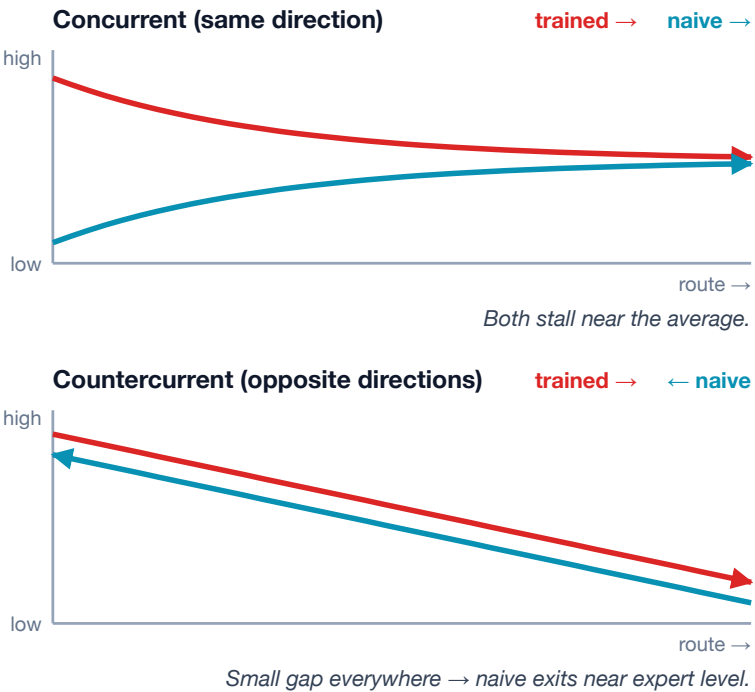


A duck keeps its core warm and its feet cold by running warm and cold blood side by side in opposite directions. Ants on a trail swap news every time they pass. Both move value *locally, along a gradient* instead of through a central pump. NemoFleet can do the same with knowledge and GPU spend.

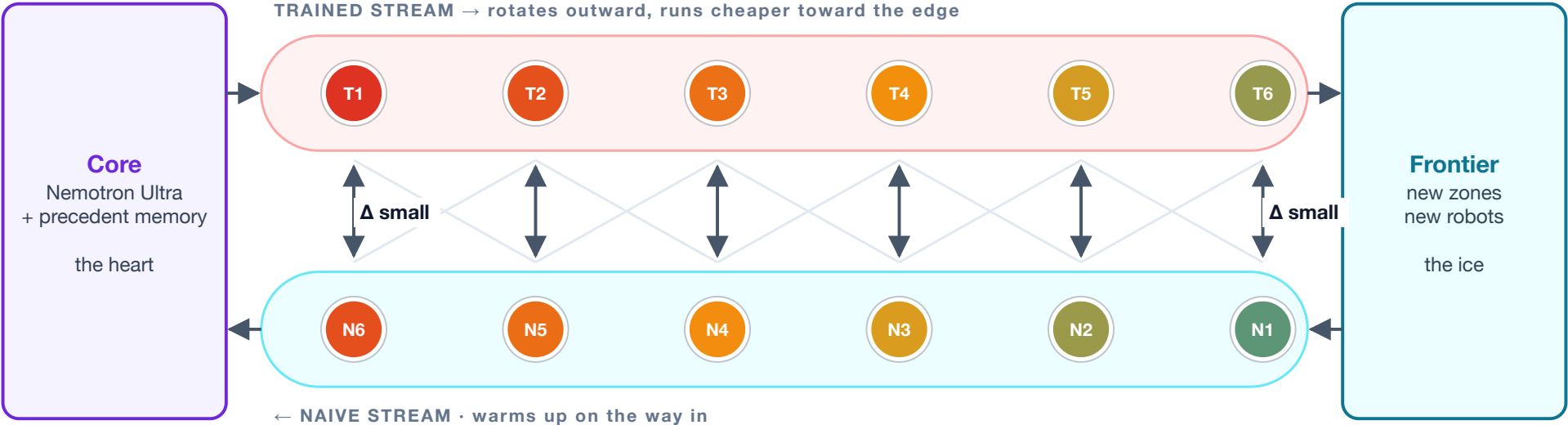
BIOLOGY	NEMOFLEET
Heat	Competence: precedents, calibrated policies, resolved anomalies
Warm arterial blood	Trained (“hot”) robots, or knowledge flowing out from the core
Cold venous blood	Naive (“cold”) robots, or raw unresolved experience flowing back
Body core and heart	Cloud: Nemotron Ultra + precedent memory (expensive heat)
Foot on ice	Frontier: new zones and new deployments
Heat lost to the ice	GPU and tokens spent re-deriving what the fleet already knows
Countercurrent flow	Gradient-ordered pairing: every encounter transfers something new
Rete mirabile	Dense peer-to-peer exchange mesh that multiplies contact points
Ant antennation	Handshake when robots pass: “what’s ahead” ⇔ “what’s behind”



