Building Production-Ready ML-Based Network Routing

Lessons from Real-World Platform Engineering Deployments

Rahul Tavva

Kairos Technologies Inc.

As traditional routing protocols struggle to meet the demands of modern distributed applications, machine learning approaches offer a promising alternative. This presentation shares practical insights from implementing ML-based routing in production environments, focusing on the technical architecture, deployment patterns, and operational considerations platform teams need to understand.

Agenda

01

Introduction to ML-Based Network Routing

Core concepts, value proposition, and current landscape

02

Technical Architecture & Components

Data pipelines, model training, integration patterns

03

ML vs. Traditional Routing Protocols

Decision-making, convergence behavior, and failure handling

04

Implementation & Deployment Patterns

Infrastructure requirements and integration approaches

05

Monitoring, Observability & Operations

Performance benchmarking, incident response, security

06

Real-World Case Studies

Deployment challenges, ROI measurements, lessons learned

07

Practical Guidance & Next Steps

Evaluation frameworks, tooling recommendations, adoption roadmap

This session is designed for platform engineers, SREs, and infrastructure architects considering ML-based routing technologies for their production environments. All insights are drawn from real-world implementation experiences across multiple industries.

The Evolving Network Routing Landscape

Why Traditional Routing Falls Short

Traditional routing protocols like BGP, OSPF, and EIGRP were designed decades ago for different network conditions and traffic patterns. They face significant limitations in modern environments:

Static Decision Models

Reliance on predefined metrics and weights that don't adapt to changing network conditions

Limited Context Awareness

Inability to incorporate application requirements, traffic patterns, or historical performance

Slow Convergence

Extended periods of suboptimal routing during network changes or failures

Manual Optimization

Heavy reliance on operator expertise for tuning and troubleshooting

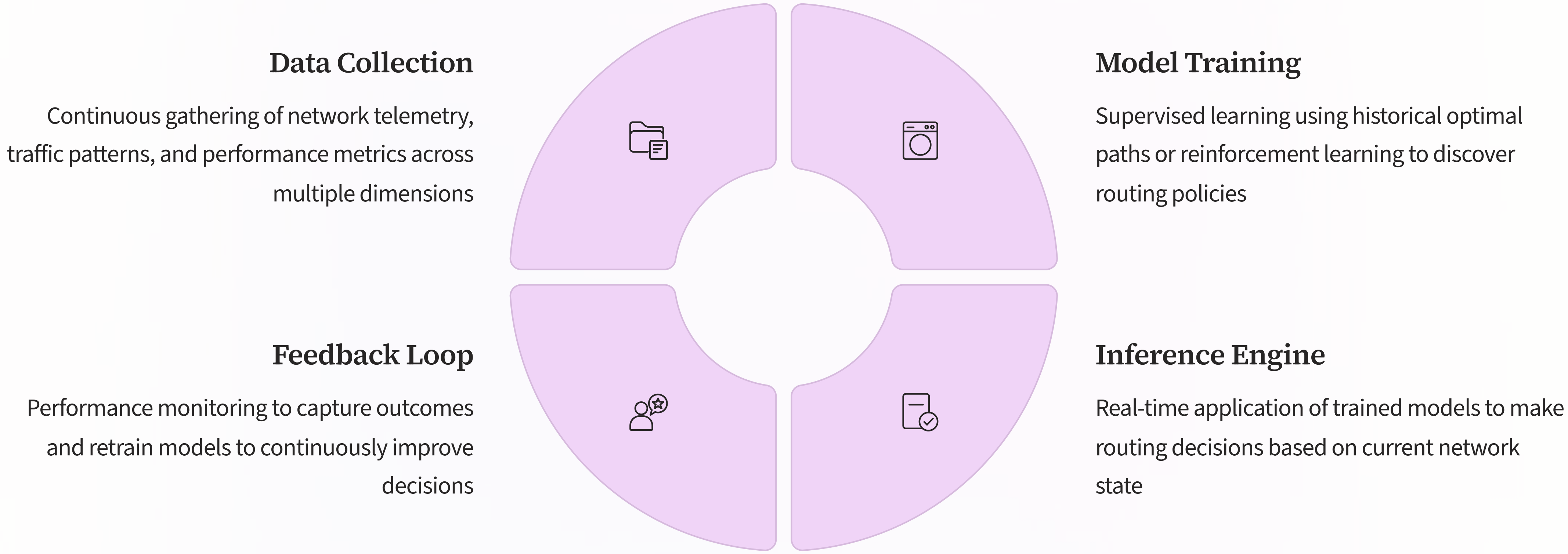
The explosion of east-west traffic in microservices architectures, coupled with increasing demands for low-latency performance and reliability, creates an environment where traditional routing protocols struggle to deliver optimal results.

