A Link That Actually Falls

Communication Loss as an Event Rather Than a Declaration, and What a Map Is Once Two Robots Stop Agreeing

Haniel Ulises

Epistemic Robotics

September 10, 2026

Every fleet in this series models restricted connectivity the same way. An agent is oblivious to an announcement because the domain declares it oblivious; the declaration is written before the run and no link ever falls. This report cuts the radio between two robots during a run and measures what the declared version cannot express.

The apparatus is small and two findings from building it are not. The announcement channel does not carry belief: it is read by a transcript recorder and by nothing else, and the epistemic model advances through a service the executor calls when an action completes, so a fault injector placed on the channel measures nothing at all. And the reconciliation of two occupancy maps, which passes forty-three tests over grids built in memory, refused the first pair that two robots actually built, because a test that builds both grids builds them the same shape and two maps that grow independently never are.

A third finding is about measurement and not about the system. The first version of the patrol driving the partner ran open loop and drove into the racking, and a partner held against a rack does not move, so the propagated belief tracked it perfectly. Correcting the collision raised the worst estimation error by a factor of four. The defect was flattering the result.

Contents

1 Scope

A fleet in this series coordinates by speech. One robot senses, tells another what it found, and a third is excluded; the domain declares who observes what, the planner reasons over those declarations, and the executor dispatches the actions they license. Restricted connectivity appears in that account exactly once, as a property of an action type: an agent is *oblivious* to an event, which is a statement about a column of a relation and not about a radio.

The declaration is made before the run. It cannot be false during one, it cannot be repaired, and nothing in the system can notice that the world stopped matching it. This report removes that by cutting the link between two robots while they work, and by measuring two things the declaration has no way to express: what one robot's estimate of the other is worth while nothing arrives from it, and what the two maps they built apart are worth once put together.

Three claims are established and are stated narrowly.

1. An outage can be injected into a running fleet without modifying the fleet, and the cut is real in the only sense that matters, which is that the messages are counted as they are discarded.
2. The belief propagation the requirement specifies exists, grows its covariance without decreasing it, and reaches the threshold past which it declares its own estimate unfit. Its acceptance criterion is met over outages of thirty, sixty and a hundred and twenty seconds against a partner in motion.
3. The reconciliation of two occupancy maps runs on maps two robots built from two lasers in a simulator, which required one operation the layer did not have and which its unit tests could not have provoked.

What is not established is set out in Section 8, and one item there is structural and not incidental: the epistemic models do not diverge under partition, because there is one model for the fleet and not one per robot, and a column of a shared relation is not a perspective held on a vehicle.

2 What existed before

The requirement names two topics and an engine. The first thing the work required was to find out what of that was code, and the answer determined the design.

Table 1: Occurrences of the artefacts the requirement names, across the three repositories, excluding build trees. The count is of appearances in source, not of definitions.

Term	`eplansys`	`eplansys-rmf`	This repository
`comm_monitor`	0	0	0
`BeliefUpdateEngine`	0	0	0
`link_down`	0	0	1, a string literal in a test
`link_up`	0	0	7, an atom of a Kripke model

The last row carries the whole difficulty. `link_up` appears in the warehouse scenario's model snapshots as `labels: {"w0": ["link_up"]}`: a proposition that a world makes true, fixed when the problem was written, and a conjunct of a safety constraint elsewhere. What existed of connectivity was the static declaration, and the requirement was specification without implementation.

2.1 The announcement channel does not carry belief

The obvious injector subscribes to the channel two robots speak on and stops forwarding. It would have measured nothing, and the reason is worth setting out because it is a property of the architecture and not an accident of this scenario.

The bridge publishes each speech act on `/eplansys/channel/private/<listener>`. The only subscriber is the transcript recorder, which exists so that who was addressed is a matter of record and not of inference. The planner package contains no reference to that channel. The model advances through a service, `epistemic_state/apply_action`, which the executor calls when an action completes, and in the performer for a speech act the two statements are `speak()` and `conclude(true, ...)`, in that order.

