Challenges for Private Agents within Agentic Frameworks

Prof. Dr. Petre Dini, ex-AT&T/Cisco Systems, Inc., USA (retired)
Prof. Dr. Eugen Borcoci, National University of Science and Technology POLITEHNICA Bucharest, Romania

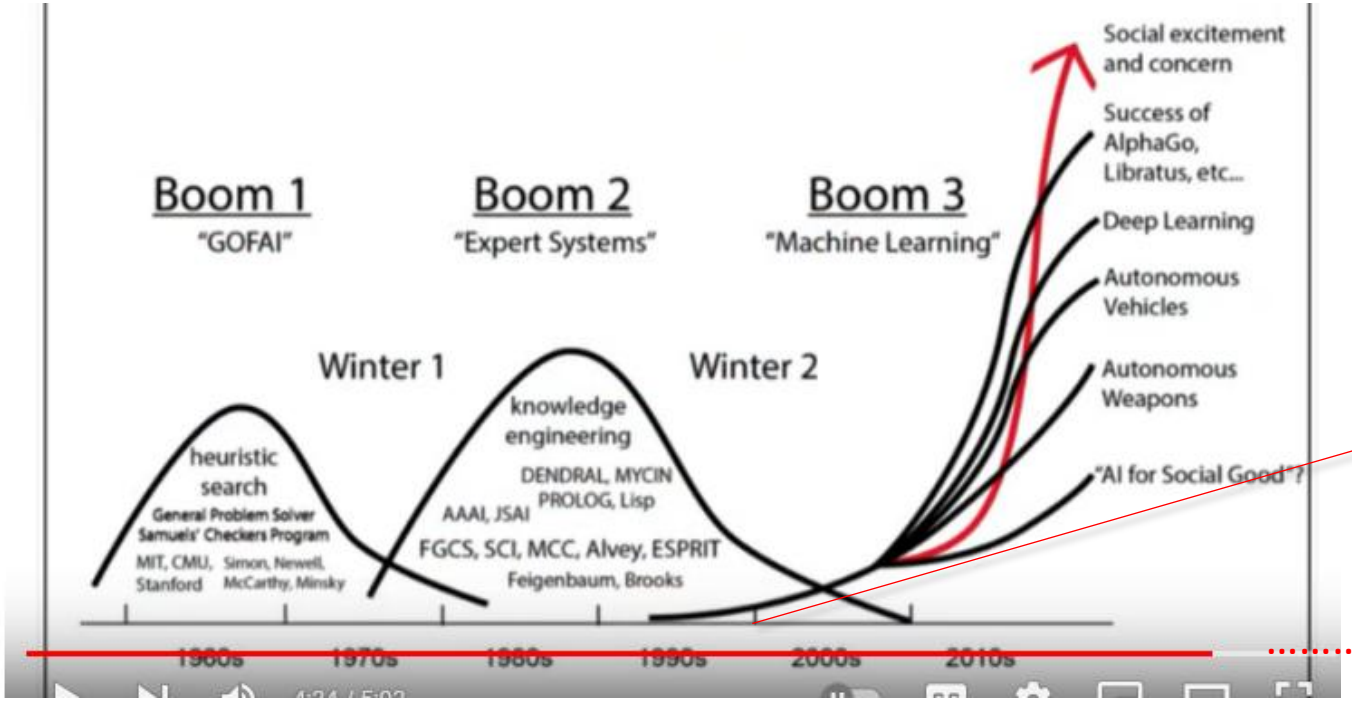
Themes

- Building Agents
- Corporate and Private Agents
- Methodology
- Agentic Frameworks
- Services and Supporting Frameworks
- Use Case – HealthCare Agents
- Synchronization
- Coding and Library Conflicts

AI and Intelligent Agents

Rules, Expert systems, → ChatGPT, Agentic frameworks, Private Agents, etc.

Symbolic methods, Lisp, Prolog, A*, → LLMs, LCMs, Hybrid AI, etc.