The ML-Based Routing Value Proposition

Machine learning approaches to routing offer the potential to overcome these limitations by learning from historical patterns, adapting to changing conditions, and considering a much richer set of inputs when making routing decisions.

ML-Based Routing: Core Concepts

Unlike traditional protocols that use fixed algorithms, ML-based routing systems learn optimal paths based on dynamic network conditions and historical performance data.



Data Collection

Continuous gathering of network telemetry, traffic patterns, and performance metrics across multiple dimensions

Model Training

Supervised learning using historical optimal paths or reinforcement learning to discover routing policies

Feedback Loop

Performance monitoring to capture outcomes and retrain models to continuously improve decisions

Inference Engine

Real-time application of trained models to make routing decisions based on current network state

Key Differentiators

ML-based routing systems differ fundamentally from traditional protocols in how they approach the routing problem:

- **Predictive vs. Reactive:** ML models can anticipate congestion or failures before they impact performance
- **Multi-dimensional Optimization:** Balance latency, bandwidth, reliability, and cost simultaneously
- **Application-Aware:** Consider specific application requirements when making routing decisions
- **Continuous Learning:** Improve over time as they observe network behaviors and outcomes
- **Anomaly Detection:** Identify unusual patterns that may indicate security issues or misconfigurations
- **Adaptive Policies:** Automatically adjust to changing network conditions without manual intervention

These capabilities enable ML-based routing to potentially deliver significant improvements in network performance, reliability, and operational efficiency compared to traditional approaches.

Technical Architecture of ML-Based Routing Systems

Core Architectural Components

Telemetry Collection Layer

High-performance agents deployed across the network to gather real-time metrics including:

- Interface statistics (throughput, errors, drops)
- Path metrics (latency, jitter, packet loss)
- Traffic characteristics (protocols, applications, flow sizes)
- External data (weather, events, maintenance windows)

Data Processing Pipeline

Stream processing infrastructure that:

- Normalizes and correlates metrics from diverse sources
- Performs feature engineering to extract relevant signals
- Manages data retention policies for historical analysis
- Handles missing or inconsistent telemetry data

ML Training Infrastructure

Specialized systems for model development and training:

- Offline training environments for model development
- Feature stores for consistent model training
- Versioned model repository and experiment tracking
- Validation frameworks for model performance assessment

Inference & Control Systems

Components that apply ML decisions to actual routing:

- Low-latency inference engines deployed near routing decision points
- Adapters to translate ML outputs to routing protocol configurations
- Safety mechanisms to prevent harmful routing decisions
- Fallback systems for graceful degradation during failures

ML vs. Traditional Routing: Technical Comparison

Decision-Making Process

Traditional Protocols

- Fixed algorithms with predetermined metrics
- Limited input variables (hop count, bandwidth, delay)
- Deterministic outcomes based on current state only
- Manual policy configuration for traffic engineering

ML-Based Routing

- Probabilistic models trained on historical data
- Hundreds of potential input features considered
- Predictions based on patterns and historical outcomes
- Autonomous policy adjustment based on feedback

Convergence Behavior

Traditional Protocols

- Reactive response to topology changes
- Fixed convergence algorithms with set timers
- Potential for temporary routing loops
- Count-to-infinity problems in distance vector protocols

ML-Based Routing

- Potential to predict and pre-compute alternate paths
- Variable response based on confidence in predictions
- Can incorporate stability as an optimization goal
- May proactively reroute before failure occurs

Failure Handling

The approaches differ significantly in how they handle network failures:

Traditional Protocols

Rely on timeout-based detection, with recovery times ranging from seconds to minutes. Failure recovery paths are predetermined based on topology, often resulting in suboptimal performance during degraded conditions.

