From Network Automation to Trustworthy Autonomous Networking in the LLM Era: A Network Control Intelligence Perspective

Tianzhu Zhang, Changgang Zheng, Shanshan Wang, Yarui Zhang, Lina Shi, Yue Jin, Xiaofei Wang, Meikang Qiu

***Abstract*—Since the inception of modern communication networks, the quest for operations automation has never ceased. Yet the evolution of network automation is difficult to characterize with a single maturity ladder. Throughout this history, network control systems have expanded their capabilities for observation, decision support, routine execution, and operator interaction, but these capabilities have not advanced uniformly. Such uneven progress makes the degree of automation an unreliable proxy for trustworthy network-side actuation. The unresolved question is not simply how much automation a system provides, but under what conditions it can be entrusted to change the network state. This paper examines that question through Network Control Intelligence (NCI), a five-axis framework spanning Decision Logic, Adaptability, Knowledge, Control Delegation, and Interface. We use NCI to organize the evolution of network-control systems into three eras: rule-based and scripted automation, programmable and data-driven control, and Large Language Model (LLM)-enabled network operations. Viewed through this framework, the three eras reveal a persistent asymmetry. Earlier automation made monitoring and routine control repeatable; programmable and data-driven control expanded optimization, telemetry, and domain-specific closed loops; and LLM-enabled operations now strengthen intent interpretation, evidence synthesis, change authoring, and workflow coordination. None of these gains, however, automatically determines when network control should be trusted to change the network state. We frame trustworthy autonomy as a governed alignment between what a system can infer, what it can verify, and what it is authorized to execute. On that basis, the paper develops a reference architecture that separates proposal generation from governed execution, identifies recurring integration patterns for LLM-enabled operations, and derives a research agenda for higher network autonomy under explicit assurance, safety, and governance constraints.**

***Index Terms*—Autonomous networking, network control intelligence, large language models, and closed-loop automation.**

I. Introduction

Network operations have long been driven by a persistent ambition: to reduce human intervention while maintaining network reliability and service assurance [1]. Over decades, this movement has reshaped network infrastructure from hardware-centric architectures built around dedicated network appliances toward software-defined, virtualized, and programmable systems [2]–[5]. In parallel, network operations and management have shifted from rule-based, operator-driven workflows toward programmable control loops informed by telemetry and data-driven analytics [6], [7]. Beyond these technical shifts, the same operational ambition has been framed through several related visions, including autonomic networking [8], self-driving networks [9], zero-touch network and service management (ZSM) [10], and autonomous networking [11]. These visions share a practical concern: how much intelligence and authority can be moved into the control loop without compromising network reliability, accountability, or service quality? This paper uses the term "autonomous networking" to refer to network control systems that perceive the operational state, reason over objectives and constraints, and recommend or execute actions within a governed operational envelope.

With the global rollout of 5G and the expected progression toward 6G, network systems are expected to face greater scale, heterogeneity, and service dynamism [12]. This growing operational complexity has elevated autonomous networking from a long-term research vision into a strategic priority for many network operators [13]. At the technical level, however, current automation techniques can boost operational efficiency but remain largely constrained in their ability to interpret operator intent, synthesize heterogeneous operational evidence, and coordinate decisions across domains and timescales [14]–[17]. These limitations suggest that network autonomy cannot be judged solely by the sheer number of operations automated. It also depends on how decisions are formed, how behavior adapts, what operational knowledge is available, where authority is placed, and how objectives and evidence are exchanged. We refer to this set of capabilities as Network Control Intelligence (NCI), the structured control capability through which network systems transform observations, objectives, policies, and operational knowledge into recommendations or actions under explicit authority, safety, and auditability constraints [18]–[24].

In this paper, we use NCI as a profiling lens to compare network control systems in terms of their underlying control properties, including how they make decisions, adapt over time, represent and use operational knowledge, delegate control authority, and exchange objectives, evidence, and feed-

T. Zhang, L. Shi, and Y. Jin are with Nokia Bell Labs, Massy, France. (Email: tianzhu.zhang@nokia-bell-labs.com, lina.1.shi@nokia.com, yue.1.jin@nokia-bell-labs.com)

C. Zheng is with the University of Oxford, Oxford, UK. (Email: changgang.zheng@eng.oxfordalumni.org)

S. Wang is with Telecom Paris, Institut Polytechnique de Paris, Palaiseau, France. (Email: shanshan.wang@telecom-paris.fr)

Y. Zhang is with École Normale Supérieure Paris-Saclay, Université Paris-Saclay, Gif-sur-Yvette, France. (Email: yarui.zhang@ens-paris-saclay.fr)

X. Wang is with Tianjin University, Tianjin, China. (Email: xiaofeiwang@tju.edu.cn)

M. Qiu is with Augusta University, Augusta, Georgia, USA. (Email: qiumeikang@gmail.com)

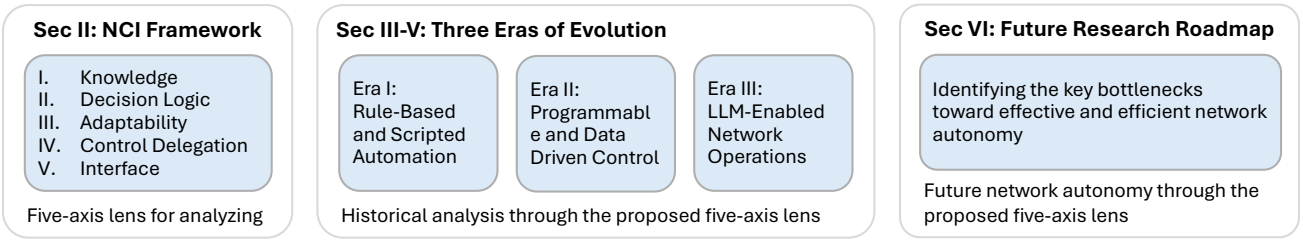


Fig. 1. Organization of the paper. The paper first introduces the Network Control Intelligence (NCI) framework, then uses its five axes to analyze three stages in the evolution of autonomous networking, and finally derives a roadmap for trustworthy progress toward higher levels of autonomy.

back. The five-axis view also gives the historical development of autonomous networking a more precise structure. The development of NCI unfolded through successive changes in how networks were observed, reasoned about, programmed, and governed. In its early days, network operations and management, often termed Operations, Administration, and Maintenance (OAM), relied on human operators working through Operations Support Systems (OSS) and Network Management Systems (NMS), via static procedures, scripts, and control logic specified by domain experts [25]. These approaches made routine work more repeatable, but their rule sets and workflows remained limited by what human teams could engineer, audit, and maintain. As topologies, service portfolios, and traffic patterns became more diverse, handcrafted control logic struggled to keep pace with operational complexity. This limitation motivated the evolution from scripted procedures toward programmable control. The subsequent wave of network softwarization and programmability gave the long-standing automation agenda a stronger technical foundation. Logically centralized control, virtualized services, extensive telemetry, and data-driven analytics enabled finer-grained traffic steering, elastic service composition, and policy-driven orchestration [6]. This shifted network operations from script-based management to domain-specific optimization, e.g., traffic engineering, energy management, and Radio Access Network (RAN) resource allocation [7]. Nonetheless, many resulting control loops remained domain-specific, with limited support for end-to-end decision-making aligned with operator intent [26]. Human oversight still plays a central role in orchestrating complex workflows, handling unanticipated situations, and reconciling competing objectives.

The rise of large language models (LLMs) adds another chapter to this history. Unlike conventional optimizers or domain-specific controllers, LLM-enabled systems are well suited to tasks that require intent interpretation, evidence synthesis, and multi-step workflow composition. When integrated with retrieval systems, external tools, and operational interfaces, they can interpret requests using relevant operational records, invoke appropriate tools, and assemble workflows [27], [28]. In network operations, they can generate structured, auditable outputs, such as diagnostic interpretations [18], operational procedures [19], and candidate configuration changes [22]. These outputs can improve human- or controller-mediated decision-making, but they do not remove the need for governed execution. In practice, LLM-enabled operational systems are seldom isolated prompt-response chatbots. They are increasingly embedded in agentic workflows that retrieve operational context, invoke external tools, maintain state across steps, validate intermediate outputs, and recover from failed actions. In such settings, long-horizon reliability depends not only on model capability, but also on how these workflow mechanisms are designed, regulated, and evaluated [29]–[32]. In network operations, such agents can improve semantic mediation across operational interfaces and support workflows aligned with operator policies and Service-Level Objectives (SLOs), while execution remains mediated by domain controllers, verifiers, authorization mechanisms, rollback procedures, and change-management systems [18]–[20]. Recent LLM-enabled network-operation systems and industry prototypes point to an incremental architecture: LLMs help assemble intent, evidence, and candidate workflows, while authoritative control remains anchored in specialized control loops and governed execution paths [33]–[37].

This progression motivates the three-era organization used in this paper. The eras are not separated by chronology alone, but by changes in how NCI is realized and governed. Rule-based and scripted automation refers to systems in which operational intelligence is primarily encoded in human-defined procedures, scripts, alarms, and management rules. Programmable and data-driven control refers to systems in which software-defined control, virtualization, telemetry, optimization, and learning shift decision-making to programmable controllers and analytics pipelines. LLM-enabled network operations refer to systems in which language-model components assist with intent interpretation, evidence synthesis, and workflow composition, while authoritative execution remains outside the model. Although earlier mechanisms often persist even after newer ones emerge, the three-era structure captures successive shifts in how NCI is expressed and constrained.

Within each era, this paper reviews representative network-control systems. The selected systems satisfy one or more of four criteria: they introduce or exemplify a mechanism that changes at least one NCI axis; they have influenced research, standards, or operational practice; they provide sufficient public technical detail to support axis-level profiling; or they illustrate a distinct role in the control stack, such as monitoring, configuration, traffic engineering, fault diagnosis, orchestration, or LLM-enabled workflow composition. This selection strategy allows the paper to show how NCI has evolved across its five axes and why progress toward autonomy

has remained uneven.

The main contributions of this paper are as follows:

- It introduces the Network Control Intelligence (NCI) framework for profiling network control systems across five coupled axes: Decision Logic, Adaptability, Knowledge, Control Delegation, and Interface.
- It applies this framework to representative network-control systems across three eras, selected for their influence, technical specificity, and ability to illustrate shifts along the NCI axes, showing that the evolution of network-control intelligence has been uneven and is not reducible to a single maturity trajectory.
- It characterizes the LLM-enabled era by separating proposal generation from governed execution, in which LLM-enabled components improve intent interpretation, evidence synthesis, change authoring, and workflow coordination, while execution remains mediated by controllers, verification, authorization, and change-management workflows.
- It derives a forward-looking roadmap for trustworthy autonomous networking, showing how future progress depends on the concerted maturation of operational knowledge, decision logic, adaptability, the Control Delegation axis, and operator interfaces under explicit assurance constraints.

As illustrated in Fig. 1, the remainder of this paper is organized as follows. Section II introduces the NCI framework and situates it relative to standards, industry frameworks, and related literature. Sections III–V trace the evolution of autonomous networking across three eras through the five-axis NCI lens. Section VI develops the forward-looking roadmap for trustworthy progress toward higher levels of network autonomy. Section VII concludes the paper.

II. Network Control Intelligence for Autonomous Networking

To characterize network control systems, we use an abstract control model. In this model, a network is represented as a collection of domains, such as the RAN, transport, core, and data center or cloud/edge domains. Each domain provides telemetry channels that expose *observables*, including counters, events, key performance indicators (KPIs), logs, traces, alarms, and topology state. It also provides control and management interfaces that expose *actuators*, such as configuration knobs, application programming interfaces (APIs), and orchestration hooks. Within this model, an *NCI component* denotes any control or management function that consumes observables, interprets them in light of explicit objectives, policies, and operational knowledge, and produces recommendations or actions for the relevant actuators. Depending on its authorized role, the component may assist human operators, propose candidate changes, execute scoped control actions, or participate in end-to-end orchestration. NCI profiles the role that such a function plays in a control process; it is not an intrinsic intelligence score for a software artifact.

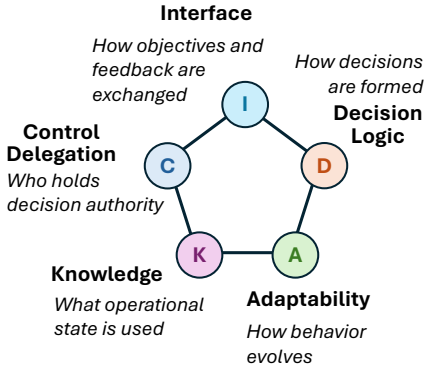
From a control perspective, autonomous networking can be viewed as the gradual enrichment of the classical sense-think-act loop. In this paper, *open-loop automation* denotes predefined execution without runtime feedback correction. *Closed-loop automation* denotes feedback-based operation in which monitored outcomes can trigger corrective actions. *Control-loop autonomy* further implies that feedback is interpreted against explicit objectives, constraints, and predictive or adaptive models within an authorized operational envelope. The network is sensed through telemetry, events, topology, inventory, and service-state observations; these observations are interpreted using operational knowledge and decision logic; and candidate actions are applied via controllers, orchestrators, or management interfaces, subject to policy and assurance constraints. NCI extends this control view by making five properties explicit: how decisions are formed, how behavior adapts, how operational knowledge is represented, how control authority is delegated, and how objectives and evidence are exchanged, as illustrated in Fig. 2.

A. Network Control Intelligence: a five-axis view

The NCI framework consists of five analytically distinct but operationally coupled axes: **Decision Logic**, **Adaptability**, **Knowledge**, **Control Delegation**, and **Interface**. Each axis captures a structural dimension of an autonomous network system, as shown in Fig. 2. The framework classifies systems by evolvable control properties, regardless of implementation technology. Legacy scripted tools, optimization-based controllers, learning-driven systems, and LLM-enabled agents can thus all be described within the same five-dimensional space. This makes it possible to compare, in a common vocabulary, how different systems form decisions, adapt to changing conditions, represent operational knowledge, delegate control authority, and interact with external actors.

Framework Design Rationale: The five-axis framework recasts the control-loop view as a set of profiling dimensions. The axes do not correspond to separate implementation modules; instead, they describe how decisions are formed, how behavior changes over time, how operational knowledge is represented, how control authority is delegated, and how objectives and feedback are exchanged. This distinction is important because two systems may use similar controller architectures yet differ sharply in delegation, knowledge scope, adaptability, or interface abstraction. Conversely, systems built from different technologies may occupy similar positions in the NCI space if they expose comparable control properties.

The choice of these five axes follows from the minimal functional path of a network-control process. Classical feedback and cybernetic views emphasize sensing, state, feedback, and action in regulated systems [38], [39], while agent-oriented models describe intelligent behavior as the mapping from perceptual input and objectives to action [40]. In operational settings, this mapping also depends on situation awareness [41], the system's ability to adapt to changing conditions [42], the allocation of authority between humans and automation [43], and the interfaces through which objectives, evidence, and feedback are exchanged. These considerations motivate the five NCI axes: *Knowledge*, *Decision Logic*, *Adaptability*, *Control Delegation*, and *Interface*. Other important concerns, including security, assurance, governance, scalability, interoperability, explainability, cost, and performance, are treated



Levels for Each Dimension

Interface	Decision Logic	Adaptability	Knowledge	Control Delegation
I0: Device-level interfaces	*D0: Reactive rules*	*A0: Static behavior*	*K0: Observational data*	*C0: Manual operation*
I1: Consolidated tools and APIs	*D1: Programmatic policies*	*A1: Configurable behavior*	*K1: Situation awareness*	*C1: Human-in-the-loop*
I2: Intent-based interfaces	*D2: Learned policies*	*A2: Periodic data-driven updates*	*K2: Domain models*	*C2: Supervised control*
I3: Conversational / explainable interfaces	*D3: Deliberative planners*	*A3: Continuous self-adaptation*	*K3: Meta/external knowledge*	*C3: High delegation under constraints*
				C4: Self-governing network

Fig. 2. The five-axis framework for Network Control Intelligence. The axes describe complementary control properties: how decisions are formed, how behavior adapts, what operational knowledge is available, how authority is delegated, and how objectives, evidence, and feedback are exchanged. In a control loop, observations are interpreted through operational knowledge and refined through interfaces. Decision logic and adaptation use this knowledge to produce recommendations or actions, while Control Delegation defines whether a system's outputs remain advisory, require human approval, or can be executed automatically within an authorized scope.

as cross-cutting constraints rather than additional axes. They determine whether movement along the five axes is safe, reliable, and operationally defensible, but they do not describe a separate functional step in the control loop [44].

The purpose of the framework is diagnostic rather than to assign a single ordered autonomy level. It asks which control properties are present, which are missing, and which assumptions must hold for a given NCI profile to be operationally credible. This makes it possible to compare systems without forcing them onto a single autonomy ladder. The axes are analytically distinct but not independent; richer knowledge can change what decision logic can verify, stronger interfaces can change what operators can safely approve, and broader autonomy can expose weaknesses in adaptation or assurance.

Decision Logic: The **Decision Logic** axis describes how decisions are formed. It characterizes how an NCI component maps observations, objectives, and constraints to recommendations or actions, ranging from fixed reactive rules to explicit optimization, learned policies, and deliberative planning. This axis should not be read as a simple quality ladder: optimization-based controllers can be more auditable and constraint-aware than learned policies for some tasks. In contrast, learned policies may offer faster inference or better empirical generalization in others. For profiling, we distinguish four representative rungs:

- **D0 (Reactive Rules):** Hand-crafted conditions, thresholds, or event-condition-action rules that map current observations to immediate responses.
- **D1 (Programmatic Policies):** Structured control programs, policy engines, or optimization mechanisms operating over explicit objectives, models, or constraint sets.
- **D2 (Learned Policies):** Data-driven mappings from observations to actions or recommendations, trained from historical, simulated, or online experience.
- **D3 (Deliberative Planners):** Goal- and plan-based controllers that evaluate alternatives, reason over future network states, and revise plans as conditions evolve.

Note that D3 systems may integrate D1-style solvers and D2-style learned components into a broader planning loop; the defining property is the ability to reason about alternatives and multi-step consequences.

Adaptability: This axis describes how behavior changes over time. It captures how an NCI component changes its behavior as network conditions, traffic patterns, objectives, or policies evolve. It distinguishes systems whose behavior is fixed at design time from systems whose parameters, models, or policies are updated periodically or continuously. For profiling, we distinguish four regimes:

- **A0 (Static Behavior):** Fixed rules, policies, or parameters; any change requires manual reconfiguration.
- **A1 (Configurable Behavior):** Predefined modes, profiles, or parameters that operators can select or tune, possibly aided by basic analytics.
- **A2 (Periodic Data-Driven Updates):** Models, parameters, or configurations are periodically re-optimized from accumulated data, typically under human oversight.
- **A3 (Continuous Self-Adaptation):** The NCI detects drift and updates its models, policies, or strategies online within an authorized operational envelope.

Higher adaptability can improve robustness under non-stationarity, but it also increases the need for monitoring, regression control, and safety constraints to prevent unreliable updates. Adaptability also depends on what state and context the component can observe or represent.

Knowledge: The **Knowledge** axis describes what operational context is available to an NCI component and how that context is structured. It includes raw telemetry, inferred state, topology, and service models, policies, constraints, external documentation, and higher-order knowledge about model scope or uncertainty. For profiling, we distinguish four layers:

- **K0 (Observational Data):** Raw or lightly processed measurements, such as counters, system logs, events, flow records, and link metrics.
- **K1 (Situation Awareness):** Structured views of current network state, such as topology, active traffic matrices, detected anomalies, alarms, and service-impact context.
- **K2 (Domain Models):** Predictive, causal, policy, or simulation models that represent how a domain behaves and how candidate actions may affect it.

- **K3 (Meta and External Knowledge):** Cross-domain or external knowledge sources, such as knowledge graphs, digital twins, documentation, historical incident records, and explicit knowledge about model limitations, uncertainty, or freshness.

The Knowledge axis is not only about data volume. Its central concern is whether the NCI has a faithful, structured, and up-to-date representation of the operational context required for safe decision-making.

