International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

Architecting Scalable Intent-Based Networking Platforms for Enterprise Automation

Manevannan Ramasamy

Software Engineering Senior Technical Leader Milpitas, California, United States of America

manevannan3@gmail.com

ABSTRACT: Enterprise intent-based networking requires more than translating high-level objectives into device configurations. A platform must coordinate planning, execution, state management and assurance across heterogeneous domains while preserving policy constraints and traceability. This paper proposes a conceptual architecture that separates intent admission, semantic normalization, hierarchical planning, distributed execution, state management and continuous assurance, and governance. An authoritative, versioned intent record defines the desired outcome; bounded domain contracts delegate planning and execution to controllers with local knowledge. Stable ownership, asynchronous work queues, idempotent operations, incremental recomputation and local reconciliation are proposed to reduce unnecessary global coordination. The architecture distinguishes globally significant invariants, which require appropriately coordinated decisions, from telemetry and intermediate operational observations that may tolerate explicitly bounded staleness. The contribution is a structured synthesis of intent-based networking, model-driven management and distributed-systems principles into a testable enterprise control-plane design. A proposed evaluation framework compares centralized planning and execution, centralized planning with distributed execution, and hierarchical planning with local reconciliation under controlled workloads. It identifies admission latency, planning cost, execution throughput, convergence, telemetry overhead, conflict detection, invariant violations and recovery behavior as evaluation measures. The paper presents architectural propositions and an experimental protocol; it does not report an implemented prototype, measured performance gains or completed experimental validation.

KEYWORDS: Intent-based networking; enterprise automation; hierarchical planning; distributed orchestration; continuous assurance; network state management

I. INTRODUCTION

Enterprise networks span campuses, data centers, wireless infrastructure, wide-area networks, cloud environments and security domains. These environments differ in device capabilities, administrative ownership, failure behavior and operational constraints. Scripts and templates can automate individual configuration tasks, but coordinating a service across several domains also requires an explicit account of the intended outcome, the dependencies between changes and the conditions under which an outcome can be considered achieved. As workload and topology complexity increase, a controller that performs every planning and reconciliation task centrally may become a coordination bottleneck.

In the proposed architecture, a versioned intent record specifies the service conditions to be maintained, and domain contracts assign responsibilities for realizing and checking those conditions. Model-driven management provides structured representations of services, resources, configuration and operational state [1,4,5]. These representations do not by themselves determine how computation is distributed or changes are coordinated across enterprise domains.

The central design question is how to retain authoritative global semantics while delegating detailed work. Completely centralized planning offers a direct place to enforce enterprise policy, but may incur high computational and communication costs. Unconstrained local autonomy can reduce coordination, but may produce incompatible decisions when an intent spans domains or affects shared resources. A practical architecture therefore needs explicit ownership boundaries, verifiable domain responsibilities and a way to escalate decisions whose consequences exceed a local controller's authority. It must also preserve the distinction between a command completing successfully and the requested service outcome being verified.

This paper proposes an intent-to-outcome control plane built around a versioned intent record and bounded domain contracts. Global services retain responsibility for authorization, intent semantics, cross-domain dependencies and enterprise invariants. Domain controllers perform local planning, execution and reconciliation using their own topology

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

and resource knowledge. The architectural contribution lies in specifying how these responsibilities interact through asynchronous execution, differentiated consistency and evidence of outcome satisfaction. The design synthesizes established concepts rather than claiming to originate each underlying mechanism. Its scalability and correctness remain propositions to be evaluated through the protocol developed in this paper; no numerical improvement or production deployment is asserted.

II. RELATED WORK

Model-driven management provides structured representations and interfaces for automation. RFC 8969 describes a framework for automating service and network management with YANG [1]. RFC 7950 specifies the YANG 1.1 data modeling language [5], while the Network Management Datastore Architecture distinguishes configuration-related and operational state concepts [4]. These sources support canonical service representations and explicit treatment of desired and observed state. RFC 8969 is informational; citing its framework does not establish conformance to a standard or validate a particular implementation. In the proposed architecture, models make domain contracts and state exchanges explicit, while planning ownership and consistency rules determine how those models are used operationally.

Software-defined networking separates control-plane functions from packet forwarding and supports programmable network behavior. The survey by Kreutz and colleagues describes the broader SDN architecture and its technical challenges [2]. Logical centralization does not require all computation to execute in one process or location, but a distributed implementation still needs coordination and failure-handling rules. The present design adopts programmable control as a foundation and adds explicit intent versions, domain-level responsibilities and continuous outcome verification. These additions define the proposed architecture's operating model; their performance advantage over alternative control arrangements must be demonstrated rather than assumed.

The Borg cluster-management work illustrates mechanisms for managing large distributed workloads, including scheduling, resource allocation and failure handling [3]. Such mechanisms inform the use of ownership keys, work queues and bounded execution responsibilities in this proposal. Enterprise network changes introduce additional constraints: operations may affect shared policies, depend on changes in other domains, or leave partially modified infrastructure after a failure. Consequently, workload distribution must be combined with dependency tracking, idempotency, checkpoints, compensating actions where appropriate and explicit verification of the resulting service state. Borg provides relevant distributed-systems experience, not direct evidence that these network-specific choices are validated.