Proposition 1 (Gating the channel is epistemically inert)**.** Let G be a filter that discards every message on the announcement channel during $[t_d, t_r]$. Then for every action the policy dispatches in that interval, the sequence of product updates applied to the epistemic state is the same with G as without it.

Proof. The state is advanced only by `apply_action`, which the executor calls on completion of an action. Completion is reported by the performer, which calls `conclude` unconditionally after `speak`. Neither call reads the channel, and no subscriber of the channel calls `apply_action`. Hence the argument to every `apply_action` call, and the order of the calls, are independent of G. ☐

The channel is an audit path. That is a reasonable thing for it to be, and it is worth knowing before an experiment is built on the assumption that it is a transport. The cut has to be placed where belief actually travels, which in this system is the pose one robot receives from another and the map it is handed.

2.2 There is one epistemic model for the fleet

The bringup runs one planner and one epistemic state. An agent's accessibility relation is therefore a column of a shared structure and not a perspective held on a robot, and two robots that stop communicating do not thereby acquire two models to disagree with each other.

This bounds the report. What is measured here is the metric part of the requirement, an estimate of a partner's pose and its covariance, and the reconciliation of two maps. The divergence of modal structures is not measured, because there are not two structures to diverge.

3 The apparatus

Five components, in a package of their own. The fleet, the world and the per-robot mapping are those of the companion report and are included unchanged.

Table 2: The nodes of the scenario. Only the first three are required by the requirement; the fourth exists because of what the first real pair of maps did, and the fifth because the fleet's own driving does not work in this workspace.

Node	What it does
`comm_monitor`	Owns the state of the link and is the only thing that does. Publishes `link_down` and `link_up`, latched, at the configured instants, and answers a service for the state now.
`link_gate`	The injector. A type-erased relay over the generic publisher and subscription, so that one node cuts an odometry and an occupancy grid without a case for either. It holds no timer and subscribes to the monitor like everything else.
`belief_update`	The propagation. Holds the last received pose while the link is up; integrates the last received twist forward while it is down, growing the covariance. Reads the *gated* odometry.
`grid_align`	Places two maps on the union of their extents at their common resolution, by whole-cell translation.
`patrol`	Drives the partner, because the measurement needs a partner in motion and the fleet's own navigation cannot plan here.

Two decisions in the construction are load-bearing.

The gate is installed alongside, not in the middle. It reads the topic the fleet already publishes and writes a second name that only the believing side subscribes to. Nothing that exists is remapped, rebuilt or told that a gate is present. Interposing properly, by renaming the producer's output and taking over its name, also works and is what would be required to cut a topic that existing nodes already read; it is not required here and would mean editing a launch file four other demonstrations share.

The schedule is open loop. The link falls at t_d and returns at t_r, both measured from the monitor's first clock tick, and neither depends on where the robots are. An outage triggered by separation would make its own duration a function of the trajectory it perturbs, and the measurement could not then separate the two.

Remark 1 (The count is the evidence)**.** The gate counts what it discards and reports the count when the link is restored. A gate reporting zero discarded messages was gating a topic nobody publishes on, which is a launch error that otherwise resembles a successful run exactly. The propagation makes the symmetric check: it refuses a partner pose that arrives while it believes the link is down, and says so, because an arrival at that moment means the gate is on a different topic from the one being read.

3.1 The partner has to move, and has to not crash

The requirement is a claim about a belief held while a partner is moving and nothing is arriving from it. Measuring it needs the partner to move; it does not need the partner to be pursuing the mission, and a trajectory that repeats between runs is a better subject than one a planner improvises.

The first version drove a fixed square open loop and drove into the racking. A warehouse is not empty, and open loop on a floor with shelving on it is a collision with a timer attached. The square is now the intent and the laser is the veto: below the clearance in the forward arc the robot turns instead of advancing and keeps turning until the way is clear, with hysteresis on the way out so that a robot clearing an obstacle by a millimetre does not advance, block on the next scan, and shuffle along the face of the rack it is leaving.