Control Delegation: This axis describes where control authority resides. It captures how decision authority is delegated between human operators and automated components. It is distinct from decision quality: a system may have sophisticated reasoning but low delegation if its outputs remain advisory, whereas a simpler controller may have higher delegation within a narrow, well-verified domain. For profiling, we use five delegation rungs:

- **C0 (Manual Operation):** The system provides information or scripted assistance, but humans select and approve all substantive actions.
- **C1 (Human-in-the-Loop):** The system can propose actions with explicit human approval for execution.
- **C2 (Supervised Control):** The system autonomously handles routine tasks within predefined guardrails and escalates unusual, ambiguous, or high-risk cases.
- **C3 (High Delegation Under Constraints):** The system makes and executes decisions across a broader scope without per-action approval, while operating within explicit policy, safety, and escalation constraints.
- **C4 (Self-Governing Network):** The system is self-governing within a declared operational design domain, planning, executing, and adapting strategies under high-level human intent and explicit assurance assumptions.

Note that higher delegation must be justified by capabilities on the other axes and by the surrounding assurance envelope. In practice, increases in delegation require stronger evidence that decision logic, adaptability, and operational knowledge are adequate for the authorized scope. Verification, rollback, and escalation mechanisms help bound the associated risk, but they cannot substitute for inadequate control capability or operational knowledge.

Interface: The **Interface** axis describes how objectives, constraints, commands, state, explanations, and feedback are exchanged between the NCI and external actors, including operators, controllers, management systems, and peer automation components. It captures the abstraction level of interaction, from device-level commands to intent-based and conversational interfaces. For profiling, we distinguish four levels:

- **I0 (Device-Level Interfaces):** Low-level command-line interfaces, device-specific configuration scripts, or narrow management interfaces.
- **I1 (Consolidated Tools and APIs):** Centralized management tools, controllers, dashboards, and structured APIs that aggregate state and expose programmable operations.
- **I2 (Intent-Based Interfaces):** Policy- or intent-level specifications that describe desired outcomes, constraints, or service objectives instead of concrete commands.
- **I3 (Conversational and Explainable Interfaces):** Natural-language, multimodal, or dialogue-based interfaces that support clarification, explanation, and human-facing supervision of autonomous behavior.

As interfaces move upward, they reduce the burden of translating high-level objectives into low-level commands, but they also increase the need for precise intent semantics, evidence-bearing explanations, and machine-checkable constraints. Higher interface abstraction is not always more operationally mature: a machine-checkable intent interface may be safer than a conversational interface when ambiguity, authorization, or verification is difficult to resolve.

The framework is most useful when applied comparatively. For example, an SNMP-based alarm system is typically placed near *D0/A0/K0–K1/C0–C1/I1*: it reacts through fixed thresholds, relies on narrow observational state, and usually requires human interpretation and approval. A traffic-engineering controller such as Google's B4 is closer to *D1/A1/K2/C2/I1*: it optimizes over explicit objectives and domain models and can execute within a scoped authority envelope. In comparison, an LLM-based change-authoring assistant may reach *D2/K3/I3* in proposal generation and evidence synthesis while remaining at only *C0–C1*, because commit authority remains outside the model. These profiles clarify why network-control systems that all appear "intelligent" can differ sharply in operational autonomy. Stronger observation, reasoning, or interface support should not be equated with broader control authority.

B. Standards Landscape

The NCI framework is not intended to replace existing autonomous-network standards, maturity models, or management architectures. It provides a finer-grained analytical vocabulary for comparing the control-intelligence properties that these initiatives only partially emphasize. Standardization bodies and industry fora have developed several related frameworks for autonomous, self-driving, zero-touch, or intent-based networking, but their purposes differ. Some define maturity levels for operational transformation, some specify closed-loop management architectures, some standardize domain-specific analytics functions, and others provide protocols and data models for network automation. Table I summarizes how these initiatives relate to the five NCI axes.

The first group focuses on how far operational authority can be delegated to automated systems. TM Forum's Autonomous Networks program defines a staged Level 0–Level 5 model for assessing the degree of autonomous operation [17], [45], [46]. From the NCI perspective, such maturity levels provide a coarse-grained view of the Control Delegation axis. They indicate the extent to which operational authority is transferred from humans to automated systems. Regional progress material further shows why a single scalar level is often insufficient. Different operators, domains, and operational processes may advance at varying rates, with service assurance, fulfillment, transport, RAN, core, and service delivery occupying distinct maturity levels [11]. The remaining NCI axes help characterize how a given maturity level is realized in practice, for example, through particular forms of decision logic, knowledge

TABLE I
HOW MAJOR STANDARDS AND INDUSTRY FRAMEWORKS RELATE TO THE FIVE-AXIS NCI FRAMEWORK. CELLS INDICATE WHETHER AN INITIATIVE EMPHASIZES AN AXIS (STRONG), COVERS IT IN A LIMITED OR DOMAIN-SCOPED MANNER (PARTIAL), ASSUMES IT INDIRECTLY (IMPLICIT), OR LARGELY TREATS IT AS OUT OF SCOPE (–). THE MAPPING IS INTERPRETIVE: IT CHARACTERIZES THE MAIN EMPHASIS OF EACH INITIATIVE WITHOUT ASSIGNING FORMAL COMPLIANCE LEVELS.

Initiative	Primary emphasis	Decision Logic	Adaptability	Knowledge	Control Delegation	Interface
TM Forum [17], [45], [46]	Maturity levels and operational transformation	Implicit	Implicit	Partial	Strong	Partial
ETSI ZSM [10]	Closed-loop management across domains	Partial	Partial	Partial	Partial	Strong
ETSI ENI [47]	Cognitive management and context-aware policy	Partial	Partial	Strong	Partial	Partial
3GPP SON [48]	RAN automation loops	Partial	Partial	Partial	Partial	Partial
O-RAN [49]	RAN automation platform with programmable control apps and management interfaces	Partial	Partial	Partial	Partial	Strong
3GPP NWDAF [50]	Standardized analytics services for 5G core	Implicit	Implicit	Strong	–	Strong
ITU-T FG-AN [51]	Pre-standardization guidance for autonomous-network evolution	Implicit	Implicit	Partial	Partial	Implicit
ITU-T Y.317x [52]	Reference architecture for ML in future networks	Partial	Partial	Strong	Implicit	Partial
NETCONF/RESTCONF/YANG, OpenConfig/gNMI [53]–[57]	Management protocols, data models, and telemetry interfaces	–	–	Partial	–	Strong
IETF intent RFCs [58], [59]	IBN concepts and intent classification methodology	Partial	Implicit	Partial	Implicit	Strong
IETF ANIMA ACP, BRSKI [60], [61]	Secure the autonomic control plane and bootstrap the infrastructure	Implicit	Implicit	Partial	Partial	Strong
OpenFlow/ONOS [4], [62]	SDN programmability and controller platforms	Partial	Partial	Partial	Partial	Strong
ONF CORD/SEBA [63], [64]	Reference designs for virtualized access/edge	Partial	Implicit	Partial	Partial	Strong

representation, adaptability, or interface abstraction. ITU-T activities on autonomous networks and intelligence levels play a related role by framing autonomy and machine-learning integration at a cross-technology level [51], [52].

The second group concentrates on closed-loop management and domain-specific automation. ETSI ZSM and ENI define architectural patterns for zero-touch and cognitive management, including management domains, data fabrics, assurance functions, and policy-driven closed loops [10], [47]. 3GPP SON, NWDAF, and O-RAN provide more domain-specific mechanisms for RAN and 5G core automation, including self-configuration, self-optimization, standardized analytics exposure, and RIC-based control applications [48]–[50]. In NCI terms, these initiatives cover selected parts of Decision Logic, Adaptability, Knowledge, Control Delegation, and Interface. They specify important architectural and domain-specific mechanisms, but they do not provide a general framework for comparing systems across all five axes.

The third group foregrounds *interfaces, protocols, and programmability*. The IETF/IRTF works on NETCONF [53], RESTCONF [54], YANG [55], intent-based networking [58], [59], and autonomic networking [60], [61] clarify how network state, configuration, and intent can be represented and exchanged. Operational practice around OpenConfig/gNMI similarly strengthens structured configuration and telemetry exchange [56], [57]. ONF efforts, including OpenFlow [4], ONOS [62], CORD [63], and SEBA [64], provide concrete examples of controller-centered programmability and open operational interfaces. These efforts primarily strengthen the Interface axis and, in controller-centered settings, parts of Decision Logic and Knowledge.

Taken together, these initiatives point toward network operations that are more programmable, more closed-loop, more intent-aware, and more autonomous within scoped authority. Their emphasis, however, remains uneven across the five NCI axes. A system may be described as highly autonomous in a maturity model while still relying on narrow knowledge, static adaptation, or low-level interfaces. Conversely, a system may expose rich intent interfaces and analytics while retaining limited commit authority. The NCI framework is complementary to these initiatives. It does not propose another maturity ladder. Instead, it provides a technology-agnostic characterization space for comparing how standardized components, industrial systems, and research prototypes differ in their concrete control-intelligence properties.

Several prior surveys review important parts of this landscape, including software-defined networking and network softwarization [2], network function virtualization [65], machine learning for networking and cognitive network management [6], [7], network digital twins [66], [67], and recent applications of LLMs to networking [26]. These works provide broad coverage of enabling technologies and application domains. This paper takes a different perspective. It uses NCI to compare how successive generations of network-control systems strengthened decision-making, adaptation, operational knowledge, authority allocation, and operator interaction, and where their outputs still remained advisory, scoped, or constrained before changing network state. This lens structures the historical analysis that follows, tracing the uneven accumulation of control-intelligence properties across the three eras.

III. RULE-BASED, SCRIPTED AUTOMATION

The 1960s–early 2000s formed the operational foundation for modern network control. In this era, network operations shifted from largely manual, human-centered workflows to Network Operations Centers (NOCs), centralized monitoring, computer-assisted data collection, standardized management protocols, and deterministic, rule-based automation. Computerized systems enabled more observable network state and more consistent execution of predefined operational procedures. However, complex tasks and high-stakes decisions, such as incident interpretation, remedy selection, and approval of high-impact changes, were still managed by operators.

Before the 1970s, large-scale telecommunications operations relied heavily on labor-intensive, paper-based workflows,

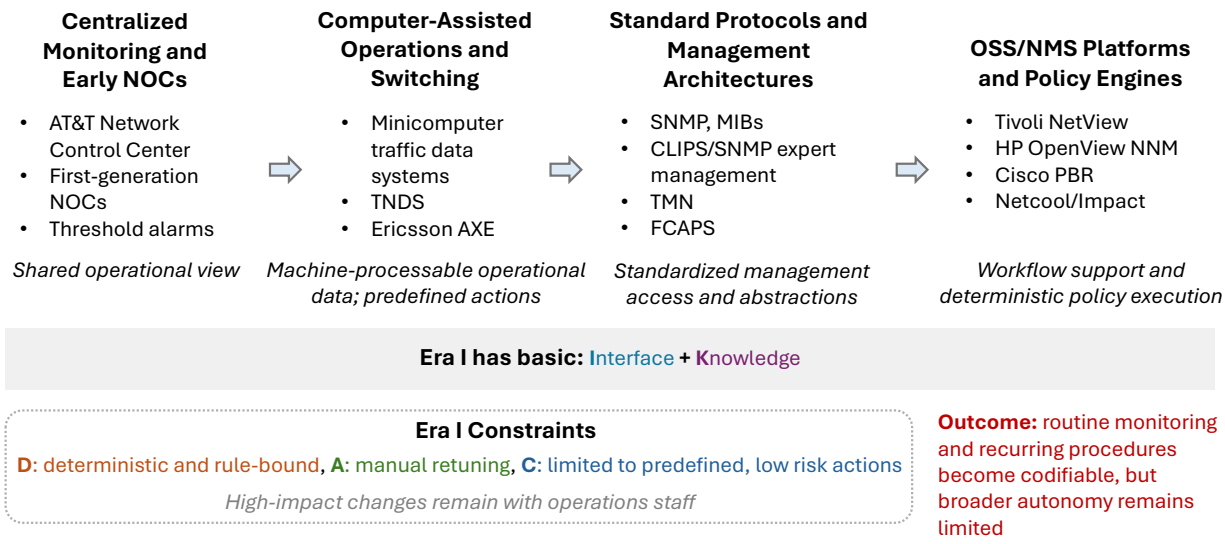


Fig. 3. Era-I progression from centralized monitoring to rule-based operational execution. The dominant advances were along the Interface and Knowledge axes: network state became more visible, machine-processable, standardized, and actionable through operations platforms. Control Delegation expanded only for predefined actions, while Decision Logic remained deterministic, and Adaptability depended on manual retuning.

including manual record-keeping, shift reports, physical inspections, and reactive maintenance [25], [68]. Although electromechanical control and alarm mechanisms already existed, they were typically attached to specific equipment, circuits, or fault conditions without a network-wide control process. The network's operative "intelligence" still relied largely on human expertise with little autonomous interpretation or decision-making. The transition toward rule-based network management unfolded gradually as centralized monitoring, traffic-data systems, software-controlled switching, standard management protocols, and operations platforms entered operational practice. This section examines this first era as the operational foundation of network automation, including centralized monitoring, traffic data systems, software-controlled switching, management protocols, and early OSS/NMS platforms, which made network operations more observable and repeatable. The cumulative progression of this era is summarized in Fig. 3.

A. From Local Oversight to Centralized Monitoring

AT&T's opening of a Network Control Center in New York in 1962 was an early milestone in centralized network monitoring. The center addressed the growing complexity of long-distance telephone operations by moving beyond local oversight of geographically dispersed network elements. It provided centralized, near-real-time visibility into switching, routing, and traffic-status information for major toll network elements [69]. This centralized view anticipated the operating pattern of later Network Operations Centers (NOCs), where operations staff could monitor distributed infrastructure from a shared operational picture.

First-generation NOCs used large electronic status boards and displays to present network status, allowing operations staff to track conditions through visual indicators and decide when to intervene. As software-based monitoring matured, threshold-based alarms and predefined event-handling rules reduced the manual monitoring burden. Specific events or metric thresholds could trigger predefined notifications or low-risk actions [70]. Compared to paper-based and equipment-local practices, centralized monitoring advanced the *Interface* axis from local inspection toward a shared operational view, and the *Knowledge* axis from dispersed equipment signals toward consolidated network-state evidence. It also contributed to *Control Delegation* by enabling the automatic triggering of alarms and low-risk notifications, though non-routine diagnosis and intervention remained with operations staff.

B. From Manual Procedures to Computer-Assisted Control

In the early 1970s, minicomputers began moving network operations from manual procedures toward computer-assisted control. By the mid-1970s, minicomputer-based systems could collect and process operational data in support of traffic analysis, engineering, and management. Bell System's Total Network Data System (TNDS), whose planning began in the mid-1960s and culminated in a family of systems documented in 1983 [71], mechanized portions of traffic-data collection and processing. Digitally controlled switches also began to replace traditional electromechanical systems in the telephone infrastructure. Ericsson's AXE system, a telephone exchange deployed in the late 1970s, used modular software control to manage calls and network resources more flexibly [72].

Compared with centralized monitoring, these systems made operational knowledge more machine-processable. TNDS mechanized traffic-data collection and processing, while AXE moved selected switching functions into software-controlled execution. From the NCI perspective, they advanced the *Interface* axis from visual operational displays toward computer-assisted analysis, and the *Knowledge* axis from centrally displayed state toward operational data that could be stored, processed, and reused. *Control Delegation* also increased slightly, but only for predefined operational or switching actions.

TABLE II
NCI PROFILE OF REPRESENTATIVE ERA-I DEVELOPMENTS. SCORES ARE APPROXIMATE QUALITATIVE POSITIONS WITHIN THE FIVE-AXIS FRAMEWORK: D = DECISION LOGIC, A = ADAPTABILITY, K = KNOWLEDGE, C = CONTROL DELEGATION, AND I = INTERFACE.

Development	Representative mechanisms	D	A	K	C	I	NCI synopsis
Centralized monitoring	AT&T Network Control Center; first-generation NOCs; threshold alarms	D0–D1	A0	K1	C0–C1	I1	Local inspection moved toward shared operational visibility; knowledge remained descriptive and alarm-centered.
Computer-assisted control	Minicomputer-based traffic-data systems; TNDS; Ericsson AXE	D0–D1	A0	K1–K2	C1	I1	Operational data became more machine-processable, and network control actions became executable under predefined logic.
Standard protocols and management architectures	SNMP; MIBs; CLIPS/SNMP expert management; TMN; FCAPS	D1	A0	K2	C0–C1	I1–I2	Managed objects, alerts, rule state, and management functions became more explicit and portable across systems.
Operations platforms and policy engines	IBM Tivoli NetView; HP OpenView NNM; Cisco PBR; Netcool/Impact	D1	A0	K2	C1	I2	Standardized management information became actionable through workflow support and deterministic policy execution.

C. Standard Protocols and Management Architectures

Another major development in this era was the standardization of network management protocols, with the Simple Network Management Protocol (SNMP) becoming the de facto standard for querying network devices and, where permitted, setting writable management variables in the late 1980s and early 1990s [73], [74]. Emerging from the Internet community's effort to unify management across a rapidly expanding infrastructure, SNMP evolved from the Simple Gateway Monitoring Protocol into a general manager/agent model for monitoring and limited control. The early SNMP specifications (RFC 1065-1067 and later RFC 1155-1157) defined the manager/agent model and basic protocol operations, including reading and setting device parameters and receiving asynchronous TRAP alerts [73], [74]. Together with Management Information Bases, SNMP enabled reactive, vendor-agnostic monitoring and broadened the reach of centralized, software-driven network management [75].

Early operational expert systems in IP networks coupled SNMP telemetry with CLIPS (a forward-chaining rule engine) to codify operational knowledge as production rules. NASA's implementation polled SNMP-managed elements for status and performance metrics, stored the returned data for rule evaluation, and applied problem-detection and problem-correction rules to adjust selected parameters or produce diagnostic output [76]. Architecturally, the CLIPS working memory and data files served as an inspectable state shared between SNMP access routines and the rule base, enabling repeatable yet rule-bound operations. The need for IP network management also catalyzed lightweight monitoring tools: Multi Router Traffic Grapher used SNMP data to chart traffic statistics [77], while Big Brother checked distributed systems and emitted operational alerts for monitored services [78].

The formalization of network management frameworks constituted another critical foundation for control. ITU-T Recommendation M.3010 defined the Telecommunications Management Network (TMN) reference architecture and its layered management model, spanning element, network, service, and business management abstractions [79]. ITU-T Recommendation M.3400 then specified TMN management functions across performance, fault, configuration, accounting, and security management, aligning closely with the fault, configuration, accounting, performance, and security (FCAPS) categories used in telecom-management practice [80]. With respect to earlier computer-assisted systems, SNMP, MIBs, CLIPS-based rules, and TMN/FCAPS frameworks made the *Interface* and *Knowledge* axes less device-local and less vendor-specific. Managed objects, telemetry access, alerts, rule state, and management functions could be represented through more explicit protocols and architectural structures. This also supplied the standardized management layer on which later OSS/NMS platforms and policy engines could operate.

D. Operations Platforms and Policy Engines

Early rule-based automation matured along two converging tracks. OSS and NMS platforms consolidated alarms, topology, and workflows across FCAPS functions, while policy engines encoded deterministic, pre-approved behaviors in management, control, or forwarding processes. Together, these tracks moved network operations from isolated monitoring and manual procedures toward rule-governed operational workflows. OSS/NMS platforms such as IBM Tivoli NetView [81] and Hewlett-Packard OpenView Network Node Manager (NNM) [82], [83] exemplified this shift. They supported centralized alarm handling, topology-aware monitoring, event forwarding, and operator workflow support, often building on standard management protocols and architectures such as SNMP and TMN. They consolidated operational context into platforms through which operations staff could inspect events, correlate symptoms, and coordinate responses.

Meanwhile, deterministic policy mechanisms began to move selected decisions closer to managed devices and event streams. Policy engines in this era appeared both inside devices and at the OSS/NMS layer. With Cisco IOS Release 11.0 in the mid-1990s, Policy-Based Routing (PBR) was operationalized as an on-device policy interpreter that used route maps to match packet attributes, such as IP access-list membership or packet length, and then set a next hop or output interface. This allowed routers to send selected packets along paths other than the default shortest path [84]. Netcool/Impact 3.1, documented in the mid-2000s, connected event sources to a rule-driven policy engine for event handling and contextual

enrichment. Upon receiving an alarm, "Impact" could apply filters, run policies, enrich events with contextual data, correlate or suppress alerts, and invoke configured automations through management integrations [85].

