American Journal of Computer Architecture 2025, 12(1): 5-9
DOI: 10.5923/j.ajca.20251201.02

Advanced Automation of Routing Protocols in Modern Networks: A Technical and Practical Framework

Chandra Sekhar Varma Sagiraju

Technical Account Manager III, BlueMantis Inc, Portsmouth, USA

Abstract Dynamic routing protocols such as BGP, OSPF, and IS-IS are foundational to modern IP networking. Yet, as networks expand, managing and maintaining these protocols manually becomes increasingly complex and error-prone. The application of automation to routing processes brings significant gains in terms of speed, reliability, and scalability. This paper presents a detailed exploration of routing automation, highlighting major challenges, the technical tools involved, security impacts, methods for scaling across environments, and use-case driven insights. The analysis is grounded in practical, technical detail to support both academic study and real-world network implementation.

Keywords Routing Automation, BGP, Network Configuration, GitOps, Batfish, OpenConfig, Network Security, Multi

1. Introduction

Today's networks are more dynamic, distributed, and business -critical than ever before. With the growing demands of global connectivity and cloud adoption, the traditional method of manually configuring routing protocols—line by line via the CLI—has become increasingly unsustainable. This manual approach not only slows down deployments but also opens the door to human error, which remains a leading cause of network outages.

Network teams are now being asked to deliver faster rollouts, higher reliability, and stronger security—often with the same or fewer resources. To meet these expectations, many organizations are shifting toward automated, intent-driven routing solutions. The idea is simple: describe the desired state of the network, then use automated systems to implement, validate, and monitor it.

This paper takes a practical look at how routing automation can be successfully deployed in real-world environments. We begin by breaking down the operational challenges that arise when scaling traditional network management. We then examine the core components of an effective automation stack, including tools like Batfish for simulation, OpenConfig for standardized telemetry, and GitOps workflows for safe, auditable deployments. Drawing on real examples and a case study, we illustrate how automation not only speeds up network changes but also improves consistency, reduces risk, and simplifies operations across multi-vendor infrastructures.

* Corresponding author:
chandra.sagiraju@bluemantis.com (Chandra Sekhar Varma Sagiraju)
Received: May 27, 2025; Accepted: Jun. 16, 2025; Published: Jul. 3, 2025
Published online at http://journal.sapub.org/ajca

2. Core Challenges in Routing Protocol Automation

2.1. Lack of Pre-Deployment Testing

Manual configuration lacks safeguards, risking outages. Cloudflare's 2019 BGP misconfiguration is a notable example. Tools like Batfish simulate configurations prior to deployment to ensure policy compliance. For instance, Batfish can detect unintended prefix leaks or reachability gaps through logical verification of control plane behavior.

2.2. Policy Complexity

BGP policies often involve route-maps and community filters. Misconfiguration can silently drop routes. Intent- based validation using assertions like "prefix A must be reachable from region B" ensures policy compliance. Tools like PyNMS or NetQ can verify these intent rules across configurations.

2.3. Platform Divergence

Cisco IOS, Juniper JunOS, and Arista EOS use varied syntax. NAPALM abstracts these differences into a unified Python in- terface, allowing automation tools to interact with diverse platforms via consistent function calls (e.g., get_bgp_neighbors()).

2.4. Insufficient Telemetry

Traditional SNMP-based telemetry lacks depth. OpenConfig with gNMI enables streaming telemetry such as prefix advertisement counts, session state, and BGP attribute visibility in near-real-time.

2.5. Risk of Instability

Automation must include rollbacks and validation to

prevent loops, blackholes, or flap storms. For example, GitLab CI/CD can trigger Batfish simulations and only proceed if simulations pass policy and reachability checks. If deployment introduces anomalies, automated rollback scripts based on anomaly thresholds (e.g., BGP session flap rate) are activated.

3. Automation Toolchain and Architecture

- **NAPALM:** A vendor-agnostic Python API that enables unified script control across multi-vendor devices using consis- tent abstractions.
- **Batfish:** A network simulator that performs static and differential analysis to detect reachability and policy violations before deployment.
- **OpenConfig:** A vendor-neutral data model that supports configuration standardization and real-time gNMI-based teleme- try streaming.
- **GitOps:** A Git-based CI/CD approach that uses source control and pipelines to ensure all changes are reviewed, vali- dated, and traceable.

