

What a proposal declares

```
# svc/db: the database service  
  
TRANSACTION  
Drain          svc/db  
Restart       svc/db  
RestoreTraffic svc/db  
  
CONFLICT KEYS  
svc/db  
  
PRECONDITIONS  
svc/db in current recovery group  
  
ON FAILURE  
Compensate: RestoreTraffic
```

THE MEDIATOR ASKS

- **Action and target:** in scope, with the capability?
 - **Effect type:** retried, reversed, or compensated?
 - **Conflict keys:** who else holds this service?
 - **Preconditions:** still true at commit time?
 - **Failure policy:** what runs if a step fails?
- ⇒ **The mediator must be trusted: small, auditable, and able to explain each decision**



Irreversible actions are refused unless a human grants break-glass approval

THE RULE IN ACTION

The same incident, mediated

ILLUSTRATIVE · AGENT PROPOSES		MEDIATOR DECIDES
Plan 1	Restart the database · refused	Dependents still serving traffic
Plan 2	Drain, restart, restore · done	Run as a transaction, compensated on failure
Plan 3	Restart it again · refused	Rate limit reached
<hr/>		
The loop produces proposals, not damage		



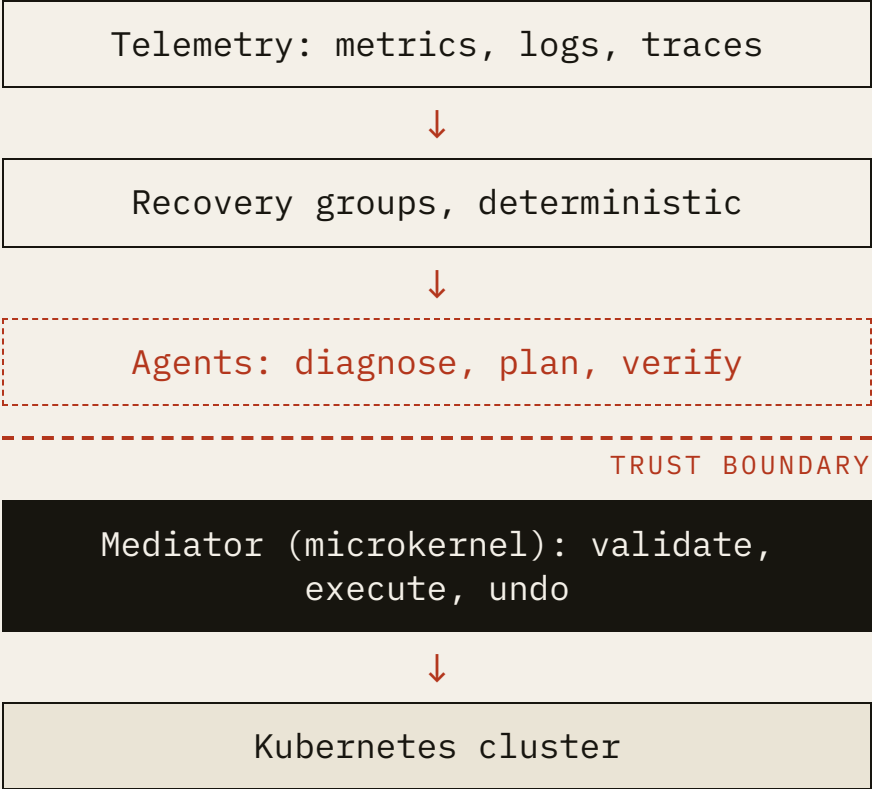
A looping agent can only keep proposing

⇒ The checks run in a small, fixed mediator

⇒ Swap the model, or use a typed decision model: the checks stay the same

PROTOTYPE · ARCS 2026

Safe recovery for cloud services



ACTION	HOW THE MEDIATOR RECOVERS
Restart	Retry, it is idempotent
Drain	Compensate with RestoreTraffic
RestoreTraffic	Inverse: Drain
CircuitBreak	Reset to default
RateLimit	Remove the limit
Scale	Scale back by the same amount
RollbackConfig	Restore the previous configuration

