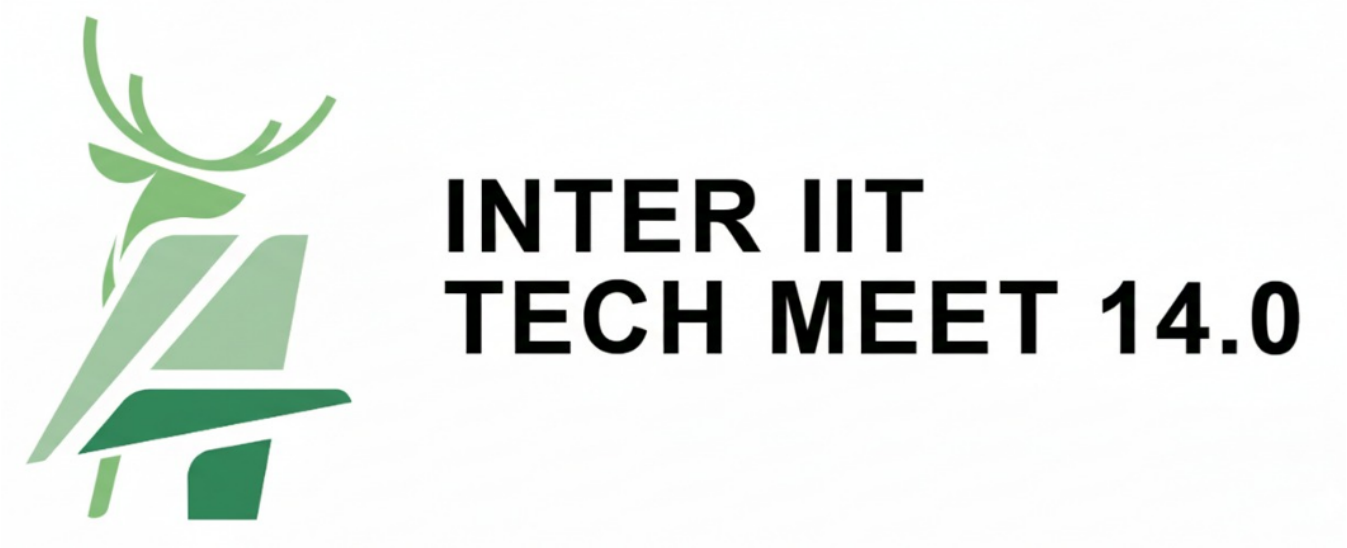

Client Centric WiFi RRM

INTER IIT TECH MEET 14.0

RRM-Plus Complete Architecture

Team 33

ARISTA

Contents

1 Executive Summary

This document presents the comprehensive architecture design for **RRM-Plus**, an AI-assisted, client-centric Radio Resource Management system developed for Arista Networks' enterprise WiFi infrastructure. The system represents a paradigm shift from traditional AP-centric RRM approaches by leveraging:

- **Graph Neural Networks (GNNs)** for interference topology learning
- **Safe Reinforcement Learning** with Conservative Q-Learning and production guardrails
- **Multi-timescale control loops** (fast/slow/event-driven)
- **Causal inference** for reward validation against counterfactual holdouts
- **NS-3 simulation** with realistic 5GHz noise modeling

Key Innovation: Unlike traditional RL systems that rely on simple delta-QoE rewards, RRM-Plus employs **causal reward validation** through difference-in-differences estimation. This ensures that observed QoE improvements are causally attributable to RL actions rather than confounding factors (time-of-day effects, client mobility, external interference).

Performance Targets:

- +25-35% median throughput for edge clients
- -30% P95 retry rate network-wide
- <0.2 configuration changes per AP per day (stability)
- >90% BSS-TM steering acceptance

2 System Overview

2.1 Architecture Philosophy

RRM-Plus adopts a **hybrid control architecture** that combines:

1. **Physics-based simulation** (NS-3) for realistic network dynamics
2. **Graph-structured learning** (GNN) for spatial interference relationships
3. **Safe offline RL** (Conservative Q-Learning) for policy optimization
4. **Production guardrails** (change budgets, blast-radius control)
5. **Causal validation** (difference-in-differences, propensity matching)

This architecture describes both the NS-3 simulation pipeline and its intended real-world deployment counterpart. Sections 3–6 refer to the simulation system, while Section 7 outlines the production-level architecture.

