Ontology of Global Storage: Canonical Order and Naming

Alexey A. Nekludoff

AstraVerge Research

E-mail: an@astraverge.org
ORCID: 0009-0002-7724-5762

17 February 2026

Abstract

Distributed systems theory commonly incorporates local epistemic limitations (such as partial observability and node failure) into the ontology of storage. As a result, properties of observation and implementation are frequently treated as properties of stored data itself.

This work formulates an ontology of global storage in which a file is identified by a globally unique name, and truth is determined exclusively by a non-divergent canonical order of operations over that name. Failure and unavailability are treated as epistemic or implementation-level conditions and do not constitute ontological states of storage.

The ontology excludes partial failure, degraded modes, and divergent histories from its domain of being. Storage is defined instead in terms of global naming, canonical order, admissible histories, and horizon-dependent interpretation.

The result is a minimal, scale-invariant ontology of global storage that separates ontological structure from conditions of observation and implementation.

Contents

1 Introduction

Distributed storage systems are commonly described in terms of local computation. Message delay, partial visibility, replication lag, and node failure are treated as primary characteristics of distributed operation.

From these local conditions, global statements about storage are often derived. Data is described as temporarily absent, truth as eventually consistent, and conflicting views as natural system states.

Such descriptions conflate two distinct levels. Local computational limits characterize conditions of observation and access. They determine what a particular agent can observe or infer at a given time. They do not, by themselves, constitute properties of storage as such.

When epistemic constraints are incorporated into the ontology of storage, they give rise to additional modes of being: partial failure, degraded operation, and divergent histories. These notions describe implementation conditions, yet they are frequently treated as intrinsic properties of stored data.

This work adopts a different descriptive level.

Storage is defined independently of local execution, access latency, or replication topology. At the global level, naming and order are primary, while implementation mechanisms are treated as secondary realizations.

Within this framework:

- A file is identified by a globally unique name.
- Truth is determined by a non-divergent canonical order of operations.
- Failure and unavailability do not introduce ontological states.
- Observation may fail without modifying being.

The scope of this work is deliberately restricted to the ontology of global storage. It does not describe fault-tolerance mechanisms or prescribe architectural solutions. Its objective is to specify the minimal ontological structure required for storage to be meaningful at a global level, and to separate this structure from conditions of implementation and observation.

2 From Epistemic Limitation to Ontological Discipline

The persistent difficulty in distributed storage theory does not arise from implementation complexity, but from a shift in descriptive level.

Local computational limits — such as message delay, partial observability, or node failure — are epistemic constraints. They describe what a particular observer can or cannot determine at a given moment.

However, these constraints are frequently elevated into ontological properties of storage itself. Unavailable data is treated as absent. Delayed knowledge is treated as indeterminate reality. Conflicting local views are treated as divergent truths.

This paper rejects that elevation.

We adopt the following methodological principle:

> Ontological statements must not be derived from limitations of observation.

Storage is therefore described at a level invariant under local failure, network partition, and implementation detail.

The following axioms formalize that level.

3 Axioms

The following axioms define the Canonical Global File Ontology. They are independent of implementation details and describe only the logical conditions under which storage is meaningful.

Axiom 1: Global Name

A file is a globally unique name. The segmentation of that name is interpretative and has no ontological status.

Formally, for any two files F_1 and F_2:

$$F_1 = F_2 \iff \text{name}(F_1) = \text{name}(F_2)$$

Axiom 2: State

A file has a state. State is defined exclusively by the ordered sequence of operations applied to its name.

Content is not primitive; it is the result of state.

Axiom 3: Existence of Total Order

For every name N, there exists a total order relation $\prec_N$ over the set of operations $\mathcal{O}(N)$.

This order is postulated as an ontological invariant of meaningful storage.

Truth of state is determined solely by this order.

Axiom 3a: Order–Time Compatibility (optional)

If a canonical global time is constructed for a chosen horizon, the order $\prec_N$ over operations on any name N may be realized as a monotonic restriction of that global canonical order. No assumption is made that such a realization is unique or mandatory.