 Only the mediator changes the cluster

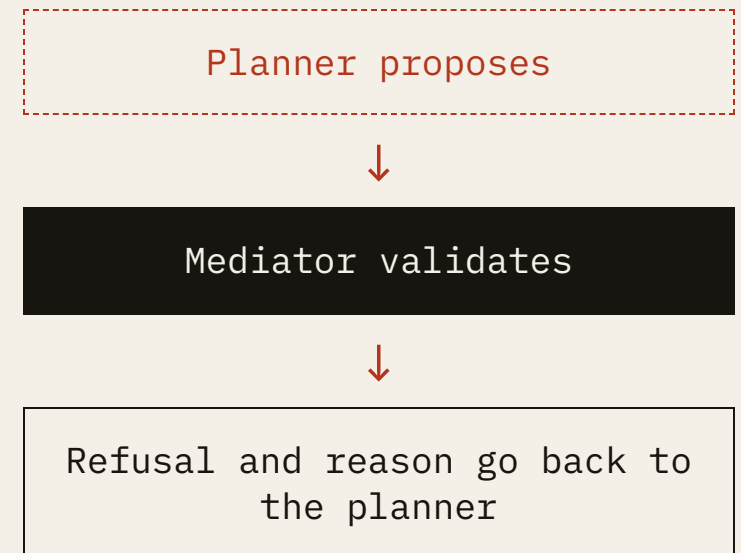
PROTOTYPE · ARCS 2026

A refusal says what to fix

MEDIATOR REPLIES

```
REJECT missing_capability("restart:svc/payment")  
REJECT out_of_scope("svc/cart" not in recovery_group)  
REJECT irreversible_effect("drop_table") requires break_glass  
REJECT conflict(resource="namespace/prod")
```

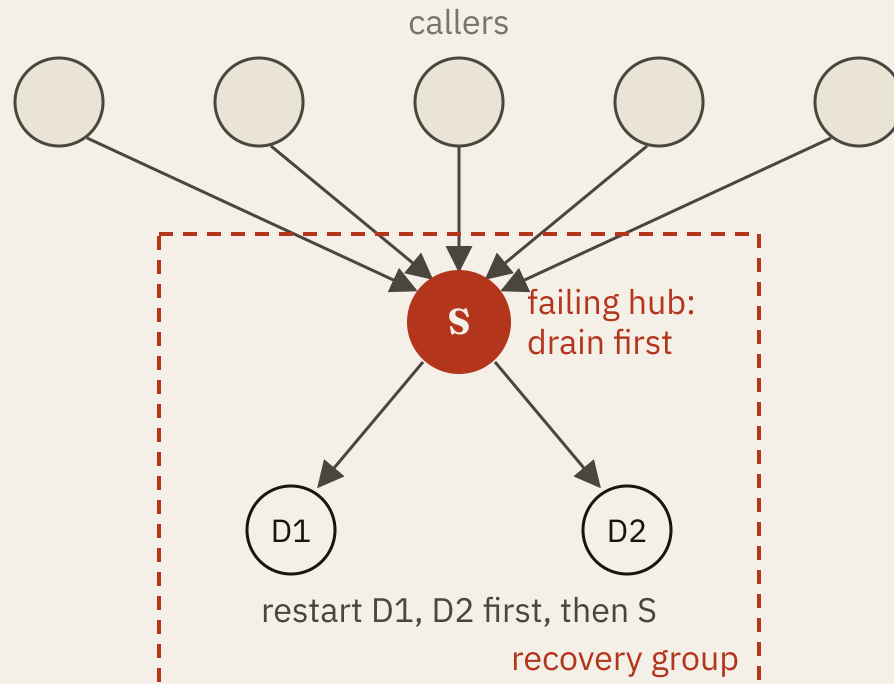
In 90 incidents over 3 runs, every refused plan was repaired within three attempts



Typed actions let the mediator say why it refused, so it steers the agent to a valid plan