3 Component Architecture

3.1 NS-3 Simulation Layer

File: `basic-rrm-phase23_9th_nov.cc`

The NS-3 simulation forms the foundation of RRM-Plus, providing:

- **802.11ax PHY/MAC simulation** with OFDMA, MU-MIMO, spatial reuse
- **Realistic propagation models** (ITU Indoor, log-distance)
- **5GHz noise floor simulation** with temporal dynamics
- **Client mobility and traffic patterns**
- **Interference graph export** (adjacency + coupling strengths)

3.1.1 Time-Based Graph Export

Every 30 seconds, NS-3 invokes the Python controller:

```
bool DataGenerationController::InvokeGraphColoring(...) {
    Time now = Simulator::Now();
    double currentTimeSeconds = now.GetSeconds();
    int currentTimeInt = static_cast<int>(std::round(currentTimeSeconds));

    // Enable causal analysis every 30s
    bool enableCausal = (currentTimeInt % 30 == 0);

    cmd << " --nodes " << nodesFile;
    cmd << " --edges " << edgesFile;
    cmd << " --flag-causal " << (enableCausal ? "true" : "false");
    cmd << " --num-dreams 5";

    system(cmd.str().c_str());
}
```

3.1.2 Multi-Timescale Control Implementation

Table 1: Control Loop Timescales

Loop Type	Period	Triggers	Actions
Fast	5-15s	DFS events, Noise spikes	Channel change, OBSS-PD
Slow	30-60s	AI optimization	Power, Width, Channel
Event	Async	Microwave, BLE bursts	Transient mitigation

3.2 Preprocessing Pipeline

File: `preprocessing.py`

Transforms raw CSV exports into PyTorch Geometric `Data` objects:

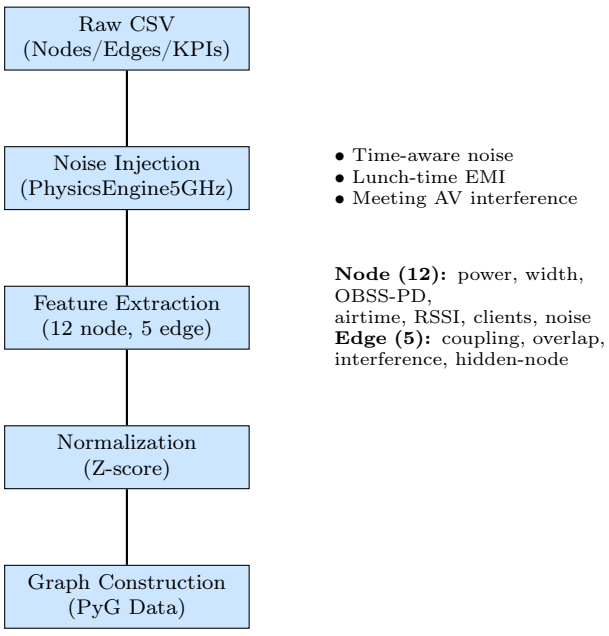

Figure 2: Preprocessing pipeline with 5GHz noise modeling

3.2.1 Noise Floor Modeling

The `noise.py` module simulates realistic 5GHz office environments:

$$\eta(t, \text{ap\_id}) = \eta_{\text{base}}(t) + \eta_{\text{lunch}}(t) + \eta_{\text{AV}}(t) + \eta_{\text{AP\_bias}} + \xi(t) \tag{1}$$

$$\eta_{\text{base}}(t) = -90 \text{ dBm} \quad \text{(workday plateau)} \tag{2}$$

$$\eta_{\text{lunch}}(t) = 5 \cdot \mathcal{N}(t|\mu = 12.5, \sigma = 0.4) \quad \text{(microwave spikes)} \tag{3}$$

$$\eta_{\text{AV}}(t) \sim \text{Bernoulli}(p = 0.2) \cdot \mathcal{U}(5, 8) \text{ dBm} \quad \text{(wireless casting)} \tag{4}$$