1992 IBM Smart Phone

Apple Inc.
OpenAI
Microsoft

2022, ChatGTP, LLMs, LCMs, Agentic

2020 |.. 2026

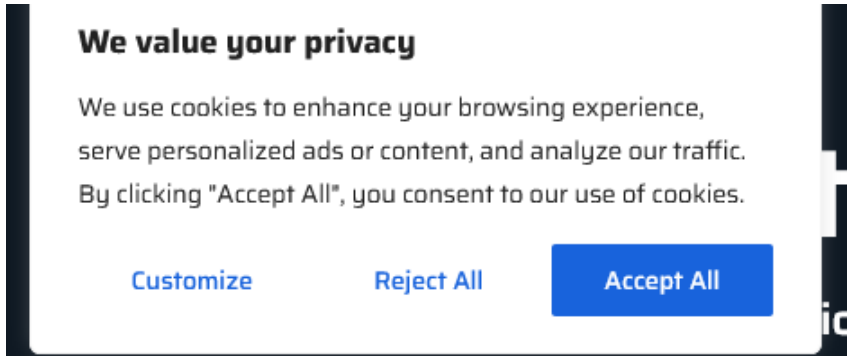
Building Agents

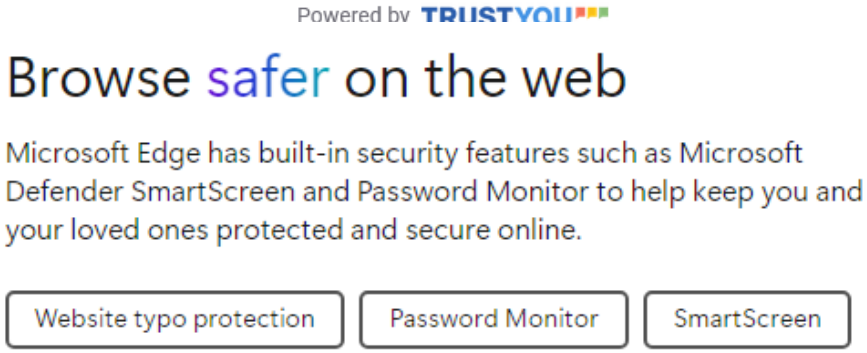
- **Gradual negotiation with the technologies around us - Tom Chatfield**
- **Societal evolution**
  - Paper, clock, phone, (multiple-devices), smartphone, chat bot, .. private agent
- **Co-evolving with Technologies**
  - Reluctance, attempts, learning curve, digital hassle, gradual use, gradual delegation
- **Agentic approach**
  - Agentic frameworks, standard agents, speciated agents, …private agents. Personal assistants
  - **Q:** if, when, on what, how to, what if, unexpected situations, damage control, trust, ethics, benefits
- **OpenClaw Personal AI Assistant case study**
  - AI agents with real system access represent a genuinely new risk category.
  - The approach isn't to avoid them; it is to be deliberate about the ***boundaries*** we set.
  - **OpenCloos-Secure**: it runs OpenClaw in a fully isolated VM + Docker environment, where it is ***physically impossible*** for the agent to touch personal files, emails, credentials, or messaging accounts by design, not just by configuration. (***limited delegation*** and ***full owner control***)

OpenClaw is currently the most starred project on hashtag#GitHub; a self-hosted AI agent that connects to your WhatsApp, calendar, email, and runs commands on your machine. The promise is remarkable. The security record, less so: 40,000+ exposed instances, plaintext credential leaks, a critical WebSocket hijacking vulnerability, and up to 41% of its community skill marketplace found to contain malicious code.