PROTOTYPE · ARCS 2026

Scope comes from traces, not configuration



WHY: ONE RESTART REACHES MANY SERVICES

Alibaba traces: median 8, P99 59, maximum 177 services

- Dependencies shift with feature flags and traffic
- Recovery group: failing service and callees
- Order: callees before callers
- ⇒ **Groups are computed online, for each incident**
- ⇒ **Scope is only as good as the dependency data**



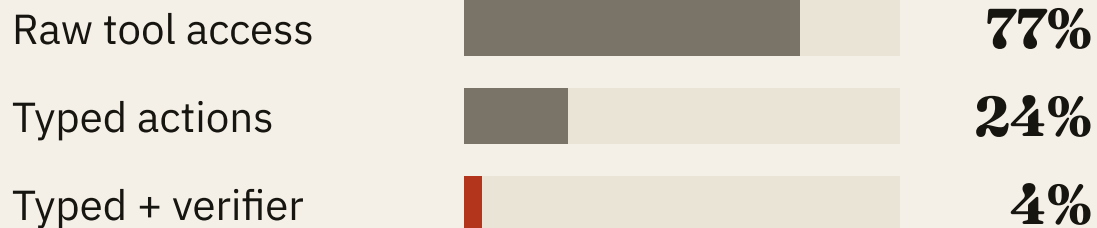
Real production traces: scope computed in 21 ms at P99 over 5,459 services

PROTOTYPE · ARCS 2026

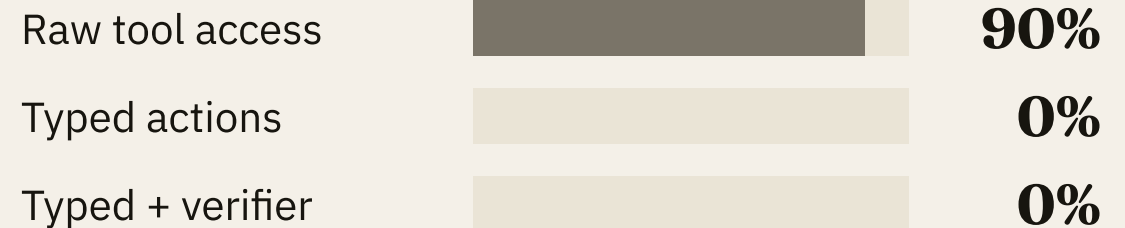
Results: harm caused by the agent

Share of incidents in which the agent caused harm

SIMULATED INCIDENTS, ALIBABA TOPOLOGIES



LIVE CLUSTER, INJECTED FAULTS



Live runs had 30 to 50 incidents each: too few to separate the two typed setups

⇒ **Typed actions remove most of the harm. A verifier agent removes more**

Harm: an agent action that worsens latency or error rate by more than 10% for more than 30 s



No harm means no action crossed this threshold. It does not mean the best recovery

PROTOTYPE · ARCS 2026

The price is recovery time, not mediation

WHERE THE MEDIAN RECOVERY TIME GOES



AGAINST KUBERNETES RESTARTING A FAST SERVICE ON ITS OWN



⇒ The extra time is the model's 13 s, not the mediator's 0.1 s



The value is safety, not speed. The mediator itself costs almost nothing

RELATED WORK · DATABASES

Databases already generate code behind a language

⇒ Both check the generated code against what the query asks

Bespoke OLAP

An LLM writes a whole engine for one fixed workload. Automated validation checks it.
Order-of-magnitude speedups over DuckDB

Wehrstein, Eckmann, Jasny, Binnig · [arXiv:2603.02001](https://arxiv.org/abs/2603.02001)

JetStream

An LLM writes accelerators inside an existing database, which keeps storage and transactions. Each is tested against the database or proved against the SQL