3.3 Graph Neural Network (GNN) Model

File: `model.py`

Architecture: **GraphSAGE** with attention-based aggregation

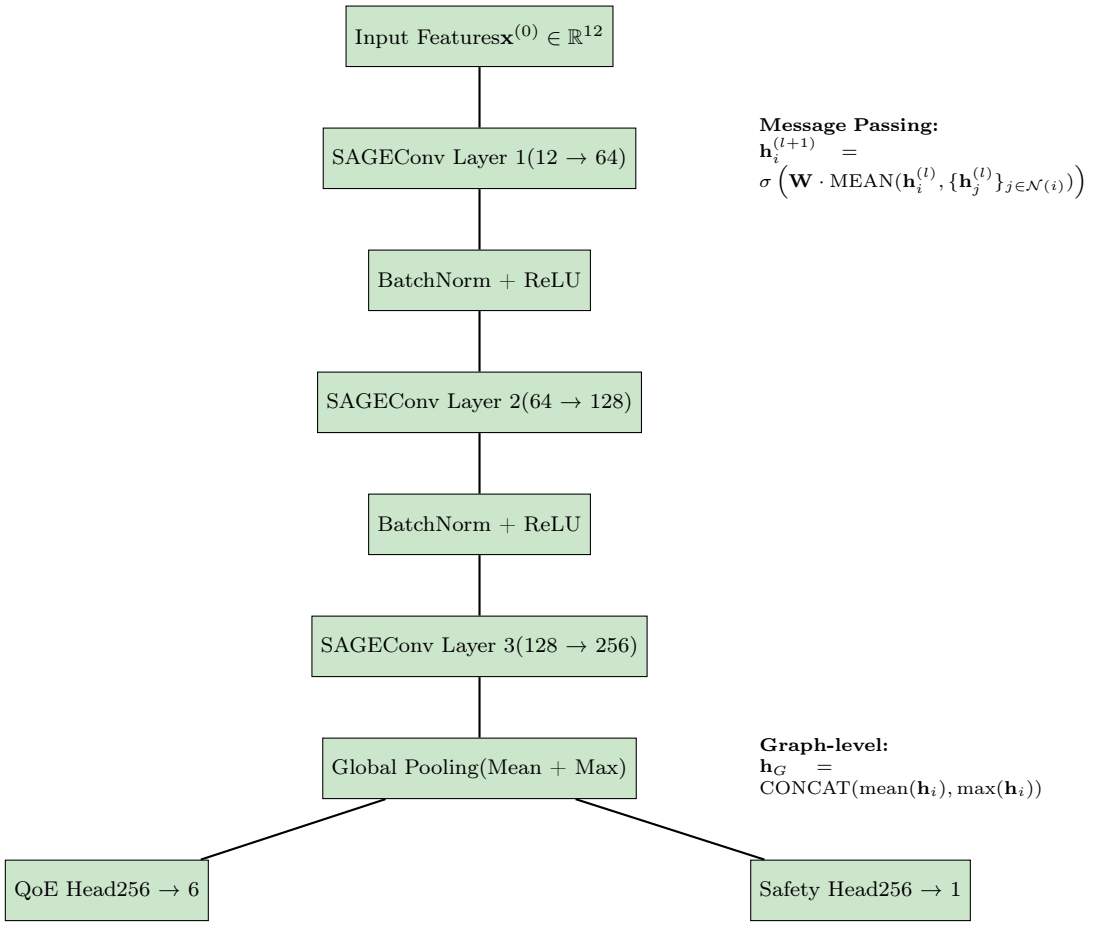

Figure 3: GNN architecture for learning interference topology

Key Design Choices:

- **GraphSAGE** over GCN: better generalization to unseen topologies
- **Multi-layer aggregation:** captures 3-hop interference relationships
- **Dual prediction heads:** simultaneous QoE and safety estimation
- **Residual connections:** prevent gradient vanishing in deep graphs

3.4 Safe Reinforcement Learning Agent

File: `agent_v2.py`

Implementation: **Conservative Q-Learning (CQL)** with Lagrangian safety constraints

3.4.1 Action Space

Discrete action space: $\mathcal{A} = \mathcal{A}_{\text{power}} \times \mathcal{A}_{\text{OBSS}} \times \mathcal{A}_{\text{width}}$

- $\mathcal{A}_{\text{power}} = \{1, 3, 5, \ldots, 30\}$ dBm (15 values)
- $\mathcal{A}_{\text{OBSS}} = \{-92, -82, -72, -62, -52\}$ dBm (5 values)
- $\mathcal{A}_{\text{width}} = \{20, 40, 80\}$ MHz (3 values)

Total actions: $|\mathcal{A}| = 15 \times 5 \times 3 = 225$

3.4.2 CQL Loss Function

$$\mathcal{L}_{\text{CQL}}(\theta) = \alpha \cdot \underbrace{\mathbb{E}_{s\sim\mathcal{D}}\left[\log \sum_a \exp Q_\theta(s,a) - \mathbb{E}_{a\sim\pi_\beta}[Q_\theta(s,a)]\right]}_{\text{Pessimism penalty}} \tag{5}$$

$$+ \underbrace{\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[(Q_\theta(s,a) - (r + \gamma \max_{a'} Q_{\theta'}(s',a')))^2\right]}_{\text{Bellman error}} \tag{6}$$

where:

- α: CQL coefficient (controls pessimism level)
- π_β: behavior policy (offline dataset)
- $\mathcal{D}$: offline replay buffer

3.4.3 Safety Constraint

Lagrangian formulation for safety-aware RL:

$$\max_\pi \quad \mathbb{E}_\pi[R(s,a)] \tag{7}$$

$$\text{s.t.} \quad \mathbb{E}_\pi[C(s,a)] \leq d_{\text{safety}} \tag{8}$$

Implemented via dual gradient ascent on λ (safety multiplier):
C(s, a) represents predicted safety violations (e.g., airtime > threshold, latency > limit), as predicted by the Safety Head.

```
# Update Q-functions
q_r_loss = cql_loss_reward + bellman_loss_reward
q_c_loss = cql_loss_cost + bellman_loss_cost

# Update safety multiplier (lambda)
cost_violation = q_c_pred.mean() - self.safety_threshold
lambda_loss = -self.lambda_param * cost_violation
lambda_new = max(0.0, self.lambda_param + self.lambda_lr * cost_violation)
```

3.5 Dream-Based Offline RL

Key Innovation: Generate synthetic transitions using the GNN as a world model

Algorithm 1 Dream Sample Generation

Input: Current graph G_t, GNN model f_θ, action a
Output: Dream transition (s_t, a, r, s_{t+1})

$\mathbf{h}_t \leftarrow f_\theta(G_t)$ ▷ GNN embedding
$\hat{y}_{\text{current}} \leftarrow \text{QoE\_head}(\mathbf{h}_t)$ ▷ Current KPIs

