Transport and Substrate

A Meta-Conceptual Comparison of OpenTelemetry and Substrates

INTRODUCTION

OpenTelemetry (OTel) and **Substrates** both operate in the broad domain of how computational systems relate to their own behavior. Both define APIs. Both concern themselves with signals, with the flow of information about what a system is doing.

And yet, reading them side by side produces a sense of dislocation, as though two cartographers had mapped the same territory and produced charts of entirely different landscapes.

> The word substrate is itself a declaration. In biology, a substrate is the medium on which an organism lives. In chemistry, a substrate is the substance that an enzyme acts upon. A substrate that identifies itself as such establishes itself as a foundation for something else—a structure that will be built upon, signals will pass through it, and judgments will be made using it. It defines circulation and defers interpretation.

OTel carries no such self-limiting declaration. Telemetry, the remote measurement of quantities, defines the entire pipeline from instrumentation to delivery. It aims for self-contained completeness.

This paper examines the divergence at its deepest level: the meta-conceptual architecture of each specification. What each presupposes about the nature of computational systems, the role of signals within them, the location of intelligence, and the relationship between a system and its own state.

These presuppositions are encoded in structural decisions: what gets a data model, what gets a behavioral contract, what gets deferred to an external backend. They are legible only when the two specifications are placed in dialogue, so that the silences in one become audible against the assertions of the other.

ONTOLOGICAL COMMITMENT

> The most revealing way to see the ontological split is to ask: what is each specification's theory of waste?

OpenTelemetry's answer is implicit in its architecture: waste is missing data.

The specification aims to instrument comprehensively so that nothing of operational significance goes unrecorded. Sampling exists, but as a concession to cost, and the specification treats it with visible discomfort, devoting careful attention to ensure that even the act of not recording is itself recorded faithfully.

The primary ontological unit is the record, which is a structured container of attributes, timestamps, and identifiers. It's designed to be serialized, exported, and stored. The specification's architectural surface is devoted to moving these records outward: from creation through processing to export. The universe of OTel is a universe of things that have happened.

Substrates inverts this definition of waste. Waste is unnecessary propagation.

The processing pipeline is explicitly designed to suppress, filter, deduplicate, threshold, and compress signals before they reach downstream consumers. The diff operator drops repeated values. The integrate operator accumulates silently until a fire condition is met.

The first design principle of the specification, "determinism over throughput," asserts that the quality of signal processing takes precedence over the volume of data transferred.

The primary ontological unit is the emission, which is a typed value traversing a governed topology. The specification's architectural surface is devoted to the conditions under which values circulate: queue ordering, dispatch semantics, temporal contracts. The universe of Substrates is a universe of things that are happening.

OTel builds a wide pipeline. Capture everything, export everything, let the backend sort it out.