RFC 8993 describes a reference model for autonomic networking and the interaction of autonomic functions with their supporting infrastructure [6]. It is also informational. Its treatment of managed autonomy provides context for allocating authority between enterprise policy and local control. Taken together, the cited work supplies foundations for model-driven management, programmability, distributed workload coordination and autonomic operation. The proposed synthesis makes their interaction explicit through an authoritative intent record, hierarchical domain contracts, differentiated consistency and local reconciliation. The evaluation framework is intended to test the trade-offs of that synthesis, including the overhead and failure modes introduced by the hierarchy itself.

III. RESEARCH METHODOLOGY

The methodology is a design-oriented conceptual study. It derives architectural requirements from the cited networking and distributed-systems literature, assigns responsibilities to control-plane layers and formulates propositions that can be examined experimentally. The resulting architecture is a proposed design, not a report of a completed implementation. The evaluation plan identifies comparisons, workloads and measurements needed to assess the design without treating an architectural argument as empirical proof.

3.1 Requirement Identification

The proposed design requires intent submission, authorization, semantic interpretation, conflict checking, decomposition, planning, execution, state tracking, assurance, reconciliation and auditability. Model-driven management and datastore concepts inform the service and state representations used for these responsibilities [1,4,5]. Each execution decision and verification result must identify the applicable intent version. Domain contracts must specify local authority and the conditions that require cross-domain coordination or escalation.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

Scalability requirements concern workload as well as device count. Relevant dimensions include the number of domains, concurrent intents, topology changes, dependency relationships, telemetry events and recovery operations. The aim is to avoid unnecessary global work while preserving correctness. Bottleneck-free operation is not assumed: the evaluation must examine where planning, queues, state storage, assurance or cross-domain coordination become limiting factors.

3.2 Architectural Decomposition

The proposed control plane separates intent admission, semantic normalization, hierarchical planning, distributed execution, state management and assurance, and governance. These are logical responsibilities; the design does not require one service or deployment unit per layer. Global coordination retains enterprise authorization, intent semantics, cross-domain constraints and decisions affecting shared invariants. Domain controllers retain detailed topology, resource and capability knowledge within defined boundaries.

Interfaces between these responsibilities should state their inputs, outputs, authority and failure behavior. A domain contract should identify the assigned outcome, constraints, dependencies, intent version and evidence required for verification. The design analysis should check that a responsibility is neither left unassigned nor independently exercised by conflicting authorities.

3.3 Intent and State Modeling

An intent record should capture the desired outcome, scope, priority, constraints, service-level objectives and authorization context. Accepted changes create new versions so that planning, execution and assurance can be associated with an identifiable request. Canonical service representations can use model-driven principles to reduce dependence on vendor-specific syntax [1,5]. Desired outcomes remain distinct from device configuration and observed operational state. RFC 8342 distinguishes intended device configuration from operational state; a high-level enterprise intent is not itself the NMDA intended configuration datastore [4].

The model should also represent uncertainty, incomplete observations, verification status and provenance. Staleness tolerances must be explicit where they affect decisions. Comparing desired and observed state can identify a candidate deviation, but reconciliation should proceed only after the relevant policy, authority and operational constraints have been checked.

3.4 Distributed Planning and Execution

A global planner decomposes an accepted intent into bounded domain contracts. Domain planners then use local knowledge to construct implementation plans. Stable ownership keys, such as site, tenant, fabric or device group, determine which executor may act on a resource. Asynchronous queues, priorities and back-pressure regulate work admission and execution. Idempotent operations should ensure that retrying the same logical operation does not create unintended duplicate effects.

Long-running workflows need persistent progress records, checkpoints and a defined response to partial completion. Depending on the operation, recovery may involve retry, reconciliation, a compensating action or escalation; rollback is not assumed to be universally possible. Cross-domain dependencies and invariant checks constrain local autonomy. The design should therefore specify both the normal execution path and behavior under stale information, duplicate events, delayed messages and controller failure.

3.5 Proposed Scalability Evaluation

A future evaluation should compare three arrangements under equivalent workloads: centralized intent-to-device planning and execution; centralized planning with distributed execution; and hierarchical planning with local reconciliation. Synthetic enterprise topologies may vary device density, domain count, concurrent intents, topology churn and telemetry rate. Recorded operational traces may supplement these workloads if their provenance, representativeness and use permissions are established. No such traces or experiments are reported here.

Measurements should include admission latency, planning cost, recomputation scope, execution throughput, convergence time, assurance delay, telemetry volume, conflict detection, invariant violations and recovery time. Median and tail behavior should be reported across repeated runs with comparable resource allocations. Fault scenarios should include delayed telemetry, stale inventory, duplicate events, partial execution, executor loss and domain-controller failure. Analysis should test whether reduced global coordination offsets the hierarchy's additional contract,

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

queueing and verification overhead. The detailed evaluation framework in Section 6 specifies how these propositions could be investigated.

IV. HIERARCHICAL INTENT TO OUTCOME ARCHITECTURE

The proposed control plane translates enterprise objectives into governed network actions and continuously evaluates whether the intended outcomes remain satisfied. It separates six functional layers: intent interface and admission control, semantic normalization, hierarchical planning, distributed execution, state management and assurance, and governance. These are logical responsibilities rather than a requirement for six independent services. Versioned records and explicit contracts connect the layers.

The central architectural principle is to retain authoritative intent semantics and cross-domain constraints while delegating detailed planning, execution and recovery to bounded domains. Distribution is permitted only where it preserves the required correctness conditions. Figure 1 summarizes the responsibilities and feedback paths. The design is a conceptual proposal; its performance and correctness properties require evaluation.