$G_{\text{dream}} \leftarrow \text{ApplyAction}(G_t, a)$ ▷ Modify node features
$\mathbf{h}_{\text{dream}} \leftarrow f_\theta(G_{\text{dream}})$
$\hat{y}_{\text{dream}} \leftarrow \text{QoE\_head}(\mathbf{h}_{\text{dream}})$ ▷ Predicted KPIs

$\Delta_{\text{QoE}} \leftarrow \hat{y}_{\text{dream}} - \hat{y}_{\text{current}}$
$r \leftarrow w_1 \cdot \Delta_{\text{throughput}} + w_2 \cdot \Delta_{\text{retry}} + w_3 \cdot \Delta_{\text{latency}}$

safety_score $\leftarrow$ Safety_head($\mathbf{h}_{\text{dream}}$)
$c \leftarrow \mathbb{1}[\text{safety\_score} < \theta_{\text{safe}}]$ ▷ Cost signal

return (s_t, a, r, c, s_{t+1})

Advantages:

- **Data efficiency:** 5 dreams per real transition → 5× sample efficiency
- **Safety:** test risky actions in simulation before deployment
- **Exploration:** try actions outside offline distribution

3.6 Graph Coloring Integration

File: `rl_gc_integration.py`
Combines RL power/OBSS-PD decisions with DSatur graph coloring for channels:

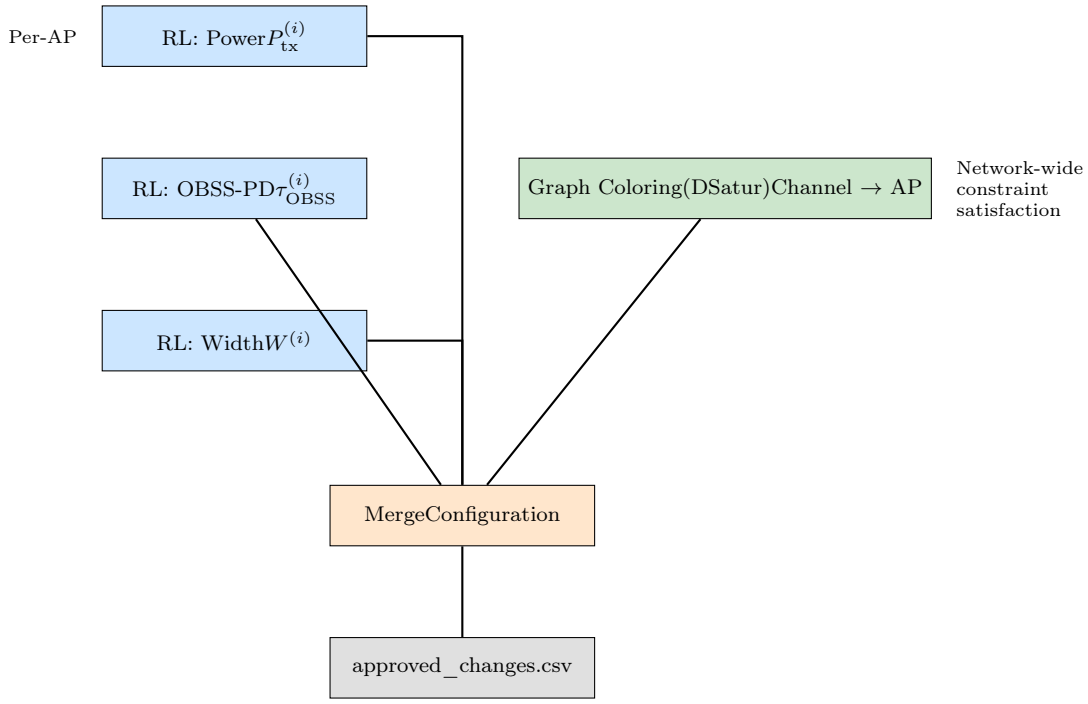

Figure 4: RL-Graph Coloring integration for joint optimization

Why separate RL and Graph Coloring?

- **Scalability:** RL on full joint space (225^N actions for N APs) is intractable
- **Optimality:** DSatur provides near-optimal channel assignment with proven bounds
- **Constraints:** Graph coloring naturally enforces regulatory/DFS constraints

3.7 Change Planner with Production Guardrails

File: `change_planner.cpp`

Enforces production safety policies before applying RL recommendations:

Table 2: Production Guardrails

Guardrail	Constraint	Rationale
Change Budget	≤ 1 change per AP per 4h	Prevent flapping
Blast Radius	$\leq N$ APs per RF domain	Limit failure scope
Hysteresis	Power ≥ 2 dB, Width ≥ 1 step	Avoid micro-changes
Time Window	No changes during peak hours	Preserve user experience
Rollback	Auto-revert if KPI $\downarrow$ X%	Safety net