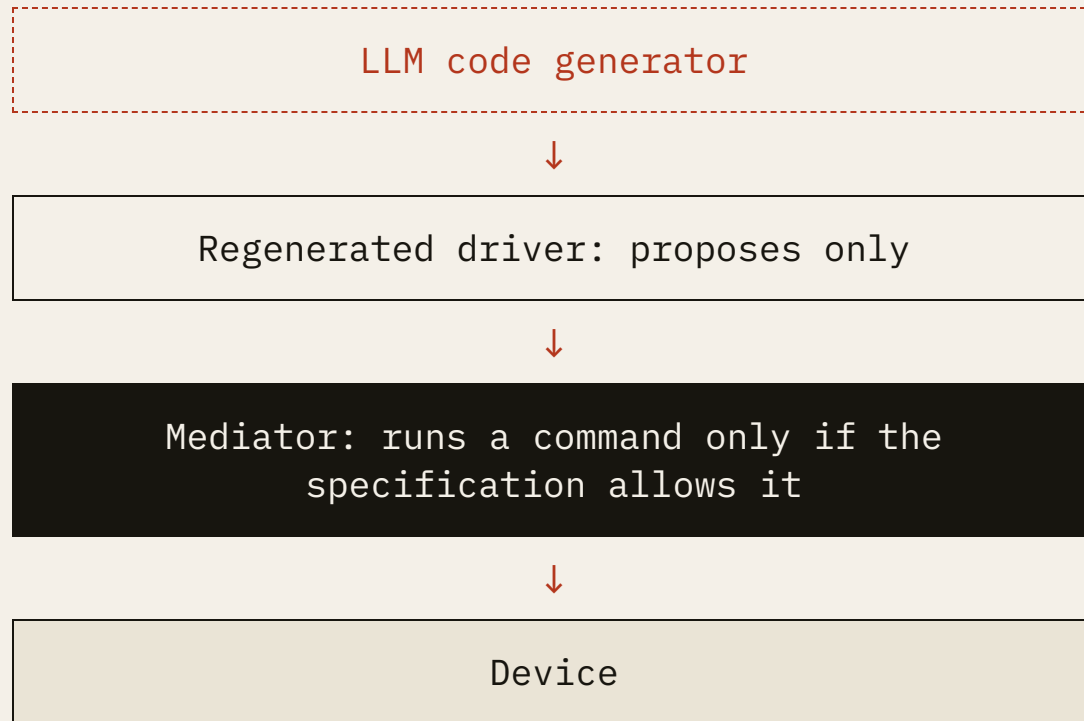
Balasingam, Ngom, Yu, Kraska · [arXiv:2610.20527](https://arxiv.org/abs/2610.20527)



A language that says what, not how, is where a check can be placed

POSITION PAPER · PACMI 2026 · WITH J. HU AND A. GOEL

The hard case: generated code runs in the system



Drivers are regenerated per device and workload. Device writes cannot be recalled

VERIFICATION HELPS, BUT A PROOF

- Is only as good as the specification
- Rests on assumptions at the edges
- Covers one binary, not the next one
- Covers safety, not speed

Mediator built, not yet released



The guarantee lives in the mediator, so it survives regeneration

Catching a wrong write

KERNEL ASKS		DRIVER PROPOSES
Request	Write blocks 100–107 from buffer B	
Proposal	Write blocks 200 –207 from buffer B	
<hr/>		
Overwrites the wrong blocks.		
No later regeneration can undo it		