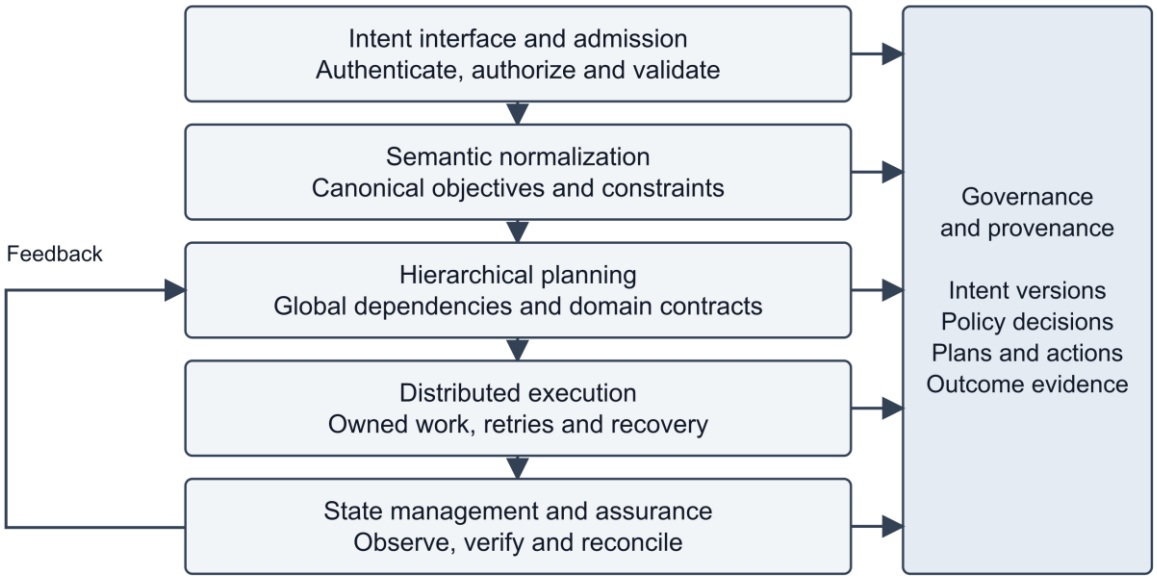


Figure 1. Six logical control-plane responsibilities and their feedback relationship.

4.1 Intent Interface and Admission Control

Administrators, applications and enterprise services submit desired outcomes through an intent interface. Examples include secure connectivity between application groups, minimum bandwidth, segmentation, availability and controlled access across domains. Each request identifies its scope, priority, service-level objectives, timing constraints, authorization context and applicable policies. These attributes make the request sufficiently explicit for validation and subsequent interpretation.

Admission control checks syntax, authorization, resource feasibility and policy compatibility using the available inventory and active intents. Malformed or unauthorized requests are rejected. Ambiguous requests require clarification or an explicitly authorized interpretation policy. Resource checks remain conditional on the freshness of the underlying information; admission does not guarantee that later execution will succeed. Conflicts discovered during planning or execution must therefore return to the appropriate decision authority.

An accepted request becomes a versioned intent record with a unique identifier. Each committed version is immutable; an authorized change creates a new version rather than silently overwriting the existing one. The record supplies a traceable reference for planning, verification and comparison with earlier objectives. There is one logical authority for each intent, even if its storage is replicated. Local views must identify their source version and cannot independently redefine the authorized objective.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

4.2 Semantic Normalization and Model-Driven Abstraction

Semantic normalization converts an admitted request into a canonical, machine-readable description of objectives, constraints, dependencies, resource requirements and verification conditions. Equivalent user expressions should map to equivalent representations. This step separates the meaning of an intent from the device commands eventually used to implement it.

YANG-based service and network models provide structured representations for this mapping [1,5]. They can reduce dependence on vendor-specific command syntax, but a modeling language does not itself resolve ambiguous business language or guarantee that a device supports a requested function. Translation therefore depends on explicit mappings, capability information and validation rules. Unsupported requirements must be reported rather than omitted.

Any unresolved ambiguity remains visible in the intent lifecycle. The platform should request clarification from an authorized user or apply a documented interpretation rule. It should not silently choose an implementation when alternative interpretations imply materially different access, resource use or service behavior. The normalized record preserves the interpretation and the conditions against which the outcome will be assessed.

4.3 Hierarchical Planning and Domain Contracts

The global planner decomposes a normalized intent into domain contracts instead of directly expanding every objective into all device operations. A domain may correspond to a campus, data center, wireless environment, wide-area network, security boundary, cloud region or tenant. Its boundary should reflect operational authority, resource dependencies and failure behavior, rather than an arbitrary division of the device inventory.

Each contract specifies the outcome assigned to a domain, relevant constraints, resource dependencies, priority and required completion evidence. The global planner retains responsibility for relationships between contracts and for enterprise-wide invariants. Examples include uniqueness within a shared address namespace, segmentation requirements and authorization boundaries. A domain cannot override these constraints merely because a local plan is feasible.

Domain planners use local topology, inventory, capabilities, policy and resource information to construct implementation plans. They may use algorithms suited to their technologies while preserving the contract's semantics. An infeasible contract or a dependency beyond local authority must be escalated. Independently accepted local plans are insufficient when their combined effects compete for a shared resource or violate a cross-domain invariant.

The planner maintains dependencies between intents, contracts and resources to support incremental replanning. A local change may require reconsidering only affected contracts, provided the dependency information is complete and no global constraint has changed. A wider dependency or policy change can legitimately require broader replanning. Locality is therefore a conditional design benefit rather than an unconditional guarantee.

