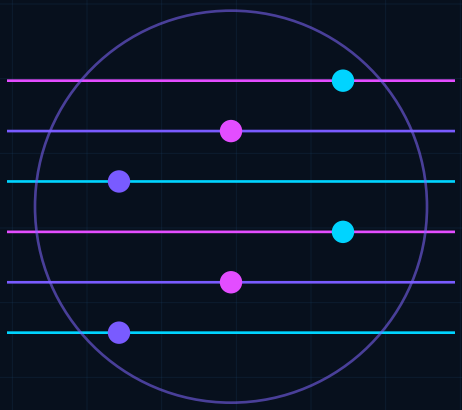
NETWORK STATE AND AUTOMATION

Desired vs. Observed Network State (Drift Dynamics)

Current-source review, evidence-bounded adoption decision, explicit limitations, reproducibility controls, and preserved source research note.



PERMANENT ID	EVIDENCE	ADOPTION	FRESHNESS
RES-001	A-	ADOPT AS FOUNDATION	STABLE WATCH

Freshness review: 2026-09-17 / Next scheduled review: 2027-03-16

RESEARCH CONTROL SURFACE

Required Research Fields

Each field remains independently reviewable; evidence strength does not imply production authority.

EVIDENCE GRADE — **A-**

Strength of the retained source set and technical basis.

SOURCE AUTHORITY — **2 registered authorities**

RFC 8342 - Network Management Datastore Architecture (NMDA); OpenConfig gNMI Specification

FRESHNESS — **STABLE WATCH**

Reviewed 2026-09-17; dynamic sources require event-driven revalidation.

CONFLICTS — **VISIBLE / NOT SILENT**

Differences between specification, vendor behavior, research claims, and reproduced evidence must remain explicit.

LIMITATIONS — **EXPLICIT**

No certification, procurement approval, legal conclusion, or unbounded production claim is created by this report.

REPRODUCIBILITY — **REQUIRED**

Material claims require exact versions, representative data, failure cases, artifacts, and repeatable acceptance criteria.

ADOPTION POSTURE — **ADOPT AS FOUNDATION**

Adopt semantic desired-versus-observed reconciliation as a core network and infrastructure control. Do not enforce automatic remediation without classification, authorization, and post-change verification.

REVIEW TRIGGER — **2027-03-16**

Scheduled at 180 days; earlier on material source, vulnerability, implementation, or contradictory-evidence change.

SOURCE AUTHORITY / FRESHNESS

Authority Register and Freshness Delta

Primary-source lineage is preserved; current-source changes are separated from unperformed implementation claims.

AUTHORITY 01

RFC 8342 - Network Management Datastore Architecture (NMDA)

https://www.rfc-editor.org/rfc/rfc8342

AUTHORITY 02

OpenConfig gNMI Specification

https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-sp

2026-09-17 FRESHNESS DELTA

OpenConfig gNMI reference now records v0.11.0 (2026-03-05); RFC 8342 remains the stable NMDA authority.

Freshness policy: scheduled review + immediate review on source supersession, material release, vulnerability, retraction, or contradictory evidence.

Evidence Must Survive Contact With Reality

- Source authority is not implementation evidence. Product behavior, scale, security, interoperability, and operator fit remain environment-specific.
- Conflicting evidence is recorded rather than averaged away. Specification behavior, vendor implementation, preprint results, and production observations are separate evidence classes.
- Quantitative claims are informative until reproduced under documented hardware, software, data, topology, load, and failure conditions.
- Adoption posture is reversible. A stronger source set can coexist with a pilot-only or research-only recommendation when maturity, safety, economics, or operability remain unresolved.

ADOPTION / VALIDATION / REVIEW

Decision Gate and Validation Plan

ADOPTION POSTURE

ADOPT AS FOUNDATION

Adopt semantic desired-versus-observed reconciliation as a core network and infrastructure control. Do not enforce automatic remediation without classification, authorization, and post-change verification.

1 / SOURCE	Pin exact versions, publication state, retrieved artifacts, and source hashes.
2 / PROTOTYPE	Demonstrate the core mechanism with representative data and instrumented execution.
3 / FAILURE	Exercise malformed input, dependency loss, stale state, overload, recovery, rollback, and misuse.
4 / BASELINE	Compare against an incumbent, classical, or simpler architecture using the same workload.
5 / ACCEPT	Record acceptance criteria, residual limitations, owner, expiration, and revalidation triggers.