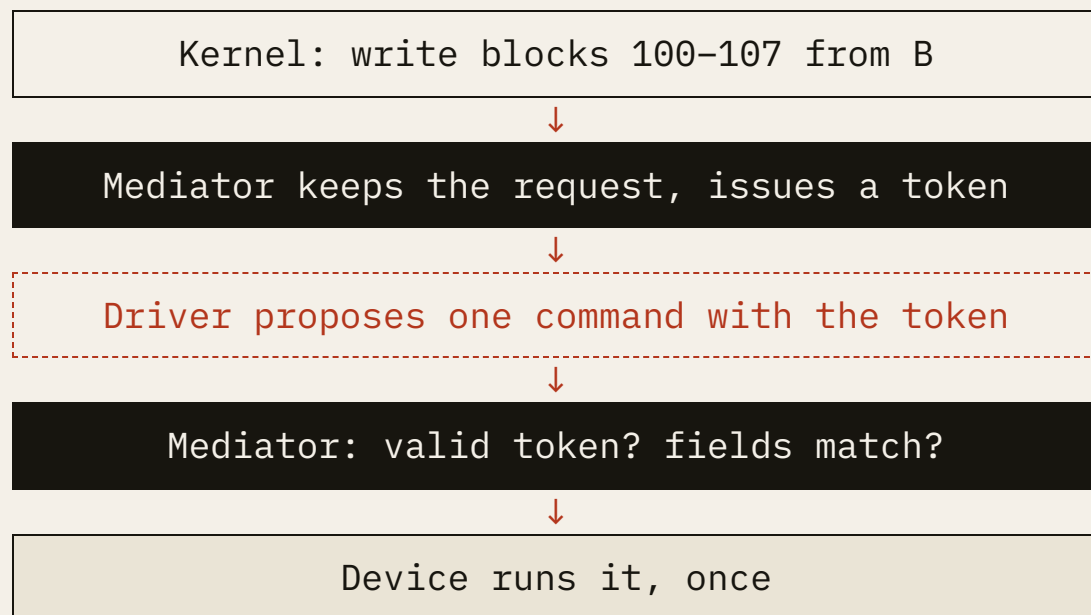
CHECK	SEES	VERDICT
Memory protection	Buffer B is allowed	passes
Device-safety monitor	Block 200 is legal	passes
Tests	Never ran this case	passes
Mediator	Request says 100	refuses

 Only a check that holds the request at run time can see the mistake

⇒ In operations: access control passes the wrong restart for the same reason

POSITION PAPER · PACMI 2026

How the mediator knows the request



- **Token:** a one-time ticket for this request. The driver cannot forge it or use it twice
- **Fixed fields:** the request sets the blocks, length, and buffer. The driver picks only the queue slot
- **Specification:** describes the interface, not the driver's code



The mediator takes the request from the kernel, never from the driver

THE LESSON FOR OPERATIONS

Mediate at the interface, never below it

	AT THE INTERFACE	BELOW IT
File systems	<code>mkdir(path, mode)</code>	writes to directory blocks
Operations	<code>Restart(svc/db)</code>	<code>bash -c "kubectl ..."</code>
The mediator	checks intent directly	must rebuild intent from bytes

⇒ Add a shell action, and the language is a shell again

? What is the right interface for your operations?

BEYOND ONE ACTION · OPEN PROBLEMS

Checking one action is not enough

Human approvals

An approval becomes a condition the commit checks, tied to the exact artifact, policy, and path that were approved

Guarded Commits · [arXiv:2610.00037](https://arxiv.org/abs/2610.00037) · with F. Kossmann, C. Liu, J. Mohoney

Effects left behind

A canceled agent leaves branches, payments, and messages behind. The runtime records each one and decides how it ends

Effect Domains · ongoing work

Fleet-wide limits

Per-agent caps do not add up: in simulation, 50 agents overdraw a shared limit 2.4× on average. A shared ledger bounds declared charges

Irreversibility Budget · Agentic OS, SOSP 2026 · with B. Mohammadi



Each needs the same boundary: every effect passes through the mediator

FOR DISCUSSION

Three open questions

01 Which operations can be described precisely enough to mediate?

Job submission, configuration changes, data deletion, access control

02 Who writes the specification, and who checks it?

A wrong specification is enforced faithfully

03 Where should humans step in?

Approving plans, single effects, or handling refusals

? What would you let an agent do today, and what never?

SUMMARY

Agents propose, mediators act

- ✓ Agents decide, but never execute
 - ✓ A small trusted mediator checks every effect
 - ✓ Mediate at the interface where intent is visible
 - ✓ Built for cloud recovery, explored for generated code
- ⇒ **Less harm, sometimes slower recovery**

**“The agent decides.
The mediator acts.”**

Laurent Bindschaedler · bindsch@mpi-sws.org · binds.ch



Questions

Backup slides for likely questions

BACKUP · QUESTION 1

Is this not just admission control?