Remark 2 (The collision was flattering the measurement)**.** This is the finding of the three, and it is a finding about measurement rather than about the system. A partner held against a rack does not move. The propagation integrates the last twist forward, so a stationary partner is tracked perfectly and the estimation error stays near zero for reasons that have nothing to do with the propagation. Correcting the collision raised the worst error of the sixty-second outage from 1.774 to 7.632 m. The run looks worse and is worth more: the defect was producing agreement between the belief and the partner by stopping the partner.

4 The criterion, measured

Three runs differing only in how long the link stays down. The partner drives its patrol throughout and knows nothing of the outage. The bound is the requirement's,

$$\varepsilon(\Delta t) \;=\; v_{\max}\,\Delta t \;+\; \sigma_{\text{prop}}, \qquad v_{\max} = 0.22\text{ m/s}, \qquad \sigma_{\text{prop}} = 0.05\text{ m},$$

with $v_{\max}$ taken from the maximum speed the fleet's own controller enforces. Taking it from anywhere else would be measuring against a speed the platform cannot reach.

Table 3: The three runs. *Margin* is the smallest difference between the bound and the error over the outage, and *trace* the positional covariance when the link returned. Every figure is read from the record the run wrote.

Outage	Samples	Worst error	At	Margin	Breaches	Final trace
30.0 s	325	3.634 m	30.0 s	0.040 m	0	0.600 m^2
60.0 s	632	6.967 m	60.0 s	0.050 m	0	1.200 m^2
119.9 s	1 247	19.199 m	119.9 s	0.040 m	0	2.398 m^2

The criterion is met in all three. The covariance behaves as specified: the trace grows without decreasing and crosses $\sigma^2_{\max} = 1.0\,\mathrm{m}^2$ at fifty seconds, at which point the engine reports that where the partner is should no longer be believed.

Two features of the table are worth more than the verdict.

The worst error is at the end, and grows with the outage. It was not, before the patrol was corrected. A partner intermittently held against a rack produced a worst error of 1.774 m at sixty seconds and a smaller one at a hundred and twenty than at sixty, which is the signature of a measurement whose subject was not moving. With the collision removed the three worst errors are monotone in the duration and each occurs at the last sample.

The margin is tightest at the start. In all three runs the smallest difference between the bound and the error is at $\Delta t = 0$, where the bound is σ_{prop} alone and the error is a few millimetres of odometry noise. It is not tightest at the end, where the error is largest.

4.1 The bound is loosest where the estimate is worst

The second observation is a property of the criterion and not of this implementation, and it is worth stating precisely because a passing verdict invites the opposite conclusion.

Proposition 2 (The criterion cannot fail against a partner that holds its course)**.** Let the partner be a differential-drive body that keeps the pair (v, ω) it had at t_d, and let the propagation integrate that same pair from the same pose. Then the estimation error is identically zero and the criterion is satisfied for every Δt. If instead the partner reverses at full speed at t_d, the error is $2v_{\max}\Delta t$ and the criterion fails for every $\Delta t > \sigma_{\text{prop}}/v_{\max}$.

Proof. For the first part, both integrations start from the same pose with the same twist and are the same function of time, so their difference vanishes and lies below any positive bound. For the second, the belief advances $v_{\max}\Delta t$ along the original heading and the partner the same distance against it, so the separation is $2v_{\max}\Delta t$. Requiring $2v_{\max}\Delta t > v_{\max}\Delta t + \sigma_{\text{prop}}$ reduces to $v_{\max}\Delta t > \sigma_{\text{prop}}$. □

With the values used here the threshold of Proposition 2 is 0.23 s. The bound is therefore broken within the first quarter second of the outage for the worst case it admits, and cannot be broken at all for the best. The measured runs sit between the two: the patrol holds its course along the straight legs, which are most of the path, and turns at the ends, and the error that accumulates is the turning. At a hundred and twenty seconds it reaches 19.2 m against a bound of 26.5 m, which is the closest any of the three comes.

What the runs establish is that the propagation tracks a partner behaving as this one behaves. They do not establish that the bound is a useful guarantee, and Proposition 2 says it is not one in the case it was presumably written for.