4.4 Distributed Execution and Workload Ownership

Execution workers apply domain plans through asynchronous work queues. A partition key is a stable logical identifier used to group related work and assign responsibility; possible components include domain, site, tenant, fabric or resource group. Keys should preserve a clear owner for operations that affect the same resource. Partitioning is not safe if two independent workers can concurrently make incompatible changes without coordination. Ownership changes must prevent an outdated worker from continuing to act as the current owner.

Retries are necessary when requests or acknowledgments are lost. An idempotent operation has the same intended effect when repeated as when applied once. Where the target operation lacks that property, execution requires an appropriate deduplication or reconciliation mechanism. A stable operation identifier and durable outcome record can support that mechanism. Neither message delivery nor retry alone establishes that an external action occurred exactly once; uncertain outcomes must be checked before further action.

Long-running workflows retain checkpoints so that recovery can identify completed and pending steps. Compensating actions may address partial execution where the affected system supports them, but compensation is not a universal rollback guarantee. Some changes may be irreversible or require operator approval. Recovery must therefore account for the observed state and the current intent version. Priority scheduling, admission limits and backpressure are proposed to control bursts and protect constrained controllers or devices. Distributed execution increases available concurrency only while these ownership and safety conditions remain satisfied.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

4.5 State Management and Continuous Assurance

State management distinguishes the authorized desired outcome, observed operational conditions and provenance records. The desired outcome is derived from the active intent and its contracts. Observations describe what devices and services report, including their timestamps, scope and source. Provenance connects those observations and actions to the decisions that produced them. These categories should not be conflated.

RFC 8342 provides a useful distinction between intended configuration and operational state [4]. The intended configuration datastore contains configuration derived from the running configuration after applicable transformations; the operational state datastore includes configuration in effect and system state. A high-level enterprise intent is distinct from both: it requires translation into configuration and verification of service behavior. Successful configuration delivery therefore does not, by itself, establish satisfaction of the original objective.

Assurance compares observed conditions with contract-specific verification requirements throughout the lifecycle. It identifies drift, unsuccessful actions, unintended changes and potential service-level violations. High-volume telemetry can be processed near its source, with summaries, relevant events and verification evidence shared across domains. Summarization must retain enough context to assess an outcome; missing, delayed or stale observations should produce an explicit uncertainty state rather than an unsupported success decision.

A domain may reconcile a local deviation within its authorized scope. Deviations involving shared resources, conflicting contracts or enterprise-wide policy require escalation or hierarchical replanning. The architecture proposes this division to limit unnecessary global work while preserving accountability for cross-domain outcomes.

4.6 Governance, Provenance and Consistency Management

Governance records intent versions, authorization decisions, policy checks, plans, approvals, execution events, state changes and assurance findings. These records support explanation, audit and investigation of why an action was permitted and which evidence supports its outcome. If assurance uses probabilistic findings, their uncertainty and decision thresholds should remain visible; an inference is not equivalent to a verified invariant.

Consistency requirements depend on the consequence of disagreement. Shared-resource allocation, authorization changes and other global invariants require an authoritative coordination mechanism before incompatible actions can be committed. Local consensus can protect decisions within a domain, but cannot independently establish a global invariant across domains. The design must identify the responsible authority and required coordination boundary. If that coordination is unavailable, affected operations may need to wait rather than proceed with an unverified safety assumption.

Telemetry, intermediate progress and other observations may tolerate bounded staleness when their use permits it. Such bounds must be defined by the relevant decision or contract. Replicated intent records and local views remain subordinate to the authoritative version; they are not competing sources of intent. Differentiated consistency is intended to avoid imposing strong coordination on every observation while retaining it where correctness requires it. Its costs and practical limits remain evaluation questions.

4.7 End-to-End Control Flow

The lifecycle begins with an authenticated request, followed by admission checks and semantic normalization. An accepted, versioned record supplies the objective for global decomposition into domain contracts. Domain planners validate local feasibility and produce execution plans, subject to cross-domain constraints. Workers receive identified operations through controlled queues and report progress, uncertain outcomes and failures.

After execution, observations are evaluated against the contract and original intent. Sufficient evidence supports an outcome decision; deviations trigger authorized local reconciliation, broader replanning or human review. Governance preserves the decision history throughout this feedback loop. Figure 2 illustrates this sequence. The proposed architecture is thus a continuing intent-to-outcome process, with local autonomy bounded by authoritative semantics, explicit ownership and verifiable responsibilities.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