Definition (Operation as an ontological atom).
An *operation* is an indivisible ontological event belonging to exactly one name. Formally, each operation o carries: (i) an immutable identifier $\text{id}(o)$, (ii) an owner name $\text{own}(o) = N$, and (iii) an optional payload $\text{val}(o) = v$. Two operations are identical iff $\text{id}(o)$ is identical.

Axiom 3b: Operation Identity.

Operation identity is immutable: $\text{id}(o)$ does not change, and no two distinct operations share the same identifier.

Axiom 3c: Ownership.

Each operation belongs to exactly one name: $\forall o \, \exists! N \ (\text{own}(o) = N)$. Accordingly, $O(N) := \{ o \mid \text{own}(o) = N \}$.

Axiom 3d: Admissible Operations.

Only admissible operations may occur in histories. Admissibility is a predicate $\text{AdmOp}_H(o)$ fixed by the ontology of horizon H. Non-admissible operations are undefined within the ontology.

Definition (State interpretation).
Let $H(N)$ be the canonical history of N. The *state* of N is defined as an interpretation of history:

$$\text{state}(N) := F_H(H(N)),$$

where F_H is the horizon-dependent interpretation operator.

Axiom 3e: Determinism of Interpretation.

For any horizon H, the interpretation operator F_H is deterministic on admissible histories:

$$H_1(N) = H_2(N) \ \Rightarrow \ F_H(H_1(N)) = F_H(H_2(N)).$$

Axiom 3f: Horizon-locality of Interpretation.

F_H is part of the ontology of horizon H (a normative interpretive component), not a mutable property of the file. Changing F_H defines a different ontological model, not an alternative state of the same file.

Axiom 4: Canonicality

A history is canonical if and only if it contains no conflicting order for the same name.

Canon is a logical predicate over histories, not a physical entity.

Axiom 5: Non-Ontological Failure

Failure is not a state of a file.

If an operation cannot be observed, this does not constitute a modification of the file's state.

Axiom 6: Implementation Independence

The ontology does not prescribe how canonical order is achieved.

Any implementation that preserves Axioms 1–5 is valid.

Axiom 7: Scale Invariance

Globality is relative to the chosen horizon. Within a given horizon, name uniqueness must hold absolutely.

Extending the horizon must not invalidate previously canonical histories.

Ontological Closure

The ontology introduced in this work is complete with respect to the global storage layer. All ontological entities required to define identity, state, truth, and admissibility at the global level are explicitly specified.

The primitive ontological commitments of the model are: (i) globally unique names, (ii) operations as ontological atoms, (iii) admissible histories ordered by a total canonical order, (iv) horizons as ontological contexts fixing name domains and admissibility, and (v) horizon-dependent interpretation operators F_H.

No additional ontological primitives are assumed. Phenomena such as partial reads, buffering, replication lag, overwrite, and access failure are treated as implementation-level or observational conditions and do not extend the ontology of global storage.

Any system requiring additional ontological primitives realizes a different ontology and falls outside the scope of the present model.

4 The File as a Globally Unique Name

Within the proposed ontology, a file is defined exclusively as a globally unique name.

No additional structure is required to establish the existence of a file. Neither location, nor transport protocol, nor physical storage medium constitutes part of its identity.

Formally, identity is determined solely by equality of names. For any two files F_1 and F_2:

$$F_1 = F_2 \iff \text{name}(F_1) = \text{name}(F_2)$$

The internal segmentation of a name, if present, is interpretative rather than ontological. Human-readable decomposition into components does not imply structural hierarchy within the model.

Consider the example:

```
gf://earth/web/astraverge.org/research/fdb
```

This string denotes a single global name. It does not ontologically decompose into zone, path, protocol, or directory. Such interpretations belong to specific access methods or implementations, not to the ontology of storage.

The model therefore rejects the notion that hierarchy is primitive. If hierarchy exists, it exists as content of a file, not as a structural property of names.