	CAN REFUSE	HOLDS THE CREDENTIALS	CHECKS CURRENT STATE	RUNS COMPENSATION	SAYS WHY
Access roles (RBAC)	yes	no, the caller does	no	no	no
Admission webhook, OPA	yes	no, the caller does	can query it	no	a message
Human approval	yes	no, usually the agent	by eye	no	sometimes
Mediator	yes	yes, for agent actions	yes, at commit	yes	structured reason

⇒ A policy engine with sole authority to act and these checks is a mediator

Requires that the agent has no other path to the system

BACKUP · QUESTION 2

Who writes the action language?

ARCS: SEVEN ACTIONS, THREE RUNBOOK CATEGORIES

CATEGORY	ACTIONS
Restart	Restart
Traffic	Drain, RestoreTraffic, CircuitBreak, RateLimit
Resources	Scale, RollbackConfig

A NEW ACTION DECLARES THREE THINGS

- **Effect type:** retried, reversed, or compensated
- **Undo:** its inverse or compensation
- **Locks:** the resources it holds

ILLUSTRATIVE, NOT BUILT

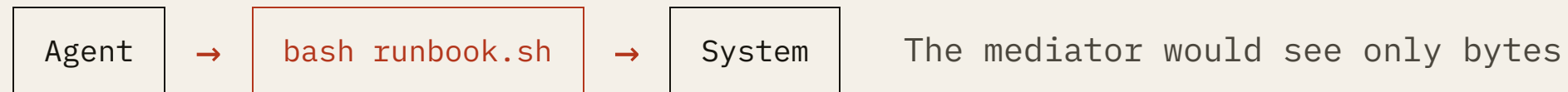
```
DrainNode(site, node) · undo: UndrainNode  
PauseTransfer(link) · undo: ResumeTransfer  
DeleteReplica(file, site) · irreversible
```

⇒ One small language per kind of effect. Each backend still needs a checked implementation

BACKUP · QUESTION 3

What happens to our shell scripts?

TODAY



WITH A MEDIATOR



⇒ **The script becomes an action's implementation, not the agent's interface**

Lift common steps one at a time. Scripts that do not fit stay with humans

BACKUP · QUESTION 4

Can an agent ever delete data?

EFFECT TYPE	EXAMPLE	WHAT THE MEDIATOR DOES
Retried	Restart a service	Runs it, retries on failure
Reversed	Scale, rate limit	Runs it, undoes it on failure
Compensated	Drain, then restore traffic	Compensates on failure, alerts operators if that fails
Irreversible	Delete a replica	Refuses, unless a human grants break-glass

⇒ **The approval binds to the exact deletion: file list, sites, policy version**

Binding the approval does not make the deletion safe. Replica and retention checks are still needed

How realistic is the evaluation?

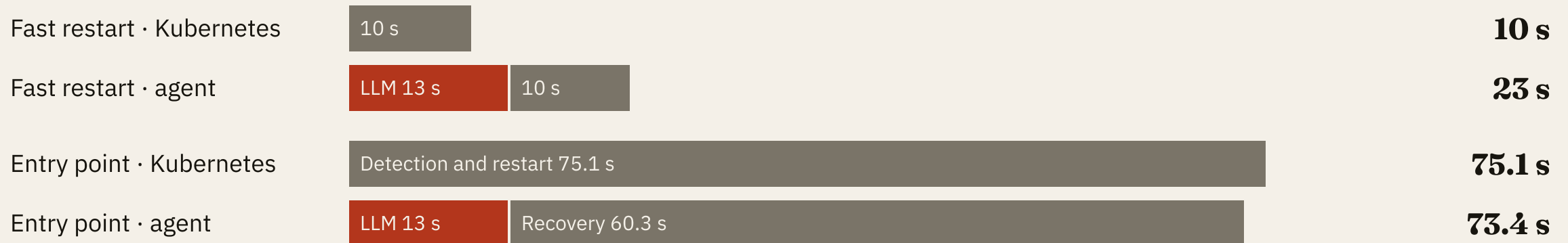
	PRODUCTION TRACES	SIMULATION	LIVE CLUSTER
Setup	Alibaba, 5,459 services. Meta, 392	Synthetic incidents from Alibaba topologies, 100 per setup	DeathStarBench, 10 and 8 services, one K3s cluster
Measures	Scope inference, blast radius	Harm, with a calibrated harm model	Harm under injected faults (343+ online incidents in all)
Result	21 ms at P99 (Alibaba)	77 → 24 → 4%	90 → 0 → 0%
Not shown	High traffic, multiple clusters, harm in production		

⇒ Real traces for scale, a small live cluster for harm

BACKUP · QUESTION 6

Why put a model in the loop at all?