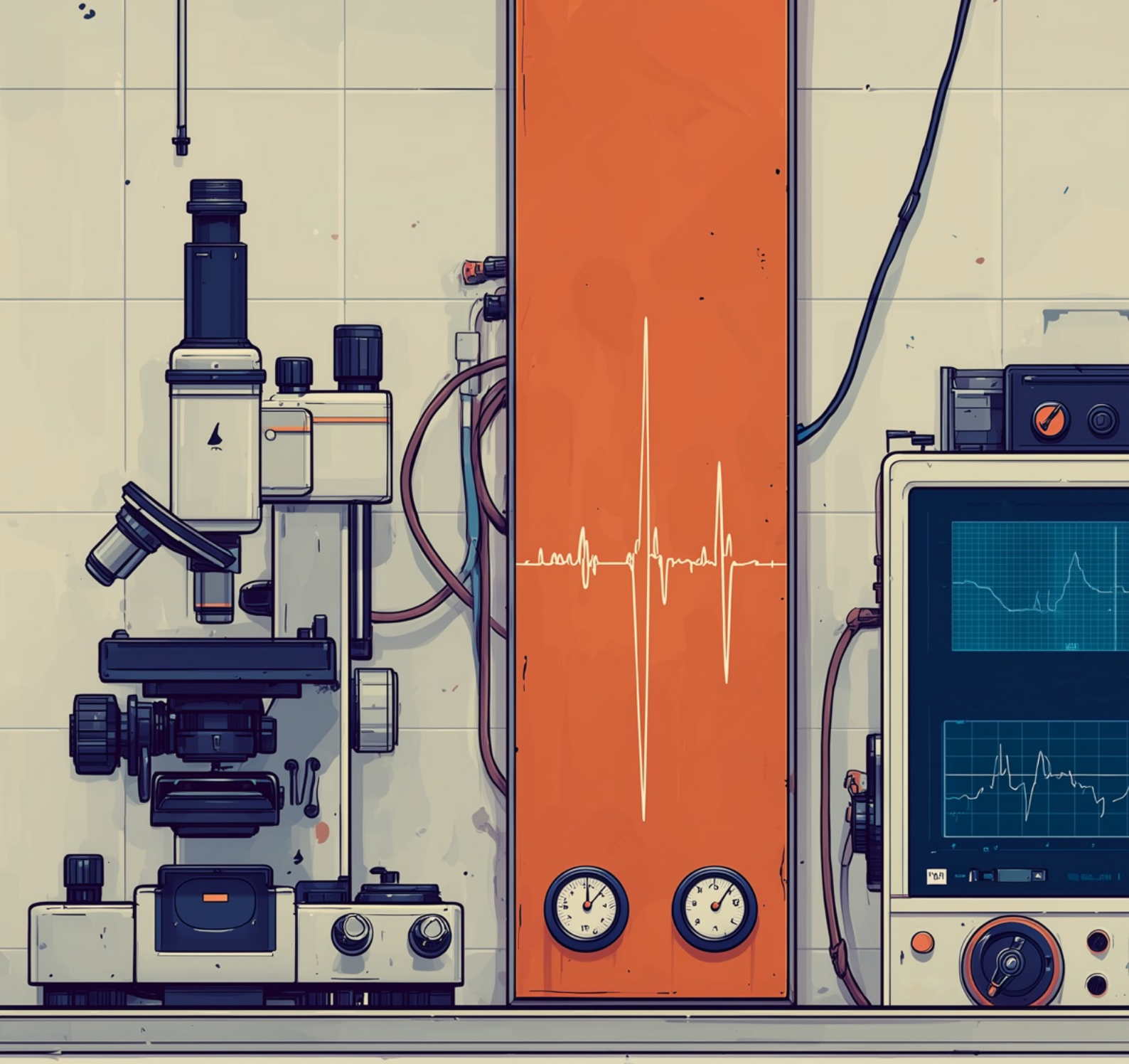
Substrates builds a selective circuit. Judge early, propagate only what matters.

> The ontological split extends to identity.

OTel's identity is correlational. Random tokens whose purpose is grouping records for later retrieval. Identity is a tag attached from outside, and an identity failure is a broken correlation: a record that exists but cannot be associated with its siblings.

Substrates' identity is constitutive. Hierarchical interned names that serve routing and pooling. Identity is part of what the entity is, and an identity failure is a routing corruption: the emission path itself is broken.

A record ontology produces correlational identity. A flow ontology produces constitutive identity.



EPISTEMOLOGICAL STANCE

Each specification embeds a theory of how a system comes to know something about itself.

OTel forms knowledge after the fact, through reconstruction.

A span is created, populated, ended, batched, exported, received by a collector, and eventually analyzed by a backend.

The knowing agent, such as the operator, dashboard, or alerting engine, operates outside the system boundary, downstream of multiple serialization and transport steps. The system itself generates records but forms no judgments.

Semantic Conventions standardize the vocabulary of these records, but interpretation is deferred entirely to the receiver.

The system produces evidence. Something else produces understanding. The system is a witness, not a knower.

Substrates forms knowledge at the point of reception.

A receptor receives a value within the circuit context, in the same execution boundary where the emission originated. The processing pipeline is an interpretive apparatus: operators filter, accumulate, compress, and threshold signals before delivery. The system processes its own signals and forms operational states within its own boundary.

The `integrate` operator demonstrates this commitment with particular clarity. It employs the *accumulate-until-fire* pattern, a biological model of perception where input accumulates until a threshold triggers a distinct recognition event.

Substrates, by including `integrate` as a first-class primitive, asserts that a computational system should perceive its signals within its own boundary.

The epistemological commitment is encoded in where each specification draws its architectural boundaries.

OTel enforces a separation between the code that emits and the code that interprets, through distinct package hierarchies with strict dependency rules. The separation encodes the assumption that signal production and signal interpretation are architecturally distinct, organizationally separate, and connected only by the transport of records across a process boundary.