A directory is thus a file whose state encodes references to other file names. No ontological distinction is introduced.

Similarly, content type is not primitive. Whether a name resolves to text, HTML, binary data, or a structured listing is a matter of interpretation. The ontology remains invariant.

A file does not reside somewhere. It is not located within a transport layer. It is not bound to a protocol.

It is defined by its name, and by the canonical order of operations applied to that name.

Naming precedes location.

Order precedes state.

Observation does not define being.

5 Order as Truth

Let N be a globally unique name.

Operations

An operation over N is an element of a set $\mathcal{O}(N)$. Operations may include write, replace, delete, or any state-transforming action defined by implementation.

The ontology does not restrict the internal semantics of operations. It constrains only their ordering.

Definition: History

A history over N is an admissible finite or countable sequence:

$$H(N) = \langle o_1, o_2, \ldots, o_k \rangle, \quad H(N) \in \text{AdmHist}_H(N),$$

where each $o_i \in O(N)$.

Order

For every name N, there exists a total order relation $\prec_N$ over $\mathcal{O}(N)$ such that:

$$\forall o_i, o_j \in \mathcal{O}(N), \quad o_i \neq o_j \Rightarrow (o_i \prec_N o_j) \vee (o_j \prec_N o_i).$$

This total order is postulated as an ontological invariant. It is not derived from timestamps or observation.

State

The state of a file with name N is a function of its ordered history:

$$\text{state}(N) = F_H(H(N)).$$

Truth of the file is determined solely by its canonical order of operations.

Time

Physical or logical timestamps may assist in constructing an order, but time is not primitive in the ontology.

If $t(o_i)$ denotes a timestamp associated with operation o_i, then:

$$t(o_i) \not\equiv \text{identity}(o_i)$$

and does not determine truth independently of order.

Relation to Canonical Global Time. The total order $\prec_N$ postulated in this ontology does not presuppose physical time as a primitive. However, it is compatible with a *canonical global time* constructed as

an observer-side ordering derived from admissible reception sequences. In such a realization, $\prec_N$ corresponds to the restriction of the global canonical order to operations over name N. Time, in this sense, is a parametrization of an already established order, not its source [2], [3].

Canonicality

A history $H(N)$ is canonical if and only if it respects the total order $\prec_N$ and contains no contradictory ordering relation.

Lemma (Uniqueness of Canon)

If two canonical histories $H_1(N)$ and $H_2(N)$ exist for the same name N, then:

$$H_1(N) = H_2(N).$$

Proof. By definition of canonicality, no two distinct histories may disagree on the order of any pair of operations. Therefore, canonical history is unique. □

Theorem 1 (Failure Has No Ontological Status)**.** *Let N be a globally unique name, and let $H(N)$ be its canonical history.*

Suppose an operation $o \in \mathcal{O}(N)$ is not observed by a particular agent.

Then the non-observation of o does not modify $H(N)$, nor does it introduce a new ontological state of N.

Proof. Truth of N is determined solely by the canonical order $\prec_N$.

Observation is not part of the definition of canonical order.

Therefore, failure of observation cannot alter $H(N)$. □

6 Implications

By the preceding theorem, failure does not constitute an ontological state of storage.

It is reclassified as an implementation condition, not a property of the file.

If bytes are not obtained, this indicates a mismatch between implementation and model, not a modification of the file's state.

The ontology therefore excludes:

- degraded states,
- partial existence,
- temporary non-being.

Such notions belong to implementation, not to storage ontology.

7 Geometry of Names

Globality is relative to scale. A name is global not in an absolute metaphysical sense, but relative to a chosen horizon of uniqueness.

Definition (Horizon as an ontological context).
A *horizon* H is an ontological context that fixes: (i) the domain of admissible names N_H, (ii) the admissibility predicate for operations and histories, and (iii) the interpretation operator F_H. A name exists in the ontology only relative to a horizon:

$$N \text{ is defined in } H \iff N \in N_H.$$

Axiom 7a: Name-domain Existence.