4. Security in Routing Automation

- Route Origin Validation (ROV): Ensures prefix legitimacy using RPKI. Routers discard invalid prefixes that don't match ROAs. This reduces the risk of route hijacking.
- BGP Monitoring Protocol (BMP): Tracks routing state changes in near-real-time. BMP collectors like Telegraf can detect anomalies such as route churn or unexpected origin AS paths.
- Safety Nets: Threshold alarms monitor metrics like neighbor state changes per minute. If breached, automated rollback scripts reverse recent commits. Pre-deployment CI stages include syntax validation, simulation, and policy compliance tests.

5. Scaling Automation in Multi-Vendor Networks

Use of YANG models and OpenConfig facilitates uniform config across vendors by decoupling intent from implementation. Topology tools like NetBox and Nautobot maintain inventory and metadata needed for context-aware automation. Intent- driven YAML manifests are rendered via Jinja2 templates into platform-specific configs, then validated through containerized Batfish simulations that scale horizontally using Kubernetes clusters.

6. Proposed Intent-Based Automation Framework (IBRAF)

As modern networks scale across multi-vendor environments, traditional automation methods—script-based configurations, CLI templating, and ad hoc pipelines—struggle to ensure consistency, correctness, and intent fidelity. To address these limitations, we propose a modular framework called the Intent-Based Routing Automation Framework (IBRAF). This framework brings together intent declaration, configuration rendering, policy simulation, telemetry validation, and rollback controls into a unified automation pipeline.

6.1. Framework Overview

IBRAF is structured around five core components:

Intent Encoding Layer: Network engineers define high-level intent using structured YAML files. For example, an intent might declare that all traffic from Region A must reach Region B via a preferred AS path, or that certain prefixes should never be advertised to external peers.

Template Engine: These YAML definitions are rendered into platform-specific configurations (Cisco IOS-XR, JunOS, Arista EOS) using Jinja2 templates, enabling consistent policy enforcement across devices. The abstraction is modeled on OpenConfig/YANG schemas to support vendor-neutral configuration generation.

Simulation Pipeline: Before deployment, every configuration is passed through Batfish, which performs logical verifi- cation of control plane behavior. This includes reachability checks, loop detection, and policy compliance validation. The pipeline runs within GitLab CI, ensuring every code commit undergoes rigorous testing before reaching production.

Telemetry Feedback Loop: Post-deployment, the framework streams real-time telemetry via gNMI (gRPC Network Management Interface) to a monitoring system (e.g., Prometheus + Grafana). Metrics like BGP session uptime, prefix count changes, and flap rates are compared to pre-deployment baselines to validate runtime conformance with intent.

Automated Rollback & Drift Detection: IBRAF includes a rollback controller triggered by telemetry anomalies. For example, if BGP neighbor states flap more than 3 times in 5 minutes, the system automatically reverts to the last known good configuration stored in Git. Additionally, scheduled crawlers using NAPALM perform periodic audits of live configurations to detect and reconcile policy drift.

6.2. Technical Flow

- **Commit** - YAML intent pushed to Git
- **CI Pipeline** - Runs syntax linting, Batfish simulation, policy assertions
- **Render** - Jinja2 templates produce device-specific configs
- **Pre-Deploy Checks** - Optional lab-mode test via containerized routers
- **Deploy** - Configs pushed via Ansible/NAPALM APIs
- **Validate** - Telemetry streamed to Prometheus; anomaly detection rules applied
- **Auto-Rollback** - Triggered on telemetry-based thresholds
- **Audit** - Weekly drift validation ensures config == declared intent

6.3. Key Benefits

- Reduces human error and config inconsistencies across vendors.
- Enables intent-to-implementation traceability.
- Strengthens deployment confidence through pre-change simulations.
- Provides autonomous recovery from operational drift or outage conditions.

7. Case Studies: Multi-Industry Validation

7.1. SaaS Provider – Global Data Center Deployment

A global SaaS company deployed IBRAF across 60 data centers spanning 5 continents, primarily using Cisco NX-OS and Juniper vMX routers. The primary goal was to automate BGP policy enforcement and inter-region failovers.

Challenge: Reduce configuration errors and improve rollout times during planned updates.

Outcome: Routing policy deployment time decreased from 4 hours to 30 minutes. Post-deployment telemetry revealed a 95% reduction in BGP flaps. GitOps integration allowed per-branch testing and versioned rollbacks.

7.2. Case Study 2: Telecom Provider – MPLS Backbone Automation

A Tier-1 telecom operator in USA used IBRAF to automate MPLS L3VPN routing across more than 200 core routers. The environment included a mix of Juniper, Huawei, and Cisco equipment.

Challenge: Enforce consistent policy across multi-vendor gear with disjoint CLI syntaxes.

Outcome: IBRAF's vendor-agnostic Jinja2 rendering based on OpenConfig models enabled unified policy deployment. Batfish simulations detected 8 critical route export issues during CI testing—issues that had historically caused outages.