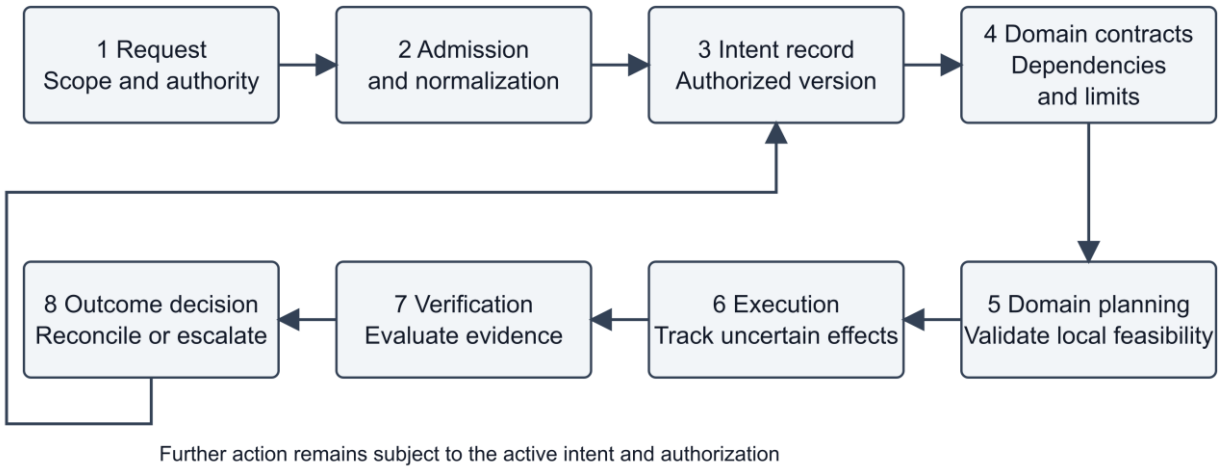


Figure.2. Proposed intent lifecycle from admission to verification and further authorized action.

V. SCALABILITY PROPOSITIONS AND ARCHITECTURAL CONSTRAINTS

The architecture motivates four propositions about planning, communication, semantic coordination and change propagation. They are conditional hypotheses for the evaluation framework, not measured findings or formal complexity proofs. Figure 3 illustrates the ownership boundaries and domain responsibilities on which they depend. Each proposition must be assessed against comparable workloads while checking correctness and recovery, because a reduction in processing is not beneficial if it weakens the required outcome.

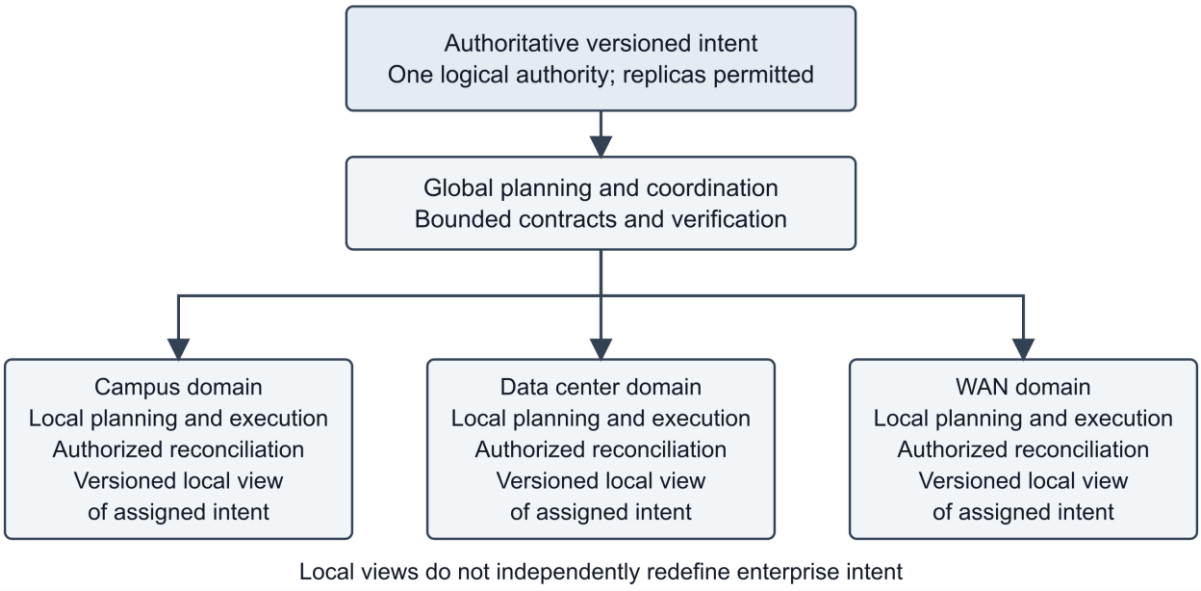


Figure.3. One logical intent authority with bounded domain responsibilities and versioned local views.

5.1 Proposition 1 Bounded Hierarchical Planning May Reduce Global Planning Work

When an intent can be represented by a bounded set of domain contracts, the global planner may perform less detailed work than a planner that expands every request directly into device actions. Domain planners then absorb technology-specific computation. The expected benefit depends on contract size, domain count, shared-resource dependencies and the cost of decomposition. A dense set of cross-domain constraints can remove the advantage or increase total work. Evaluation should therefore distinguish global planning load from aggregate computation and compare both with direct centralized expansion.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

5.2 Proposition 2 Authorized Local Reconciliation May Reduce Coordination Traffic

A domain that can diagnose and correct a deviation within its contract need not send every device-level observation through a global reconciliation loop. Contract summaries, exceptions and verification evidence may replace some continuous cross-domain state exchange. The benefit depends on the locality of failures, evidence requirements and the frequency of changes affecting multiple domains. It is not enough to reduce message counts: evaluation must also determine whether summaries preserve relevant information and whether reduced communication delays conflict detection, assurance or convergence.

5.3 Proposition 3 Authoritative Versioned Intent May Reduce Semantic Divergence

An immutable record for each committed intent version provides planners, workers and assurance components with a common reference. Replicas and domain views can distribute access without acquiring independent authority to redefine that reference. This may reduce inconsistent interpretations and improve traceability across retries and recovery. The proposition requires explicit version selection, authorized update rules and handling of stale views. Versioning alone does not prevent concurrent execution of incompatible plans, ensure correct translation or make restoration of an earlier objective safe; these require additional coordination and validation.