For each horizon H, the set N_H is well-defined as the domain of names to which the ontology applies. The ontology makes no finiteness assumption about N_H (it may be finite, countable, or potentially infinite).

Axiom 7b: Membership Criterion.
Membership $N \in N_H$ is normative: it is fixed by the definition of the horizon. Names outside N_H are undefined within H.

Horizon

A horizon $\mathcal{H}$ is a domain within which name uniqueness holds absolutely.

Let $N_{\mathcal{H}}$ denote the set of names defined within $\mathcal{H}$.

A name N is global within $\mathcal{H}$ if no other element of $N_{\mathcal{H}}$ shares the same identifier.

Global qualification is therefore not intrinsically required. It is required only to the extent that uniqueness is preserved.

Scale and Qualification

Full cosmic qualification (e.g., universum/milkyway/solarsystem/earth/...) is unnecessary whenever the active horizon already guarantees uniqueness.

For example, within the solar system, the prefix

```
universum/milkyway/solarsystem
```

may be omitted without loss of identity, provided no ambiguity is introduced.

Identity is therefore independent of the maximal descriptive scope.

Geometric Delegation

Uniqueness may be maintained through geometric delegation.

A namespace may be partitioned into subspaces, each responsible for preserving uniqueness within its own domain.

Such partitioning is not ontological hierarchy. It does not introduce layered being. It is a structural mechanism for preserving global uniqueness without requiring centralization.

Delegation does not create multiple truths. Canonical history $H(N)$ remains unique for each name N, independently of geometric subdivision.

Invariance Under Extension

If a horizon $\mathcal{H}_1$ is extended to $\mathcal{H}_2$ such that $\mathcal{H}_1 \subset \mathcal{H}_2$, then for every name $N \in \mathcal{H}_1$, the canonical history $H(N)$ remains unchanged.

Formally:

$$\mathcal{H}_1 \subset \mathcal{H}_2 \Rightarrow \forall N \in N_{\mathcal{H}_1}, \quad H(N) \text{ remains unchanged.}$$

Extending the naming scope must not reorder operations nor invalidate identity.

Definition (Admissible horizon extension).
An extension $H_1 \subset H_2$ is admissible iff it preserves all previously defined names: $N_{H_1} \subseteq N_{H_2}$ and does not change F_{H_1} on histories of names in N_{H_1}.

Web as Virtual Geography

A segment such as

```
earth/web/astraverge.org
```

is interpreted as a coordinate within the geometric space of names.

It is not a transport protocol, not a storage medium, and not a directory structure. It is a virtual geographical marker within the horizon of Earth.

Locality Without Ontological Fragmentation

Locality concerns implementation and access. It does not fragment ontology.

A file remains identical across all horizons in which its name is valid. Changes in observation scope do not generate new histories.

Lemma (Horizon Reduction)

If a prefix does not contribute to uniqueness within a given horizon, its omission does not alter identity.

Proof. Identity is determined solely by name equality. If omission of a prefix does not introduce ambiguity, the resulting identifier refers to the same name. □

8 Canon as Logical Predicate

Canonicality is a property of history, not of infrastructure.

Definition (Admissible history).
For a name $N \in N_H$, an *admissible history* is any (finite or countable) sequence

$$H(N) = \langle o_1, o_2, \dots \rangle$$

such that: (i) $\forall i\, (\text{own}(o_i) = N)$, (ii) $\forall i\, \text{AdmOp}_H(o_i)$, and (iii) the sequence is compatible with the postulated total order $\prec_N$. Denote the class of admissible histories by $\text{AdmHist}_H(N)$.

Remark. The ontology does not require the existence of a set of *all* histories as a completed object; it specifies admissibility conditions under which a history is meaningful within horizon H.

Definition

Let $H(N)$ be a history over name N.

A history $H(N)$ is canonical if and only if:

1. it respects the total order $\prec_N$, and
2. no alternative history $H'(N)$ exists such that $H'(N)$ is also canonical and disagrees on the order of any pair of operations.