Where the analogy breaks. Heat is used up when it moves; knowledge is not, since sharing a precedent doesn’t make the trained robot forget it. So the quantity actually conserved is **compute and energy**: every exchange that settles a question locally is an Ultra call not made. What is scarce in each encounter is contact time and context budget. That is why the gradient matters: each pass should carry a small, relevant difference.

Rete mirabile: every robot passes many peers, each only slightly more (or less) trained

Peer exchange settles low-confidence calls locally; only unresolved residue reaches Ultra



1 · Enter cold

New robots enter at the frontier and travel toward the core. Trained robots rotate outward, toward the edge.

2 · Small steps

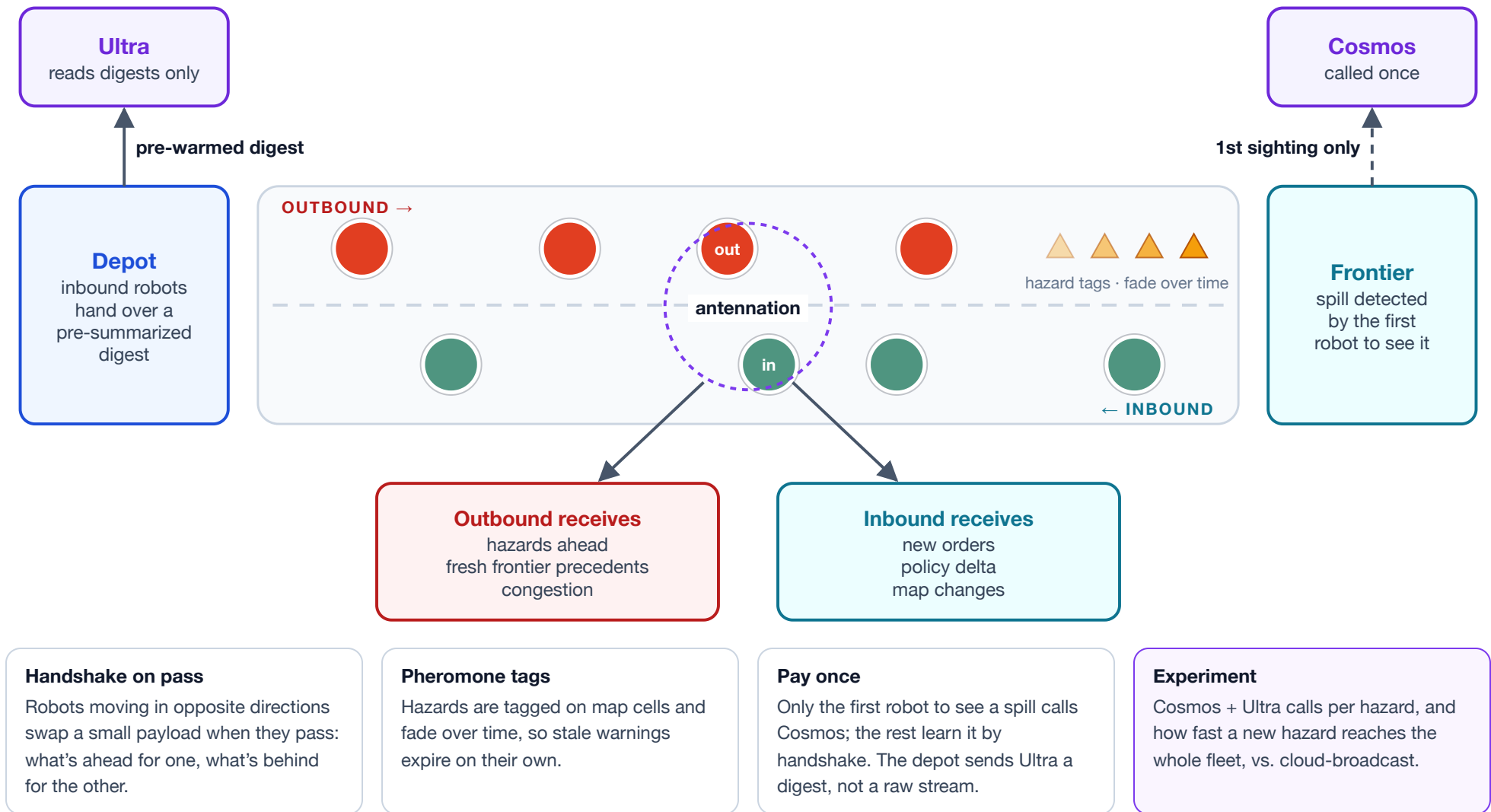
Each robot first meets peers only slightly more trained, then progressively more expert ones: a built-in curriculum.