5.4 Proposition 4 Dependency-Aware Replanning May Limit Change Propagation

If dependencies are recorded accurately, a topology, capability or policy change may invalidate only the contracts and plans that depend on it. Recomputing those elements could avoid rebuilding unaffected plans. The benefit depends on dependency accuracy, maintenance overhead and fan-out. Missing dependencies can leave an apparently unchanged plan incorrect, while conservative dependencies can trigger extensive recomputation. Evaluation should compare the amount of recomputed work and resulting convergence with complete replanning, including changes whose effects legitimately span the enterprise.

5.5 Constraints and Conditions for Evaluation

Work distribution principles from large-scale cluster management provide useful context [3], but network operations also affect external, stateful resources. Poor partition keys can concentrate work on a few owners, create hot shards or separate operations that require joint decisions. Too many small domains may increase coordination, whereas excessively large domains may overload local planners and enlarge failure scope. Ownership and dependency boundaries must therefore be evaluated together.

Unbounded telemetry retention, repeated global policy checks, queue growth and stale observations can shift the bottleneck from planning to assurance or recovery. Capacity analysis should consider concurrent intents, telemetry rates, topology churn, dependency fan-out, retained state and coordination demand, alongside device count. Instrumentation must distinguish planner, executor and assurance load. Horizontal replication is consequently a design option rather than proof of scalability. The proposed evaluation should identify where the architecture offers a benefit, where its assumptions fail and which safety requirements limit concurrency.

VI. PROPOSED EVALUATION FRAMEWORK

The evaluation should test the architectural propositions under controlled and comparable conditions. Its purpose is to determine whether delegating work reduces global coordination without weakening outcome verification, policy enforcement or recovery. This section specifies an experimental protocol; it does not describe experiments already performed or supply measured results. Figure 4 shows the proposed comparison pipeline.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

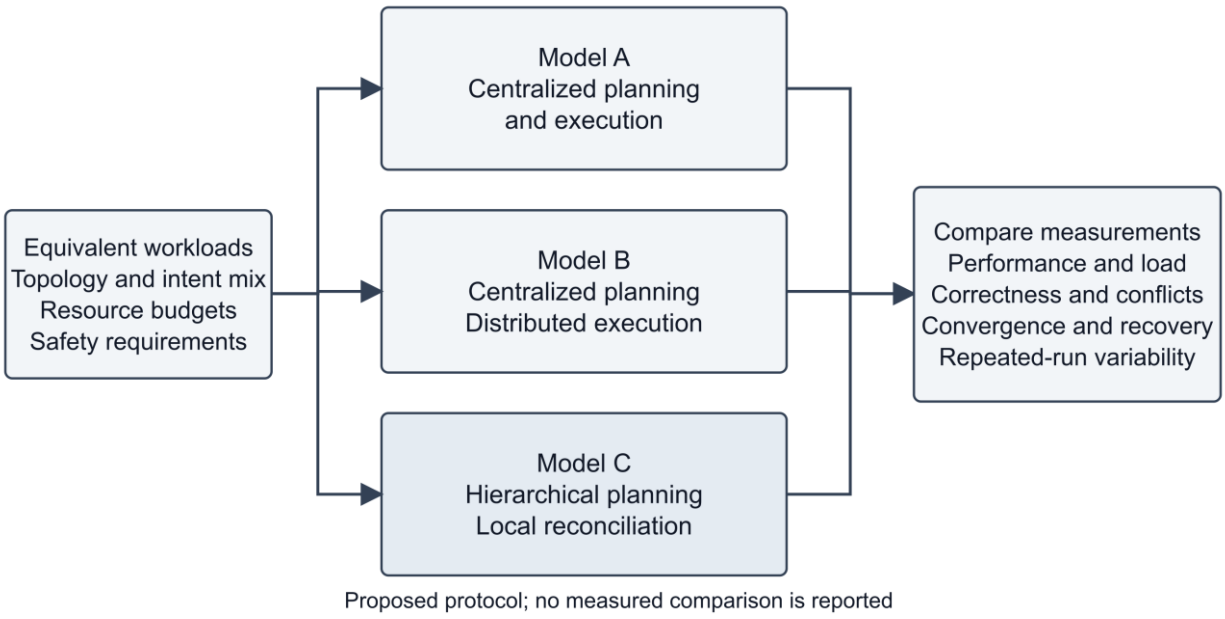


Figure.4.Proposed comparison of three architectures under equivalent workloads and safety requirements

6.1 Experimental Setup and Workload Generation

A future implementation should define the software versions, controller resources, device or simulator capabilities, storage configuration and communication conditions used in the test environment. All compared designs should receive equivalent resource budgets, supported operations, policy requirements and verification criteria. Any differences needed to implement a design should be disclosed so that an apparent improvement is not simply the consequence of additional resources or weaker checks.

Synthetic topologies should vary device count, domain count, connectivity, service dependencies and the distribution of resources across sites. Workloads should vary concurrent intents, request arrival rates, topology churn, dependency fan-out and telemetry volume. Cases should include local and cross-domain intents, conflicting requirements, urgent corrections and changes affecting shared resources. Each generated workload should retain its parameters and random seed where applicable. Recorded operational traces could supplement the synthetic workloads if their provenance, permissions and limitations are documented; this paper does not supply such traces.

The protocol should include stable-load and burst scenarios, repeated runs and a stated approach to warm-up, run duration and stopping conditions. Workload generation must preserve comparable offered demand across alternatives. Overload, rejected requests, timeouts and incomplete operations should be retained in the analysis rather than excluded as unsuccessful samples. The chosen workload ranges would define the scope of any resulting claim; they would not establish performance for every enterprise environment.