RECOVERY TIME AGAINST KUBERNETES RESTARTING THE SERVICE ON ITS OWN

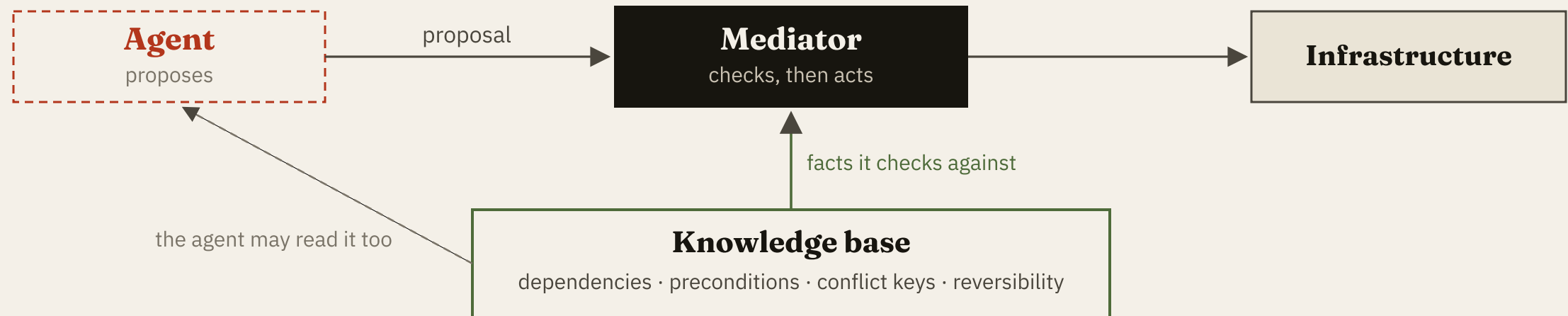


⇒ **Known fault: keep the runbook, and route it through the mediator**

⇒ **Novel fault: the agent proposes, the mediator checks**

BACKUP · QUESTION 7

Where does a knowledge base fit?

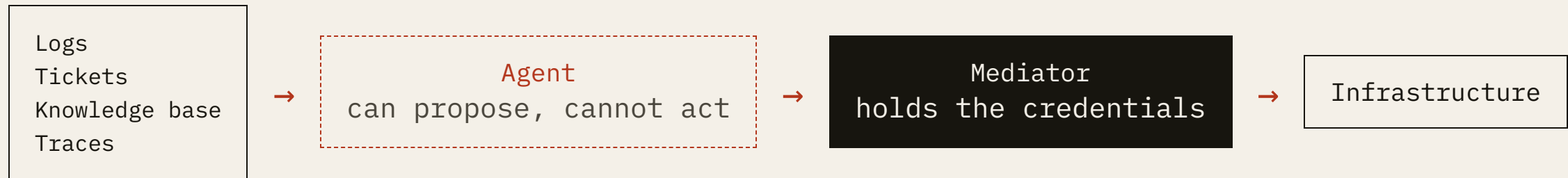


⇒ In ARCS, traces supplied the scope. Preconditions and conflict keys came with each plan

! A stale entry gives a wrong scope, enforced faithfully

BACKUP · QUESTION 8

What about prompt injection?



- An injected agent still only proposes, within the language and the scope
- It can still pick a bad action inside the allowed scope
- Traces and the knowledge base also feed the mediator's scope. Poisoned entries can widen it



ARCS does not defend against prompt injection or compromised telemetry