Arash: https://arash-hajikhani.com/

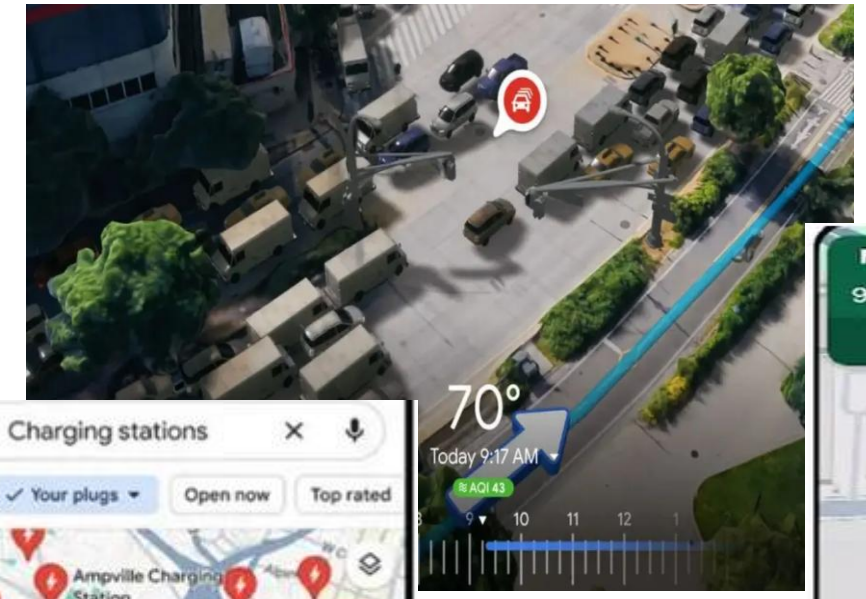
Learning and accepting the digital environment (trusting others' agents)





GPS Embedded Agents

Google Maps gets a massive **AI upgrade** with 5 new features
The latest updates to Google Maps makes it smarter and more helpful
https://www.foxnews.com/tech/google-maps-gets-massive-ai-upgrade-5-new-features



3D with Immersive View

Corporate/Private Agents

Awareness: Uninformed (ignorance) and Informed (hesitation)

- **Digital/AI harassment**
  - Extra features
  - New apps
  - New enrollments
- **Interfaces technologies (self-embedded behavior)**
  - Multilevel interaction design (multiple clicks), not always intuitive
  - Y/N positions
  - Setting personalized cookies
  - Handling backups
- **The power of Digital/AI enforced by**
  - Practicing, learning, trusting
  - Mental upgrade and digital acceptance (through Digital Literacy)

- **Downsides on relying on agents**
  - Digital/AI dependency
  - Digital/AI addiction
  - Digital/AI-infused deskilling
  - Mental and health disturbance
  - Frustration by interaction complexity
  - Frustration by losing control

- **Classical AI Assistants** (reactive tools)

  answer questions
  execute commands
  provide recommendations

- **Private Agents** persistent autonomous entities

  accumulate contextual knowledge
  learn user preferences
  negotiate with other agents
  make delegated decisions
  represent user interests over time.

Q: Will future humans use private agents as tools, or gradually evolve into humans represented by agents?

- Identity and Representation
- Privacy and Knowledge Ownership
- Trust and Verification
- Multi-Agent Negotiation and Conflict
- Autonomy versus Human Control
- Economic and Societal Asymmetry (examples)

  wealthy users obtaining superior agents
  asymmetry of negotiation capabilities
  cognitive augmentation inequality
  corporate-controlled ecosystems
  algorithmic influence monopolies

Methodology (i)

Methodological Approach for Acquiring or Building a Private Agent within an Agentic Framework

Before choosing a framework "I want a personal agent." vs. "I want an agent with specified operational, cognitive, privacy, autonomy, and interoperability characteristics."

STEP I (Requirements space)

Functional requirements: (roles): scheduling, information retrieval, negotiation, document management, purchasing, healthcare assistance, coding, financial monitoring. nput: use cases, operational scenarios, user stories, etc.

Cognitive requirements: (expected intelligence behavior): reasoning depth, memory persistence, contextual awareness, personalization level, long-term learning, emotional simulation, explainability, etc.

Autonomy requirements: (boundaries): assistant only, recommendation engine, supervised executor, delegated negotiator, semi-autonomous operator (+ escalation conditions)

Privacy and Data requirements: local-only storage, cloud synchronization, memory retention duration, encryption, data portability, right-to-forget mechanisms, etc.

Multi-Agent Interaction requirements: negotiate with other agents? access marketplaces? delegate subtasks? cooperate with institutional agents? (via trust protocols, identity verification, and behavioral governance compatibility).