EXPIRATION / REVIEW TRIGGER

Next scheduled review: 2027-03-16 (180-day cadence). Review sooner after material source revision, breaking implementation release, critical vulnerability, retraction, benchmark contradiction, changed workload, or changed legal/safety boundary.

8. Complete Source Research Note

SOURCE-LINEAGE NOTICE

The following material preserves the complete original source note, excluding only the conversational wrapper and outer Markdown fence. Legacy status labels and absolute language are retained for traceability and do not override the candidate status, limitations, or adoption decision in this edition.

Document ID: RES-001 Version: 1.0.0 (Definitive Registry Edition) Status: Published Research Note Classification: Public Organization: Mythos Systems (MYTHOS AI) Owner: Aaron Stovall Effective Date: 2026-07-16 Applies To: Network architects, Site Reliability Engineers (SREs), and platform engineers designing continuous reconciliation loops, drift detection algorithms, and Network-as-Code (NaC) pipelines Companion Standards: MYTHOS-NET-0005 (Source-of-Truth & Drift Reconciliation), MYTHOS-NET-0001 (Network Architecture), MYTHOS-PLT-0002 (IaC & Configuration)

The history of network engineering is a battle against entropy. Organizations attempt to declare their network's desired state in a Source of Truth (SoT) like NetBox or Git, but the observed state on the physical devices constantly diverges due to human error, emergency break-glass changes, and autonomous system reactions.

Traditional network management treats this delta as an anomaly to be investigated manually. The Mythos architecture treats the delta between Desired State (S_D) and Observed State (S_O) as a continuous, measurable physical property known as Drift (Δ).

This research note defines the mathematical dynamics of network state drift. It categorizes drift into syntactic, semantic, and temporal classes, and exposes the fallacy of raw-text `diff` tools. True drift reconciliation requires Abstract Syntax Tree (AST) parsing and YANG data models to mathematically prove that two configurations, though syntactically different, are semantically identical. Understanding these dynamics is a prerequisite for building the self-healing, zero-touch networks mandated by the Mythos Standards.

To engineer a self-healing network, we must first define state mathematically.

- Desired State (S_D): The declarative intent. The configuration as it exists in the Source of Truth (NetBox) and version control (Git). This is the architectural truth.
- Observed State (S_O): The running configuration on the physical or virtual network device at time T. This is the operational reality.
- Drift (Δ): The asymmetric delta between S_D and S_O.

$$ \Delta = S_D - S_O $$

If $\Delta \neq 0$, the network has drifted. The goal of a continuous reconciliation loop (as defined in MYTHOS-PLT-0002) is to minimize the half-life of Δ—the time it takes for a drift to be detected and autonomously reverted to S_D.

Not all drift is created equal. An engineer manually typing a command is different than a routing protocol dynamically updating a list. Drift must be classified to determine the correct remediation action.

2.1 Intentional / Emergency Drift (Break-Glass)

The most common form of drift. During a P1 outage at 3 AM, an engineer bypasses the GitOps pipeline and uses SSH to change an ACL or shut down an interface to restore service.

- Dynamic: S_O changes, S_D remains static.
- Remediation: The reconciliation loop will detect this drift and revert it, potentially breaking the emergency fix. Solution: The SoT pipeline must allow engineers to apply a "Drift Ignore" tag to specific objects for a 24-hour TTL, after which the change must be codified in Git or reverted.

2.2 Autonomous / Protocol Drift (Ephemeral)

Network protocols (OSPF, EIGRP, IS-IS) dynamically alter the running configuration. For example, OSPF injects learned routes into the RIB, or a dynamic access-list updates its hit-counters.

- Dynamic: S_O changes constantly, but these changes are not part of the intended architectural baseline.
- Remediation: Naive `diff` engines constantly flag this as drift, causing alert fatigue. Solution: The reconciliation engine MUST utilize YANG models or AST parsing to filter out ephemeral, protocol-specific state. Only baseline configuration blocks should be evaluated.

2.3 Organic Decay (Silent Failure)

A device experiences a memory leak or a process crash. It reboots, but a volatile configuration command was not saved to NVRAM, or a line card fails and the OS drops the interface configuration.

- Dynamic: S_O reverts to a subset of S_D without human intervention.
- Remediation: The reconciliation loop detects the missing configuration and pushes it back to the device. This is the true power of continuous reconciliation—self-healing hardware failures.