6.2 Comparative Architectures

Model A is centralized planning and execution: a global service expands intents into device-level plans and manages their execution and reconciliation. Model B is centralized planning with distributed execution: the global service retains detailed planning responsibility while workers execute assigned operations. Model C is hierarchical planning with local reconciliation: a global service defines bounded domain contracts and cross-domain constraints, while domain controllers plan, execute and reconcile within their authority. These labels are used consistently in Figure 4.

The comparison should isolate responsibility placement rather than equate every centralized design with SDN or assume that SDN requires a single physical controller. Alternative designs may themselves use replicated services. Each model should preserve equivalent semantics, safety requirements and outcome checks. A secondary comparison could disable individual mechanisms, such as incremental replanning or telemetry summarization, to examine their contribution. Such comparisons would require implementations and measurements; no superiority of Model C is established by the diagram.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

6.3 Performance and Scalability Metrics

Admission latency should measure the interval from request receipt to an acceptance or rejection decision. Planning cost should distinguish global computation from the total work performed across planners, including contract generation, dependency maintenance and recomputation. Execution throughput should count completed operations or verified intents using an explicitly stated unit. Convergence time should measure the interval from an accepted intent or relevant disturbance to satisfaction of a defined outcome. Assurance delay should measure the time required to reach a verification decision after the relevant observations become available.

Additional measures should include queue depth, rejected or expired work, controller CPU and memory, stored state, telemetry traffic, cross-domain messages and the number of recomputed objects. Median and specified tail percentiles should be reported across repeated runs, together with sample counts and an appropriate account of variability. Results should distinguish offered load from completed work and local resource savings from aggregate system cost. These definitions prevent a reduction in traffic or CPU load from being interpreted as an improvement when the system has instead deferred or dropped required work.

6.4 Correctness and Conflict Handling

Correctness checks should accompany performance measurements. Test cases should specify the authorized intent version, expected service outcome, applicable invariants and permitted actions before execution. Known conflicting intents and resource constraints can then be used to assess missed conflicts, false conflict reports and the timing of detection. Outcome verification should distinguish satisfied, violated and indeterminate conditions; insufficient observations should not be counted as success.

The evaluation should record invariant violations, unauthorized changes, execution against obsolete intent versions and differences between configuration completion and verified service behavior. Shared-address allocation, segmentation and authorization provide examples of requirements whose coordinated enforcement can constrain concurrency. These requirements should be checked consistently across all models. The objective is to determine the cost of preserving correctness as scale increases, rather than reward a design for omitting checks.

6.5 Fault Injection and Recovery Evaluation

Fault scenarios should include executor loss, domain-controller loss, delayed or duplicated messages, stale inventory, delayed telemetry, network partitions and partial success of a multi-device change. The timing and duration of injected faults should be specified relative to the intent lifecycle. Where a test cannot safely use real infrastructure, an explicitly described simulator or testbed can represent the affected behavior. Figure 5 shows the proposed relationship between disturbance, observation, governed reconciliation and verification.

Recovery measurements should include detection delay, time to a verified acceptable state, residual inconsistency, duplicate external effects and the number of operations requiring compensation or human intervention. An acceptable state may be the requested outcome, a verified prior state or another authorized safe condition, depending on the operation. Rollback should not be assumed to restore all external effects. Tests should determine whether local recovery remains within authority and whether faults involving shared constraints are escalated correctly. The proposition that local reconciliation can limit unnecessary global recovery remains to be tested.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

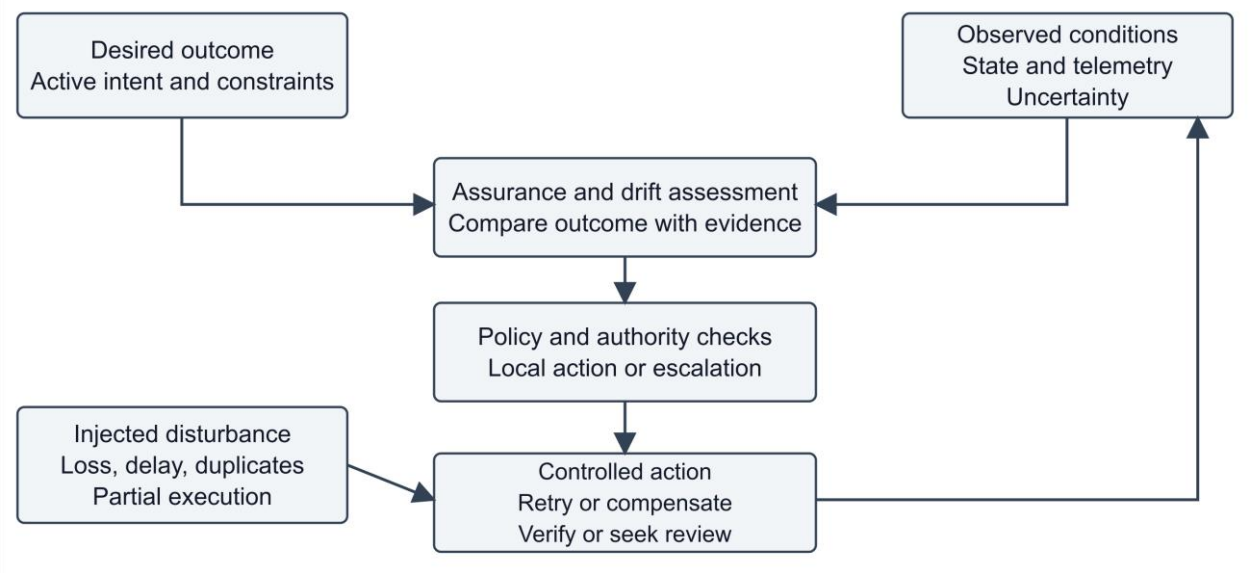


Figure.5. Proposed fault-injection, assurance and governed-recovery loop.