Formally:

$$H(N) \text{ is canonical } \iff \neg\exists H'(N) \neq H(N) \text{ such that } \prec_N \text{ diverges.}$$

Logical Status

Canonicality is a logical predicate over histories. It does not imply the existence of a physical coordinator, central authority, or synchronization mechanism.

The ontology does not prescribe how canonical order is constructed. It only defines the condition under which a history is meaningful.

Uniqueness of Canon

Lemma 1. *For any name N, there exists at most one canonical history.*

Proof. By definition of canonicality, no two distinct histories may disagree on the order of any pair of operations. Therefore, if $H_1(N)$ and $H_2(N)$ are canonical, then $H_1(N) = H_2(N)$. □

Interpretation

Canon is not an entity. It is not a location. It is not an implementation role.

Canon is the logical condition that excludes divergent histories.

Corollary

Split-brain is undefined within this ontology, since divergent canonical histories are logically excluded.

9 Minimal Operational Semantics

The ontology does not prescribe implementation, but it admits a minimal operational interface.

Primitive Operations

For a name N, let the following abstract operations exist:

- Put(N, v) — append a state-transforming operation
- Get(N) — observe the current state

- List(N) — observe referenced names

Put appends an admissible operation to an admissible history, i.e., $H(N)$ is extended only within $\text{AdmHist}_H(N)$.

These operations do not describe transport, replication, or storage mechanisms. They describe only logical interaction with the ordered history of a name.

Lemma 2 (Uniqueness of the State-Transforming Primitive)**.** *Let N be a globally unique name and let $\mathcal{O}(N)$ denote the set of operations applicable to N.*

Any operation that modifies the state of N can be represented as an instance of Put(N, v) *for some admissible payload v.*

In particular, no additional primitive operation is required to express creation, modification, or termination of state.

Proof. By definition, the state of a file with name N is determined exclusively by its canonical history $H(N) = \langle o_1, o_2, \dots \rangle$.

Any modification of state therefore corresponds to the addition of a new element to this history.

Put(N, v) is defined precisely as an operation that appends a new state-transforming element to $H(N)$.

Consequently, every admissible state transition is representable as a Put operation, and no additional state-transforming primitive is ontologically required. □

Theorem 2 (Semantic Reducibility of Storage Operations)**.** *All commonly used storage operations, including **create**, **write**, **delete**, **rename**, and **move**, are semantically reducible to compositions of the primitive operation* Put(N, v) *and observational projections.*

None of these operations constitute distinct ontological primitives.

Proof. We consider each operation class:

- **Create**(N) corresponds to the first admissible Put(N, v_0) in the canonical history of N, introducing an initial state.
- **Write**(N, Δ) corresponds to a Put(N, v) that transforms the current state according to Δ.
- **Delete**(N) corresponds to a Put(N, v_{term}) that terminates admissible future state transitions without erasing history.
- **Rename**$(N \to N')$ corresponds to the introduction of a new name N' with its own canonical history, optionally referencing the history of N.
- **Move**$(N \to N')$ is a semantic specialization of rename under a particular namespace interpretation.

Operations such as **read** and **list** do not modify state and are therefore observational:

$$\text{Read}(N) \equiv \text{Get}(N), \quad \text{List}(N) \equiv \text{Get}(N) \text{ under directory interpretation.}$$

Thus, all storage operations are reducible to Put and observation, and no additional ontological primitives are required. □

Corollary 1 (Impossibility of Partial State)**.** *Let N be a globally unique name and let $H(N)$ be its canonical history. The state of N is $state(N) = F_H(H(N))$.*

Then there exists no ontological notion of a partial state *of N. In particular, between two consecutive canonical histories $H_k(N) = \langle o_1, \ldots, o_k \rangle$ and $H_{k+1}(N) = \langle o_1, \ldots, o_k, o_{k+1} \rangle$, there is no intermediate state of N in the ontology.*

Proof. By definition, state is a function of the canonical history: $state(N) = F(H(N))$. Canonical history is an ordered sequence of operations, hence discrete. Therefore, admissible states correspond only to admissible histories.