Substrates collapses this separation. The entity that discovers a signal source defines the full processing pipeline for that source, within the same callback, in the same execution context. Production and interpretation share a boundary because perception, in this epistemology, is local.

CAUSAL MODEL

OTel models causation as lineage.

The trace is a directed acyclic graph of spans connected by parent-child relationships. Causation is represented by structure, specifically the topology of the graph, which indicates the relationships between different entities. Identity tokens propagate across process boundaries so that a backend can later reassemble the scattered spans into a coherent graph.

This is genealogical causation. Causation as ancestry.

The trace records ancestry, rendered after the fact from collected records.

Substrates models causation as propagation.

The dual-queue architecture, comprising one queue for external emissions and another for cascading emissions within the circuit, functions as a causal engine.

When processing an emission triggers further emissions, those effects drain completely before the next external emission begins. The *happens-before* relations formalize this: temporal ordering guarantees that causation is maintained in the act of processing.

OTel requires an external system to infer causal relationships from recorded data.

The quality of the reconstruction depends on the completeness of instrumentation, the reliability of context propagation, and clock synchronization across distributed nodes.

Substrates enacts causal relationships through its execution semantics.

There is no reconstruction step because causation was never disassembled.

One is archaeology. The other is physics.

THE ARCHITECTURE OF ATTENTION

OpenTelemetry is architecturally omnivorous.

The design principle is maximum coverage. The attention model prioritizes capturing information first, with filtering occurring later, downstream.

This filtering happens during sampling decisions, collector pipelines, and the backend's query engine.

The architectural consequence is a centrifugal system.

Data moves outward, away from the point of emission, toward external backends where analysis occurs.

The completeness is horizontal: breadth across the full data journey.

Substrates is architecturally selective.

The processing pipeline is an attention mechanism. Guards filter by predicate. Diffs suppress duplicates. Limits cap volume permanently. Integrate-and-fire accumulates silently until a threshold is met. These operators execute within the circuit context, before any downstream consumer sees the signal. The attention model is judge first, propagate selectively.

The architectural consequence is a centripetal system.

Data moves inward, toward the point of reception, through governed processing stages that shape, filter, and compress it before delivery. Every component is a stage in an inbound processing path.

The completeness is vertical: depth within the processing boundary.

Centrifugal architectures produce reports.

Centripetal architectures produce awareness.

TEMPORAL MODEL

OTel's temporality is stamped into data.

Time is a property of data, represented as a field on a record, a coordinate in a time series, or a boundary defining the duration of a span. Ordering is recoverable from these stamps, subject to the well-known challenges of clock synchronization across distributed nodes. **Time is extrinsic**: attached to data from outside, by the system clock at the instrumentation point.

Substrates' temporality is enacted through execution.