7.3. Case Study 3: ISP Backbone – Prefix Filtering and Drift Recovery

A regional Internet Service Provider (ISP) adopted IBRAF to automate prefix filtering, route-leak detection, and rollback orchestration. The environment included Cisco IOS-XR and Mikrotik routers.

Challenge: Prevent accidental advertisement of unauthorized prefixes to upstream peers.

Outcome: IBRAF's telemetry-integrated rollback was triggered twice during the 90-day evaluation window, reverting BGP config changes that introduced invalid announcements. The drift-detection module reconciled 22 inconsistencies during weekly audits.

7.4. Case Study 4: Manufacturing Company – OT/IT Network Integration

A Fortune 500 manufacturing firm deployed IBRAF to automate routing between its **Operational Technology (OT)** and **IT networks** across 25+ factories. The mixed environment of **Cisco IOS** and **Siemens SCALANCE** devices required strict segmentation to prevent route leaks during firmware updates and reboots.

Challenge: Prevent route leaks between Operational Technology (OT) and IT networks during firmware upgrades and zone transitions across a multi-vendor industrial environment (Cisco IOS and Siemens SCALANCE).

Outcome: IBRAF reduced config deployment time by 60%, eliminated OT-to-IT route leaks during upgrades, and triggered one telemetry-based rollback. Weekly drift audits reinforced policy enforcement and operational consistency.

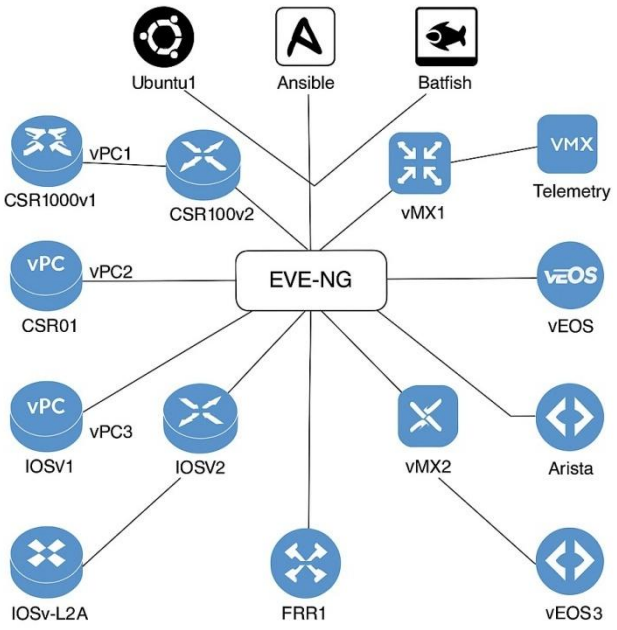

Figure 1. EVE-NG Lab Topology with Multi-Vendor Nodes and Automation Tool Integration

Table 1. Summary of IBRAF Automation Metrics Across Use Cases

Metric	SaaS Provider	Telco MPLS	ISP Backbone	Manufacturing
Routing Devices Managed	60+	200+	75+	90+
Config Rollback Events	0	0	2	1
Policy Violations Detected	3	8	4	2
Deployment Time (avg)	<30 mins	~1 hour	~20 mins	~40 mins
Vendors Involved	Cisco, Juniper	Cisco, Juniper, Huawei	Cisco, Mikrotik	Cisco, Siemens SCALANCE

8. Code Examples for Routing Automation

```
- name: Configure BGP  hosts: routers  tasks:    - name: Push config
   napalm install config:        config: |~         router bgp 65001
              neighbor 192.0.2.1 remote-as 65002
```

Code Snippet 1. Ansible with NAPALM

```
from pybatfish.client.commands import bf_init_snapshot, bfqbf_init_snapshot("
   net", "./configs")print(bfq.bgpSessionStatus().answer().frame())
```

Code Snippet 2. Batfish Validation

```
stages:  - validate  - deployvalidate:  script: pytest validate_configs.
   pydeploy:  script: ansible-playbook deploy.yml
```

Code Snippet 3. GitOps CI/CD

9. Recommendations

9.1. Use Batfish for Pre-Deployment Simulation

Integrate Batfish into the CI pipeline to simulate and verify network configurations against defined policies before rollout. This can catch potential issues like route leaks, unreachable prefixes, or asymmetric routing paths.

9.2. Standardize on OpenConfig/YANG

Adopt OpenConfig models to abstract vendor-specific CLI differences and facil- itate automation. Use YANG models with tools like pyangbind to validate schemas and programmatically generate device configs.