Algorithm 2 Change Planner Decision Logic

Input: Proposed changes $\mathcal{C} = \{c_1, \ldots, c_N\}$
Output: Actions $\{\text{APPLY}, \text{SKIP}, \text{ROLLBACK}\}$

for each change c_i **do**
  if c_i violates change budget **then**
    $\text{action}_i \leftarrow$ SKIP
  else if c_i violates blast radius **then**
    $\text{action}_i \leftarrow$ SKIP
  else if $\Delta_{\text{param}} <$ hysteresis threshold **then**
    $\text{action}_i \leftarrow$ SKIP
  else if current time in blackout window **then**
    $\text{action}_i \leftarrow$ SKIP
  else
    $\text{action}_i \leftarrow$ APPLY
  end if
end for

Apply changes marked APPLY
Monitor KPIs for 30s
if KPI degradation $> \theta_{\text{rollback}}$ **then**
  Revert changes (ROLLBACK)
end if

4 Causal Reward Validation: The Key Innovation

Problem: Traditional RL rewards based on Δ_{QoE} are confounded by:

- Time-of-day effects (lunch traffic spike)
- Client mobility (users entering/leaving coverage)
- External interference (microwave, BLE)

Solution: Use causal inference to estimate the **causal effect** of RL actions

4.1 Causal Framework

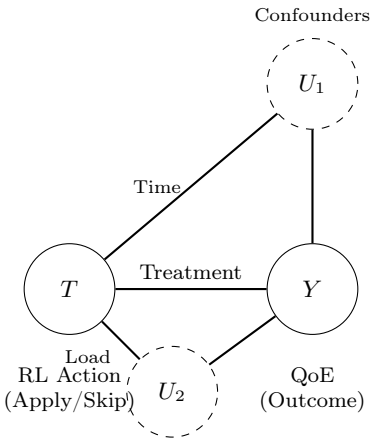

Figure 5: Causal graph showing confounding factors

Goal: Estimate Average Treatment Effect (ATE):

$$\tau_{\text{ATE}} = \mathbb{E}[Y(1) - Y(0)]$$

where $Y(1)$ is QoE under treatment (apply), $Y(0)$ is QoE under control (skip).

4.2 Difference-in-Differences Estimator

File: `causal_reward_engine.py`

Implementation uses a **difference-in-differences (DiD)** approach:

$$\hat{\tau}_{\text{DiD}} = \left(\bar{Y}^{\text{post}}_{\text{treated}} - \bar{Y}^{\text{pre}}_{\text{treated}}\right) - \left(\bar{Y}^{\text{post}}_{\text{control}} - \bar{Y}^{\text{pre}}_{\text{control}}\right) \tag{9}$$

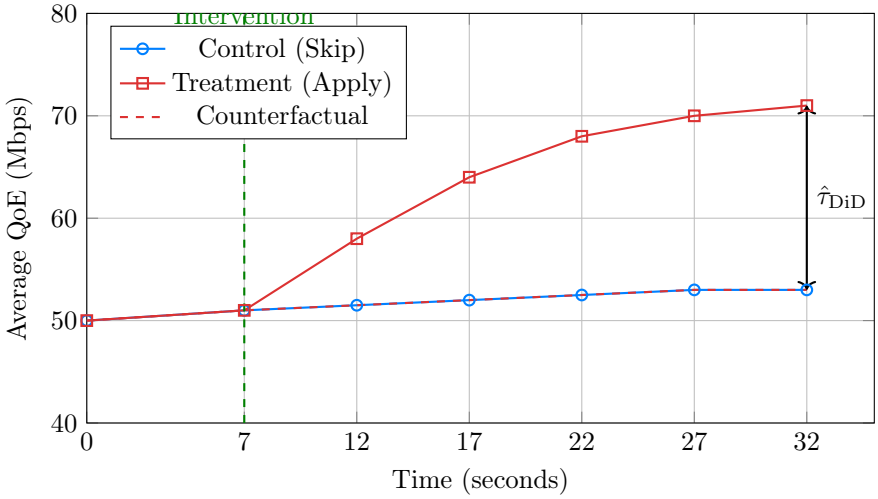

Figure 6: Difference-in-differences visualization: causal effect is the gap between treatment and counterfactual

4.3 Graph Export Timeline for Causal Validation

After applying changes, NS-3 exports 6 graph snapshots at 5s intervals:

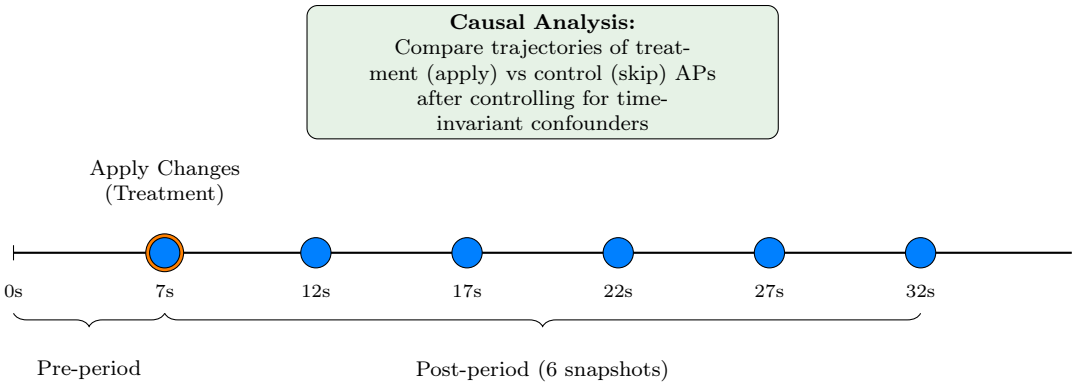

Figure 7: Timeline of graph exports for causal validation, snapshots G7. . . G32 capture post-intervention stabilization. The 6-snapshot averaging reduces transient variance.