Any claim of an intermediate or partial state would require a history that is neither $H_k(N)$ nor $H_{k+1}(N)$, contradicting the definition. What may be partial is observation (e.g., incomplete retrieval), but observation does not modify $H(N)$ and thus cannot introduce a new ontological state. □

Remark. "Partial state" is an implementation-level symptom (e.g., partial reads, streaming, buffering, or incomplete replicas), not a property of N. Ontologically, either a canonical operation is in $H(N)$, or it is not.

Corollary 2 (Non-existence of In-place Mutation)**.** *Let N be a globally unique name. Any modification of the state of N is an extension of its canonical history by a new operation.*

Therefore, in-place mutation *of N does not exist as an ontological primitive.*

Proof. By the uniqueness of the state-transforming primitive, state changes occur only by appending an operation to $H(N)$. Thus any change is represented as $H(N) := H(N) \parallel \langle o \rangle$ for some o.

An "in-place mutation" would imply a state change without a new canonical operation, i.e., without modifying $H(N)$. This contradicts the definition $state(N) = F(H(N))$. Hence in-place mutation is not an ontological notion in this framework. □

Remark. Low-level overwrites of blocks, pages, or objects may occur in implementations, but they are not ontological mutations of N. They are merely a physical technique for realizing the effect of an appended operation.

Theorem 3 (Availability Has No Ontological Status)**.** *Let N be a globally unique name and let $H(N)$ be its canonical history. Assume $N \in N_H$ and $H(N) \in \text{AdmHist}_H(N)$. Let an agent a attempt to observe the current state via* $\text{Get}(N)$.

If $\text{Get}(N)$ *fails for a (timeout, disconnection, missing replicas, etc.), then this failure does not modify $H(N)$, does not introduce a new ontological state, and does not license the claim that N is partially absent.*

Availability is therefore an epistemic predicate about an access channel, not an ontological predicate about the file.

Proof. By definition, truth of N is determined solely by the canonical order $\prec_N$ and the induced history $H(N)$. The act of observation $\text{Get}(N)$ is not part of $\mathcal{O}(N)$ and does not contribute operations to $H(N)$.

Therefore, failure of $\text{Get}(N)$ cannot alter $H(N)$. Since state is $F(H(N))$, no new ontological state can be introduced by non-observation. The failure characterizes only the agent's access conditions. □

Corollary 3 (Non-ontological Status of Degraded Modes)**.** *Terms such as "degraded mode", "partially available file", or "temporarily non-existent data" do not denote ontological states of N. They denote implementation-level conditions of observation and access.*

Proof. Immediate from Theorem "Availability Has No Ontological Status": since failure of observation does not modify $H(N)$, it cannot generate additional states of being. □

Remark. Trade-offs between consistency and availability describe implementation behavior under constrained access. They do not extend the ontology of N with additional modes of being.

Remark (History immutability). Since canonical history determines state, already established prefixes of $H(N)$ are not subject to retroactive revision; new access can only extend observation, not rewrite truth [2], [3].

Remark (Names are not mutable). Renaming and moving do not modify the state of a name. They introduce new names with independent canonical histories. Any interpretation of renaming as in-place mutation belongs to implementation-level indirection mechanisms, not to storage ontology.

Remark (Deletion is terminal, not destructive). Deletion does not erase canonical history. A delete operation terminates admissible future state transitions while preserving the immutability of past operations.

Remark (Observation is not operation). Observational acts do not belong to $\mathcal{O}(N)$. They neither modify canonical history nor introduce new ontological states.