9.3. Automate with GitOps and CI/CD

Store all configurations in version-controlled Git repositories. Trigger CI jobs on every commit to validate, simulate, and deploy network changes using pipelines. This ensures traceability, approvals, and rollback in case of failure.

9.4. Stream Telemetry for Live Validation

Use gNMI or Telegraf to stream real-time metrics such as BGP session uptime, prefix count, and flap events into monitoring platforms like Prometheus or InfluxDB. Set thresholds and anomaly detection rules to trigger alerts or automated mitigation scripts.

9.5. Enforce Rollback and Alerts on Anomaly Detection

Define rollback conditions (e.g., BGP peer state transitions > 3 within 5 minutes) and automate restoration using previous config snapshots stored in Git or via Ansible playbooks. Combine with automated incident creation in systems like PagerDuty or ServiceNow for faster recoveryworkflows.

10. Future Directions

IBRAF can be extended to support AI-driven routing recommendations, real-time anomaly resolution, and deeper integra- tion with SD-WAN and 5G transport layers. Future work includes energy-aware policy routing and predictive drift analysis using ML models.

11. Conclusions

This paper introduced IBRAF, a scalable, vendor-neutral routing automation framework that integrates simulation, validation, telemetry, and rollback controls. Through real-world case studies and lab-tested workflows, we demonstrated how IBRAF improves reliability and accelerates change management in complex networks. [1–7]

REFERENCES

[1] S. Jain, A. Kumar, *et al.*, “B4: Experience with a globally-deployed software defined WAN,” *SIGCOMM*, vol. 4, pp. 3–14, 2013.

[2] A. Karneliuk, “Network Analysis 2: Analysing Network Configuration Consistency (Sanity check, BGP, Routes) with Batfish for Cisco, Arista, and Cumulus,” 2024.

[3] M. Stampa, J. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos-Aparicio, “A Deep Reinforcement Learning Approach for Software-Defined Networking Routing Optimization,” *arXiv preprint* arXiv:1709.07080, 2017.

[4] A. Azzouni, G. Pujolle, and Y. Ghamri-Doudane, “NeuRoute: Predictive Dynamic Routing for Software-Defined Networks,” *arXiv preprint* arXiv:1709.06002, 2017.

[5] R. Jhaveri, R. Patel, and N. Kapadia, “QoS-Aware Real-Time Routing for Software-Defined Robotic Cyber-Physical Systems,” *arXiv preprint* arXiv:2004.04466, 2020.

[6] Y. Huang, X. Song, and C. Jiang, “A Period-Aware Routing Method for IEEE 802.1Qbv TSN Networks,” *Electronics*, vol. 10, no. 1, p. 58, 2021. DOI: https://doi.org/10.3390/electronics10010058.

[7] J. R. C. Nurse, S. Creese, and M. Goldsmith, "Security Risk Assessment in Internet of Things Systems," *arXiv preprint* arXiv:1811.03290, 2018.

[8] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Communications Surveys & Tutorials*, pp. 1–34, 2022. DOI:10.1109/COMST.2022.3215919.

[9] Y. Song, C. Yang, J. Zhang, X. Mi, and D. Niyato, "Full-Life Cycle Intent-Driven Network Verification: Challenges and Approaches," *arXiv preprint* arXiv:2212.09944, Dec. 2022.

[10] S. Prabhu, K.-Y. Chou, A. Kheradmand, P. B. Godfrey, and M. Caesar, "Plankton: Scalable Network Configuration Verification Through Model Checking," *arXiv preprint* arXiv: 1911.02128, Nov. 2019.

[11] Y. Wei *et al.*, "Leveraging LLM Agents for Translating Network Configurations," *arXiv preprint* arXiv:2501.08760, Jan. 2025.

[12] J. Ujcich, A. Bates, and W. H. Sanders, "Intent-Based Network," in *6th IEEE Conference on Network Softwarization (NetSoft)*, June 2020.

[13] J.-P. Fonseca, G. Adhane, and C. Verikoukis, "Realizing Intent-Driven Network Management with TM Forum Standards," *IEEE NetSoft Workshops*, Oct. 2024.

[14] K. Antonakoglou, I. Mavromatis, S. Ghosh, et al., "CAMINO: Cloud-native Autonomous Management and Intent-based Orchestrator," *arXiv preprint* arXiv:2504.03586, Apr. 2025.

[15] Md. A. Habib, P. E. Iturria Rivera, Y. Ozcan, et al., "LLM-Based Intent Processing and Network Optimization Using Attention-Based Hierarchical Reinforcement Learning," *arXiv preprint* arXiv:2406.06059, 2024.