6.6 Analysis and Reproducibility

Analysis should examine how each metric changes with workload and topology rather than report only one aggregate improvement. Relevant questions include when global planning saturates, whether contract coordination outweighs local savings, how dependency fan-out affects recomputation and whether telemetry summarization delays assurance. Comparisons should include adverse cases in which the hierarchy offers little benefit or performs worse. Performance differences must be interpreted alongside correctness, uncertainty and recovery outcomes.

Any future empirical report should provide enough implementation, configuration, workload and measurement detail to support reproduction or a clearly bounded independent assessment. It should identify unavailable data and explain the resulting limits without substituting architectural diagrams for measured evidence. This manuscript supplies the design and protocol only.

VII. DISCUSSION AND LIMITATIONS

The proposal treats scalability as a problem of workload placement and coordination. Increasing device count is one source of load, but concurrent intent processing, telemetry, topology changes and cross-domain dependencies may dominate in a smaller but more dynamic environment. The architecture therefore retains a global account of meaning and constraints while delegating detailed work where authority and dependencies permit. Its expected benefits depend on the locality of those responsibilities and on the cost of maintaining their contracts.

This arrangement does not eliminate central responsibility. Authorization, cross-domain resource decisions and authoritative intent versions still require a defined decision authority. Physical replication can improve availability or read access, but cannot by itself resolve conflicting decisions. During partitions or loss of coordination, some operations may need to wait to preserve safety. The architecture should make that trade-off visible instead of implying simultaneous guarantees of unrestricted availability and global correctness.

Domain boundaries introduce their own risks. Small domains can increase contract traffic and dependency management; large domains can concentrate load and enlarge failure scope. Inaccurate dependency models can miss required replanning, while conservative models can erase the benefit of incremental work. Similarly, local telemetry aggregation can reduce communication but may discard information needed to detect cross-domain effects. These are design variables for evaluation, not assumptions that the hierarchy automatically handles.

The cited sources establish relevant foundations, but they do not validate this particular synthesis. The manuscript does not establish historical priority over every related architecture, formal complexity bounds, implementation correctness or measured improvement. Its contribution is a structured description of responsibilities, contracts and testable

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 4, Issue 3, May-June 2022 |

DOI:10.15662/IJEETR.2022.0403008

propositions. The proposed evaluation protocol identifies evidence needed to assess those propositions. Comparisons with additional implementations, formal analysis of selected safety properties and field-specific requirements would be appropriate subsequent work.

Security and governance are also implementation-dependent. An authorized request can still be translated incorrectly; an apparently successful action can still violate a service objective. The proposed separation between interpretation, permission to act, execution and verification is intended to make these failures detectable and attributable. It is not a claim that every error can be prevented or automatically repaired. The same caution applies to recovery: compensation, reconciliation and operator intervention must reflect what the affected systems can actually support.

VIII. CONCLUSION

This paper proposes a hierarchical control-plane architecture for enterprise intent-based networking. It separates intent admission, semantic normalization, planning, execution, state management and assurance, and governance. An authoritative record for each intent version connects these responsibilities, while bounded domain contracts delegate work without granting independent authority to redefine the enterprise objective.

The design combines stable workload ownership, asynchronous execution, idempotent or deduplicated operations, dependency-aware replanning and authorized local reconciliation. Differentiated consistency reserves coordinated decisions for invariants that require them and permits explicitly bounded staleness where the decision context allows it. Four conditional propositions identify possible reductions in global planning, communication, semantic divergence and change propagation.

The proposed evaluation compares centralized planning and execution, centralized planning with distributed execution, and hierarchical planning with local reconciliation under equivalent workloads and safety requirements. It specifies measurements for performance, correctness, convergence and recovery. No implemented prototype or measured benefit is reported. Empirical work is needed to determine where the architecture improves operation, where its assumptions fail and how its coordination costs affect enterprise-scale behavior.

REFERENCES

[1] Q. Wu and M. Boucadair, Eds., D. Lopez, C. Xie, and L. Geng, "A Framework for Automating Service and Network Management with YANG," RFC 8969, Jan. 2021, doi: 10.17487/RFC8969. https://doi.org/10.17487/RFC8969

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.
https://doi.org/10.1109/JPROC.2014.2371999

[3] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15), ACM, 2015, doi: 10.1145/2741948.2741964.
https://doi.org/10.1145/2741948.2741964

[4] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, and R. Wilton, "Network Management Datastore Architecture (NMDA)," RFC 8342, Mar. 2018, doi: 10.17487/RFC8342. https://doi.org/10.17487/RFC8342

[5] M. Bjorklund, Ed., "The YANG 1.1 Data Modeling Language," RFC 7950, Aug. 2016, doi: 10.17487/RFC7950. https://doi.org/10.17487/RFC7950

[6] M. Behringer, Ed., B. Carpenter, T. Eckert, L. Ciavaglia, and J. Nobre, "A Reference Model for Autonomic Networking," RFC 8993, May 2021, doi: 10.17487/RFC8993. https://doi.org/10.17487/RFC8993