Resilience requirements: offline mode, fallback operation, degraded safe mode, rollback capability, hallucination containment, adversarial resistance, etc.

Methodology (ii)

STEP II Existing Similar Services (classify existing solutions)

Assistant vs Agent: reactive assistant, workflow orchestrator, autonomous agent, multi-agent participant, etc.
Closed vs Open Ecosystem: proprietary platform, API-extensible, plugin-based, federated ecosystem, etc.
Local vs Cloud: fully cloud, hybrid, edge-based, fully local/private, etc.
Generic vs Domain-specialized: general-purpose, healthcare, finance, legal, industrial, research-oriented, etc.
Build an Evaluation Table: candidate, agent level, memory, multi-agent, local control, explainability, extensibility

STEP III - Compare Agent-Building Frameworks

Architectural Comparison: centralized or distributed? event-driven? orchestration-based? tool-calling architecture? agent-to-agent protocol support?
Memory Model: short-term vs long-term memory? vector memory? symbolic memory? episodic memory? memory pruning? contextual persistence?
Governance and Safety: role-based control? bounded autonomy? auditability? policy enforcement? human override? behavioral constraints?
Extensibility: plugin architecture? external APIs? sensor integration?ustom tools? multimodal support?
Interoperability: standard protocols? multi-agent communication? MCP-like interfaces? federated identity? agent marketplaces?
Explainability and Observability: can you inspect reasoning? replay decisions? trace actions? inspect memory evolution?
Economic Model: subscription, token consumption, compute-based billing, per-agent licensing, enterprise orchestration fees.

Methodology (iii)

STEP IV (Identify Service Levels) Agentic Service-Level Agreements (A-SLA)

Basic. Level 0 - Conversational Assistance
Reactive only. Level 1 - Persistent Personalization
Memory and preference adaptation. Level 2 - Tool-Oriented Execution
Can execute constrained tasks. Level 3 - Delegated Operational Autonomy
Can act independently within policies. Level 4 - Multi-Agent Representation
Can negotiate and cooperate with external agents. Level 5 - Adaptive Cognitive Companion

STEP V - Subscription and Operational Prerequisites

Identity Requirements: verified identity?, enterprise identity?, agent certificates?, trust credentials?, etc.
Infrastructure Requirements: cloud dependency, GPU requirements, edge hardware, bandwidth, latency constraints.
Governance Acceptance: (user to accept): monitoring, behavioral logging, liability clauses, delegated permissions.
Regulatory Constraints: (compliance): healthcare, finance, defense, education, children-related systems

STEP VI — Suggested additional methodological layers (how is long-term alignment preserved?)

Exit Strategy *(Can the user):* migrate memories? export knowledge? transfer identity? revoke permissions completely?
Dependency Risk Assessment *(what if):* the provider disappears? APIs change? subscriptions end? governance policies change?
Cognitive Dependency Assessment decision dependency, memory dependency, autonomy erosion, etc. *(user progressively losing capability due to delegation):*
Ethical Alignment Drift (over time): user changes, agent changes, providers change, policies evolve.

Agentic Frameworks

There are plenty of agentic frameworks to choose from (random examples and their main approach/feature

Frameworks	Main approach/feature
1. LangChain / LangGraph	1. Stateful graph-based orchestration
2. CrewAI	2. Role-based multi-agent collaboration
3. Microsoft AutoGen	3. Conversational agent ecosystems
4. OpenAI Agents SDK	4. Lightweight tool-oriented agents
5. Google Agent Development Kit (ADK)	5. Scalable multimodal agent infrastructure
6. Microsoft Semantic Kernel	6. AI orchestration for enterprise software
7. Pydantic PydanticAI	7. Structured, validated AI execution
8. Letta	8. Persistent memory-centric agents
9. NVIDIA ACE / Agentic stack	9. High-performance agent infrastructure
10. Anthropic agent workflows	10. Constitutional/safety-oriented orchestration