ML-Based Routing

Can leverage anomaly detection for faster failure identification. Models can be trained specifically on failure scenarios to optimize recovery. Capable of degraded-state optimization by considering partial failures and performance constraints.

These fundamental differences create both opportunities and challenges for teams implementing ML-based routing in production environments.

Data Collection & Model Training Approaches

Data Requirements for Effective ML Routing

Telemetry Types

- **Network-level metrics:** Link utilization, error rates, queue depths
- **Path characteristics:** End-to-end latency, jitter, packet loss
- **Traffic profiles:** Protocol distribution, flow sizes, application patterns
- **Infrastructure state:** CPU/memory utilization, temperature, power metrics
- **External factors:** Time of day, scheduled maintenance, weather events

Collection Considerations

- **Sampling rate:** Balance between detail and system overhead
- **Storage requirements:** Typically 30-90 days of historical data
- **Distribution:** Edge collection with central aggregation
- **Consistency:** Synchronized timestamps and normalization
- **Privacy:** Data anonymization and compliance considerations

Model Training Methodologies

Supervised Learning

Training on historical "optimal" paths identified retrospectively

Common algorithms: Random forests, gradient boosting, neural networks

Challenge: Requires labeled training data that may be difficult to generate

Reinforcement Learning

Models learn routing policies through exploration and feedback

Common algorithms: Deep Q-Networks, PPO, A3C

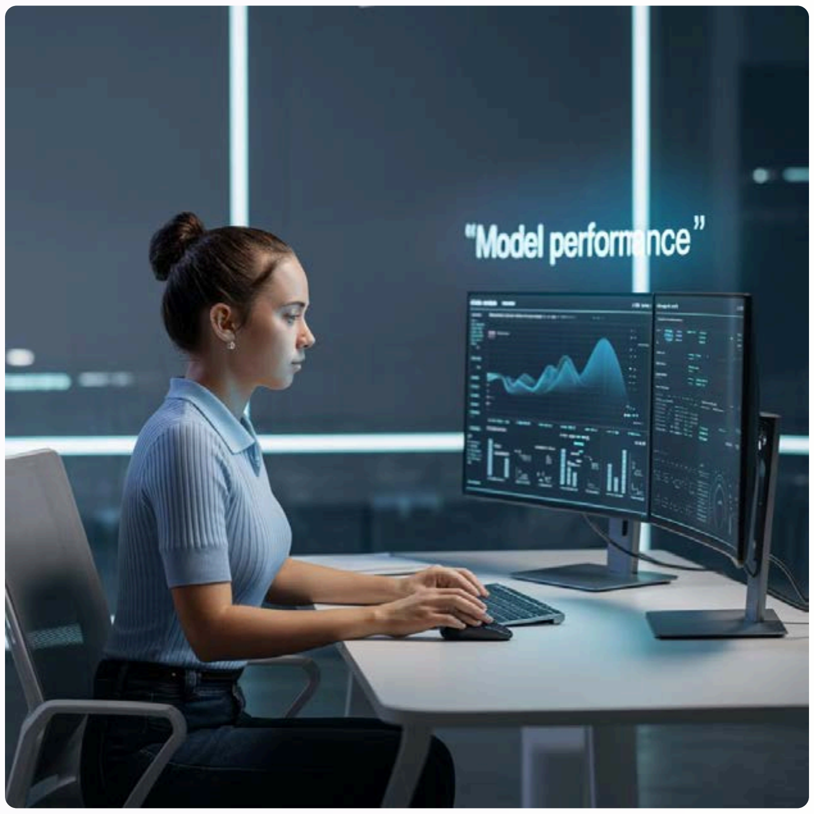
Challenge: Requires safe exploration mechanisms in production