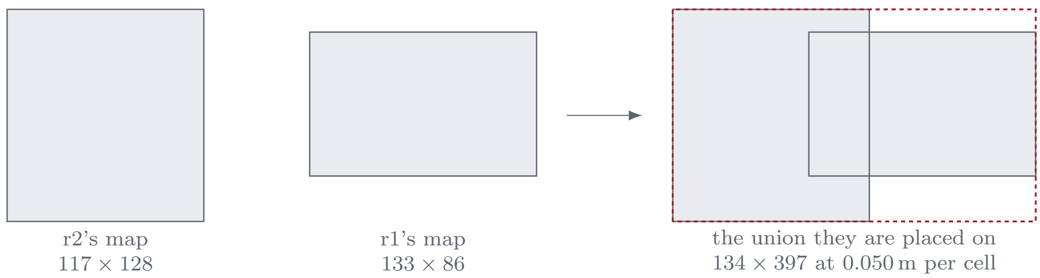
5 Reconciliation on maps two robots built

The collaborative layer classifies cells against thresholds, computes per-robot coverage, detects contradictions and reconciles. It passes forty-three tests. Every one of them builds both grids in memory, and a test that builds both grids builds them the same shape.

Two maps that two robots actually built are never the same shape. Each grows as its own robot discovers floor, and the first pair this reconciliation was ever handed came back `grids differ in size: 118x128 and 134x86`.

Remark 3 (A refusal its own tests could not provoke)**.** The refusal is correct and is not weakened here. Two maps whose discretisation does not coincide are declined and not approximated, because a wrong registration produces a merged map that is confidently wrong, which is worse than one that is obviously incomplete. What is worth recording is that the refusal path had never executed: the suite covered the fusion and not its negative, and not through carelessness. The bench cannot produce the situation that provokes it, because the situation is two independent map processes and the bench has none.

The missing operation is placement, not registration. Both maps come from the same simulator at the same resolution, so putting them on a common extent is a translation by whole cells: no interpolation, no rotation, no scaling, and a cell keeps the value it had. Maps at differing resolutions are still refused, because putting those on one grid would be the approximation the layer declines to make.



Extents, not contents. The overlap is where the two robots both have an opinion and is where a contradiction could arise; in the runs reported here it is empty of disputes, because the believing robot holds its dock and the partner patrols elsewhere. The union grows as both maps grow, so the figure is of one instant and the cell counts of Table 4 are of the instant the link returned.

Figure 1: What `grid_align` does. It is a whole-cell translation onto the union of the two extents, and it exists because the layer refuses a pair it cannot place, which is the right refusal and had never been reached.

Table 4: What the fusion reported at reconnection. The counts are the fusion's own, from its reply, and are not a recount: an earlier version of the recorder differenced cell counts it had subscribed to itself and reported a merged map with fewer decided cells than one of its inputs, because the aligner keeps publishing as both maps grow and the pair in hand when a reply arrives need not be the pair the fusion was given.

Outage	Cells	Believer learns	Partner learns	In conflict
30.0 s	65 505	5 537	4 917	0
60.0 s	65 505	5 556	4 872	0
119.9 s	65 505	4 879	4 903	0

Each robot learns from the other close to the count the other had decided, and nothing is in conflict. That is not the reconciliation being clever; it is what a disjoint pair of observations looks like. The believing robot holds its dock and maps what it can see from there, the partner patrols elsewhere, and the two sets of observed cells barely meet. A conflict requires two robots to have observed one cell and disagreed about whether it is passable, and this pair never did.

The zeros are therefore a property of the trajectory and not a result about the fusion. Producing a genuine contradiction requires routing the two through one aisle from opposite ends, which is the next run and not this one.

6 Defects

Three, and the third is the one worth carrying.

Table 5: What was found by running the integration, and how each presented.