These systems expanded the scale and consistency of rule-based operations. They made alarms, topology, and event context more accessible through management platforms, and they allowed predefined rules to trigger selected forwarding or event-handling actions. Compared with earlier centralized monitoring, they advanced the *Interface* axis from display-oriented visibility toward workflow support, and the *Knowledge* axis from isolated device status toward consolidated alarms, topology, and event context.

NCI Synthesis of this era

This era established the operational foundation of modern network control. Centralized monitoring and NOCs made distributed network state visible from a shared operational view; traffic-data systems and software-controlled switching made selected operational data and actions machine-processable; SNMP, MIBs, TMN, and FCAPS categories standardized management interfaces and operational abstractions; and early OSS and NMS platforms connected alarms, topology, workflows, and deterministic policies. These developments made routine monitoring feasible and allowed routine procedures to be codified as scripts, runbooks, and event-handling rules.

From the five-axis perspective, the systems in this era occupy a low-autonomy, low-adaptation region of the NCI space, as summarized in Table II. **Interface** and **Knowledge** advanced most visibly. Interfaces evolved from local inspection and status displays toward centralized dashboards, management protocols, OSS/NMS consoles, and workflow support. Knowledge evolved from dispersed equipment signals and paper records toward machine-readable counters, alarms, topology information, managed objects, and management-function categories. However, this knowledge remained primarily descriptive rather than predictive or causal, and was usually organized around devices, links, alarms, and procedures instead of service-level intent or cross-domain models.

The remaining axes advanced more modestly. **Decision Logic** was predominantly reactive and deterministic, relying on threshold triggers, route maps, event filters, production rules, and scripted procedures that produced local or one-step responses rather than explicit planning or multi-objective reasoning. **Adaptability** was limited because changes in traffic patterns, failures, or operational policies usually required manual retuning of thresholds, scripts, rules, and workflows. **Control Delegation** remained tightly scoped: automated mechanisms could trigger routine notifications, execute predefined event-handling actions, or apply selected device-level policies, but non-routine incidents and high-impact changes still required operations-staff judgment, review, and approval.

The limiting factor of this era was not the absence of automation. Rather, the automation mechanisms lacked expressive decision logic, safe adaptation under non-stationary conditions, rich operational state models, and assurance mechanisms to circumscribe the risk of autonomous control actions. These limitations motivated the next era of network control, as network scale, heterogeneity, and service dynamism exceeded what centralized monitoring, deterministic rules, static workflows, and manual retuning could sustain.

IV. Programmable and Data-Driven Control

From the 2000s to the mid 2020s, network control evolved from device-local scripts and rule engines toward programmable and data-driven control loops supported by increasingly fine-grained and timely telemetry. Compared with the rule-based and scripted era, the control and service-delivery layer became more programmable. Forwarding behavior could be controlled through logically centralized controllers. Network functions could be instantiated and orchestrated as software in parallel, with operational observation expanding from periodically polled device counters to flow-level, packet-level, in-band, and infrastructure-level measurements, while learning-based analytics increasingly supported prediction, diagnosis, and optimization. This section traces these evolutions through network softwarization and virtualization, measurement and telemetry, data-driven analytics, and cognitive and autonomic architectures that organize monitoring, reasoning, and actuation into closed-loop control, as depicted in Fig. 4.

A. Programmable Actuation through Network Softwarization

Since the mid 2000s, network softwarization and virtualization have reshaped how control logic and network functions are implemented. In particular, Software-Defined Networking (SDN) separated control logic from packet forwarding and exposed programmable controller interfaces, while Network Functions Virtualization (NFV) decoupled network functions from dedicated appliances and introduced software-based lifecycle management over shared infrastructure.

Software-Defined Networking: SDN introduced the decoupling of the control plane (decision logic) from the data plane (packet-forwarding fabric), thereby enabling logically centralized control [2]. The intellectual roots of SDN date back to the "active networks" initiative in the late 1990s [86], which proposed in-network programmability through code injection. That initiative was later discontinued because of practical deployment constraints [87]. The modern SDN movement began with the Ethane project at Stanford in 2007, which demonstrated the benefits of separating control logic from switch hardware in enterprise networks [88]. This project led to the OpenFlow protocol [4], which defined the standard interface for controllers to program forwarding devices. These developments reduced dependence on vendor-specific control planes and catalyzed a broader movement toward open programmability. Once that architectural split was established, the surrounding ecosystem supplied controllers, emulators, virtualization layers, and programming languages. Early SDN systems permeated the operational control stack: NOX supported global coordination [89], FlowVisor enabled isolation-preserving slicing [90], Mininet made large-scale emulation accessible [91], and Frenetic introduced higher-level programming abstractions [92]. VMware's acquisition of Nicira in 2012 [93] signaled the commercial potential of SDN, and

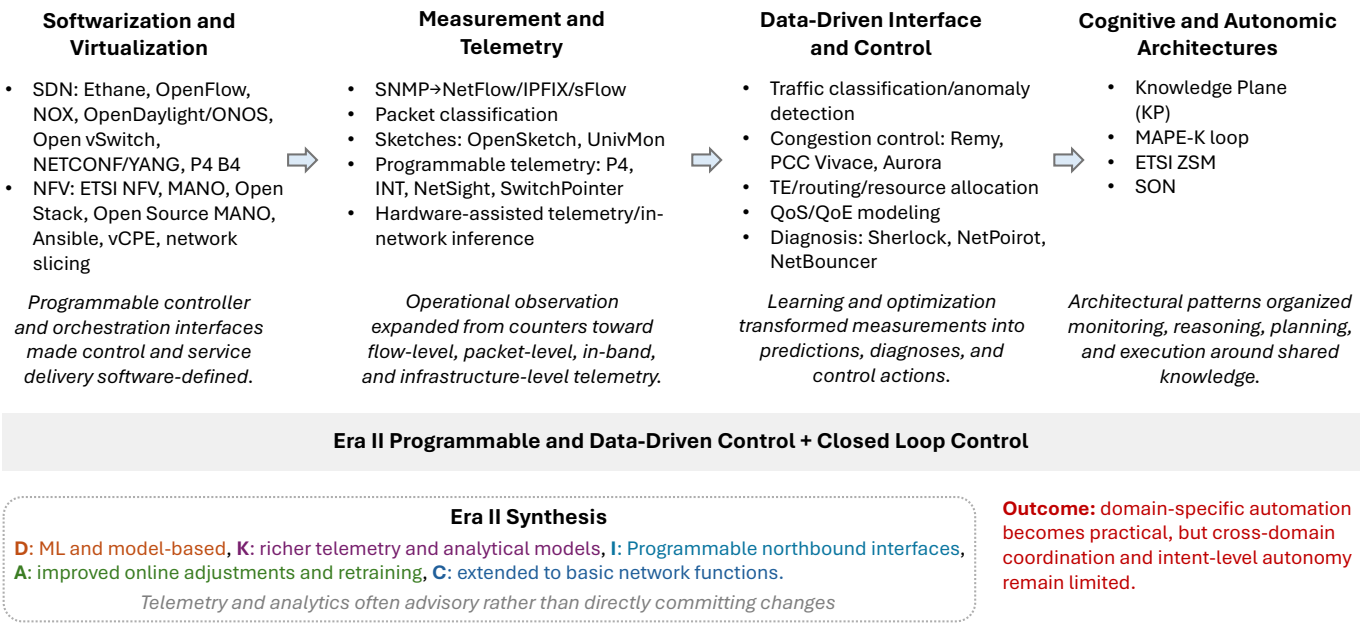


Fig. 4. Era-II progression from programmable actuation to data-driven closed-loop control. Network softwarization and virtualization exposed controller and orchestration interfaces; measurement and telemetry expanded operational knowledge from counters toward flow-level, packet-level, in-band, and infrastructure-level observations; data-driven inference transformed measurements into predictions, diagnoses, and optimization inputs; and cognitive/autonomic architectures organized these capabilities into closed-loop control patterns. Through the NCI lens, the era produced strong advances in *Decision Logic*, *Knowledge*, and *Interface*, with selected gains in *Adaptability* and bounded expansion of *Control Delegation*. The residual need to coordinate domain-specific controllers, telemetry pipelines, and learned models sets the stage for the LLM-enabled era.

Google's B4 deployment [94] showed that centralized traffic engineering could operate at scale in a private WAN.

These advances greatly galvanized institutional and operational consolidation. In 2011, the Open Networking Foundation (ONF) was launched by major network operators and web-scale companies, with vendors joining as additional members to promote SDN standardization and interoperability [95]. Projects such as OpenDaylight [96], ONOS [62], and Open vSwitch [97] broadened the SDN software ecosystem across controllers and programmable switching. In contrast, the open disaggregation movement and systems such as SONiC [98] made vendor-neutral programmability a practical design objective. During the 2010s, interfaces such as NETCONF, standardized earlier and widely adopted alongside YANG-based management, complemented OpenFlow with persistent configuration management [53], [56], while P4 extended programmability directly into the data plane [5].

Network Function Virtualization: NFV transformed service provisioning through virtualization and cloud technologies. Its origins are commonly traced to a seminal ETSI position paper published in 2012, in which a consortium of global CSPs formally proposed replacing traditional middleboxes with Virtual Network Functions (VNFs) running on commercial off-the-shelf (COTS) servers [3]. ETSI then established the NFV Industry Specification Group (ISG) to expedite standardization, while early deployments such as virtual Customer Premises Equipment (vCPE) demonstrated operational feasibility [99]. Telcos adopted cloud platforms such as OpenStack as Virtual Infrastructure Managers (VIMs) [100], and the NFV 001–004 specifications introduced the Management and Orchestration (MANO) framework for lifecycle control, orchestration, and OSS/BSS integration [101]. Open-source implementations such as Open Source MANO [102] and playbook-style automation such as Ansible [103] provided practical mechanisms for repeatable VNF and device lifecycle actions.

NFV also exposed a practical constraint for autonomous control: virtualized services can be orchestrated effectively only when their data-plane performance remains predictable. This motivated optimized packet-I/O and software data-plane techniques, including kernel-bypass frameworks, poll-mode drivers, batching, hugepages, high-performance virtual switching, and, later, in-kernel fast paths such as eBPF/XDP and AF_XDP [65], [97], [104]–[107]. These techniques are not themselves NCI mechanisms, but they made virtualized network functions sufficiently performant to participate in programmable orchestration loops. By the second half of the 2010s, operator programs such as AT&T's Domain 2.0 [108] and Telefónica's UNICA platform [109] had framed large-scale virtualization and orchestration as central telecom transformation objectives. Notably, the programmability introduced by SDN/NFV also enabled network slicing, in which compute, transport, and network function resources are assembled and managed as logically isolated service instances. In 5G systems, this made the dynamic allocation and lifecycle management of virtualized resources a central objective of orchestration [110].

With SDN and NFV, control actions no longer need to be embedded only in device-local procedures; they can be explicitly programmed through software interfaces. Compared with the rule-governed management era, this advanced the *Interface* axis from management consoles and protocols toward controller and orchestration APIs, and the *Decision Logic* axis from local rules toward centralized policies, optimization

programs, and lifecycle-management workflows.

B. Richer Knowledge through Measurement and Telemetry

While SDN and NFV established programmable actuation, a parallel evolution expanded what network-control systems could observe. Measurement and telemetry progressed from periodic counter polling toward packet-level, flow-level, in-band, and infrastructure-level instrumentation.

From counters to packet- and flow-level telemetry: Traditional mechanisms such as SNMP exposed management variables and counters through a polling-oriented management model [74]. Operators could infer end-to-end behavior from such counters, but sparse samples were unfit for transient microbursts or millisecond-scale congestion in multi-tenant cloud environments. In response, the early 2000s saw the rise of flow-export protocols that bridged the gap between per-device counters and per-packet traces. Cisco's NetFlow and IETF's IPFIX [111] introduced structured flow records summarizing traffic by 5-tuples, while sFlow [112] provided statistically sampled packet and counter data. These mechanisms enabled scalable traffic monitoring across carrier backbones and data centers. The shift from periodic polling to asynchronous flow export marked the first step toward continuous telemetry, reducing data collection latency from minutes to seconds and laying the foundation for more advanced analysis [113].

Packet classification and header processing: As flow-level telemetry matured, operators sought finer visibility into sub-flow behaviors, such as application classes, policy domains, or security signatures. This need made *packet classification* a core dataplane primitive, in which packets had to be matched against large, multidimensional rule sets at line rate. Early work moved beyond linear search and hashing through tuple-space indexing [114], hierarchical geometric partitioning [115], [116], and bit-parallel representations [117].

As packet processing moved into high-performance software dataplanes, rule update efficiency became the primary concern. Classifiers had to sustain line-rate lookups while absorbing frequent rule changes driven by policy and service churn. TupleMerge [118] revisited tuple-space methods for this online setting, and NPC [119] made classifier construction network-aware by incorporating measured traffic characteristics. Together, these methods show how classification co-evolved with measurement and control in programmable dataplanes, enabling selective mirroring, per-class telemetry, and event filtering at high packet processing rates.

In-network computing: As programmable data planes matured, a set of measurement and classification functions can be recast as lightweight in-network inference under strict pipeline constraints. Iisy [120], Planter [121], and Homunculus [122] demonstrated this trajectory by mapping classifiers or machine-learning pipelines onto programmable devices subject to constraints on features, state, latency, and throughput.

Sketch-based streaming measurement: To capture statistical traffic characteristics without maintaining per-flow state, researchers introduced probabilistic sketches, compact data structures that approximate counts, frequencies, or cardinalities within fixed memory; HyperLogLog is a representative cardinality estimator [123]. OpenSketch [124] and UnivMon [125] integrated sketching with programmable data paths, while subsequent systems addressed constraints on configuration, implementation, and reconfiguration [126]–[128]. These efforts bridged the gap between software and hardware telemetry, showing that streaming summarization could coexist with packet-level instrumentation.

Programmable data-plane telemetry: In parallel with advances in sketch-based summarization, programmable switching enabled direct, in-band measurement within forwarding pipelines. P4 enabled programmable packet parsing and header processing across protocol-independent packet processors [5], and In-band Network Telemetry (INT) demonstrated how packets could carry hop-by-hop metadata such as timestamps, queue occupancy, and egress ports [129]. Complementary postcard or report-based schemes, such as NetSight [130] and SwitchPointer [131], collect per-hop visibility without requiring the same packet-header modification model as INT. Hybrid designs then combined in-band telemetry with sketches or sampling techniques [132], [133]. Together, these systems made fine-grained packet-level introspection practical for operational monitoring and feedback control.

Hardware-assisted telemetry: Beyond packet-centric and in-band mechanisms, a complementary line of inquiry examined hardware-assisted telemetry, leveraging low-level device and microarchitectural signals to indicate network dynamics. Dobrescu et al. [134] helped establish this direction by instrumenting software packet-processing platforms to expose system-level signals as predictors of performance. Their work demonstrated that observing fine-grained system events can support inference of throughput and latency variability without explicit packet sampling, thereby bridging the gap between host-level instrumentation and end-to-end telemetry. Building on this principle, later systems exploited host-level counters, queueing signals, and infrastructure-level performance indicators to infer contention and service degradation in virtualized network functions [134]–[136]. Together, these works revealed a complementary trend in telemetry: they treated the underlying infrastructure itself as a source of operational observability, beyond exporting packet statistics alone. This systems-resident perspective links packet-level measurements to performance inference, providing low-overhead insights valuable for network automation.

In summary, flow export, packet classification, sketches, programmable telemetry, hardware-assisted measurement, and bounded in-network inference expanded the operational evidence available to network-control systems. Compared with the counters and alarms of the rule-based and scripted era, these mechanisms provided finer-grained, timelier, and more policy-relevant representations of network state. Their principal NCI contribution was along the *Knowledge* axis, supplying the evidence needed for anomaly detection, traffic engineering, capacity planning, diagnosis, and feedback control. They improved what a control system could observe and infer.

C. Data-Driven Inference and Learning-Based Control

As richer telemetry became available since the 2010s, machine learning (ML) techniques, especially Deep Learning

(DL) and Reinforcement Learning (RL), have been increasingly adopted to support inference, prediction, diagnosis, and optimization in network operations [6], [7]. In application domains such as traffic classification, congestion control, traffic engineering, resource allocation, and fault diagnosis, ML models have transcended traditional fixed-threshold rules by estimating hidden states, forecasting future conditions, and, in some cases, recommending or selecting control actions.

Data-driven control models were first widely applied in traffic classification and cybersecurity analytics [137]–[139]. Traditional approaches based on static port numbers or signature-based deep packet inspection became less reliable as encryption and application multiplexing increased, creating demand for classifiers that could infer application or traffic categories from statistical flow features [140]. ML-based classifiers learned traffic classes from flow features [137], while later neural models learned packet representations with less manual feature engineering [141] and generated compact classification structures for specific rule sets [142]. Related supervised and unsupervised methods supported anomaly and intrusion detection [143]. Through the NCI lens, these systems advanced the *Knowledge* and *Decision Logic* axes by transforming raw traffic measurements into inferred application classes, traffic types, and security-relevant labels.

Beyond traffic classification and security analytics, data-driven control models also reshaped congestion control at the transport layer. Classical Transmission Control Protocol (TCP) variants such as Reno [144] and CUBIC [145] relied on handcrafted control rules. Remy [146] showed that congestion-control rules could be synthesized through offline optimization; PCC Vivace [147] combined online learning with convex optimization to adapt sending rates from observed loss and latency; and Aurora [148] used deep reinforcement learning to adjust congestion windows from network feedback. These systems marked a shift from fixed heuristics to controllers that adapt sender behavior based on measured performance. Through the NCI lens, they advanced *Decision Logic* by replacing fixed heuristics with learning-based optimization, and advanced *Adaptability* by allowing sender behavior to change in response to measured loss, latency, and throughput.

Learning-based control has also expanded beyond sender-side adaptation to encompass network-wide traffic engineering (TE). In this setting, ML models were used to predict path performance and to support routing, scheduling, and resource allocation decisions over explicit network topologies and traffic demands. Specifically, "Learning to Route" [149] reframed routing configuration as a learning problem, and subsequent work has combined graph neural networks (GNNs), reinforcement learning, and optimization. RouteNet [150], AuTO [151], Bernardez et al. [152], Teal [153], RedTE [154], and LOTE [155] illustrate the progression from topology-aware prediction toward learned TE controllers. Through the NCI lens, these systems advanced *Knowledge* by learning topology- and traffic-dependent performance models. They also advanced *Decision Logic* by incorporating prediction, policy search, and optimization into routing decisions.

In addition to TE, data-driven control also supported resource allocation under stochastic demand. In cluster and data-center settings, Decima [156] used graph-structured reinforcement learning for job-DAG scheduling. In wireless and 5G systems, GNN-based approaches modeled interference and resource dependencies: Shen et al. [157] proposed a GNN architecture for scalable radio-resource management, while Salaün et al. [158] applied a GNN to cell-free massive-MIMO resource allocation. Through the NCI lens, these systems advanced *Knowledge* by representing workload or interference dependencies as graphs, and advanced *Decision Logic* and selected forms of *Adaptability* by learning allocation policies that responded to changing workloads, topologies, and conditions.

Data-driven control also broadened the objectives represented in network decision loops. Quality-of-service (QoS) and quality-of-experience (QoE) management used machine learning to relate network-level measurements to user-perceived outcomes. Measurement studies showed that video quality could be inferred even from encrypted traffic [159], while Pensieve [160] used reinforcement learning to optimize adaptive-bitrate decisions under time-varying network conditions. Through the NCI lens, these systems advanced *Knowledge* by estimating user-perceived service quality from network observations and advanced *Decision Logic* by incorporating those estimates into control and action selection.

A parallel line of work used ML methods to diagnose abnormal behavior. In particular, Sherlock [161] modeled causal dependencies between faults and symptoms, NetPoirot [162] distinguished network- from host-level faults in hyperscale data centers, and NetBouncer [163] combined active probing with latent-factor learning to localize device and link failures in production data-center networks. Learned surrogate models were also used to approximate expensive simulation and performance-analysis pipelines. For instance, MimicNet [164], DeepQueueNet [165], and m3 [166] learned surrogates for packet-level simulation, queueing behavior, or flow-level performance estimation. Through the NCI lens, these systems advanced the *Knowledge* axis from measured and inferred network state toward explanatory and predictive domain models that could support fault localization, scenario evaluation, planning, and optimization.