Hybrid Approaches

Combining traditional routing rules with ML for specific optimizations

Common approach: ML for path ranking, traditional protocols for execution

Challenge: Defining clear boundaries between systems



Feature Engineering Best Practices

- Time-based features (moving averages, trends)
- Topological features (centrality, redundancy)
- Cross-metric correlations
- Domain-specific transformations

Production Deployment Patterns

Common Integration Approaches

Shadow Mode

ML system runs alongside traditional routing but decisions are only logged, not implemented. Allows for performance comparison without operational risk.

Best for: Initial validation and model tuning

Duration: Typically 4-8 weeks

Advisory Mode

ML system generates routing recommendations that operators can manually review and apply. Provides human oversight while leveraging ML insights.

Best for: Building operational confidence

Duration: 2-6 months

Selective Automation

ML system directly controls routing for specific traffic classes or network segments, while traditional protocols handle the rest.

Best for: Targeted optimization of critical traffic

Duration: Ongoing operational model

Full Automation

ML system makes all routing decisions with traditional protocols serving only as fallback mechanisms during system failures.

Best for: Mature deployments with proven reliability

Duration: End-state for most deployments

Infrastructure Requirements

Compute Resources

- **Training infrastructure:** High-performance GPU/TPU clusters for model development (can be cloud-based)
- **Inference endpoints:** Distributed computing resources near routing decision points
- **Data processing:** Stream processing infrastructure for real-time telemetry
- **Storage:** Time-series databases for metrics, object storage for training data

Network Considerations

- **Telemetry bandwidth:** Dedicated collection paths to avoid interference
- **Control plane capacity:** Sufficient bandwidth for routing updates
- **Out-of-band management:** Separate control paths for system recovery
- **Latency requirements:** Decision time budgets typically 10-100ms

Successful deployments typically follow a phased approach, gradually increasing the scope and autonomy of the ML system as confidence builds. This requires careful planning of infrastructure scaling to support growing demands.

Monitoring & Observability Challenges

Unique Monitoring Requirements

Model Performance Metrics

- Prediction accuracy against ground truth
- Inference latency and throughput
- Feature importance drift over time
- Model confidence scores per decision

Data Quality Monitoring

- Telemetry completeness and consistency
- Feature distribution shifts
- Missing or anomalous input signals
- Feedback loop integrity

System Health Indicators

- End-to-end decision pipeline latency
- Fallback activation frequency
- Model version distribution across network
- Training/serving skew metrics

Recommended Visualization Approaches

Effective dashboards for ML routing systems typically include:

- Side-by-side comparisons with traditional routing decisions
- Confidence interval visualization for predictions
- Feature importance heat maps
- Decision tree path visualization for interpretability
- Network topology maps with ML-influenced routing overlays

Performance Benchmarking Methodologies

Measuring the effectiveness of ML-based routing requires specific benchmarking approaches:

- ### A/B Testing

  Directing identical traffic through ML-routed and traditionally-routed paths to compare performance metrics such as latency, jitter, and packet loss.

- ### Synthetic Workloads

  Generating controlled traffic patterns to test system response to specific conditions, including flash crowds, microbursts, and failure scenarios.

- ### Historical Replay

  Simulating how the ML system would have routed traffic during past network events, comparing against actual outcomes from traditional routing.