Long-running private agents, healthcare workflows, enterprise orchestration

Research assistants, SME automation, collaborative agents

Multi-agent reasoning, enterprise copilots

Practical production assistants

Large distributed intelligent services

Business processes, enterprise copilots

Safety-critical workflows

Personal assistants, longitudinal interaction

Robotics, avatars, digital humans

Sensitive-domain assistants

A. Workflow-Centered Agentic Systems

Examples: LangGraph, Semantic Kernel, PydanticAI

Characteristics: bounded autonomy, deterministic checkpoints, governance-friendly, easier certification, safer for regulated domains

B. Emergent/Social Agentic Systems

Examples: CrewAI, AutoGen, Letta-like persistent agents

Characteristics: negotiation, adaptive interaction, evolving memory, agent societies, higher creativity, higher unpredictability

Use Case: Healthcare - Agents

PORTO 2026

Telehealth triage & e-prescription scenario; **agentic coordination under strict safety, privacy, and auditability constraints**

The increasing pressure on healthcare systems to deliver rapid, accurate, and equitable medical responses has intensified interest in **telehealth triage augmented by intelligent agents**. In such settings, patients interact remotely with AI-assisted services capable of preliminary symptom evaluation, prioritization, and e-prescription. The incentive lies in creating an **agentic ecosystem** where distributed digital actors — diagnostic agents, privacy guardians, audit agents, and prescribing modules — collaborate autonomously yet transparently to ensure that each clinical and administrative action respects medical ethics, data protection laws, and accountability standards.

Building this environment demands **traceable goal alignment** among agents: safety goals constrain diagnostic decisions, privacy goals regulate data access and transfer, and audit goals enforce non-repudiation and explainability. The overarching motivation is to design an intelligent, self-monitoring telehealth process that maintains human oversight while improving timeliness and consistency. From these drivers emerge the system requirements — for secure reasoning pipelines, consent-aware data flow, verifiable e-prescription issuance, and adaptive escalation to human clinicians — forming the structural and behavioral blueprint of an agentic telehealth framework.

13

Healthcare – Goals (i)

PORTO 2026

Translating Requirements → Goals → Agents (mini example)

B1. Requirements (excerpt)

R1 Patient safety first; critical symptoms must escalate to human within 2 min.
R2 HIPAA privacy; minimum necessary data, consent tracking, audit logs.
R3 e-Rx authenticity; cryptographic signing + pharmacy verification.
R4 Latency: triage advice ≤ 5 s p95; video intake stable at 30 fps.
R5 Bias control in triage recommendations; explanations required.
R6 Data provenance for training/QA; no raw PHI leaves region.
R7 Fallback: degraded service during outages (phone/SMS, cached rules).

B2. Goals (what?)

G1 SafeTriage (critical→human ≤2 min; advice ≤5 s; explainable).
G2 PrivacyCompliance (HIPAA, consent, minimization, DSR).
G3 AuthEPrescription (sign e-Rx; verify at pharmacy).
G4 MeetLatencyBudget (p95 ≤5 s; video QoS maintained).
G5 Provenance&Audit (append-only, region-bound).
G6 BiasMonitoring (drift/inequity alarms; periodic fairness reports).
G7 Resilience (graceful degradation path active on SLO breach).

B3. Agents (roles)

Orchestrator/Goal Manager — decomposes & routes requests, enforces priorities.
Triage Agent — symptom NLP + rules/ML; advisory on severity.
Clinical Safety Agent (Runtime Assurance) — safety authority; escalates to human; Privacy & Compliance Agent — consent checks, field minimization, policy gate.
e-Rx Crypto Agent — HSM-backed signing/verification, key rotation.
QoS Agent — monitors latency/video metrics; triggers adaptive bitrate/throttling.
Bias & Drift Agent — monitors cohort outcomes; gates model updates.
Provenance/Ledger Agent — immutable logs, regional storage policy.
Fallback/Resilience Agent — activates phone/SMS workflows; caches top rules.