Across these applications, data-driven models changed how operational evidence was interpreted and incorporated into network decisions. Compared with telemetry mechanisms that primarily exposed the measured state, these models inferred traffic classes, anomalies, likely root causes, performance, and candidate actions. The main NCI advances, therefore, occurred in *Knowledge* and *Decision Logic*: operational state became more predictive and explanatory, while decision-making increasingly incorporated learned models, policy search, and optimization. *Adaptability* also improved in selected domains through online rate adjustment, periodic retraining, re-optimization, and controlled policy updates. These gains nevertheless remained domain-bounded: many systems provided decision support, while direct controllers operated within predefined objectives, state representations, and execution scopes rather than exercising broad operational authority.

TABLE III
NCI PROFILE OF PROGRAMMABLE AND DATA-DRIVEN CONTROL DEVELOPMENTS. SCORES INDICATE APPROXIMATE QUALITATIVE POSITIONS WITHIN THE FIVE-AXIS FRAMEWORK: D = DECISION LOGIC, A = ADAPTABILITY, K = KNOWLEDGE, C = CONTROL DELEGATION, AND I = INTERFACE. RANGES REFLECT VARIATION AMONG THE REPRESENTATIVE MECHANISMS GROUPED IN EACH ROW.

Development	Representative mechanisms	D	A	K	C	I	Main NCI reading
Softwarization and virtualization	SDN/OpenFlow; controllers; NFV MANO; VNF lifecycle orchestration	D1	A0–A1	K1–K2	C1–C2	I1	Advanced *Interface* and *Decision Logic* by exposing programmable controller and orchestration functions, while enabling scoped execution within controller-defined domains.
Measurement and telemetry	NetFlow/IPFIX; sFlow; packet classification; sketches; INT; hardware-assisted telemetry	D0	A0	K0–K1	C0	I1	Advanced *Knowledge* by making traffic, forwarding, queue, and infrastructure state more fine-grained and timely; the mechanisms remained primarily observational.
Data-driven analytics and control	ML-based traffic classification; congestion control; TE; resource allocation; QoE; diagnosis; learned surrogates	D1–D2	A1–A2	K1–K2	C0–C2	I1	Advanced *Knowledge* and *Decision Logic* by converting measured state into inferred, predictive, and optimization-relevant representations; *Adaptability* is improved in selected control loops.
Cognitive and autonomic architectures	Knowledge Plane; MAPE-K; SON; ZSM; NWDAF analytics	D1	A1	K2	C0–C2	I1–I2	Made the composition of *Knowledge*, *Decision Logic*, and *Interface* explicit; authority ranged from analytics-only services to bounded closed-loop actuation.

D. Closed-Loop Cognitive and Autonomic Architectures

The preceding developments advanced specific capabilities, including programmable control, fine-grained observation, prediction, diagnosis, and optimization. Cognitive and autonomic architectures addressed a complementary question: how should these capabilities be organized around shared operational knowledge and connected to decision-making and actuation through observed feedback?

The Knowledge Plane (KP) offered an influential architectural vision [167]. It envisaged a logically integrated yet physically distributed layer to maintain network-wide models, combine operational context, and inform the control/data planes. It frames network control around a knowledge layer to support inference, planning, and coordinated decision-making.

Autonomic computing provided a control pattern through the Monitor-Analyze-Plan-Execute-over-shared-Knowledge (MAPE-K) loop [8]. MAPE-K structured self-management around monitoring, analysis, planning, and execution functions connected through a shared knowledge base. Its agenda encompassed self-configuration, self-optimization, self-healing, and self-protection under high-level policies rather than continuous low-level intervention. Early autonomic-networking prototypes explored how distributed autonomic elements could share operational knowledge, invoke rule engines and optimizers, and execute actions [168].

In essence, KP emphasized the network-wide organization of operational knowledge and reasoning, while MAPE-K specified a reusable feedback structure connecting observation, analysis, planning, and action. These ideas later influenced telecom-management architectures. Notably, ETSI Zero-touch Network and Service Management (ZSM) defined management services, closed-loop functions, and interfaces for coordinating automation across management domains [10]. These principles also appeared in more domain-specific settings. Self-Organizing Network (SON) functions implemented local self-configuration and self-optimization loops [169], whereas the Network Data Analytics Function (NWDAF) in 5G exposed analytics services to other network functions without directly committing network changes [50].

Through the NCI lens, this line of work made the composition of control intelligence explicit. KP emphasized the integration of operational knowledge and reasoning; MAPE-K organized monitoring, analysis, planning, and execution around shared knowledge; ZSM translated related principles into a telecom management architecture with defined functional and interface boundaries; and SON and NWDAF illustrated how closed-loop control and analytics services could be instantiated within bounded mobile network domains. Collectively, these developments advanced *Knowledge*, *Decision Logic*, and *Interface* by clarifying how observation, reasoning, and actuation could be connected. They also showed that *Control Delegation* depends on where execution authority is allocated: SON functions may operate within restricted control scopes, whereas analytics functions, such as NWDAF, are primarily tasked to inform other network functions.

NCI Synthesis of this era

In this era, the scope and capability of network control have been materially expanded through the integration of programmable actuation with richer operational state and data-driven decision support. Network softwarization and virtualization exposed controller and orchestration interfaces through which policies, optimization programs, and lifecycle-management workflows could affect network state. Measurement and telemetry supplied finer-grained and timelier representations of traffic, performance, faults, and resource usage. Data-driven models transformed these observations into inferred states, predictions, diagnoses, and optimization guidance. Cognitive and autonomic architectures then provided reusable patterns for composing observation, shared

knowledge, analysis, planning, and execution into a closed-loop control system. Table III summarizes this progression in the five-axis NCI space. The strongest advances occurred along *Decision Logic*, *Knowledge*, and *Interface*. *Decision Logic* moved beyond threshold rules and static scripts toward centralized policy programs, optimization-based controllers, learned models, and hybrid combinations of prediction and optimization. *Knowledge* expanded from counters, alarms, and basic topology records toward streaming telemetry, inferred operational state, predictive domain models, and shared knowledge structures. *Interface* evolved from device-oriented commands and management protocols toward programmable controller APIs, orchestration frameworks, telemetry services, and higher-level management abstractions.

Progress along *Adaptability* and *Control Delegation* was more uneven. Adaptability improved in systems that updated sending rates, routing decisions, resource-allocation policies, or learned models in response to changing conditions. Such adaptation was often confined to a particular control loop or implemented through periodic re-optimization, offline retraining, or controlled policy updates. Control Delegation also expanded into congestion control, traffic engineering, virtualized-service lifecycle management, SON functions, and closed-loop management frameworks. Notably, execution authority remained regulated by explicit objectives, policy scopes, controller-defined guardrails, and, for higher-impact actions, human supervision. Telemetry and analytics functions often remained advisory because they supplied operational evidence or recommendations without directly committing changes.

These advances led to substantial capability within individual control loops, but limited coordination across them. Programmable controllers, telemetry pipelines, and learned models were typically designed for particular domains, timescales, and operational tasks. Operators continued to translate high-level objectives into domain-specific inputs, reconcile evidence across heterogeneous systems, and coordinate multi-step workflows spanning multiple control and management domains. These residual interpretation and coordination burdens created a natural role for LLM-enabled network operations. LLMs can assist with intent interpretation, evidence synthesis, and workflow assembly, but they do not by themselves make adaptation safe or broader autonomy defensible. As discussed in Section V, authoritative actuation consequently remains anchored in policy-governed controllers, verification mechanisms, authorization, staged rollout, and rollback.

V. LLM-Enabled Network Operations

Since the early 2020s, the burdens of interpretation and coordination have become more pronounced with the proliferation of 5G network slicing, cloud-native network functions, and RAN disaggregation. Services and OAM responsibilities no longer align neatly with individual devices or single domains. Instead, they increasingly span the core, wide-area network (WAN), data center, and RAN [170]. As a result, operational incidents and network changes have increasingly become cross-domain concerns. A configuration error or runtime anomaly in one domain may propagate through service dependencies, while multi-vendor environments expose heterogeneous data models, interfaces, and control semantics. Cloud-native deployment practices also increase the frequency of software and configuration changes, while operators must continue to satisfy reliability, compliance, and service-level constraints [13]. The operational challenge thus extends beyond optimizing individual control loops within a single domain to interpreting intent, assembling cross-domain operational context, and, most importantly, reliably and consistently executing changes through auditable validation, authorization, rollout, and recovery mechanisms [171]. Together, these operational pressures motivate the emergence of a third era of NCI. Programmable, data-driven control remains necessary, while network operations increasingly require intent interpretation, evidence-grounded analysis, cross-domain workflow coordination, and governed change execution [10], [14], [15].

To tackle this operational backdrop, networking research and industry initiatives increasingly explored whether LLMs could reduce the burden of interpreting requests and coordinating OAM workflows, while continuing to rely on established procedures for validating, approving, and executing network changes [13], [26], [37]. Built on the transformer architecture [172], LLMs demonstrated exceptional capabilities in instruction following, code generation, text summarization, retrieval-based analysis, and multi-step tool use [173]–[175]. These capabilities made them relevant to tasks that require interpreting intent, synthesizing heterogeneous evidence, and assembling workflows across existing control systems.

In practice, LLMs are most applicable to tasks performed before an operational change is approved and executed. For example, an LLM-enabled system may interpret an operator request; retrieve and synthesize logs, tickets, runbooks, topology information, and configuration state; draft diagnostic hypotheses, proposed configuration changes, or rollout plans; and organize an OAM procedure. To support these tasks, the LLM is typically embedded in a workflow that supplies relevant context, exposes usable tools, preserves intermediate results, and records outputs for subsequent checking [26], [28], [29], [31], [32], [176]. When the workflow enables the LLM to pursue an objective by selecting tools, maintaining intermediate state, and planning subsequent steps, it is commonly described as an LLM-enabled agent.

To explain how these workflows interact with existing network control systems, this section distinguishes among three concepts. A *proposal artifact* is an inspectable output prepared for review or validation, such as a diagnostic hypothesis, verification query, proposed configuration change, objective specification, or rollout plan. *Commit authority* is the authority to approve a proposal for application to the operational network. *Governed execution* denotes the controlled path from an approved proposal to an operational change. It includes validation and authorization before deployment, execution through established control or management systems, plus monitoring, rollback, and record-keeping afterward. Under this distinction, LLM-enabled systems interpret operator requests, assemble relevant evidence from heterogeneous OAM records, translate intent into structured operational inputs, and produce proposals that existing validation and execution systems can process.

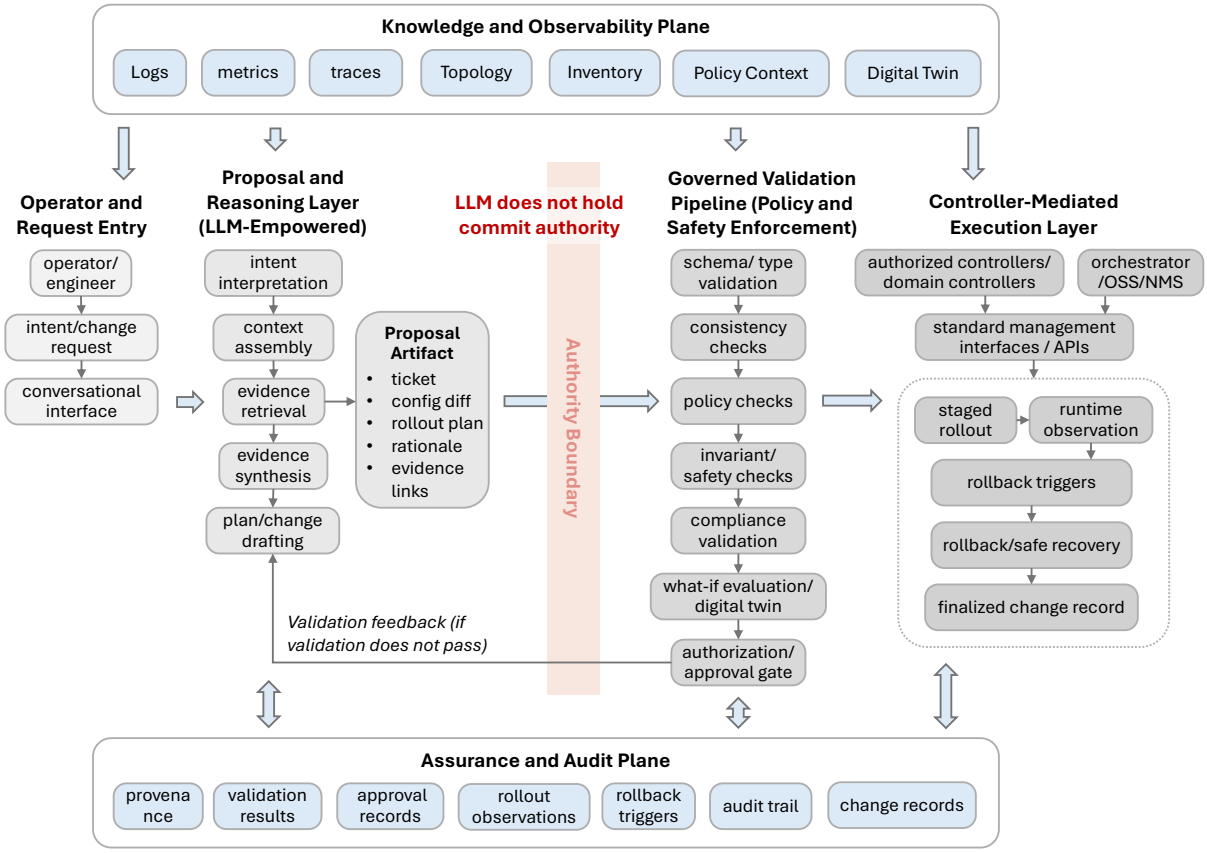


Fig. 5. Reference architecture for LLM-enabled governed network operations. The critical architectural boundary is between proposal generation and commit authority: the LLM assembles operational context and drafts a proposal artifact, while deterministic checks, authorization, staged rollout, and rollback remain in the governed execution path. Detailed pre-change checks can include schema/type validation, policy and invariant checks, reachability and compliance validation, and what-if evaluation in a simulator or digital twin.

Operators or established authorization processes determine whether a proposal may proceed, while existing control and management systems apply the approved change [18], [177].

The separation between proposal generation and authorized execution also resonates in standards and industry guidance. In particular, TM Forum's introductory guide treats governance, oversight, traceability, interoperable interfaces, policy representation, and audit records as architectural concerns for AI-enabled closed loops [16], [17]. In cellular networks, 3GPP and O-RAN define management and control interfaces through which operational tools interact with established network functions [49], [50]. Vendor and operator white papers similarly describe architectures in which higher-level functions interpret intent and coordinate service and resource management above domain controllers, supported by assurance functions, knowledge bases, digital twins, and human supervision [35]–[37], [178]–[180]. Likewise, Appledore argues that LLM-based components should be integrated into existing control loops for autonomous network operation rather than deployed as a separate automation system [181].

One related architectural consideration is the distinction between fast and slow control loops. Latency-critical control is generally better handled by deterministic controllers, compact learned policies, or optimization-based mechanisms whose timing and stability can be bounded. LLM-enabled components are more naturally placed in slower or supervisory workflows, where they can support intent interpretation, the synthesis of telemetry and logs, policy analysis, and the coordination of OAM procedures [46].

Together, these architectural and temporal considerations place the boundary between proposal generation and authorized execution at the center of LLM-enabled network operations. According to our study, available evidence does not yet support assigning LLM-enabled agents direct, end-to-end authority over operational networks. Current designs generally take a conservative approach: LLM-enabled components interpret requests, assemble evidence, and propose actions, while approval and execution remain subject to established operational controls. Broader autonomy requires evidence that proposed changes have been validated, execution remains within authorized constraints, outcomes can be monitored, and failures can be contained or reversed [182]–[190]. The remainder of this section first derives operator requirements from this authority boundary, then develops a reference architecture, identifies recurring patterns for integrating LLM-enabled components into network operations, and analyzes the emerging trend through the NCI framework.

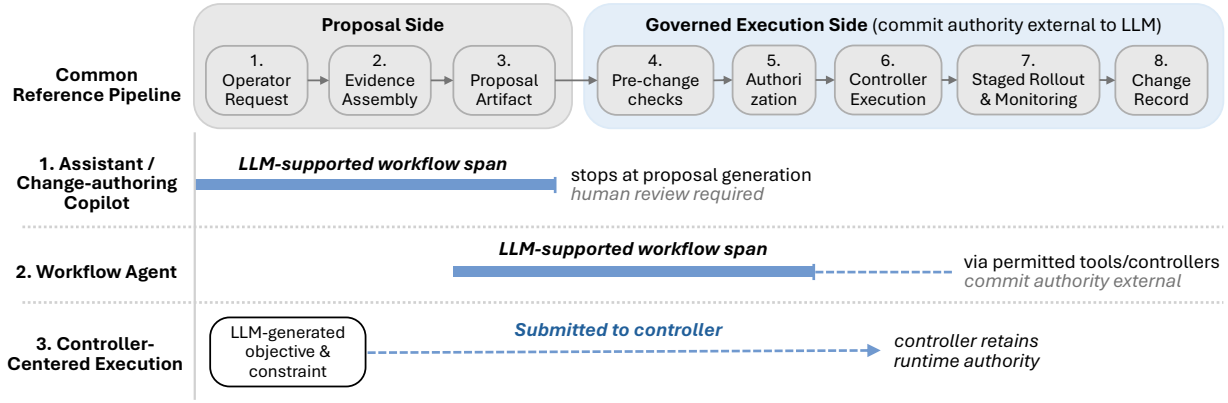


Fig. 6. Placement of the three integration patterns over the common operational pipeline. Their main architectural difference is not merely whether an LLM is present, but where LLM-supported processing terminates relative to the commit-authority boundary: assistants remain on the proposal side, OAM workflow agents reach permission-scoped procedure execution, and controller-centered designs retain runtime commitment within established controllers.

A. Operator Requirements

From the operator's perspective, LLM-enabled automation is operationally acceptable only when it is integrated into controlled OSS/OAM processes. It cannot be deployed as an unconstrained control agent. Four requirements follow. First, proposal generation must remain separate from commit authority. Second, proposed actions must be represented in forms that can be checked against schemas, policies, safety invariants, reachability constraints, and the data models exposed by control and management systems. Third, execution must support controlled deployment, post-change monitoring, and recovery when outcomes deviate from expectations. Fourth, the workflow must produce auditable records that link each request to its inputs, validation results, executed actions, and observed outcomes. These properties must remain effective across multiple vendors, integrate with existing Operations Support Systems and Network Management Systems (OSS/NMS), and fail safely when telemetry is incomplete, tools are unavailable, or operator intent remains ambiguous [10], [16].

B. Reference Architecture

Fig. 5 presents a minimal reference architecture organized around the boundary between proposal generation and authorized execution. It does not replace complete OSS/NMS, Zero-touch Network and Service Management (ZSM), or autonomous-network architectures. Instead, it isolates the functions needed to produce, validate, authorize, execute, and record an operational change.

On the proposal side of the boundary, an interaction and proposal layer interprets operator intent and combines it with logs, topology state, tickets, configuration information, and other operational evidence. It produces inspectable outputs such as diagnostic hypotheses, verification queries, proposed configuration changes, or rollout plans. Before execution, a governed validation pipeline checks whether the proposal satisfies applicable schemas, policies, invariants, authorization rules, and safety constraints. Approved proposals are then applied through established controllers, orchestrators, or management systems [16], [53], [191], [192]. A knowledge and observability plane provides the operational information needed for proposal generation and validation, including logs, metrics, traces, topology, inventory, policy context, and digital twin state. Simulation and digital-twin facilities can additionally support what-if evaluation of proposed changes [66], [193]. An assurance and audit plane preserves provenance, validation results, authorization decisions, deployment observations, rollback triggers, and finalized change records.