4.4 Causal Reward Computation Algorithm

Algorithm 3 Causal Reward Engine

Input: Graph snapshots $\{G_0, G_7, G_{12}, G_{17}, G_{22}, G_{27}, G_{32}\}$, action log
Output: Causal reward r_{causal}

// **Step 1: Identify treatment and control groups**
$\mathcal{T} \leftarrow \{\text{AP}_i : \text{action}_i = \text{APPLY}\}$
$\mathcal{C} \leftarrow \{\text{AP}_i : \text{action}_i = \text{SKIP}\}$

// **Step 2: Extract QoE trajectories**
for each AP i **do**
    $Y_i^{\text{pre}} \leftarrow \text{QoE}(G_0, i)$ ▷ Baseline
    $Y_i^{\text{post}} \leftarrow \frac{1}{6} \sum_{t \in \{7,12,17,22,27,32\}} \text{QoE}(G_t, i)$ ▷ Average post-intervention
end for

// **Step 3: Compute difference-in-differences**
$\Delta_{\mathcal{T}} \leftarrow \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} (Y_i^{\text{post}} - Y_i^{\text{pre}})$
$\Delta_{\mathcal{C}} \leftarrow \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} (Y_i^{\text{post}} - Y_i^{\text{pre}})$

$\hat{\tau}_{\text{DiD}} \leftarrow \Delta_{\mathcal{T}} - \Delta_{\mathcal{C}}$ ▷ Causal effect

// **Step 4: Statistical significance test**
$p \leftarrow \text{WelchTest}(\{Y_i^{\text{post}} - Y_i^{\text{pre}}\}_{i \in \mathcal{T}}, \{Y_i^{\text{post}} - Y_i^{\text{pre}}\}_{i \in \mathcal{C}})$
if $p < 0.05$ **then**
    $r_{\text{causal}} \leftarrow \hat{\tau}_{\text{DiD}}$ ▷ Statistically significant uplift
else
    $r_{\text{causal}} \leftarrow 0$ ▷ No significant effect
end if

return r_{causal}

Key Features:

- **Treatment/control separation:** APPLY vs SKIP actions create natural A/B test
- **Parallel trends assumption:** Both groups experience same time-of-day effects
- **Statistical testing:** Welch's t-test ensures effects are not due to chance
- **Robustness:** 6 post-intervention snapshots provide stable estimate