3 · Settle locally

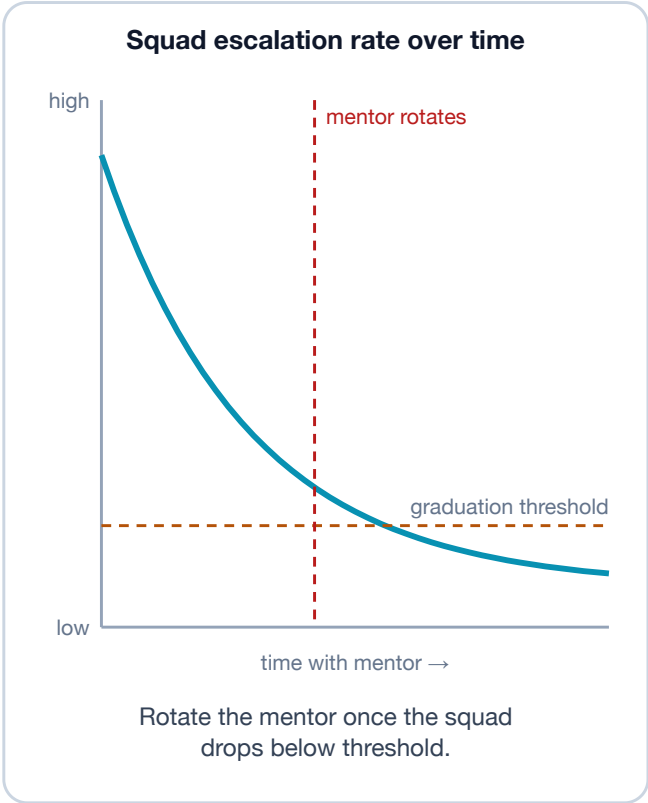
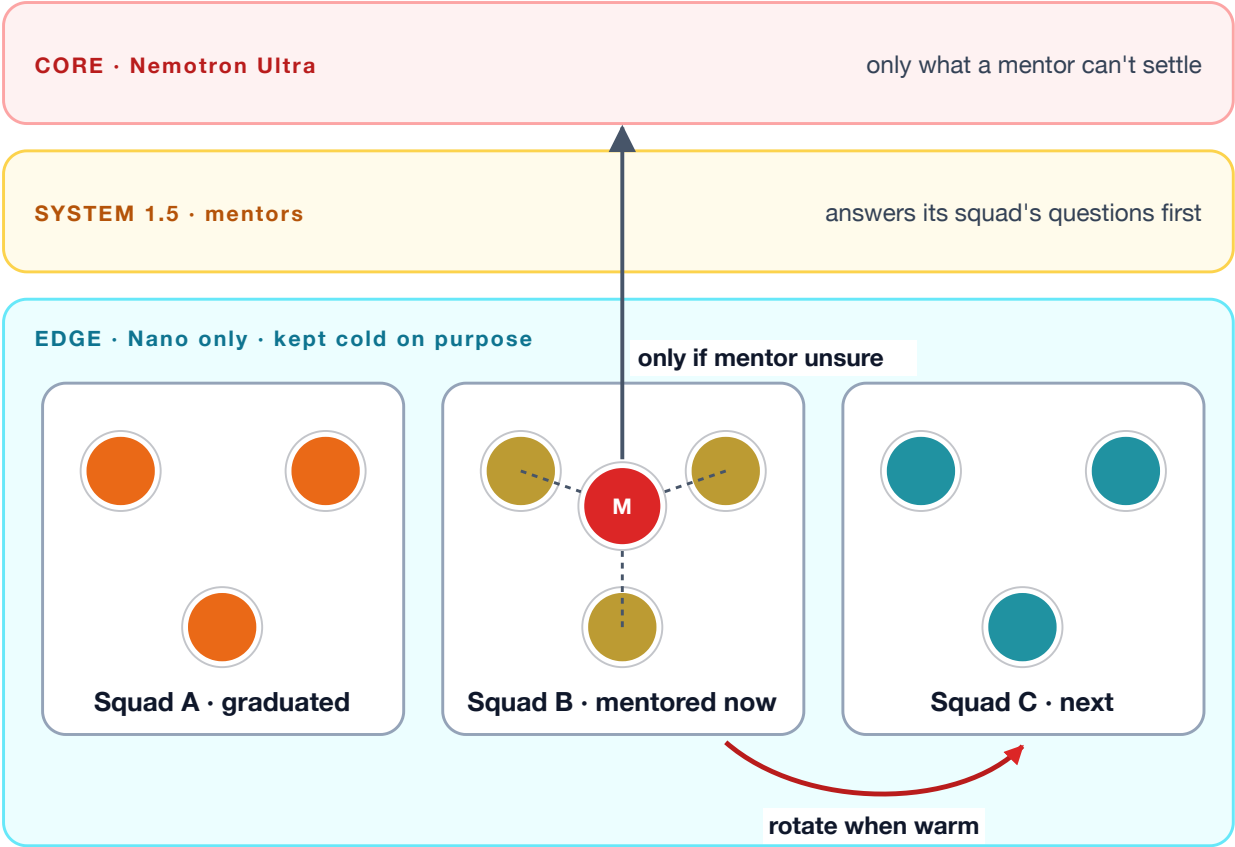
Peer exchange (Nano + shared precedents) settles low-confidence calls. Only the unresolved residue reaches Ultra.