Operationally, a request and its supporting evidence are assembled into a proposal artifact. When the proposal is intended to produce an operational change, it enters the governed validation pipeline. Proposals that fail schema, policy, invariant, or authorization checks are returned to the proposal layer with validation feedback instead of being executed. Approved proposals are applied through existing controller interfaces and management protocols, deployed in controlled stages when appropriate, and monitored for conditions that require rollback or other recovery actions. The workflow concludes by producing a *change record* that links the original request, supporting evidence, validation results, authorization decision, executed actions, and observed outcomes [18]–[20], [182], [188], [189], [191], [192], [194].

C. Integration Patterns

This subsection classifies representative systems in the LLM-enabled era by their operational role and authority boundary. Our study identifies three recurring integration patterns, namely *operator-facing assistants*, *tool-using OAM workflow agents*, and *controller-centered execution with LLM-generated objectives*. Assistants primarily improve interaction and proposal generation; workflow agents extend the LLM-enabled component to coordinate multi-step procedures via approved tools; and controller-centered designs use LLMs to express objectives or constraints, while established controllers retain runtime execution authority. The patterns differ in the role assigned to the LLM-enabled component, the artifacts or procedures it produces, the boundary at which human approval or controller-mediated execution is required, and the assurance mechanisms that constrain any proposed or executed change. Fig. 6 places these three patterns along a common operational

pipeline and highlights where LLM-supported processing terminates relative to the commit-authority boundary.

Operator-facing assistants: In this pattern, the assistant serves as a bridge between network operators and existing OAM tools. It helps translate an operator request into a relevant operational context. This context may include operational records and network state, such as logs, tickets, runbooks, topology information, and configuration data. Based on this context, the assistant may identify relevant procedures and produce diagnostic hypotheses, checklists, or draft configuration changes. When such outputs may affect network state, they remain subject to operator review and to the approval processes of existing control and management systems.

One important category is the diagnostic assistant, which uses LLMs to help operators interpret incidents and localize likely causes. NetAssistant [18] is a production-reported assistant for data-center diagnosis. It uses an LLM-backed chatbot to translate operator questions into predefined diagnostic workflows that query monitoring data and return diagnostic results. BiAn [195] is deployed in Alibaba Cloud's production network for failure localization. It organizes the localization process through LLM-assisted agents that analyze monitoring data, score candidate devices, and produce ranked failure-localization results with explanations. RCACopilot [196], although focused on cloud incidents rather than network-specific OAM, uses LLM-based reasoning over collected diagnostic information to predict a root-cause category and produce an explanatory report for human responders. In these diagnostic assistants, the LLM supports interpretation, diagnosis, and explanation; yet operational action remains with human operators and the surrounding control or management processes.

A second assistant category focuses on configuration and test authoring. Unlike diagnostic assistants, which primarily support investigation, these systems generate artifacts that may later enter validation, testing, or deployment workflows. NetConfEval [21] benchmarks natural-language-to-configuration and related network-programming tasks. Clarify [197], CEGS [198], and INTA [22] provide research prototypes for ambiguity resolution, documentation-grounded configuration synthesis, and cross-vendor configuration translation with syntax or semantic checks. LLM-NetCFG [177] illustrates an intent-to-configuration workflow that combines local LLM-based generation with verification and device configuration. EYWA [199] extends the same authoring role to protocol testing. It employs LLMs to construct intended protocol-behavior models from RFCs and related documents, after which model-based testing exercises implementations. In these assistants, the LLM-generated artifact is closer to operational action than the diagnostic ones, but its use still depends on validation and the surrounding execution process.

From the NCI perspective, assistants usually reach *D2*, and some approach *D3* when they construct or revise multi-step plans at the proposal layer. Their behavior remains configured or curated rather than self-updating (*A0–A1*). Knowledge ranges from *K2* to *K3*, depending on whether the system primarily uses structured documentation and task inputs or correlates heterogeneous operational records. Interface capability similarly ranges from intent translation to dialogue clarification (*I2–I3*). Control Delegation remains low (*C0–C1*) because commit authority stays outside the assistant.

Tool-using OAM workflow agents: This pattern extends the assistant role from producing reviewable outputs to coordinating approved operational procedures. The agent receives an operator request, breaks it into ordered steps, invokes relevant tools, and records intermediate results. The tools may retrieve information, run diagnostics, update tickets, or perform operations that have already been authorized by the surrounding process. The agent thus coordinates execution within a bounded workflow, but it does not grant itself authority to change the network. Operations that may affect network state remain subject to the permissions, checks, and approval processes of existing control and management systems.

Existing workflow agents span production settings and research prototypes. Confucius [19] reports two years of production operation with more than 60 onboarded applications. It converts operator intent into workflow graphs that reuse existing operational procedures and runbook tools. Before an operation that may change the network state is executed, the workflow is verified and validated. OSS-GPT [20] presents an experimental design for coordinating operations across support-system interfaces. The agent plans a sequence of API calls, but operations that create, update, or delete service state require human validation. ArachNet [200] applies a similar tool-using workflow to Internet measurement tasks. Its prototype implementation coordinates multiple tools and records intermediate results.

In the NCI profile, tool-using workflow agents mainly advance *Decision Logic*. They can reach *D3* when the agent constructs, compares, or revises a multi-step procedure; systems that only select a predefined workflow remain closer to *D1–D2*. Their *Adaptability* is usually limited to *A1* because most systems reuse existing procedures rather than updating models, policies, or configurations in response to operational feedback. Their *Knowledge* can reach *K3* when the workflow combines operational records, documentation, procedure repositories, and execution traces. Their *Interface* capability is typically *I2–I3*, because the agent translates operator requests into tool calls and returns intermediate results in a form operators can inspect. *Control Delegation* rises to *C1–C2*, since the agent may execute approved workflows, but commit authority remains outside the agent.

Controller-centered execution: This pattern keeps execution inside established control mechanisms, such as controllers, solvers, and compilers. The LLM does not run the control loop directly. Instead, it translates operator intent, service requirements, or scenario descriptions into structured inputs, such as objectives, constraints, parameters, or test cases, that these mechanisms can process. Existing validation and approval processes check the resulting specification before any network change is applied. Runtime execution remains with controllers, solvers, and compilers, while validators and approval processes determine whether the proposed change can be applied.

Most systems in this pattern are either benchmarks or research prototypes. For instance, IntentOpt [23] evaluates whether LLM-generated optimization formulations match the

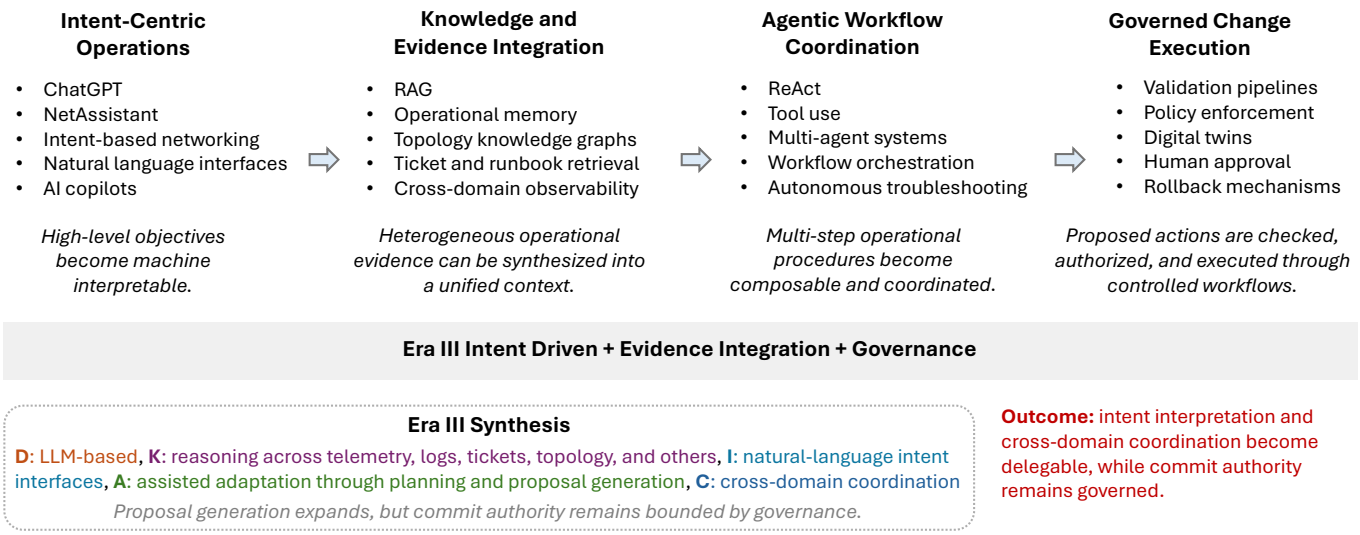


Fig. 7. This era transitions toward LLM-enabled network operations. Cross-domain operational pressure from network slicing, cloud-native functions, disaggregated RAN/core deployments, and multi-vendor environments shifts the bottleneck from domain-local control toward intent interpretation, evidence synthesis, and workflow coordination. LLM-enabled components operate mainly on the proposal side: they retrieve operational context, synthesize evidence, and generate proposal artifacts such as diagnostic hypotheses, verification queries, configuration deltas, ticket updates, rollout plans, and objective specifications. Governed execution remains separated from proposal generation through pre-change checks, authorization, controller-mediated execution, monitoring, rollback, and audit records. Through the NCI lens, Era III advances *Interface* and *Knowledge* most strongly, broadens proposal-side *Decision Logic*, and leaves *Control Delegation* and *Adaptability* bounded by assurance requirements.

requested objective and can be executed against ground-truth. It shows that textual plausibility alone is not enough; a formulation must also be compilable, feasible, and consistent with the requested behavior. Hybrid LLM-solver designs for constrained allocation [201] and solver-guided generation in LeJIT [24] follow the same division of labor. The LLM prepares a structured formulation, while solvers or compilers verify its validity and enforce the applicable constraints.

The same boundary also manifests in controller-centered systems, where the LLM is not the main source of the objective. Matryoshka [203] is a production-scale, model-driven compiler that translates high-level data-center design intent into detailed switch configurations. NetKeeper [204] combines natural-language intent, anomaly information, and traffic analysis with RL-based configuration updates in dynamic networks. NetNomos [202] uses formal rules and solver-based checks to constrain generative outputs. Network digital twins and high-fidelity simulation platforms provide another form of boundary: they allow proposed objectives or configuration updates to be tested before deployment [66], [193]. These works broaden the pattern beyond LLM-only formulation, but they support the same architectural point.

In the NCI view, this pattern mainly affects *Interface* and *Decision Logic*. The *Interface* score typically ranges from *I1–I2* because the system translates operator intent into structured objectives or constraints. *Decision Logic* is usually *D2*, as the LLM helps formulate the decision problem, but a solver, compiler, controller, or simulator makes the control decision. *Adaptability* often remains *A0–A1*, because the underlying control logic changes slowly and is not usually updated by the LLM itself. *Knowledge* can reach *K3* when the generated specification is based on topology, traffic, policies, or simulation state. *Control Delegation* remains low, typically *C0–C1*, because runtime authority stays with the established control mechanism rather than the LLM. Model-driven compilers and adaptive update engines may move further along *Control Delegation*, but their authority comes from explicit models, validators, and approval processes rather than unconstrained generative reasoning.

Capabilities and Boundaries: Across the three integration patterns, LLMs mainly strengthen the proposal side of network operations. They help interpret operator requests, assemble relevant operational context, draft reviewable outputs, and coordinate the steps of approved procedures. In doing so, they can make logs, tickets, runbooks, topology information, and configuration data easier to use in operational workflows. Their intermediate steps and outputs should remain visible to operators and existing management systems, rather than being hidden inside the model.

The same evidence also clarifies that LLMs do not remove the need for domain controllers, optimization engines, policy checks, validation mechanisms, approval processes, or recovery procedures. When a system appears autonomous, the autonomy is usually assembled from several parts: LLM-assisted proposal generation, execution through established controllers or tools, and explicit checks before a network change is applied. The scope of action remains limited by permissions and by the surrounding operational process. Fig. 7 summarizes the shift toward LLM-assisted proposal generation, evidence synthesis, and workflow coordination, with operational changes applied through governed execution paths.

The evidence behind these patterns is uneven. Production deployments are strongest for diagnostic assistants and tool-using OAM workflow agents, as illustrated by NetAssistant and Confucius. Matryoshka and Hoyan show that intent compilation and verification can become production-grade opera-

TABLE IV

FIVE-AXIS PROFILE OF REPRESENTATIVE WORKS IN THE LLM-ENABLED ERA. THE TABLE HIGHLIGHTS A RECURRING ASYMMETRY: *Interface* AND *Knowledge* ADVANCE QUICKLY, WHILE *Control Delegation* REMAINS CONSERVATIVE. ADAPTABILITY ENTRIES ABOVE *A1* ARE RESERVED FOR PERIODIC DATA-DRIVEN UPDATES OR CONTINUOUS ONLINE ADAPTATION, CONSISTENT WITH SECTION II; ORDINARY MAINTENANCE OF PROMPTS, RETRIEVAL CORPORA, WORKFLOWS, OR POLICY RULES DOES NOT, BY ITSELF, QUALIFY. THE EVIDENCE COLUMN DISTINGUISHES PRODUCTION REPORTS, RESEARCH PROTOTYPES, AND BENCHMARK STUDIES.

Work / System	D	A	K	C	I	Evidence
Operator-facing assistants: diagnosis						
NetAssistant [18]	D2	A1	K3	C0	I3	Production report
BiAn [195]	D2	A1	K3	C0	I3	Production report
RCACopilot [196]	D2	A1	K3	C0	I2	Research prototype
Operator-facing assistants: configuration and test authoring						
NetConfEval [21]	D2	A0	K2	C0	I2	Benchmark study
Clarify [197]	D2	A1	K2–K3	C1	I3	Research prototype
CEGS [198]	D2	A1	K2–K3	C1	I2	Research prototype
INTA [22]	D2	A1	K2–K3	C1	I2	Research prototype
LLM-NetCFG [177]	D2	A1	K2–K3	C1	I2	Research prototype
EYWA [199]	D2	A0	K3	C0	I2	Research prototype
Tool-using OAM workflow agents						
Confucius [19]	D3	A1	K3	C2	I2	Production report
OSS-GPT [20]	D3	A1	K3	C1	I3	Research prototype
ArachNet [200]	D3	A1	K3	C1	I2	Research prototype
Controller-centered execution with LLM-generated objectives						
IntentOpt [23]	D2	A0	K2–K3	C0	I2	Benchmark study
LLM–solver allocation [201]	D2	A0	K2	C0	I1–I2	Research prototype
LeJIT [24]	D2	A0	K2	C0	I1	Research prototype
Controller-centered systems with limited or no LLM role						
NetNomos [202]	D2	A0	K2–K3	C0	I1	Research prototype
Matryoshka [203]	D1–D2	A1	K2	C1–C2	I2	Production report
NetKeeper [204]	D2–D3	A2	K2–K3	C1–C2	I2	Research prototype

tional infrastructure, even when the LLM role is limited [189], [203]. By contrast, controller-centered uses of LLMs are still dominated by benchmarks and research prototypes, such as IntentOpt, LeJIT, and constrained LLM-solver designs.

D. Five-Axis Analysis of the LLM-Enabled Era

The integration patterns above indicate that LLM-enabled components mostly enter operational workflows before approval and execution, as summarized in Table IV. The most visible movement is along the *Interface* and *Knowledge* axes. LLM-enabled systems reduce the effort required to turn operator requests, operational records, runbooks, topology information, and configuration data into explanations, reviewable proposals, and structured inputs that existing controllers can process. Assistants and workflow agents most directly advance *Interface* and *Knowledge*. They also broaden *Decision Logic* at the proposal layer while keeping *Control Delegation* low, since approval and execution are still tied to established processes.

The five-axis profile exposes the main asymmetry of this era: LLM-enabled systems have become better at interpreting requests, assembling evidence, and drafting operational proposals, but this progress has not been matched by a comparable increase in their authority to implement changes or their ability to adapt safely during operation. Richer interfaces allow operators to express intent more naturally, but they also require clarification, grounding, and validation when intent is ambiguous. Broader operational knowledge improves proposal quality only when the source, freshness, scope, and provenance of the supporting context are visible. Likewise, stronger proposal-side decision logic can improve diagnosis, change authoring, and workflow coordination, but these gains do not by themselves justify greater authority to apply changes.

By contrast, *Control Delegation* and *Adaptability* are still the slower-moving axes. Although tool-using workflow agents move further along *Control Delegation* than assistants because they can execute approved procedural steps using existing tools, the authority to commit still lies outside the LLM-enabled component. Controller-centered designs are more conservative: the LLM prepares objectives, constraints, or structured inputs, while controllers, solvers, or compilers retain runtime authority. *Adaptability* is also limited in most systems. Operational behavior is usually changed through maintained prompts, retrieval sources, workflows, model versions, policy rules, configuration templates, or controller parameters, not through online adaptation by the LLM during operation.

This asymmetry suggests a cautious view of the LLM-enabled era. Its main contribution is not autonomous actuation, but the generation of proposals and workflows that can be inspected, checked, approved, and recorded. Consequently, evaluation should reflect the authority assigned to the LLM-enabled component. A proposal-generating assistant should be judged by whether its outputs are grounded, reviewable, and checkable; a workflow agent that approaches state-changing operations must also be judged by whether permissions, approval gates, monitoring, rollback, and audit records work as intended. Task success and linguistic plausibility are useful signals, but they are insufficient for assessing operational control. The next step is not to expand LLM autonomy simply because proposal generation has improved, but to pragmatically establish when proposal generation, controlled adaptation, and authorized execution are supported by sufficient evidence, safeguards, and accountability within the surrounding operational process, as detailed in Section VI.

VI. Toward Trustworthy Autonomous Networking

The evolution traced in Sections III–V shows the uneven progress of NCI. Specifically, rule-based automation made routine operations more repeatable; programmable and data-driven control improved observability, optimization, and domain-specific control; and LLM-enabled systems now strengthen intent interpretation, evidence synthesis, and proposal generation. However, these advances do not by themselves justify broader network autonomy. A network control system that can observe more states, generate better explanations, or prepare more structured proposals may still be unsuitable for changing network state without additional checks, authority boundaries, and recovery mechanisms.

This section translates the NCI lens into a forward-looking roadmap for trustworthy autonomy in evolving network environments. The roadmap focuses on two coupled questions. First, can NCI remain reliable as traffic, topology, software, policies, and operational practices change? Second, can autonomy increase without weakening the safeguards that make network changes accountable? The TM Forum Autonomous Networks Levels framework describes staged progress toward more autonomous operation, largely in terms of how operational tasks and authority are delegated between humans and automated systems [45]. Recent industry reports show that operators are actively pursuing this direction [11]. The NCI perspective asks a complementary question: what technical conditions must be in place before such autonomy becomes defensible? In practice, this remains difficult when integration, data governance, regulatory constraints, skills, organizational trust, and cross-domain assurance are not yet mature [13]. For this reason, stronger autonomy claims should be tied to explicit scope, authority boundaries, operational state, validation procedures, and recovery conditions [205].

On this basis, we organize the roadmap around *Adaptability* and *Control Delegation*. Adaptability determines whether NCI can remain reliable as operating conditions change. Control Delegation determines whether authority to affect network state can be delegated to automated components without weakening accountability. The other three axes operationalize this focus: *Knowledge* grounds decisions in a faithful view of the network state, *Decision Logic* makes candidate actions checkable under constraints, and *Interface* exposes objectives, evidence, uncertainty, and responsibility to operators and peer systems. Cross-domain incidents and changes illustrate why these axes must be considered together, as symptoms and dependencies may span services, topology, policies, telemetry, recent configuration changes, and rollout constraints [14], [15], [58], [66]. Trustworthy autonomous networking thus requires that the five axes mature together and be evaluated under realistic operational conditions, including pre-change validation, incident replay, rollback behavior, and production-level workflows [206]–[208].

A. Evaluation Methodology and Evidence of Progress

Before delving into the remaining technical challenges, it is necessary to clarify what would count as *evidence of progress*. In autonomous networking, benchmark accuracy, successful troubleshooting on curated tickets, or a plausible natural-language explanation can be useful, but they do not prove that a system can be safely granted more authority over network-side actuation. For instance, the systems reviewed in Section V are usually hybrid, i.e., LLM-enabled components interpret requests, assemble evidence, and prepare proposals, while controllers, solvers, validation tools, approval processes, and recovery procedures remain responsible for execution. The evaluation should state which part of this workflow is being tested and where the authority boundary is drawn.