Security Considerations for ML Routing

Threat Modeling for Intelligent Network Systems

Data Poisoning Attacks

Adversaries manipulating telemetry data to influence routing decisions

Mitigation: Input validation, anomaly detection on telemetry, diversity of data sources

Model Extraction

Inferring model behavior through systematic probing to identify exploitable patterns

Mitigation: Rate limiting probes, detecting unusual query patterns, model obfuscation

Control Plane Hijacking

Unauthorized access to ML inference systems to manipulate routing decisions

Mitigation: Strong authentication, encrypted control channels, behavior monitoring

Adversarial Examples

Crafting network conditions that trick ML models into making specific routing decisions

Mitigation: Adversarial training, ensemble models, decision bounds checking

Essential Security Controls

Data Protection

- Encrypted telemetry collection channels
- Data anonymization for sensitive traffic information
- Access controls on historical training data
- Secure deletion policies for obsolete data

System Security

- Model signing and verification
- Secure model deployment pipelines
- Least privilege access to ML systems
- Comprehensive audit logging of all decisions
- Regular penetration testing of ML infrastructure

Safety Mechanisms

Beyond security, ML routing systems require additional safety measures:

Decision Bounds

Enforcing limits on how drastically ML systems can change routing policies, particularly for critical traffic paths

Fallback Mechanisms

Automatic reversion to traditional routing when ML confidence falls below thresholds or anomalous behavior is detected

Human Circuit Breakers

Emergency override capabilities for operators to disable ML routing during incidents or unexpected behaviors

Operational Considerations

Incident Response Procedures

Detection

Specialized monitoring for ML-specific failure modes:

- Model confidence dropping below thresholds
- Unexpected routing pattern changes
- Telemetry pipeline disruptions
- Divergence between model versions

Containment

Targeted response options:

- Gradual traffic shifting away from ML routes
- Model version rollback capabilities
- Feature isolation to disable problematic inputs
- Partial to full fallback to traditional routing

Resolution

Recovery processes:

- Progressive reintroduction of ML routing control
- Post-incident model retraining
- Telemetry validation and reconstruction
- Confidence testing before full restoration

Common Operational Challenges

Model Drift

Performance degradation as network conditions evolve away from training data

Solution: Continuous retraining pipelines with automated evaluation

Explainability

Difficulty understanding why specific routing decisions were made

Solution: Feature importance tracking, decision path visualization tools

Team Skills

Hybrid expertise requirements spanning networking and ML domains

Solution: Cross-training programs, specialized team structures

Runbook Essentials

Well-documented operational procedures are critical for ML routing systems. Key runbooks should include:

- **Model deployment and rollback procedures** with specific validation steps
- **Feature isolation protocols** to disable problematic telemetry inputs
- **Traffic gradual shifting plans** for safely testing new models
- **Emergency override procedures** for complete fallback to traditional routing
- **Recovery checklists** for systematically restoring ML routing after incidents
- **Data pipeline recovery procedures** for handling telemetry disruptions

Real-World Case Studies: Successes and Challenges

Case Study: Financial Services Provider

Implementation Details

- Multi-region deployment across 12 global data centers
- Focus on latency-sensitive trading application traffic
- Gradient boosting models with 5-minute retraining cycles
- Integration with existing SD-WAN infrastructure

Results

- 68% reduction in 99th percentile latency spikes
- 42% decrease in path switching events during peak hours
- 23% improvement in overall bandwidth utilization
- Estimated $3.2M annual savings from reduced circuit costs

Challenges Overcome

- Initial resistance from networking team due to "black box" concerns
- Regulatory compliance issues with traffic metadata collection
- Integration complexity with legacy MPLS infrastructure

Case Study: E-commerce Platform

Implementation Details

- Hybrid cloud environment spanning 5 public cloud regions
- Reinforcement learning approach for dynamic traffic management
- Specialized optimization for holiday shopping traffic patterns

Results

- 31% reduction in average page load times during traffic spikes
- 52% decrease in cross-region data transfer costs
- 95% reduction in manual traffic engineering interventions

Challenges Overcome

- Significant false positives during initial deployment
- Model performance degradation during flash sales events
- Cloud provider API rate limiting affecting telemetry collection

Common Implementation Pitfalls

Insufficient Training Data Diversity

Models trained primarily on normal conditions perform poorly during unexpected events. Solution: Synthetic data generation and chaos engineering to create diverse training scenarios.