2.4 Parallel Pipeline Collision

Two separate automation pipelines (e.g., a security team pushing an emergency blocklist via Ansible, and a network team pushing a baseline update via Terraform) target the same device concurrently.

- Dynamic: S_O oscillates between the two desired states as each pipeline overwrites the other.
- Remediation: There can be only one execution engine. All changes MUST flow through the central SoT reconciliation loop.

The greatest architectural failure in legacy network automation is relying on text-based `diff` tools to detect drift. Network operating systems (Cisco IOS, Arista EOS, Junos) allow multiple syntactic variations to achieve the exact same semantic configuration.

3.1 The Syntactic Illusion

Consider a simple ACL configuration:

Desired State (Git):

```
access-list 10 permit 192.168.1.0 0.0.0.255
```

Observed State (Router):

```
access-list 10 permit 192.168.1.0/24
```

A naive `diff` engine will flag this as a critical drift and trigger a remediation (which will fail or oscillate, as the router will reject the second command because the ACL already exists).

3.2 The AST/YANG Solution

To accurately calculate Δ, the reconciliation engine MUST NOT compare raw text. It must:

1. Parse: Translate both S_D and S_O into an Abstract Syntax Tree (AST) or a structured YANG data model.
2. Normalize: Convert all IP prefixes, subnet masks, and interface names into their canonical forms (e.g., converting `0.0.0.255` to `/24`).
3. Semantic Compare: Compare the data trees mathematically.

Only by comparing the meaning of the configuration, rather than the text, can a reconciliation engine achieve zero false positives.

Once Δ is accurately detected, the system must resolve it. There are two primary architectural models for reconciliation.

4.1 Push-Based Reconciliation (CI/CD Triggered)

- Mechanism: An external orchestrator (e.g., Ansible AWX, Jenkins) periodically polls the devices via NETCONF/SSH, compares the result to the SoT, and pushes remediation configurations if drift is found.
- Latency: High (minutes to hours).
- Mythos Alignment: Fails the Mythos standard. Polling is inefficient, and latency is too high to prevent outages.

4.2 Pull-Based / Streaming Reconciliation (gNMI/Event-Driven)

- Mechanism: Devices are configured to stream their state changes in real-time via gNMI (gRPC Network Management Interface) telemetry to a local agent. The agent immediately evaluates the stream against the SoT intent. If a drift is detected, the agent pushes a remediation via gNMI in milliseconds.
- Latency: Ultra-low (sub-second).
- Mythos Alignment: This is the mandated architecture. The network must continuously push its own state, and the control plane must instantly revert it. This creates a mathematically bounded half-life for drift, ensuring manual CLI changes are erased before they can cause an outage.

The transition from manual network management to Network-as-Code (NaC) is not merely a shift in tooling; it is a shift in physics. As long as the Desired State (S_D) and Observed State (S_O) are decoupled, drift (Δ) is inevitable.

Organizations that treat configuration as a static artifact pushed via CI/CD will forever battle drift. Organizations that implement continuous, semantic, pull-based reconciliation loops transform the network into a self-healing organism. In a Mythos-compliant architecture, the CLI is no longer a tool for configuration; it is a read-only sensor, and the Source of Truth is the absolute, mathematically enforced sovereign.

```
Intent is a declaration.
Reality is a deviation.

A text diff is a lie; a semantic tree is truth.
A polling pipeline is a snapshot; a gNMI stream is continuous.

If the network is not self-healing, it is decaying.
> Intent may be declared.
> State must be enforced.
```

End of Research Note RES-001

9. Source Register

Source	Authority and Status	Freshness Control
RFC 8342 - Network Management Datastore Architecture (NMDA) https://www.rfc-editor.org/rfc/rfc8342	Primary standard Stable RFC Published: 2018	Validated: 2026-07-22 Review on material revision, withdrawal, supersession, or scheduled report review.
OpenConfig gNMI Specification https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md	Primary specification Living specification Published: Living	Validated: 2026-07-22 Review on material revision, withdrawal, supersession, or scheduled report review.

10. Lineage, Review, and Publication Record

Record	Value
Original source file	RES-001_Desired_vs_Observed_Network_State_Drift_Dynamics.md
Original source words	1275
Upgrade type	Enterprise research report; source and freshness validation; limitations; adoption decision; reproducibility plan
Generated on	2026-07-22
Current status	Candidate Research Report
Publication authority	Formal Mythos approval not yet recorded
Next review	180 days or event-driven trigger, whichever occurs first