Experiment

Warm-up speed and Ultra calls per novel event: countercurrent vs. concurrent (random pairing) vs. central-only.



Mentor mingling with thermal zones



Cold edge by design

Edge robots run Nano only. Heavy compute stays in the core, like the duck keeping heat in its body.

System 1.5

One trained robot per squad answers its peers first. Naive robots learn from the mentor's decisions and warm up.

Rotate the heat

When the squad drops below the graduation threshold, the mentor moves to the next cold squad.

Experiment

Compute cost per task by zone, and squad warm-up time vs. rotation interval.

Which one to build

	A · Countercurrent curriculum	B · Ant-trail antenation	C · Mentor mingling
Biological model	Duck rete mirabile	Ant trail antenation	Duck thermal zoning
Mechanism	Gradient-ordered peer exchange	Handshake when robots pass + fading map tags	Embedded mentor per squad, rotated
What it saves	Ultra calls on already-solved problems	Cosmos + Ultra calls per hazard; raw-stream bandwidth	Compute at the edge
Demo visibility	Medium: shows best as a chart	High: visible on the map in seconds	Medium: visible squads + rotation
Build effort	Medium: pairing scheduler + memory sharing	Medium: proximity trigger + payload + tags	Low: one extra escalation tier
Risk	Bad precedents spread too	Stale or conflicting news	Mentor becomes a bottleneck
Headline metric	Warm-up speed; Ultra calls per novel event	Calls per hazard; fleet-wide spread time	\$ per task by zone; warm-up time

Recommendation: B on screen, A in the numbers

Use ant-trail antenation as the visible demo mechanic: judges understand it in five seconds when two robots pass and the hazard spreads. Back it with Option A's countercurrent vs. concurrent experiment as the quantitative result. Both run on the existing Serverless Jobs frontier pipeline. Option C is the simplest fallback if time runs short.

Where it plugs into the architecture

`cognition/memory.py`: shareable precedents with provenance
`cognition/gate.py`: ask a peer or mentor before Ultra
`sim/world.py`: proximity events and fading map tags
`jobs/redteam.py`: pairing strategy as a sweep dimension