Defect	Presentation
Stale binary interface	Every action node died at startup with `undefined symbol: ActionExecutorClient::finish(bool, float, string const&)`. The installed binaries were built when `finish` took three arguments; it now takes four, the fourth being the outcome the epistemic extension needs. A rebuild fixes it, and nothing warns that an installed tree is older than the library it links.
The fleet cannot drive with two robots	One robot's global costmap never receives its map, keeps the default five-metre extent at the origin, reports the robot out of bounds, and the navigation cannot plan. The topic exists with one publisher and five subscribers and the costmap's own parameter names it. Not diagnosed further; the scenario drives the partner directly instead.
The patrol drove into the racking	An open-loop square on a floor with shelving on it. The consequence was not a crashed run but a flattering one: a partner held against a rack does not move, and a propagation that integrates the last twist tracks a stationary partner exactly. See Remark 2.

The third is a defect in the instrument and not in the system under test, and it is the kind this series keeps finding: not a component that fails, but an arrangement in which every component behaves correctly and the number that comes out measures something other than what was intended. The gate discarded its messages, the propagation integrated its twist, the recorder compared against the ungated truth, and the answer was still wrong, because the subject of the measurement had stopped moving for a reason nobody had asked about.

7 What the runs establish

That an outage can be injected into a running fleet without modifying it, and that the cut is real and counted: 1 764 odometry messages and 64 map messages discarded in the sixty-second run, against 1 179 and 41 that passed before it.

That the propagation exists, grows its covariance monotonically, and reaches the threshold past which it declares its own estimate unfit to be believed.

That its acceptance criterion is met over outages of thirty, sixty and a hundred and twenty seconds against a partner in motion, with the qualification of Proposition 2 attached.

And that the reconciliation runs on maps two robots built from two lasers in a simulator, which required an operation the layer did not have and which its own tests could not have provoked.

8 What they do not

They do not establish that the epistemic models diverge under partition, because there is one model for the fleet. Two robots that stop communicating do not thereby acquire two structures to disagree with each other, and an accessibility relation here is a column of a shared structure and not a perspective held on a vehicle. Section 2.2 states this and Section 9 says what it would take to remove it.

They do not establish that the fused state satisfies the goal at reconnection by the validator's account, for the same reason. There is one model, it never diverged, and a check against it would be a check that nothing was partitioned.

They do not establish anything in favour of the bound.

And they do not establish that the fusion resolves contradictions well, because it was never given one.

9 What would remove the limitation

The obstacle is one shared model, and the path to two is shorter than it looks. Three properties of the existing code decide it, and all three were checked and not assumed.

1. **The state node namespaces cleanly.** Every service and topic name in it is relative; the package contains no absolute name. Launching it under `r1` and again under `r2` yields `/r1/epistemic_state/apply_action` and its counterpart, two independent models, with no change to the planner.

2. **A reconciliation primitive already exists.** `epistemic_state/announce` takes a formula, keeps the worlds where it holds, and increments a belief version precisely because the information arrived outside the plan. Replaying into a model what it missed is one call per missed finding.

3. **The gate already cuts where it would need to.** It gates by topic and type; the path carrying an update to an absent agent's model is one more entry in a launch file.

What would have to be written is a router: today the executor calls one model, and it would have to call each, applying the observability rule the domain declares. That is new code in the scenario package, not a modification of the planner.

One caveat belongs with the proposal. `announce` implements a public announcement, so replaying a missed private one makes its content common knowledge in the model that missed it. Reconciliation by replay therefore returns a model that is *compatible* with the one that never lost the link, and not identical to it. That is the right target and not a shortcoming: what reconciliation should restore between two structures that observed different things is their compatibility, and requiring equality would falsify one of them to preserve a consistency the domain does not demand.

Reproduction

```
colcon build --packages-select epistemic_msgs epistemic_comm epistemic_slam
colcon test --packages-select epistemic_comm # 62 tests

ros2 launch epistemic_comm link_outage_launch.py \
    t_disc:=70 t_recon:=130 scenario:=baseline
```

`t_disc` and `t_recon` are seconds from the monitor's first clock tick. Two records are written per run, one of the belief error against the bound at every sample and one of the reconciliation, and the figures in Tables 3 and 4 are read from them.