To make evidence comparable, each study should specify the NCI capability being tested and the authority boundary assumed by the system. Specifically, it should state whether the system produces only advisory outputs, coordinates approved steps, or can affect the network state; it should also identify the associated checks, approvals, guardrails, and recovery mechanisms. When the system adapts its models, workflows, policies, or configuration over time, the evaluation should show whether the original evidence remains valid or whether the system must be revalidated before stronger autonomy is bestowed. Without this separation, a system that improves diagnosis may be mistaken for one that can execute remediation, and a system that coordinates approved procedure steps may be mistaken for one that holds independent commit authority. Evaluation should report both effectiveness and efficiency: whether actions remain valid and recoverable under change, and whether the control process stays within acceptable purviews for latency, cost, and operator workload.

B. Knowledge: Faithful Operational State

The first axis-specific bottleneck is *Knowledge*. Autonomous operation depends on the state over which the system reasons. In many deployments, telemetry, topology, configuration, inventory, service dependencies, tickets, and policy artifacts are distributed across tools, domains, vendors, and administrative domains [66], [167], [209]. Even when each domain is well instrumented, cross-domain context may be reconstructed from records with different timestamps, naming conventions, levels of detail, and provenance. An autonomous system can then appear competent on abundant local evidence while failing when the relevant cause spans domains.

Moreover, trustworthy autonomy requires more than collecting more data. It requires an operational knowledge base that represents what the network is doing, what it is expected to do, what constraints it must satisfy, and how reliable each piece of evidence is. The system must also track the conditions under which each piece of context is valid. A topology snapshot, ticket, policy rule, or telemetry stream may be reliable only within a particular time window, domain, or administrative scope. When those limits are unclear, an otherwise valid proposal can rely on stale evidence, misattribute responsibility, or apply a change in the wrong part of the network. Digital twins illustrate both the promise and the difficulty. They can support what-if analysis, pre-change validation, and cross-domain reasoning, but only if their assumptions and synchronization state are explicit [66], [67], [210]. A twin used for planning without

operational reality may provide a false sense of reliability. Similarly, knowledge graphs and schema-driven pipelines can encode topology, policies, service dependencies, and historical incidents, but they must remain accurate through failures, maintenance windows, and configuration changes [58], [211]. The research challenge is to build knowledge layers that can be checked against the live network and invalidated when their assumptions no longer hold.

Evidence on this axis should demonstrate that the system can reliably reconstruct the operational state under realistic conditions. This can be evaluated through replay studies on historical incidents, shadow-mode studies beside live operations, or controlled fault-injection experiments in emulation or digital-twin environments. Such studies should compare inferred state with trusted operational records, measure the impact of stale or missing inputs, and test whether incorrect context is detected before it affects a proposal or a network change. Without such evidence, stronger decision logic or more capable language models will be poorly grounded.

C. Decision Logic: Compositional Planning Under Operational Constraints

Once the system has a usable view of operational state, the next question is how it turns that state into valid actions. The *Decision Logic* challenge is not to replace existing control mechanisms with a general-purpose reasoner. Multi-domain production networks already rely on specialized mechanisms, including traffic-engineering solvers, admission-control logic, configuration compilers, verification tools, diagnosis systems, and domain-specific control loops. The challenge is to compose these mechanisms so that plans remain valid across domains, timescales, and approval and execution boundaries.

Section V suggests a recurring division of labor. LLM-enabled components are useful for interpreting requests, assembling context, drafting proposals, constructing structured formulations, and coordinating steps in approved procedures. Actions that require responsiveness, stability, or safety guarantees should remain with mechanisms whose behavior can be bounded and checked. In controller-centered systems, for example, an LLM may prepare objectives or constraints, while solvers, compilers, and controllers determine whether the resulting specification is valid and executable [23], [24]. In tool-using workflow agents, the agent may sequence approved steps, but authority to apply network changes remains with the surrounding control and approval process [19], [20].

The open problem is to make the assumptions behind each proposed action explicit. A proposal should state the conditions under which it is valid, including required inputs, preconditions, affected scope, expected outcomes, rollback conditions, and checks to be performed before execution. The same requirement applies to plans assembled from multiple tools. A sequence that is safe within one domain may become unsafe once service dependencies, maintenance windows, or policy constraints in another domain are taken into account. Decision logic should produce artifacts that can be checked, simulated, and audited in addition to explanations.

Future work should make the handoff from proposal generation to governed execution explicit. A proposal artifact should state the information it relies on, the intended scope of the change, the checks it has passed, the assumptions that still need validation, and the conditions under which rollback would be triggered. This is necessary because a change that appears safe within one domain may become unsafe when service dependencies, policy rules, maintenance windows, or shared resources in another domain are considered [183], [212], [213]. Clear handoff records allow existing control and management systems to check a proposal before execution, monitor whether its assumptions hold after deployment, and recover if the outcome deviates from the expected path.

D. Adaptability: Improving Without Destabilizing

As for *Adaptability*, the challenge is no longer simply whether a system can adjust its behavior. The harder question is whether such adjustments can be introduced without weakening the guarantees under which the system was allowed to operate. In autonomous networking, control behaviors can be adapted through several sources, including prompts that guide LLM responses, retrieval sources that supply operational context, workflow definitions that order tool use, policy rules that constrain approval, learned models that support prediction or diagnosis, configuration templates that generate candidate changes, and controller parameters that shape runtime decisions. Each adaptation can improve performance in one setting while creating regressions elsewhere, moving behavior outside tested envelopes, or degrading performance on unseen but operationally important cases [214], [215].

This distinction is important when interpreting the *Adaptability* axis. Routine maintenance of prompts, retrieval corpora, workflow definitions, or policy files should not be treated as high adaptability. Such maintenance may improve system behavior, but it does not necessarily show that the system learns from operational feedback or updates its behavior through a governed adaptation process. Higher levels on the Adaptability axis should be reserved for systems that periodically or continuously update models, policies, parameters, or configurations based on operational data, and that subject those updates to regression control and human or automated oversight.

This distinction also exposes several failure modes. Learning-enabled controllers can perform well on fixed topologies or curated fault scenarios, yet degrade under traffic shifts, new hardware, or unseen incidents [152], [154], [216]. LLM-enabled workflows can drift over long tasks when intermediate state is summarized poorly, or tool results are not preserved in durable artifacts [29], [32]. Configuration-update systems can improve one metric while violating an operational constraint that was absent from the training or update loop. In all three cases, the problem is not adaptation itself, but adaptation without adequate control over regression, scope, and recovery.

Evidence should be reported for the specific artifact or component that has been updated. For a learned model, this includes performance under shifted topologies, traffic patterns, faults, and policy settings, as well as regression results on previously validated cases. For a workflow, policy rule, prompt package, retrieval corpus, or configuration template, this includes tests showing that existing procedures still execute correctly, unsafe proposals are still blocked, required checks are

still applied, and operational records remain complete. Across these cases, the evaluation should report how failures are detected, whether rollback succeeds, how long recovery takes, and whether the system safely reduces its autonomy level until the update is corrected and revalidated. Adaptation that cannot preserve or safely lower autonomy is not operational progress; it is an unbounded change to the assurance envelope.

E. Control Delegation: Authority as an Assurance Claim

Control Delegation is the second central bottleneck because it determines when a system can move from proposal to action. Broader autonomy is not justified by better recommendations alone, and it becomes especially fragile when the system is also adapting. A proposed change must remain within scope, satisfy policy and safety constraints, be rolled out safely, be monitored after deployment, and be recoverable if the outcome is unacceptable [14], [15], [183], [206].

In many systems, these checks remain fragmented. For instance, a configuration may be syntactically valid but inconsistent with policy. A routing update may satisfy a local objective but violate a service-level constraint elsewhere. A workflow may include approval steps but leave incomplete evidence about what was checked, who or what approved the action, and why execution was allowed to proceed. Stronger autonomy requires these checks to be tied to both the proposal artifact and the execution record.

A delegated action should leave enough evidence for operators to understand why execution was allowed. The record should connect the original intent to the affected resources, the checks that constrained the action, the approval decision, the rollout strategy, and the conditions under which recovery would be triggered. This evidence can be produced through verification, staged rollout, shadow execution, production emulation, and replay [66], [185], [186], [207], [208], [217], [218]. As autonomy increases, the record must provide stronger justification that the action remained within its declared scope.

Security and governance also determine how far control can be delegated. Tool-using LLM systems expand the attack surface because retrieved content, tool outputs, and intermediate instructions can influence later actions. Indirect prompt injection, excessive tool permissions, and misuse of operational data are not only security concerns; they can invalidate the authority boundary itself [219]. Telecom-oriented agent protocols add another layer of coordination, where capabilities, permitted scope, and accountability must be explicit rather than inferred [220]. A defensible design should restrict what each tool can do, preserve the origin of retrieved evidence, isolate operations that can change state, and leave an audit trail from request to outcome [211], [221]. These controls determine whether a system can be trusted to apply changes, not merely whether it is securely implemented.

F. Interface: Evidence-Bearing Interaction

The *Interface* axis determines how objectives, explanations, decisions, and supporting records are exchanged. The LLM-enabled era makes this axis central because natural-language interaction can reduce the effort required to express intent, inspect incidents, and coordinate operations. Yet conversational fluency does not provide accountability by itself. An interface that supports stronger autonomy must make the basis for a recommendation or action visible: which information was used, which assumptions were made, which checks passed or failed, and what uncertainty remains [14], [16], [17].

For operators, the interface should evolve from a place for issuing commands into a workspace for reviewing evidence and responsibility. It should connect the original request, the assembled context, the proposal artifact, the validation results, the approval decision, and the observed outcome. This requirement follows directly from the proposal/execution boundary in Section V. When an LLM-enabled component proposes a change, the interface should make the proposal and its supporting evidence reviewable before the change is applied. When a workflow agent executes approved steps, the interface should expose the execution trace, deviations from the expected path, and any escalation.

Interoperability extends the same problem across tools and organizations. Automation components should describe their capabilities, required inputs, permitted actions, and safety scope in forms that other systems can interpret. Intent and state should be exchanged with stable semantics across integration points. Conflict handling, arbitration, and plan repair should be explicit parts of the interface [58], [220], [222], [223]. Without shared semantics, each new integration must rebuild the assurance boundary from the beginning.

The human role also changes. Operators move from writing low-level commands toward reviewing proposals, curating policies, handling exceptions, and learning from incidents. This transition requires training and organizational processes that define responsibility for review, approval, and escalation. A system may be technically capable of producing valid proposals, but autonomy will remain limited if operators cannot understand why an action was proposed, what risk remains, and who is accountable for approving it.

From Axes to Roadmap

The challenges across the five axes are not a checklist of five independent workstreams. The roadmap is illustrated in Fig. 8. Its center is the Adaptability–Control Delegation pair, as a network control system must be able to improve as conditions change, and any increase in authority must remain justified after that change. The other three axes provide the supporting conditions: Knowledge keeps adaptation and autonomy grounded in a faithful state; Decision Logic makes proposed actions checkable; and Interface makes evidence, uncertainty, and responsibility visible. Higher autonomy is limited by the weakest relationship among these axes, not by the most advanced component.

This roadmap also clarifies why the research–deployment gap persists. Emulation, replay, configuration analysis, and deployed diagnostic systems show that production environments impose constraints on scale, fidelity, reliability, and operational workflow that curated laboratory studies can miss [18], [206]–[208]. Credible progress should be demonstrated under realistic constraints, including change windows, rollback

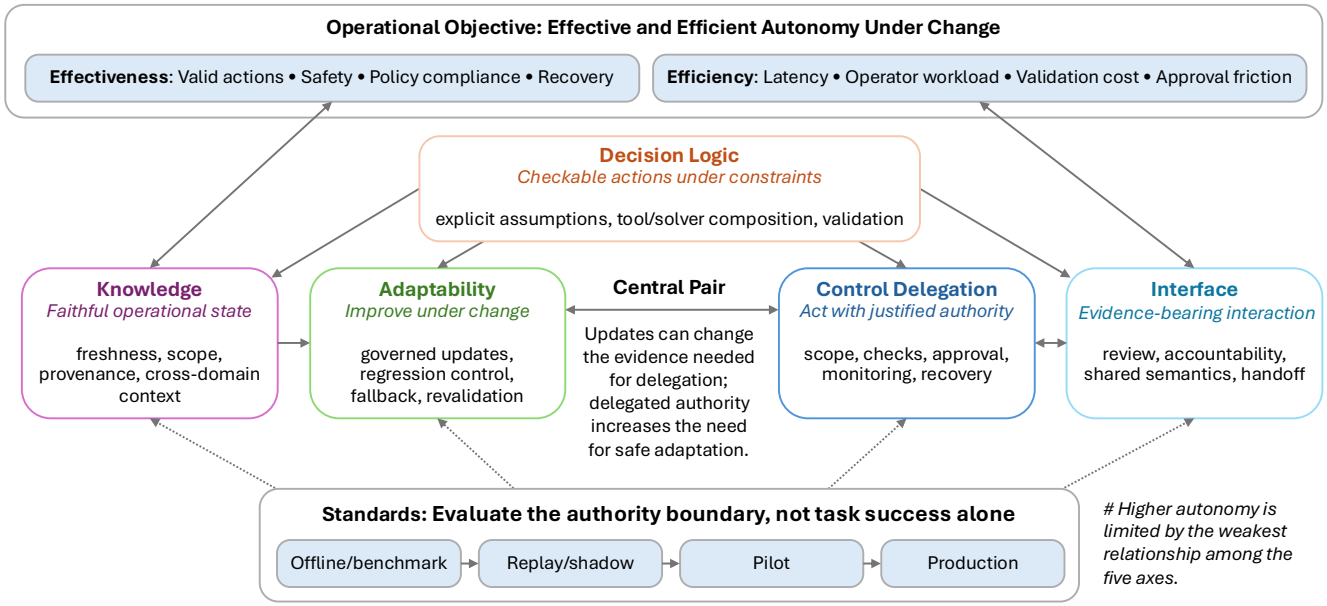


Fig. 8. Roadmap toward trustworthy autonomous networking through the five-axis NCI lens. *Adaptability* and *Control Delegation* form the central unresolved pair: a system must improve under changing conditions, and stronger authority must remain justified after such change. *Knowledge*, *Decision Logic*, and *Interface* provide the supporting conditions. Evidence standards determine whether stronger autonomy claims are credible.

requirements, multi-vendor boundaries, partial observability, and adversarial conditions. The relevant technologies should be evaluated as parts of an operational control stack rather than as isolated advances. In this sense, the roadmap is not a list of desirable features; it is an evidentiary discipline for deciding when a capability can safely support stronger autonomy.

VII. Conclusion

This paper has used Network Control Intelligence (NCI) to reinterpret the long pursuit of autonomous networking as a question of control capability, evidence, and authority. Across the eras reviewed here, progress did not follow a single ladder. It appeared instead as a sequence of partial advances: more programmable interfaces, richer operational state, stronger analytics, more expressive proposal mechanisms, and gradually broader but still bounded autonomy.

The central lesson is that network autonomy is not obtained by inserting a more intelligent component into an otherwise unchanged control stack. In credible designs, LLM-enabled components may generate proposals, explanations, candidate configuration changes, verification queries, or rollout plans, but operational changes must still be applied through established control and management systems. Authority is constrained by policy and invariant checks, authorization, staged rollout, rollback mechanisms, and change records. The enduring architectural question is not whether an agent can produce a plausible plan, but whether the surrounding system can justify committing it.

This perspective also clarifies what progress should mean. Larger models and more capable agents may improve interpretation and workflow preparation, but trustworthy autonomy will depend on control systems whose authority grows only with the evidence that supports it. Adaptation is central to this progression, yet it also creates risk: as models, policies, workflows, and network conditions change, the justification for stronger autonomy must be rechecked rather than assumed to persist. The path forward is incremental. Autonomy should expand only where actions are grounded in a faithful state, constrained by checkable decision logic, exposed through evidence-bearing interfaces, and recoverable when assumptions fail. The measure of success is not the appearance of end-to-end agency, but the ability to increase autonomy without losing safety, accountability, or operational control.

Acknowledgment

The authors used Grammarly for typographical corrections, grammar checking, and language polishing. The authors reviewed and approved the final manuscript and remain responsible for all technical content, citations, and conclusions.

References

[1] J. Schoenwaelder, "Overview of the 2002 iab network management workshop," Internet Engineering Task Force, RFC 3535, 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3535

[2] D. Kreutz, F. M. 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, 2014.

[3] ETSI Network Functions Virtualisation Industry Specification Group (NFV ISG), "Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action," ETSI, White Paper, Oct. 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[6] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[7] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[8] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[9] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," *arXiv preprint arXiv:1710.11583*, 2017.

[10] ETSI, "Zero-touch network and Service Management (ZSM); Reference Architecture," ETSI, ETSI GS ZSM 002 V1.1.1, Aug. 2019, published 2019-08; accessed 2025-09-01. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[11] R. Webb, "A Regional Guide to Autonomous Networks Progress," https://inform.tmforum.org/research-and-analysis/reports/a-regional-guide-to-autonomous-networks-progress, Jun. 2025, accessed 2026-02-14.

[12] L.-H. Shen, K.-T. Feng, and L. Hanzo, "Five facets of 6g: Research challenges and opportunities," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–39, 2023.

[13] Capgemini Research Institute, "Networks with intelligence," https://www.capgemini.com/wp-content/uploads/2024/01/CRI_Autonomous-Network.pdf, 2024, accessed on Sep 1, 2025.

[14] ETSI, "Zero-touch network and service management (zsm); intent-driven closed loops," ETSI, ETSI GS ZSM 016 V1.1.1, Oct. 2024, accessed 2026-02-14. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/016/01.01.01_60/gs_ZSM016v010101p.pdf

[15] ——, "Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers," ETSI, ETSI GS ZSM 009-1 V1.1.1, Jun. 2021, accessed 2026-02-14. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf

[16] TM Forum, "IG1251E Multi-Agent Architectures v1.1.0," https://www.tmforum.org/resources/introductory-guide/ig1251e-multi-agent-architectures-v1-1-0/, Nov. 2025, accessed 2026-02-14.

[17] ——, "IG1414 Agentic AI Closed Loops v1.1.0," https://www.tmforum.org/resources/standard/ig1414-agentic-ai-closed-loops-v1-1-0/, Sep. 2025, accessed 2025-09-22.

[18] H. Wang, A. Abhashkumar, C. Lin, T. Zhang, X. Gu, N. Ma, C. Wu, S. Liu, W. Zhou, Y. Dong, W. Jiang, and Y. Wang, "NetAssistant: Dialogue based network diagnosis in data center networks," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 2011–2024. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/wang-haopei

[19] Z. Wang, S. Lin, G. Yan, S. Ghorbani, M. Yu, J. Zhou, N. Hu, L. Baruah, S. Peters, S. Kamath, J. Yang, and Y. Zhang, "Intent-driven network management with multi-agent llms: The confucius framework," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 347–362. [Online]. Available: https://doi.org/10.1145/3718958.3750537

[20] A. Mekrache, A. Ksentini, and C. Verikoukis, "Oss-gpt: An llm-powered intent-driven operations support system for 6g networks," in *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*, 2025, pp. 155–163.

[21] C. Wang, M. Scazzariello, A. Farshin, S. Ferlin, D. Kostić, and M. Chiesa, "Netconfeval: Can llms facilitate network configuration?" *Proc. ACM Netw.*, vol. 2, no. CoNEXT2, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3656296

[22] Y. Wei, X. Xie, T. Hu, Y. Zuo, X. Chen, K. Chi, and Y. Cui, "Inta: Intent-based translation for network configuration with llm agents," in *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*, 2025, pp. 1–16.