Overoptimization For Specific Metrics

Focusing too narrowly on single metrics like latency can create unexpected side effects. Solution: Multi-objective optimization with balanced weighting across key performance indicators.

Inadequate Telemetry Infrastructure

Underestimating the scale of data collection needs leads to incomplete models. Solution: Start with overprovisioned telemetry infrastructure and right-size after understanding actual requirements.

ROI Assessment & Risk Management

Quantifying the Business Impact

15-40%

Bandwidth Utilization Improvement

Typical increase in effective circuit utilization through more intelligent traffic distribution

20-60%

Latency Reduction

Typical decrease in 95th percentile application latency during normal operations

30-80%

MTTR Reduction

Typical reduction in mean time to recover from network disruptions

40-70%

Operations Efficiency

Typical reduction in manual network engineering interventions

Cost Components to Consider

- **Implementation costs:**
  - Infrastructure for training and inference
  - Telemetry system enhancements
  - Integration development effort
- **Operational costs:**
  - Ongoing model maintenance
  - Specialized skill development
  - Increased monitoring complexity
- **Risk mitigation costs:**
  - Fallback system maintenance
  - Additional testing environments
  - Security controls specific to ML

Risk Assessment Framework

Technical Risks

- **Model performance degradation** due to changing network conditions
- **Inference system failures** affecting routing decisions
- **Telemetry collection disruptions** creating data gaps
- **Integration points** with existing routing infrastructure

Mitigation: Robust testing, staged rollout, comprehensive fallbacks

Operational Risks

- **Skills gap** in ML network operations
- **Troubleshooting complexity** for hybrid routing environments
- **Incident response readiness** for new failure modes
- **Change management** processes for model updates

Mitigation: Cross-training, specialized tooling, enhanced runbooks

Business Risks

- **Extended outages** due to novel failure scenarios
- **Unpredictable performance** affecting critical applications
- **Resource contention** with other ML initiatives
- **Vendor lock-in** with specialized ML routing platforms

Mitigation: SLAs, phased rollout, open standards adoption

A comprehensive ROI analysis should balance quantifiable performance improvements against implementation costs and risk factors, considering both immediate benefits and long-term strategic advantages.

Implementation Roadmap & Best Practices

Phased Deployment Strategy

Phase 1: Foundation (3-6 months)

- Implement comprehensive telemetry collection infrastructure
- Establish baseline performance metrics for existing routing
- Develop initial ML models in isolated lab environment
- Create monitoring dashboards for model performance tracking

Phase 2: Shadow Mode (2-3 months)

- Deploy ML routing system in parallel with traditional routing
- Compare recommendations against actual routing decisions
- Refine models based on observed discrepancies
- Validate telemetry completeness and quality

Phase 3: Limited Production (3-4 months)

- Apply ML routing to non-critical traffic segments
- Implement canary deployment for select applications
- Develop operational runbooks and incident response procedures
- Train operations team on new tools and dashboards

Phase 4: Full Deployment (4-6 months)

- Gradually expand ML routing to all traffic classes
- Implement automated retraining and model deployment pipelines
- Optimize infrastructure based on production requirements
- Develop long-term governance and maintenance processes

Team Structure & Skills

Recommended Team Composition

Successful ML routing implementations typically require a cross-functional team with diverse expertise:

Network Engineering

- Deep understanding of existing routing protocols
- Traffic engineering experience
- Network telemetry expertise

Data Science

- ML model development and evaluation
- Feature engineering for network data
- Time-series analysis experience

Platform Engineering

- Data pipeline development
- ML operations infrastructure
- CI/CD for model deployment

Operations

- Network monitoring expertise
- Incident response experience
- Runbook development skills

Key Skill Development Areas

Organizations implementing ML routing should focus training and hiring efforts on:

- Network telemetry systems and protocols
- ML model interpretation and debugging
- Time-series analysis techniques
- ML system observability practices
- Network simulation and digital twin concepts
- Graph-based machine learning approaches

Conclusion & Next Steps

Key Takeaways

ML routing is production-ready for specific use cases

While not a universal replacement for traditional protocols, ML-based routing has demonstrated significant value in production environments, particularly for performance-sensitive applications and complex multi-path networks.