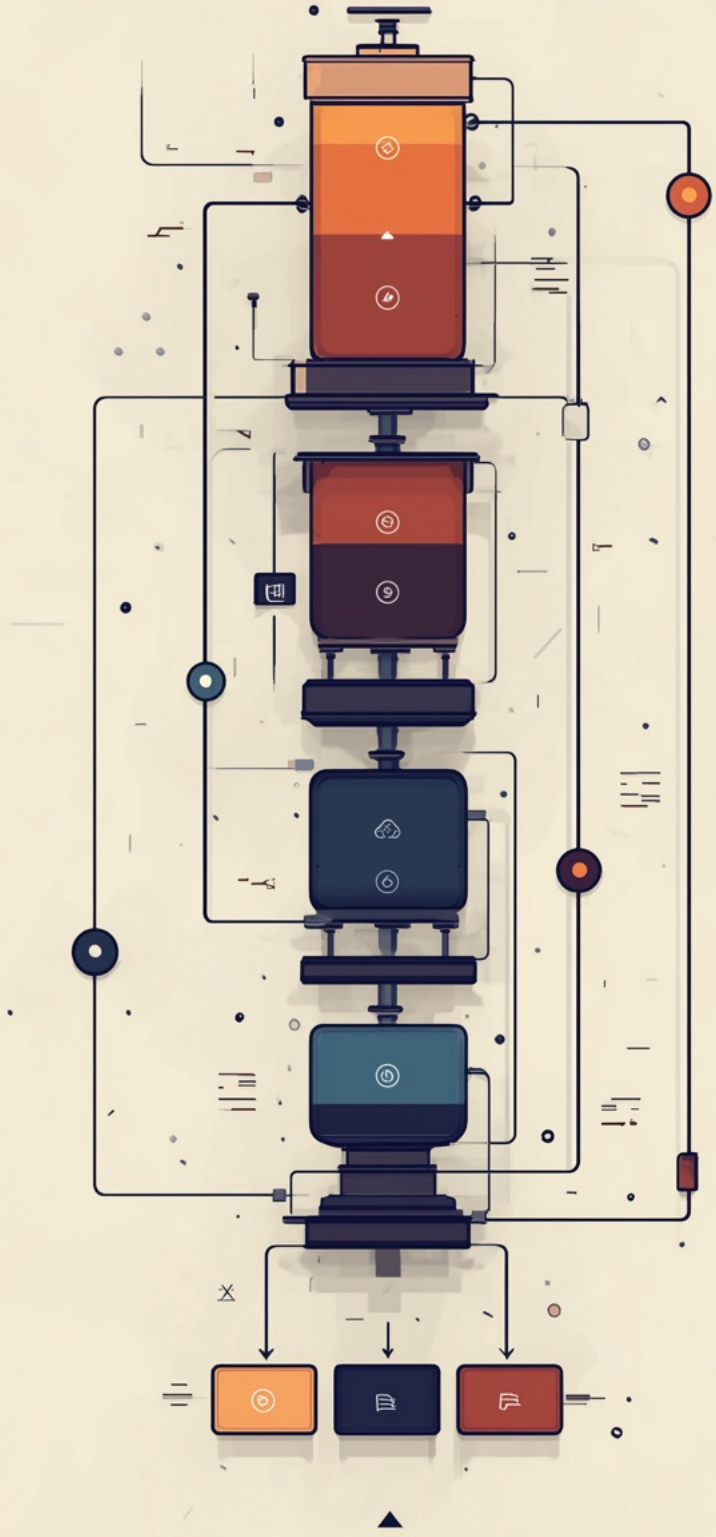
The specification contains no notion of timestamps. Ordering is guaranteed by queue discipline: emissions are observed in strict enqueue order, earlier emissions complete before later ones begin. **Time is intrinsic**: a property of the execution model itself, manifested as position in the queue.

The temporal contracts are concerned with lifetimes, providing guarantees about when something is real, expressed in terms of scope rather than clock time. Components come into being, exist within a bounded scope, and cease to be real when that scope closes. Using a component beyond its lifetime is a contract violation: an "illegal temporal use" error. This is temporality as ontology.

> The absence of timestamps in Substrates is architecturally significant.

The specification does not participate in the distributed clock synchronization problem because its temporal guarantees are expressed as ordering relations within a single sequential execution context.

The *happens-before* relations establish a partial order on operations, independent of wall-clock time. This approach is more akin to *Lamport's logical clocks*, but it eliminates the need for logical clock tokens altogether. The single-circuit-context guarantee ensures a total order within each circuit, thereby providing a complete ordering of operations.



THE GOVERNING QUESTION

> Every specification is an answer to a governing question.

OTel asks: How do we capture what happened in a distributed system and make it available for analysis?

The value proposition is forensic: the records exist to reconstruct what occurred. The system produces evidence for a downstream analyst.

Substrates asks: How do we construct a computational topology in which signals circulate with governed behavior so that a system can form its own situational awareness?

The value proposition is perceptual: the system processes its own signals, forms its own states, and produces its own recognition events. The system produces awareness for itself.

> These questions are not variations.
> They are orthogonal.

One is an instrumentation framework for data capture and export. The other is a signal substrate for cybernetic self-regulation. They share almost no conceptual surface despite both operating in the domain of "observability," because they hold fundamentally different positions on where intelligence about a system should reside.

OTel places intelligence outside.
Substrates places intelligence inside.

THE FINAL CONVERSATION

> The missing conversation is about the coupling of these two paths.

How does the data that OTel captures get connected to the governed judgment mechanisms that Substrates provides?

How does forensic evidence feed perceptual awareness?

How does the reporting system inform the awareness system?

OTel has built the afferent path. It is standardized and widely adopted.

Substrates has built the processing substrate for the efferent path. It is deterministic, governed, and compositional.

> The system that closes the loop between them, that takes OTel's sensory data and processes it through Substrates' judgment architecture to produce adaptive, self-regulating behavior, is the system that the observability industry has been groping toward without quite naming.

That system is not an observability platform. It is a nervous system.

Neither specification alone is sufficient to create it.