[23] T. Ahmed, Y. Zhu, and S. Choudhury, "Vision language models for optimization-driven intent processing in autonomous networks," 2026. [Online]. Available: https://arxiv.org/abs/2601.12744

[24] H. Hè and M. Apostolaki, "Just-in-time logic enforcement: A new paradigm of combining statistical and symbolic reasoning for network management," in *Proceedings of the 24th ACM Workshop on Hot Topics in Networks*, ser. HotNets '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 184–192. [Online]. Available: https://doi.org/10.1145/3772356.3772406

[25] AT&T Bell Laboratories. Technical Publication Department, *Engineering and Operations in the Bell System*, 2nd ed. AT&T Bell Laboratories, 1983.

[26] Y. Huang, H. Du, X. Zhang, D. Niyato, J. Kang, Z. Xiong, S. Wang, and T. Huang, "Large language models for networking: Applications, enabling techniques, and challenges," *IEEE Network*, vol. 39, no. 1, pp. 235–242, 2025.

[27] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian *et al.*, "Toolllm: Facilitating large language models to master 16000+ real-world apis," in *International Conference on Learning Representations*, vol. 2024, 2024, pp. 9695–9717.

[28] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[29] Anthropic, "Effective context engineering for AI agents," https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents/, Sep. 2025, accessed 2026-04-17.

[30] ——, "Demystifying evals for AI agents," https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents, Jan. 2026, accessed 2026-04-17.

[31] ——, "Effective harnesses for long-running agents," https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents, Nov. 2025, accessed 2026-04-17.

[32] OpenAI, "Harness engineering: leveraging Codex in an agent-first world," https://openai.com/index/harness-engineering/, Feb. 2026, accessed 2026-04-17.

[33] Y. Xiao, H. Zhou, X. Li, Y. Gao, G. Shi, and P. Zhang, "Sannet: A semantic-aware agentic ai networking framework for multi-agent cross-layer coordination," *arXiv preprint arXiv:2505.18946*, 2025.

[34] "Blue Planet Showcases Agentic AI Framework at DTW25," https://www.ciena.com/about/newsroom/press-releases/blue-planet-showcases-agentic-ai-framework-at-dtw25, 2025, Accessed on Sep 1, 2025.

[35] K. McDonnell, "The Agentic Network: Why Multi-Agent Systems Will Define Telecom Autonomy," https://blog.huawei.com/en/post/2025/8/14/agentic-network-multi-agent-systems-telecom-autonomy, Aug. 2025, accessed 2025-09-01.

[36] "Netcracker Agentic AI Solution," https://www.netcracker.com/portfolio/products/agentic-ai-solution.html, accessed on Sep 1, 2025.

[37] Deutsche Telekom, "Deutsche Telekom and Google Cloud Partner on Agentic AI for Autonomous Networks," https://www.telekom.com/en/media/media-information/archive/deutsche-telekom-and-google-cloud-partner-on-agentic-ai-for-autonomous-networks-1088504, Feb. 2025, accessed 2025-09-01.

[38] W. R. Ashby, *An introduction to cybernetics*. Chapman and Hall, 1956.

[39] K. J. Åström and R. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[40] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 4/E*. Pearson, 2021.

[41] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," in *Situational awareness*. Routledge, 2017, pp. 9–42.

[42] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[43] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "A model for types and levels of human interaction with automation," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, vol. 30, no. 3, pp. 286–297, 2000.

[44] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2016.

[45] TM Forum, "IG1392 Autonomous Networks Levels Assessment and Certification Whitepaper v2.0.0," https://www.tmforum.org/resources/standard/ig1392-autonomous-networks-levels-assessment-and-certification-whitepaper-v2-0-0/, Nov. 2025, accessed 2026-02-14.

[46] "AN manifesto - TM Forum," https://www.tmforum.org/autonomous-networks-manifesto/, accessed on Sep 1, 2025.

[47] "ETSI Experiential Networked Intelligence group releases 5 specifications," https://www.etsi.org/newsroom/press-releases/1308-2018-06-press-etsi-experiential-networked-intelligence-group-releases-5-specifications/, Accessed on Sep 1, 2025.

[48] 3GPP, *Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements*, 3rd Generation Partnership Project Std. TS 32.500, 2022, accessed 2025-09-01. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2031

[49] O-RAN Alliance, "O-RAN: Towards an Open and Smart RAN," https://mediastorage.o-ran.org/white-papers/O-RAN.White-Paper-2018-10.pdf, Oct. 2018, accessed on Sep 1, 2025.

[50] 3GPP, "3GPP TS 29.520: 5g system; network data analytics services (nwdaf); stage 3," https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3355, accessed: 2025-09-25.

[51] ITU-T, "Focus Group on Autonomous Networks (FG-AN)," https://www.itu.int/en/ITU-T/focusgroups/an/Pages/default.aspx, accessed 2026-04-17.

[52] ——, "Architectural framework for machine learning in future networks including IMT-2020," International Telecommunication Union, Recommendation Y.3172, Jun. 2019, approved 2019-06-22. [Online]. Available: https://www.itu.int/itu-t/recommendations/rec.aspx?rec=13894

[53] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (netconf)," Internet Engineering Task Force (IETF), Request for Comments 6241, Jun. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6241

[54] A. Bierman, M. Bjorklund, and K. Watsen, "Restconf protocol," Internet Engineering Task Force (IETF), Request for Comments 8040, Jan. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8040

[55] M. Bjorklund, "The yang 1.1 data modeling language," Internet Engineering Task Force (IETF), Request for Comments 7950, Aug. 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7950

[56] B. Claise, J. Clarke, and J. Lindblad, *Network programmability with YANG: the structure of network automation with YANG, NETCONF, RESTCONF, and gNMI*. Addison-Wesley Professional, 2019.

[57] OpenConfig, "OpenConfig: Vendor-neutral, model-driven network management designed by users," https://www.openconfig.net/, accessed 2026-06-01.

[58] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-based networking - concepts and definitions," RFC Editor, RFC 9315, Oct. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9315

[59] C. Li, O. Havel, A. Olariu, P. Martinez-Julia, J. Nobre, and D. Lopez, "Intent classification," Internet Research Task Force (IRTF), Request for Comments 9316, Oct. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9316

[60] T. Eckert, M. Behringer, and S. Bjarnason, "An autonomic control plane (acp)," Internet Engineering Task Force (IETF), Request for Comments 8994, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8994

[61] M. Pritikin, M. Richardson, T. Eckert, M. Behringer, and K. Watsen, "Bootstrapping remote secure key infrastructure (brski)," Internet Engineering Task Force (IETF), Request for Comments 8995, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8995

[62] "Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions," https://opennetworking.org/onos/, Accessed on Sep 1, 2025.

[63] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, "Central office re-architected as a data center," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 96–101, 2016.

[64] Open Networking Foundation, "SDN Enabled Broadband Access (SEBA) Reference Design v2.0 (ONF TS-103)," Open Networking Foundation, Technical Specification, Mar. 2021. [Online]. Available: https://lfbroadband.org/wp-content/uploads/sites/8/2024/03/ONF-Reference-Design-SEBAv2.0FINAL3.pdf

[65] T. Zhang, H. Qiu, L. Linguaglossa, W. Cerroni, and P. Giaccone, "Nfv platforms: Taxonomy, design choices and future challenges," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 30–48, 2020.

[66] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong *et al.*, "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022.

[67] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[68] C. S. Dawson, C. J. McCallum Jr, R. B. Murphy, and E. Wolman, "Operations research at bell laboratories through the 1970s: Part i," *Operations Research*, vol. 48, no. 2, pp. 205–215, 2000.

[69] AT&T, "History of Network Management," https://web.archive.org/web/20131102111637/http://www.corp.att.com/history/nethistory/management.html, accessed 2026-04-24.

[70] M. Boucadair, Q. Wu, Z. Wang, D. King, and C. Xie, "Framework for Use of ECA (Event Condition Action) in Network Self Management," IETF Internet-Draft draft-bwd-netmod-eca-framework-00, 2020.

[71] G. C. Ebner and D. G. Haenschke, "Total network data system: network management," *The Bell System Technical Journal*, vol. 62, no. 7, pp. 2239–2260, 1983.

[72] "Development of the AXE system - Ericsson History," https://www.ericsson.com/en/about-us/history/products/the-switches/development-of-the-axe-system, Accessed on Sep 1, 2025.

[73] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol," Internet Engineering Task Force (IETF), Request for Comments 1067, Aug. 1988. [Online]. Available: https://www.rfc-editor.org/info/rfc1067

[74] ——, "Simple network management protocol (snmp)," Internet Engineering Task Force (IETF), Request for Comments 1157, May 1990. [Online]. Available: https://www.rfc-editor.org/info/rfc1157

[75] D. Perkins and E. McGinnis, *Understanding SNMP MIBs*. Prentice-Hall, Inc., 1997.

[76] B. M. Faul, "Using a clips expert system to automatically manage tcp/ip networks and their components," in *NASA. Johnson Space Center, Second CLIPS Conference Proceedings, Volume 1*, 1991.

[77] T. Oetiker and D. Rand, "Mrtg: The multi router traffic grapher," in *LISA*, vol. 98, 1998, pp. 141–148.

[78] P. M. Sittler, "Linux gazette: Big brother monitoring system," *Linux Journal*, vol. 1997, no. 40es, pp. 20–es, 1997.

[79] ITU-T, "Principles for a telecommunications management network," International Telecommunication Union, Recommendation M.3010, Feb. 2000. [Online]. Available: https://www.itu.int/rec/T-REC-M.3010

[80] ——, "TMN management functions," International Telecommunication Union, Recommendation M.3400. [Online]. Available: https://www.itu.int/rec/T-REC-M.3400-200002-I/en

[81] "IBM Tivoli NetView - IBM Documentation," https://www.ibm.com/docs/en/netcoolomnibus/8.0.0?topic=integrations-tivoli-netview, Accessed on Sep 1, 2025.

[82] Hewlett-Packard Company, "Managing Your Network with HP OpenView Network Node Manager," May 2002, manual J1240-90080.

[83] ——, "A Guide to Scalability and Distribution for Network Node Manager," https://community.opentext.com/cfs-file/__key/telligent-evolution-components-attachments/00-470-01-00-00-49-10-29/12227.pdf, Apr. 1997, HP OpenView; Edition 1, J1136-90001; accessed 2026-04-24.

[84] Cisco Systems, "Configuring IP Routing [Cisco IOS Software Releases 11.0]," https://www.cisco.com/en/US/docs/ios/11_0/access/configuration/guide/aciprout.html, 1994, documents policy routing in Cisco IOS Release 11.0; accessed 2026-04-17.

[85] "Netcool/OMNIbus Administration Guide," https://www.ibm.com/docs/en/SSSHTQ_8.1.0/pdf/omn_pdf_adm_master.pdf, IBM, 2005, Accessed on Sep 1, 2025.

[86] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 2, pp. 5–17, 1996.

[87] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[88] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," *ACM SIGCOMM computer communication review*, vol. 37, no. 4, pp. 1–12, 2007.

[89] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 3, pp. 105–110, 2008.

[90] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, p. 132, 2009.

[91] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[92] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," *ACM Sigplan Notices*, vol. 46, no. 9, pp. 279–291, 2011.

[93] "VMware to Acquire Nicira," https://www.globenewswire.com/news-release/2012/07/23/1327218/0/en/VMware-to-Acquire-Nicira.html, Accessed on Sep 1, 2025.

[94] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[95] Open Networking Foundation, "Open Networking Foundation Formed to Speed Network Innovation," https://opennetworking.org/news-and

-events/press-releases/onf-formed-to-speed-network-innovation/, Mar. 2011, press release, 2011-03-21.

[96] "OpenDaylight - The Open Source SDN Platform," https://www.opendaylight.org/, Accessed on Sep 1, 2025.

[97] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar *et al.*, "The design and implementation of open {vSwitch}," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, 2015, pp. 117–130.

[98] "Sonic foundation - linux foundation project," 2017, accessed: 2025-09-25.

[99] "Why Operators Will Deploy Virtualized CPE," https://www.sdxcentral.com/opinions/why-operators-will-deploy-virtualized-cpe, Accessed on Sep 1, 2025.

[100] "Network Functions Virtualization Product Guide — Red Hat OpenStack Platform," https://docs.redhat.com/en/documentation/red_hat_openstack_platform/10/html-single/network_functions_virtualization_product_guide/index, Accessed on Sep 1, 2025.

[101] "ISG NFV," https://www.etsi.org/technical-groups/nfv/, Accessed on Sep 1, 2025.

[102] ETSI, "Open Source MANO (OSM)," https://osm.etsi.org/, accessed 2025-09-01.

[103] Ansible Project, "ansible/ansible: Ansible is a radically simple it automation system," GitHub repository, 2026, accessed: 2026-03-22. [Online]. Available: https://github.com/ansible/ansible

[104] L. Rizzo, "netmap: A Novel Framework for Fast Packet I/O," in *Proceedings of the 2012 USENIX Annual Technical Conference (USENIX ATC)*, 2012.

[105] "Intel® DPDK," https://www.dpdk.org/, accessed: 2025-09-25.

[106] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The express data path: Fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th international conference on emerging networking experiments and technologies*, 2018, pp. 54–66.

[107] F. Parola, R. Procopio, R. Querio, and F. Risso, "Comparing user space and in-kernel packet processing for edge data centers," *ACM SIGCOMM Computer Communication Review*, vol. 53, no. 1, pp. 14–29, 2023.

[108] AT&T Inc., "At&t domain 2.0 vision white paper," https://www.att.com/Common/about_us/pdf/AT&T%20Domain%202.0%20Vision%20White%20Paper.pdf?msockid=194a0776ef646e21319a11d8eb646ca5, 2014, accessed: 2025-09-25.

[109] D. Cooperson and C. Chappell, "Telefónica's unica architecture strategy for network virtualisation," https://www.telefonica.com/es/wp-content/uploads/sites/4/2021/03/Telefonica_Virtualisation_gCTO_FINAL.pdf, 2017, accessed: 2025-09-25.

[110] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[111] B. Claise, B. Trammell, and P. Aitken, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," Tech. Rep., 2013.

[112] P. Phaal, S. Panchen, and N. McKee, "Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks," Tech. Rep., 2001.

[113] T. Zhang, L. Linguaglossa, M. Gallo, P. Giaccone, and D. Rossi, "Flowatcher-dpdk: Lightweight line-rate flow-level monitoring in software," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1143–1156, 2019.

[114] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, 1999, pp. 135–146.

[115] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in *Hot Interconnects VII*, vol. 40, 1999, pp. 34–41.

[116] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting," in *Proceedings of the ACM SIGCOMM Conference (SIGCOMM '03)*, 2003, pp. 213–224.

[117] F. Baboescu and G. Varghese, "Scalable packet classification," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 199–210, 2001.

[118] J. Daly, V. Bruschi, L. Linguaglossa, S. Pontarelli, D. Rossi, J. Tollet, E. Torng, and A. Yourtchenko, "Tuplemerge: Fast software packet processing for online packet classification," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1417–1431, 2019.

[119] X. Zhang, Q. Qiu, Z. Xu, P. He, X. Liu, K. Salamatian, C. Pei, and G. Xie, "Npc: Rethinking dataplane through network-aware packet classification," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 677–691. [Online]. Available: https://doi.org/10.1145/3718958.3750511

[120] C. Zheng, Z. Xiong, T. T. Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, "Iisy: Hybrid in-network classification using programmable switches," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2555–2570, 2024.

[121] C. Zheng and N. Zilberman, "Planter: seeding trees within switches," in *Proceedings of the SIGCOMM '21 Poster and Demo Sessions*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 12–14. [Online]. Available: https://doi.org/10.1145/3472716.3472846

[122] T. Swamy, A. Zulfiqar, L. Nardi, M. Shahbaz, and K. Olukotun, "Homunculus: Auto-generating efficient data-plane ml pipelines for datacenter networks," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 329–342. [Online]. Available: https://doi.org/10.1145/3582016.3582022

[123] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," *Discrete mathematics & theoretical computer science*, no. Proceedings, 2007.

[124] M. Yu, L. Jose, and R. Miao, "Software {Defined}{Traffic} measurement with {OpenSketch}," in *10th USENIX symposium on networked systems design and implementation (NSDI 13)*, 2013, pp. 29–42.

[125] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One sketch to rule them all: Rethinking network flow monitoring with univmon," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 101–114.

[126] Q. Huang, P. P. Lee, and Y. Bao, "Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 576–590.

[127] H. Namkung, Z. Liu, D. Kim, V. Sekar, and P. Steenkiste, "{SketchLib}: Enabling efficient sketch-based monitoring on programmable switches," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 743–759.

[128] Z. Zhou, J. Lv, L. Cheng, X. Chen, T. Zhang, Q. Huang, J. Luo, L. Zhu, D. Zhang, and C. Wu, "Sketchguide: Reconfiguring sketch-based measurement on programmable switches," in *2022 IEEE 30th International Conference on Network Protocols (ICNP)*. IEEE, 2022, pp. 1–11.

[129] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, L. J. Wobker, and N. McKeown, "In-band network telemetry via programmable dataplanes," in *Proceedings of the ACM SIGCOMM*, 2015, pp. 117–122.

[130] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "I know what your packet did last hop: Using packet histories to troubleshoot networks," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)*. USENIX Association, Apr. 2014, pp. 71–85.

[131] P. Tammana, R. Agarwal, and M. Lee, "Distributed network monitoring and debugging with switchpointer," in *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)*, 2018, pp. 453–456.

[132] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, and S. Licking, "Packet transactions: High-level programming for line-rate switches," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 15–28.

[133] R. B. Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "Pint: Probabilistic in-band network telemetry," in *Proceedings of the ACM SIGCOMM Conference (SIGCOMM '20)*, 2020, pp. 662–680.

[134] M. Dobrescu, K. Argyraki, and S. Ratnasamy, "Toward predictable performance in software {Packet-Processing} platforms," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 141–154.

[135] A. Tootoonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, and S. Shenker, "{ResQ}: Enabling {SLOs} in network function virtualization," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 283–297.

[136] A. Manousis, R. A. Sharma, V. Sekar, and J. Sherry, "Contention-aware performance prediction for virtualized network functions," in *Proceedings of the Annual conference of the ACM Special Interest*

Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, 2020, pp. 270–282.

[137] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2005, pp. 50–60.

[138] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the deployment of machine learning solutions in network traffic classification: A systematic survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.

[139] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 201–206.

[140] B. Anderson and D. McGrew, "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity," in *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*, 2017, pp. 1723–1732.

[141] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[142] E. Liang, H. Zhu, X. Jin, and I. Stoica, "Neural packet classification," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 256–269. [Online]. Available: https://doi.org/10.1145/3341302.3342221

[143] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[144] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.

[145] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," vol. 42, no. 5. New York, NY, USA: Association for Computing Machinery, Jul. 2008, p. 64–74. [Online]. Available: https://doi.org/10.1145/1400097.1400105

[146] K. Winstein and H. Balakrishnan, "Tcp ex machina: computer-generated congestion control," vol. 43, no. 4. New York, NY, USA: Association for Computing Machinery, Aug. 2013, p. 123–134. [Online]. Available: https://doi.org/10.1145/2534169.2486020

[147] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "Pcc vivace: Online-learning congestion control," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2018, pp. 343–356.

[148] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3050–3059. [Online]. Available: https://proceedings.mlr.press/v97/jay19a.html

[149] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, ser. HotNets '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 185–191. [Online]. Available: https://doi.org/10.1145/3152434.3152441

[150] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[151] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 191–205. [Online]. Available: https://doi.org/10.1145/3230543.3230551

[152] G. Bernárdez, J. Suárez-Varela, A. López, B. Wu, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Is machine learning ready for traffic engineering optimization?" in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–11.

[153] Z. Xu, F. Y. Yan, R. Singh, J. T. Chiu, A. M. Rush, and M. Yu, "Teal: Learning-accelerated optimization of wan traffic engineering," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 378–393.

[154] F. Gui, S. Wang, D. Li, L. Chen, K. Gao, C. Min, and Y. Wang, "Redte: Mitigating subsecond traffic bursts with real-time and distributed traffic engineering," in *Proceedings of the ACM SIGCOMM 2024 Conference*. ACM, 2024, pp. 71–85.

[155] C. Liu, H. Deng, V. Aggarwal, Y. Yang, and M. Xu, "Shooting large-scale traffic engineering by combining deep learning and optimization approach," *Proceedings of the ACM on Networking*, vol. 3, no. CoNEXT1, pp. 1–21, 2025.