Healthcare – Goals (ii)

PORTO 2026

B4. Allocation (sample):

G1 → Triage Agent (advice) + Clinical Safety Agent (escalation decision).
G2 → Privacy & Compliance Agent (gate on every data flow).
G3 → e-Rx Crypto Agent (sign/verify).
G4 → Orchestrator + QoS Agent (shed load, degrade bitrate).
G5 → Provenance/Ledger Agent.
G6 → Bias & Drift Agent (observe-only → periodic reports; veto
G7 → Fallback/Resilience Agent (policy-driven activation).

B5. Conflict patterns & policies

- **Latency vs. Safety:** If p95 > 5 s, **short-path rules** activate; Safety Agent keeps escalation budget ≤2 min.
- **Privacy vs. Explainability:** Explanations must not reveal PHI beyond consent; Compliance Agent redacts.
- **Throughput vs. Video QoS:** QoS Agent reduces bitrate/resolution before dropping sessions.
- **Model vs. Policy:** If Triage suggests "home care" but red flags present, Safety Agent overrides → human.

B6. Example temporal guards

IF red_flag_detected THEN escalate_to_human WITHIN 120s
IF queue_length > Qmax THEN enable_short_path_triage AND alert Ops
IF consent.revoked(user) THEN purge non-essential caches WITHIN 24h

B7. Assurance snapshots

Pre-deployment: spec-based tests (STL/MTL), failure injection (network, HSM), PHI leak scanner.
Runtime: monitors + signed decisions to Ledger; fairness drift alarms weekly; Simplex-style safety override always on.
Continuous: canary updates; post-incident reviews tied to provenance.

A1) Example personalized agent [Health]

Agent name: ProgressiveTrustScorer
Purpose: Score each purchase/session for bot/abuse risk *without* hurting latency; supports progressive checks (cheap→costly).
Capability contract (WHAT + SLA):
Inputs: { user_id, device_fingerprint, payment_hash, ip, behavior_signals[], cart_value }
Outputs: { risk_score∈[0,1], risk_band∈{low,med,high}, actions: {throttle?, challenge?, block?}, rationale[] }
Non-functionals: p95 ≤ 5ms, p99 ≤ 12ms, availability ≥ 99.95%, determinism on same inputs (seeded), no PII persistence.
Trust class: Advisory (cannot directly block; Orchestrator enforces).
Policies: No use of prohibited features (e.g., demographics), explainability required (rationale with top 3 features), signed responses.

Coding: Private Agent


```
{
  "name": "ProgressiveTrustScorer",
  "version": "1.2.0",
  "inputs": {
    "user_id": "uuid",
    "device_fingerprint": "string",
    "payment_hash": "sha256",
    "ip": "ipv4|ipv6",
    "behavior_signals": "array<float>",
    "cart_value": "float"
  },
  "outputs": {
    "risk_score": "float[0,1]",
    "risk_band": "enum(low,med,high)",
    "actions": {"throttle": "bool", "challenge": "bool", "block": "bool"},
    "rationale": "array<string>"
```


17

A2) Avoiding conflicts with the "agent library"

Design-time controls

1. Capability ontology & registry

1. Each agent declares **capabilities**, **pre/post-conditions**, **trust class**, **resource budget**.
2. Namespacing: `risk.trust.progressive.v1`. Prevents accidental role overlap.

2. Conflict declarations

1. Agents state **mutual-exclusion** or **precedence**:

```
1.conflicts_with: ["risk.trust.static"],
   precedence_under: ["queue.fairness"].
```

3. Policy contracts (ABAC/RBAC)

1. Who can *decide* vs. *advise*. Your new agent stays "advisory" unless governance promotes it.

4. Static conformance suite

1. Schema validation, SLA proof (bench harness), **negative tests** (reject PII, fail closed).

5. Formal invariants (lightweight)

1. Temporal specs like "never both `block` and `low` band"; Orchestrator checks at runtime.

Run-time controls

Arbitration layer in Orchestrator

Priority graph: Safety > Compliance > Fairness > Revenue.
Rule combiner: (deny-overrides, permit-overrides, first-applicable).

Resource quotas & circuit breakers

Cap CPU/latency; isolate misbehaving agents (bulkhead pattern).

Shadow/canary & auto-revert

New agent runs shadow; if SLOs or policy drift, auto-disable and alert.