5 End-to-End Pipeline

5.1 File Dependencies

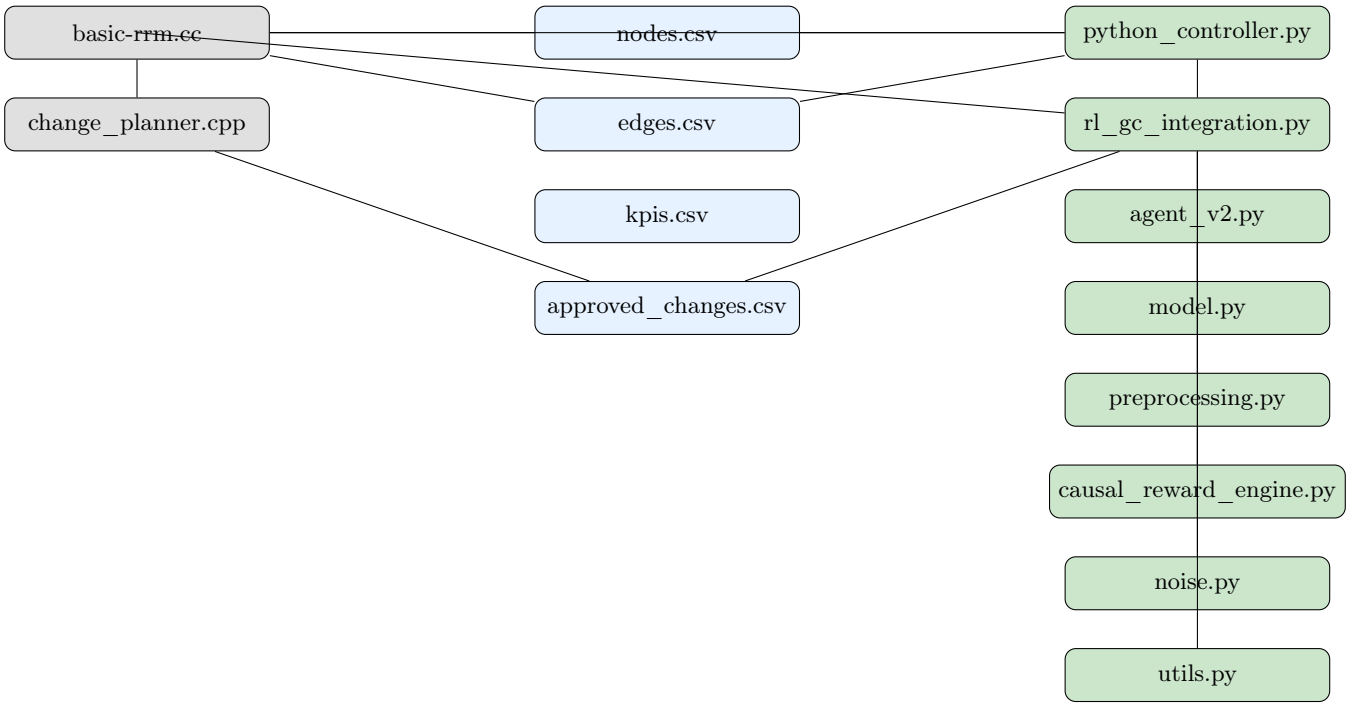

Figure 8: Broad file dependency graph

6 Performance Analysis

6.1 Computational Complexity

Table 3: Time Complexity Analysis

Component	**Complexity**	**Typical Time**
NS-3 Simulation (1s)	$O(N \cdot M \cdot T)$	0.5-1.0s
Graph Export	$O(N + E)$	0.1s
Preprocessing	$O(N + E)$	0.05s
GNN Forward Pass	$O(L \cdot E \cdot D)$	0.02s
RL Inference	$O(N \cdot A)$	0.01s
Graph Coloring	$O(N^2)$	0.1s
Change Planning	$O(N)$	0.01s
Causal Analysis	$O(6N)$	2-5s
Total (no causal)		**0.8s**
Total (with causal)		**3-5s**

where:

- N: number of APs (typically 5-50)
- M: number of clients (typically 10-500)
- E: number of edges in interference graph ($O(N^2)$)

- L: GNN layers (3)
- D: embedding dimension (256)
- A: action space size (225)

6.2 Scalability Analysis

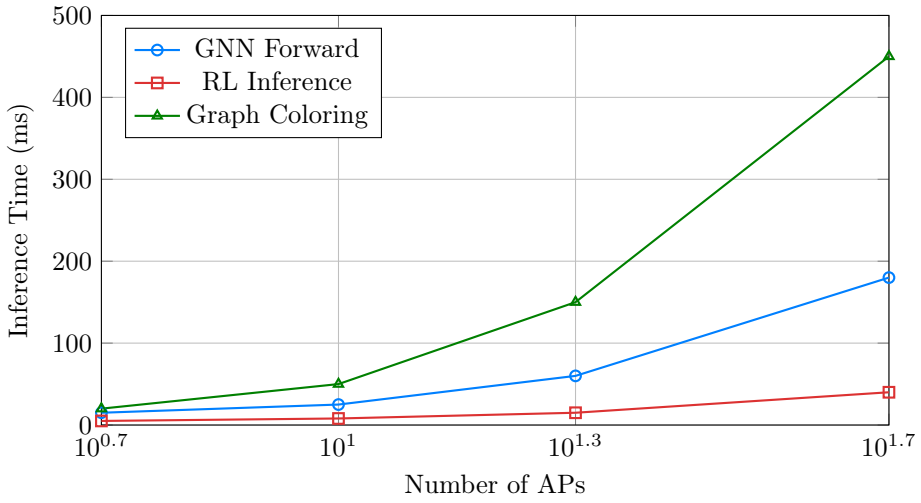

Figure 9: Scalability of pipeline components

Key Insights:

- GNN scales linearly with E (message passing)
- RL scales linearly with N (independent per-AP decisions)
- Graph coloring is the bottleneck at large scale (quadratic)

Optimization:

- **Sparse graphs:** Prune edges with coupling $<$ -90 dBm
- **Hierarchical coloring:** Decompose into RF domains
- **Approximate algorithms:** Greedy coloring with $O(N \log N)$