Observational operations. Operations such as **Read**(N) and **List**(N) do not modify canonical history and therefore are not part of $\mathcal{O}(N)$. They are observational projections of the current state:

$$\text{Read}(N) \equiv \text{Get}(N), \quad \text{List}(N) \equiv \text{Get}(N) \text{ under directory interpretation.}$$

Operational Effect

Let H(N) denote the canonical history of N. Let $\|$ denote sequence concatenation.

$$\text{Put}(N, v) \quad \Rightarrow \quad H(N) := H(N) \,\|\, \langle o_v \rangle$$

where o_v is appended according to the canonical order $\prec_N$.
The resulting state is:

$$\text{Get}(N) = F(H(N))$$

as defined in Section "Order as Truth".

Determinism

Given a canonical history, Get(N) must be deterministic.

$$H_1(N) = H_2(N) \Rightarrow F(H_1(N)) = F(H_2(N)).$$

Thus, truth depends solely on ordered history.

Non-Observation

If Get(N) fails to return a value, this does not modify $H(N)$. It constitutes an implementation-level condition, not a semantic one.

No Partial States

No Partial States. See Corollary 1.

Idempotence

Repeated application of the same canonical history does not alter state:

$$F(H(N)) = F(F(H(N)))$$

10 Related Work

This section does not attempt a comprehensive survey. Its purpose is to situate the proposed ontology relative to existing naming- and consistency-oriented systems, without engaging at the level of implementation or performance trade-offs.

The present work does not introduce new replication algorithms, consensus mechanisms, or transport protocols. Its contribution lies at the ontological level. Nevertheless, several existing systems touch adjacent aspects of naming, ordering, and distribution.

Unix and Plan 9

Unix established the principle that "everything is a file", collapsing heterogeneous resources into a unified abstraction.

Plan 9 extended this principle into distributed environments, allowing remote resources to be mounted into a unified namespace.

However, both systems remain implementation-centric. They do not formalize global naming as an ontological primitive, nor do they exclude failure states from the definition of storage.

Domain Name System (DNS)

The Domain Name System provides a globally delegated naming hierarchy. It demonstrates that large-scale uniqueness can be maintained without centralization.

However, DNS defines resolution protocols rather than a storage ontology. Failure, timeout, and replication lag are treated as operational conditions, not excluded from conceptual modeling.

The present work generalizes the geometric intuition of DNS into a name-first storage ontology.

Content-Addressed Systems

Systems such as Git and IPFS employ content-based identifiers derived from cryptographic hashes.

These approaches invert the name-first model: identity is derived from content, rather than content from ordered history.

Furthermore, divergent histories and branching are permitted as first-class states.

In contrast, the present model treats name as primary and excludes alternative canonical histories.

Distributed Consistency Models

Classical distributed systems theory analyzes trade-offs between consistency, availability, and partition tolerance.

These models typically treat partial failure, message delay, and incomplete knowledge as structural elements of distributed reality.

The current work differs methodologically. It distinguishes between epistemic limitation and ontological definition, and deliberately excludes failure states from the ontology of storage.

In summary, existing systems provide mechanisms for naming, ordering, or replication. The present work does not compete at that level. It proposes a minimal ontological framework within which such mechanisms may be interpreted.

Existing models define storage under epistemic constraint. The present work defines storage independently of epistemic constraint, treating such constraints as external to ontology.

Canonical order and operational time. The present work relies on canonical order as a primitive of storage meaning. A compatible ontological specification and a physical reconstruction framework for canonical global time are developed separately [2], [3], where global time is defined as a constructed order from reception sequences. In that view, storage canon can be treated as a name-restricted projection of a global canonical order.

11 Metatheoretical Properties

This section makes explicit the metatheoretical status of the ontology already specified. No additional ontological assumptions are introduced.

The following results characterize the ontology specified in this work at a metatheoretical level. They do not concern formal derivability in a proof-theoretic sense, but rather the structural properties of the ontology with respect to its declared domain: global storage.

Ontological Completeness

Theorem 4 (Ontological Completeness)**.** *The ontology of global storage specified in this work is complete with respect to its declared domain. All ontological properties required to define identity, state, truth,*

and admissibility at the global storage level are derivable from the primitive ontological commitments enumerated in Section 3.