Hybrid approaches offer the best near-term value

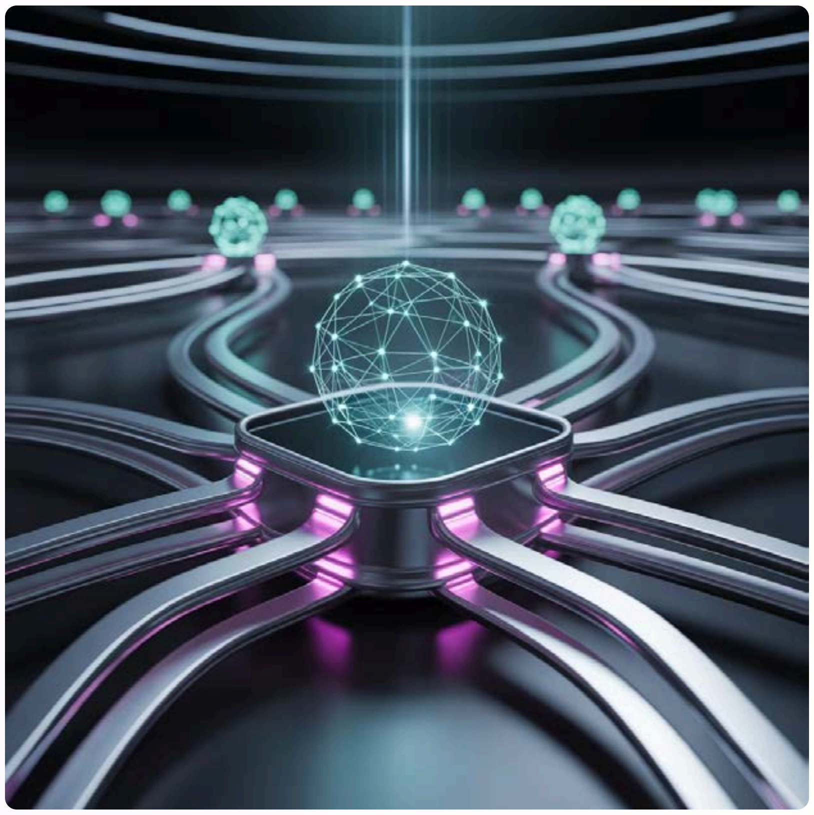
Most successful implementations combine ML-based decision-making with traditional routing protocols as execution and fallback mechanisms, leveraging the strengths of both approaches.

Operational considerations are as important as technical ones

Building effective operational processes, monitoring systems, and team capabilities is equally critical to success as the underlying ML technology and network integration.

Start small, but design for scale

Phased implementations with careful validation at each stage have proven most successful, but initial architecture should anticipate eventual network-wide deployment requirements.



Recommended First Steps

1. Assess your current network telemetry capabilities and identify gaps
2. Identify specific use cases with clear performance metrics for initial validation
3. Develop a small-scale proof of concept in a lab or non-critical environment
4. Build cross-functional team capabilities through training and partnerships
5. Establish baseline metrics for evaluating ML routing performance against current solutions

The Future of Network Routing

ML-based routing represents a fundamental shift in how networks will be managed and optimized in the coming years. As these technologies mature, we can expect:

Greater Integration

ML routing capabilities will become standard features in commercial networking equipment and cloud platforms

End-to-End Optimization

ML systems will coordinate across application, network, and infrastructure layers for holistic performance

Autonomous Operations

Networks will increasingly self-tune, self-heal, and adapt to changing conditions with minimal human intervention

Platform engineering teams that develop expertise in ML-based networking technologies today will be well-positioned to lead this transformation and deliver significant value to their organizations.