[156] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 270–288. [Online]. Available: https://doi.org/10.1145/3341302.3342080

[157] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 101–115, 2021.

[158] L. Salaün, H. Yang, S. Mishra, and C. S. Chen, "A gnn approach for cell-free massive mimo," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 3053–3058.

[159] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 513–526.

[160] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 197–210.

[161] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," vol. 37, no. 4. New York, NY, USA: Association for Computing Machinery, Aug. 2007, p. 13–24. [Online]. Available: https://doi.org/10.1145/1282427.1282383

[162] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred, "Taking the blame game out of data centers operations with netpoirot," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 440–453. [Online]. Available: https://doi.org/10.1145/2934872.2934884

[163] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "Netbouncer: Active device and link failure localization in data center networks," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

[164] Q. Zhang, K. K. W. Ng, C. Kazer, S. Yan, J. a. Sedoc, and V. Liu, "Mimicnet: fast performance estimates for data center networks with machine learning," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 287–304. [Online]. Available: https://doi.org/10.1145/3452296.3472926

[165] Q. Yang, X. Peng, L. Chen, L. Liu, J. Zhang, H. Xu, B. Li, and G. Zhang, "Deepqueuenet: towards scalable and generalized network performance estimation with packet-level visibility," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 441–457. [Online]. Available: https://doi.org/10.1145/3544216.3544248

[166] C. Li, A. Nasr-Esfahany, K. Zhao, K. Noorbakhsh, P. Goyal, M. Alizadeh, and T. E. Anderson, "m3: Accurate flow-level performance estimation using machine learning," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 813–827. [Online]. Available: https://doi.org/10.1145/3651890.3672243

[167] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 3–10.

[168] J. Strassner, J. W.-K. Hong, and S. van der Meer, "The design of an autonomic element for managing emerging networks and services," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*. IEEE, 2009, pp. 1–8.

[169] S. Feng and E. Seidel, "Self-organizing networks (son) in 3gpp long term evolution," *Nomor Research GmbH, Munich, Germany*, vol. 20, pp. 1–15, 2008.

[170] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[171] TM Forum Autonomous Networks Project, "Gb1524a wireless network change management questionnaire v1.0.0," https://www.tmforum.org/resources/guidebook/gb1524a-wireless-network-change-management-questionnaire-v1-0-0/, Sep. 2025, accessed 2026-02-16.

[172] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[173] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[174] "ChatGPT," https://chatgpt.com/, Accessed on Sep 1, 2025.

[175] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[176] F. Jiang, C. Pan, K. Wang, P. Michiardi, O. A. Dobre, and M. Debbah, "From large ai models to agentic ai: A tutorial on future intelligent communications," *IEEE Journal on Selected Areas in Communications*, vol. 44, pp. 3507–3540, 2026.

[177] O. G. Lira, O. M. Caicedo, and N. L. da Fonseca, "Large language models for zero touch network configuration management," *IEEE Communications Magazine*, vol. 63, no. 7, pp. 146–153, 2024.

[178] Nokia, "Autonomous networks — nokia," https://www.nokia.com/autonomous-networks/, 2024, accessed: 2025-09-24.

[179] C. Finelli, A. Sharaf, B. Lo, and M. Sierra Marco, "Agentic AI for RAN Optimization: Pathway to Autonomous Network Level 5," https://aws.amazon.com/blogs/industries/agentic-ai-for-ran-optimization-pathway-to-autonomous-network-level-5/, Jul. 2025, accessed 2025-09-01.

[180] F. Barreros and A. Lucchesi, "One more step towards autonomous networks: self-healing in Brazil's 5G Core," https://www.telefonica.com/en/communication-room/blog/self-healing-in-the-core-5g-brazil/, Jul. 2025, published 2025-07-17; accessed 2025-09-01.

[181] G. Lenahan, "Agentic AI in the Autonomous Networks Control Loop," https://appledoreresearch.com/report/agentic-ai-autonomous-networks-control-loop/, Jun. 2025, published 15 June 2025.

[182] A. Krentsel, R. Iyer, I. Keslassy, B. Modhipalli, S. Ratnasamy, A. Shaikh, and R. Shakir, "CrossCheck: Input validation for WAN control systems," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 647–667. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/krentsel

[183] S. Renganathan, B. Rubin, H. Kim, P. L. Ventre, C. Cascone, D. Moro, C. Chan, N. McKeown, and N. Foster, "Hydra: Effective runtime network verification," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 182–194. [Online]. Available: https://doi.org/10.1145/3603269.3604856

[184] Q. Xiang, C. Huang, R. Wen, Y. Wang, X. Fan, Z. Liu, L. Kong, D. Duan, F. Le, and W. Sun, "Beyond a centralized verifier: Scaling data plane checking via distributed, on-device verification," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 152–166. [Online]. Available: https://doi.org/10.1145/3603269.3604843

[185] C. Zhao, Y. Guo, J. Wang, Q. Qi, Z. Zhuang, H. Sun, L. Guo, Y. Xie, and J. Liao, "EPVerifier: Accelerating update storms verification with Edge-Predicate," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 979–992. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/zhao

[186] Z. Li, P. Zhang, Y. Zhang, and H. Yang, "NDD: A decision diagram for network verification," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. Philadelphia, PA: USENIX Association, Apr. 2025, pp. 237–258. [Online]. Available: https://www.usenix.org/conference/nsdi25/presentation/li-zechun

[187] D. Wang, P. Zhang, W. Sun, W. Li, X. Feng, H. Li, J. Chen, W. Jiang, and Y. Tang, "S2: A distributed configuration verifier for hyper-scale networks," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 796–808. [Online]. Available: https://doi.org/10.1145/3718958.3750516

[188] R. Yang, G. Han, H. Shao, X. Zheng, X. Fang, Z. Wang, L. You, R. Zhou, L. Kong, E. Zhai, Q. Xiang, and J. Shu, "Diagnosing and repairing distributed routing configurations using selective symbolic simulation," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 1635–1652. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/yang

[189] Y. Yuan, F. Ye, Y. Li, J. Zhang, M. Liu, Y. Sang, R. Yang, D. She, Z. Ye, T. Guo, X. Zhu, X. Tang, L. Jia, Z. Guan, L. Su, C. Wang, R. Feng, S. Wu, Z. Xie, C. Jin, P. Zhang, Q. Ma, X. Zeng, D. Cai, and E. Zhai, "New evolution of hoyan: Enhancing scalability, usability, and accuracy for alibaba's global wan verification," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 809–825. [Online]. Available: https://doi.org/10.1145/3718958.3754343

[190] T. Schneider, S. Vissicchio, and L. Vanbever, "Verifying maximum link loads in a changing world," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. Philadelphia, PA: USENIX Association, Apr. 2025, pp. 1269–1287. [Online]. Available: https://www.usenix.org/conference/nsdi25/presentation/schneider

[191] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 323–334, 2012.

[192] A. Bierman and M. Bjorklund, "Network configuration access control model," Internet Engineering Task Force (IETF), Request for Comments 8341, Mar. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8341

[193] Z. Wang, S. S. Ahuja, X. Zhang, M. Noormohammadpour, G. R. Steinbrecher, T. Fuller, X. Liu, K. Quirk, M. J. Fernandez, A. Triguna, Y. Cai, S. Politis, P. Lapukhov, N. Hasani, and Y. Zhang, "Enabling AI network Cross-Layer design and operations with arcadia: A simulation platform at scale," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 1791–1807. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/wang-zhaodong

[194] A. Nelson, S. Rekhi, M. Souppaya, and K. Scarfone, "Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication 800-61 Rev. 3, 2025.

[195] C. Wang, X. Zhang, R. Lu, X. Lin, X. Zeng, X. Zhang, Z. An, G. Wu, J. Gao, C. Tian, G. Chen, G. Liu, Y. Liao, T. Lin, D. Cai, and E. Zhai, "Towards llm-based failure localization in production-scale networks," in *Proceedings of the ACM SIGCOMM 2025 Conference*, ser. SIGCOMM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 496–511. [Online]. Available: https://doi.org/10.1145/3718958.3750505

[196] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems*, ser. EuroSys '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 674–688. [Online]. Available: https://doi.org/10.1145/3627703.3629553

[197] R. Mondal, N. Bjorner, T. Millstein, A. Tang, and G. Varghese, "Tackling ambiguity in user intent for llm-based network configuration synthesis," in *Proceedings of the 24th ACM Workshop on Hot Topics in Networks*, ser. HotNets '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 176–183. [Online]. Available: https://doi.org/10.1145/3772356.3772402

[198] J. Liu, L. Chen, D. Li, and Y. Miao, "CEGS: Configuration example generalizing synthesizer," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. Philadelphia, PA: USENIX Association, Apr. 2025, pp. 1327–1347. [Online]. Available: https://www.usenix.org/conference/nsdi25/presentation/liu-jianmin

[199] R. Mondal, R. Singha, T. Millstein, G. Varghese, R. Beckett, and S. K. R. Kakarla, "Eywa: Automating Model-Based testing using LLMs," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 777–800. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/mondal

[200] A. Ramanathan, E. Kang, D. Han, and S. Abdu Jyothi, "Towards an agentic workflow for internet measurement research," in *Proceedings of the 24th ACM Workshop on Hot Topics in Networks*, ser. HotNets '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 61–68. [Online]. Available: https://doi.org/10.1145/3772356.3772409

[201] S. Sudhakara and P. Rajak, "Constrained network slice assignment via large language models," 2025. [Online]. Available: https://arxiv.org/abs/2512.00040

[202] H. Hè, M. Jin, and M. Apostolaki, "Making logic a First-Class citizen in generative ML for networking," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 801–824. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/he

[203] Y. Cai, J. Li, K. Akpinar, T. Li, H. Morsy, J. Wilson, S. Khaunte, Y. Xia, and Y. Zhang, "Matryoshka: Realizing hyperscale data center network design for the AI era," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI 26)*. Renton, WA: USENIX Association, May 2026, pp. 2095–2110. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/cai

[204] Z. Wan, R. Han, H. Sun, Q. Qi, Z. Zhuang, B. He, L. Zhang, J. Liao, and J. Wang, "NetKeeper: Enhancing network resilience with autonomous network configuration update on traffic patterns and anomalies," in *2025 USENIX Annual Technical Conference (USENIX ATC 25)*. Boston, MA: USENIX Association, Jul. 2025, pp. 715–730. [Online]. Available: https://www.usenix.org/conference/atc25/presentation/wan

[205] F. Haysom, "Bridging the Network Autonomy Chasm," Appledore Research white paper, Dec. 2024, accessed 2026-02-14. [Online]. Available: https://appledoreresearch.com/report/bridging-the-network-autonomy-chasm/

[206] M. Brown, A. Fogel, D. Halperin, V. Heorhiadi, R. Mahajan, and T. Millstein, "Lessons from the evolution of the batfish configuration analysis tool," in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 122–135. [Online]. Available: https://doi.org/10.1145/3603269.3604866

[207] Z. Gao, A. Abhashkumar, Z. Sun, W. Jiang, and Y. Wang, "Crescent: Emulating heterogeneous production network at scale," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1045–1062. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/gao-zhaoyu

[208] W. Sentosa, B. Chandrasekaran, P. B. Godfrey, and H. Hassanieh, "CellReplay: Towards accurate record-and-replay for cellular networks," in *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. Philadelphia, PA: USENIX Association, Apr. 2025, pp. 1169–1186. [Online]. Available: https://www.usenix.org/conference/nsdi25/presentation/sentosa

[209] ETSI, "Zero-touch network and Service Management (ZSM); Network Digital Twin," https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/015/01.01.01_60/gr_ZSM015v010101p.pdf, ETSI, ETSI GR ZSM 015 V1.1.1, Feb. 2024, accessed 2025-09-01.

[210] ——, "Zero-touch network and Service Management (ZSM); Network Digital Twin for enhanced zero-touch network and service management," ETSI, ETSI GS ZSM 018 V1.1.1, Dec. 2024, accessed 2026-02-14. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/018/01.01.01_60/gs_ZSM018v010101p.pdf

[211] B. E. Ujcich, A. Bates, and W. H. Sanders, "Provenance for intent-based networking," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 195–199.

[212] D. Guo, S. Chen, K. Gao, Q. Xiang, Y. Zhang, and Y. R. Yang, "Flash: fast, consistent data plane verification for large-scale network settings," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 314–335. [Online]. Available: https://doi.org/10.1145/3544216.3544246

[213] X. Xu, Y. Yuan, Z. Kincaid, A. Krishnamurthy, R. Mahajan, D. Walker, and E. Zhai, "Relational network verification," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ser. ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 213–227. [Online]. Available: https://doi.org/10.1145/3651890.3672238

[214] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114

[215] A. Wachi, X. Shen, and Y. Sui, "A survey of constraint formulations in safe reinforcement learning," 2024. [Online]. Available: https://arxiv.org/abs/2402.02025

[216] Y. Cai, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li, "Deep reinforcement learning for online resource allocation in network slicing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7099–7116, 2023.

[217] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein, "A general approach to network configuration analysis," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 469–483. [Online]. Available: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/fogel

[218] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "A general approach to network configuration verification," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 155–168. [Online]. Available: https://doi.org/10.1145/3098822.3098834

[219] Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 10 471–10 506. [Online]. Available: https://aclanthology.org/2024.findings-acl.624/

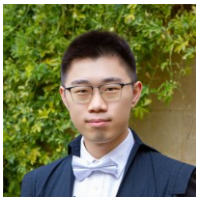
[220] TM Forum, "IG1453 Agent to Agent Protocol for Telecoms (A2A-T) v1.0.0," https://www.tmforum.org/resources/introductory-guide/ig1453-agent-to-agent-protocol-for-telecoms-a2a-t-v1-0-0/, Nov. 2025, accessed 2026-02-14.

[221] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," National Institute of Standards and Technology, NIST Special Publication 800-207, 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-207

[222] K. Mehmood, K. Kralevska, and D. Palma, "Intent-driven autonomous network and service management in future cellular networks: A structured literature review," *Computer Networks*, vol. 220, p. 109477, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622005114

[223] E. Dehghan Biyar, M. D'Angelo, J. C. n. Cisneros, A. Nascimento, M. Orlic, A. Likhyani, A. Zahemszky, A. C. Baktir, D. A. Temesgene, and D. Roeland, "Autonomous conflict handling in intent-based management," *Comput. Netw.*, vol. 271, no. C, Oct. 2025. [Online]. Available: https://doi.org/10.1016/j.comnet.2025.111561

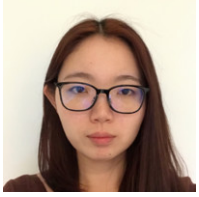
Tianzhu Zhang (Member, IEEE) received the B.S. degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2012, and the M.S. degree in computer and communication networks engineering from Politecnico di Torino, Turin, Italy, in 2014. From 2014 to 2017, he was a Ph.D. candidate in a joint program between Politecnico di Torino and Telecom Italia, supported by a Telecom Italia–PoliTo Ph.D. scholarship. From 2017 to 2019, he was a Postdoctoral Researcher at LINCS in France, supported by a research grant from Telecom ParisTech and Cisco Systems. He is currently a Research Scientist at Nokia Bell Labs and an Associate Member of LINCS. His research interests include software-defined networking, network function virtualization, high-speed network systems, AI/ML for networked systems, autonomous networking, and trustworthy network automation.



Changgang Zheng received the BEng degree in electronic and electrical engineering (first class Honours) from the University of Glasgow, Glasgow, U.K., and in communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, and the DPhil degree in engineering science from the University of Oxford, Oxford, U.K. He was a Postdoctoral Researcher with the Computing Infrastructure Group at the University of Oxford in 2024 and conducted research at Alibaba Cloud in 2025. His research interests include data center networks, space networks, AI computing networks, in-network computing, programmable network devices, and machine-learning-based approaches for efficient and scalable network systems. His work has appeared in venues including NSDI, ACM CoNEXT, IEEE/ACM Transactions on Networking, ACM SIGCOMM Computer Communication Review, and IEEE Communications Surveys & Tutorials.

Shanshan Wang (Member, IEEE) received the master's degree with distinction from the University of Bristol, Bristol, U.K., in 2014, and the Ph.D. degree in modeling wireless networks from L2S, CNRS, Université Paris-Saclay, France, in 2019. She is currently an Assistant Professor with the Chaire C2M, Télécom Paris, Institut Polytechnique de Paris, and is affiliated with the RFM^2 team in COMELEC. Before joining Télécom Paris, she was a Research Engineer at the Toshiba Telecommunication Laboratory in Bristol, U.K. After receiving her Ph.D., she worked as a Postdoctoral Researcher at Télécom Paris's Chaire C2M, focusing on AI-based electromagnetic field exposure mapping. From 2023 to 2024, she was an Assistant Professor with ETIS, CY Cergy Paris University. She has participated in several European Horizon projects, including 5GWireless, SEAWave, and GOLIAT. Her research interests include electromagnetic-field exposure assessment, AI-based prediction, stochastic geometry, system-level modeling of wireless networks, machine learning, and uncertainty quantification.

Yarui Zhang received the B.E. degree in electronic engineering from Xidian University, Xi'an, China, in 2018, and the M.S. and Ph.D. degrees in signal and image processing from Université Paris-Saclay, Gif-sur-Yvette, France, in 2019 and 2022, respectively. From 2023 to 2025, she was a Postdoctoral Researcher with the Chair C2M at Télécom Paris, France. She is currently an Assistant Professor with the SATIE Laboratory, École Normale Supérieure Paris-Saclay, France. Her research interests include inverse problems, computational imaging, and machine learning for wave-based problems.



Lina Shi (Member, IEEE) received an engineering degree in Computer Science and Electronics for Embedded Systems from the Université Grenoble Alpes, Grenoble, France, in 2017, followed by a Ph.D. in telecommunication from Sorbonne Université, Paris, France. Between 2021 and 2022, she was a postdoctoral researcher at UVSQ (Université Paris-Saclay), France. She is currently a researcher at Nokia Bell Labs France. Her research interests include performance estimation and monitoring in optical transmission systems, machine learning, optical communication, and sensing.

Yue Jin received the B.S. degree in automotive engineering and the M.S. degree in industrial engineering from Tsinghua University, Beijing, China, in 2000 and 2002, respectively, and the Ph.D. degree in industrial engineering and operations research from the University of Massachusetts Amherst, Amherst, MA, USA, in 2007. She is currently a Researcher in the Machine Learning and Systems Department at Bell Labs France, Nokia. Her research interests include optimization, stochastic processes, reinforcement learning, and continual learning.

Xiaofei Wang (Senior Member, IEEE) received the B.S. degree in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2005, and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University, Seoul, South Korea, in 2008 and 2013, respectively. From 2014 to 2016, he was a Postdoctoral Researcher with the School of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada. He is currently a Professor with the School of Computer Science and Technology, Tianjin University, China. His research interests include edge computing, edge intelligence and cloud-edge collaborative systems. He has authored or coauthored more than 220 technical papers in journals and conferences, including IEEE Communications Surveys & Tutorials, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Mobile Computing, IEEE Transactions on Services Computing, IEEE/ACM Transactions on Networking, IEEE Transactions on Knowledge and Data Engineering, IEEE INFOCOM, and so on.

Meikang Qiu (Senior Member, IEEE) received the B.E. and M.E. degrees from Shanghai Jiao Tong University, Shanghai, China, in 1992 and 1998, respectively, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2003 and 2007, respectively. He is currently a Professor in the School of Computer and Cyber Sciences at Augusta University, Augusta, GA, USA. His research interests include artificial intelligence, cybersecurity, cloud computing, big data analytics, embedded systems, Internet of Things, bioinformatics, and smart computing. He has published extensively in these areas, including books, journal articles, and conference papers. He has served as an Associate Editor for several international journals, including IEEE Internet of Things Journal, IEEE Transactions on Computers, IEEE Transactions on Cloud Computing, IEEE Transactions on Big Data, and IEEE Transactions on Systems, Man, and Cybernetics: Systems. He was selected as a Highly Cited Researcher by Web of Science in 2020 and as an IEEE Distinguished Visitor from 2021 to 2023. He received the IEEE Bio-inspired Computing Special Technical Community Life-Career Award in 2023. He is an ACM Distinguished Member.