Proof sketch. Identity is fixed by globally unique naming. State is determined by admissible canonical history and the horizon-dependent interpretation operator F_H. Truth is determined by the non-divergent canonical order. Admissibility is fixed by horizon-relative predicates on operations and histories.

No additional ontological entities are required to define these notions at the global level. Therefore, the ontology is complete with respect to the domain it claims to describe. □

Ontological Consistency

Theorem 5 (Ontological Consistency)**.** *The ontology of global storage is internally consistent. No ontological entity is required to simultaneously possess mutually exclusive modes of being within the same horizon.*

Proof sketch. Names are either elements of a horizon domain N_H or undefined within that horizon. Histories are either admissible or inadmissible, and canonical or non-canonical.

Failure and unavailability are explicitly excluded from the ontology of being and are treated as epistemic conditions of observation. Partial states and divergent histories are likewise excluded by construction.

As no entity is assigned incompatible ontological roles, the ontology admits no internal ontological contradiction. □

Ontological Minimality

Theorem 6 (Ontological Minimality)**.** *The ontology of global storage is minimal. Removing any primitive ontological commitment results in the inability to define at least one of the following global notions: identity, state, truth, or admissibility. Adding further ontological primitives does not increase expressive power at the global level, but introduces implementation-dependent distinctions.*

Proof sketch. Without global names, identity cannot be defined. Without operations and canonical order, neither change nor truth can be specified. Without admissible histories, state cannot be determined. Without horizons, global uniqueness and admissibility lack ontological grounding.

Conversely, introducing notions such as partial state, availability, or in-place mutation does not extend the ontology of global storage, but reintroduces observer- or implementation-relative conditions that belong to local storage ontology.

Therefore, the ontology is minimal with respect to its declared domain. □

Remark. These metatheoretical properties are relative to the declared ontological level of global storage. They do not preclude richer ontologies at the level of local storage, physical realization, or observation.

12 Limitations

The present model does not claim that total order can be physically realized in all environments.

It postulates total order as an ontological condition for meaningful storage.

Implementations that cannot preserve this invariant realize a different model.

Ontology vs. physical realization. The ontology remains valid even in the absence of any physically realized global time infrastructure. Conversely, a physical construction of canonical global time may provide a concrete mechanism by which canonical order can be instantiated, without modifying the ontology itself.

13 Conclusion

This work has specified an ontology of global storage in which identity is determined by global naming, and truth is fixed by a canonical, non-divergent order of operations.

The central methodological distinction is between ontological structure and epistemic limitation. Local failure, partial visibility, and implementation constraints characterize conditions of observation and access; they do not constitute states of storage itself.

Within this ontology:

- A file is identified by a globally unique name.
- State is determined by an ordered admissible history.
- Canonicality excludes divergent histories.
- Failure and unavailability have no ontological status.
- Globality is invariant under admissible horizon extension.

The ontology does not prescribe replication strategies, consensus mechanisms, or physical architectures. It specifies only the minimal ontological conditions under which storage is well-defined at a global level.

By maintaining a strict separation between being and observation, the framework clarifies the conceptual foundations of distributed storage without incorporating implementation difficulty into ontology.

References

[1] P. Mockapetris, "Domain names – concepts and facilities," Internet Engineering Task Force, RFC 1034, 1987. Accessed: Feb. 11, 2026. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc1034`.

[2] A. A. Nekludoff. "Global time as a physical theory: Canonical order, operational time, and history immutability," Accessed: Feb. 16, 2026. [Online]. Available: `https://doi.org/10.5281/zenodo.18662399`.

[3] A. A. Nekludoff. "Ontology of global time: Local observers, sources and destinations," Accessed: Feb. 16, 2026. [Online]. Available: `https://doi.org/10.5281/zenodo.18655183`.

[4] J. Postel, "Transmission control protocol," Internet Engineering Task Force, RFC 793, 1981. Accessed: Feb. 11, 2026. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc793`.