Provenance & explainability bus

All decisions with signatures → Ledger; conflicting outputs are flagged for adjudication.

Treat your custom agent as a *plugin* with a *contract.*
Declare where it sits in the *hierarchy*, what it may never do, and *how conflicts are arbitrated*.

18

Synchronization (i)

Agents' behavior is driven by feedback from the real system (disregard simulation). The events from the system reach the agents after going up from the physical elements (e.g, CPU, routers) through many levels, and each attaches to the event its own timestamp (assume they all have a clock); Now, the events reach different agents. What is the mechanism for agent synchronization?

1. Time Sources and Synchronization Layers

Clock sync protocols:

- *NTP (Network Time Protocol)* — ms-level sync, good for distributed IT.
- *PTP (Precision Time Protocol / IEEE 1588)* — µs or ns-level sync, used in telecom, finance, avionics.
- *GPS-disciplined clocks* — absolute reference, often for high-assurance systems.

Local monotonic clocks: Each node uses a monotonic counter (not wall clock) to avoid regressions when NTP shifts.

2. Event Timestamp Strategies

Source timestamping: Events stamped at origin (e.g., NIC, router ASIC, kernel driver). Highest fidelity, but requires hardware support.

Ingress timestamping: Middleware adds a timestamp on receipt. Easier, but mixes propagation delays.

Hybrid: Keep *both* (source + ingress) so later agents can reason about uncertainty.

Synchronization (ii)

3. Event Correlation Mechanisms

Lamport clocks: Logical clocks — give a partial order ("happened before") without relying on wall time.

Vector clocks: Capture causality in multi-agent settings; more overhead but resolve concurrent vs. causal.

Hybrid logical clocks (HLC): Combine physical time with logical counters — practical for distributed systems like CockroachDB.

Causal message tagging: Propagate context (e.g., trace IDs in OpenTelemetry) so events can be reassembled in order.

4. Dealing with Clock Drift and Skew

Bounded uncertainty windows: Every timestamp is expressed as `[t ± δ]` where δ is drift + jitter. Agents reason over intervals, not points.

Temporal logic with slack: Instead of "event A before event B," you specify `A → B within [0, 50 ms]` *with tolerance*.

Resequencers / buffers: Small delay queues reorder events by timestamp before passing to reasoning agents.

Synchronization (iii)

5. Practical Tools / Patterns

Syslog limitation: As you noted, fields vary — solution is a **normalization gateway** that canonicalizes all logs/events into a shared schema (`event_time`, `ingest_time`, `source_id`, `uncertainty`).

Observability frameworks: OpenTelemetry / Jaeger / Zipkin use trace & span IDs + NTP/PTP to unify timing across layers.

Event bus guarantees: Kafka, Pulsar, etc., ensure ordering *per partition*; you still need clocks for cross-partition causality.

Complex Event Processing (CEP) engines: They implement "temporal windows" with late-event handling and watermarking.

6. In Agentic Systems (your setting)

Each agent sees "causal envelopes" not raw timestamps: e.g., `Event(A) @ [12:00:01.002 ± 3ms].`

Orchestrator/Time Service agent normalizes events:

- Aligns clocks (NTP/PTP),
- Canonicalizes timestamp fields,
- Adds provenance (which layer's clock),
- Issues watermarks ("safe up to T").

Agents' reasoning is then expressed in *temporal logics with slack* (e.g., "if login-failed followed by account-locked within 10s ±1s, trigger escalation")

Synchronization is achieved by **global clock sync (NTP/PTP)** + **event normalization (canonical timestamps)** + **logical/causal clocks** for ordering + **uncertainty windows** for safety. In practice, a dedicated **"Time/Provenance Agent"** often sits in the architecture to buffer, reorder, and annotate events before downstream agents consume them.

21

TO CONSIDER:

- Security/Privacy
- Uncontrolled actions
- Features interaction with the existing framer agents
- Conflict mediation
- ROI on having personal/private agents
- Maintenance and licensing conditions
- Rights on change and termination for automatic framework-driven actions

22