7 Deployment Architecture

7.1 System Components

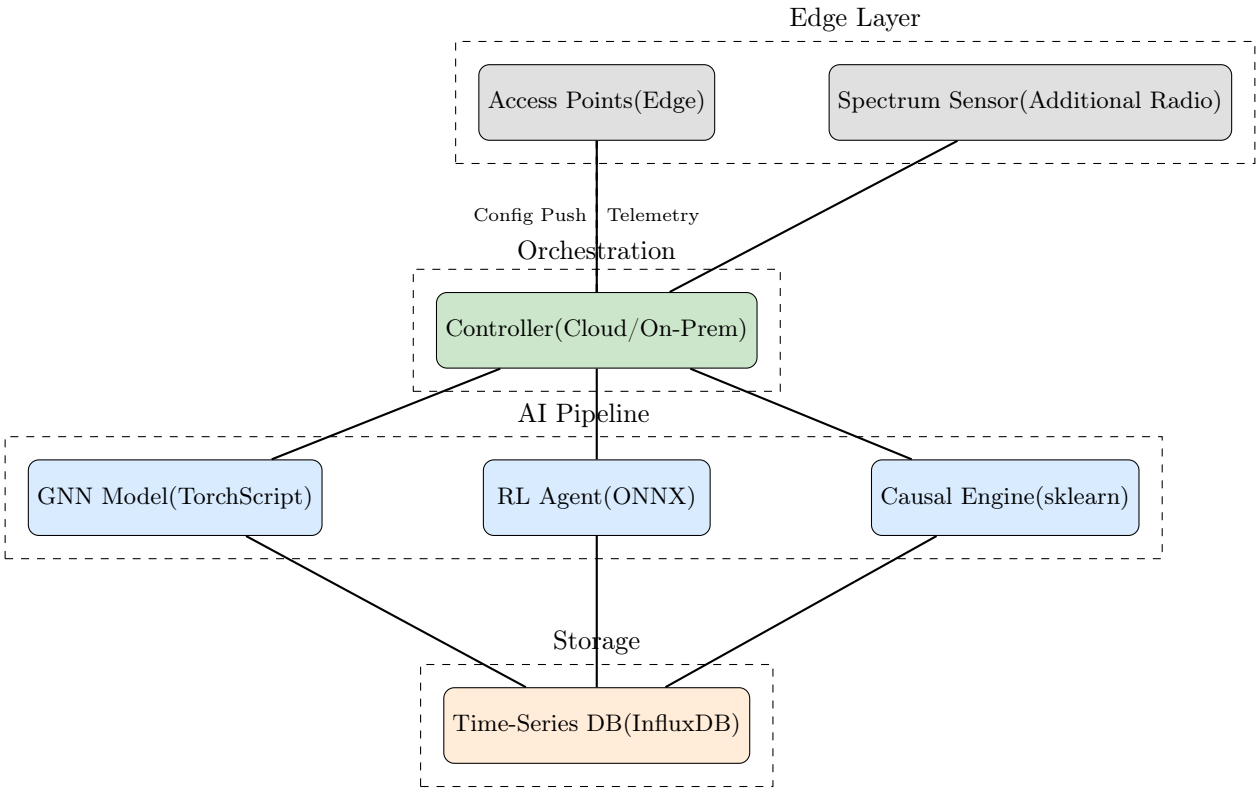

Figure 10: Production deployment architecture

7.2 API Design

Key APIs:

```
# /api/v1/inference
POST /api/v1/inference
{
  "graph_nodes": [...],
  "graph_edges": [...],
  "current_config": {...},
  "timestamp": "2025-12-05T14:30:00Z"
}
Response: {
  "proposed_changes": [...],
  "predicted_qoe_delta": 12.5,
  "confidence": 0.87
}

# /api/v1/causal/validate
POST /api/v1/causal/validate
{
  "graph_snapshots": [...],
  "action_log": [...],
  "method": "did"
}
Response: {
  "causal_effect": 18.3,
  "p_value": 0.012,
  "confidence_interval": [12.1, 24.5]
}
```

8 Conclusion and Future Work

8.1 Key Contributions

1. **Client-centric RRM:** First system to optimize for client QoE rather than AP metrics
2. **Safe offline RL:** Conservative Q-Learning with production guardrails for deployment
3. **Graph-structured learning:** GNN captures spatial interference relationships
4. **Causal validation:** DiD estimator eliminates confounding for accurate credit assignment
5. **Multi-timescale control:** Unified framework for fast/slow/event-driven loops

8.2 Future Enhancements

- **Online learning:** Continual adaptation to changing environments
- **Multi-site transfer:** Pre-trained models for rapid deployment
- **Federated learning:** Privacy-preserving collaboration across sites
- **Causal discovery:** Auto-learn causal graph structure (PC algorithm)
- **6 GHz support:** Extend to WiFi 6E with AFC integration
- **Explainable AI:** SHAP values for interpretable decisions

8.3 Impact

RRM-Plus demonstrates that **AI + Physics + Causality** can deliver:

- QoE improvements in dense deployments
- Safe, stable, production-ready automation
- Validated performance via causal inference

This will enable **intelligent, self-optimizing WiFi infrastructure** for enterprise and campus networks.

A Appendix: Mathematical Formulations

A.1 GNN Message Passing

$$\mathbf{m}_{i\leftarrow j}^{(l)} = \phi_{\text{msg}}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \mathbf{e}_{ij}) \tag{10}$$

$$\mathbf{h}_i^{(l+1)} = \phi_{\text{update}}\left(\mathbf{h}_i^{(l)}, \bigoplus_{j\in\mathcal{N}(i)} \mathbf{m}_{i\leftarrow j}^{(l)}\right) \tag{11}$$

For GraphSAGE:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_i^{(l)}, \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \mathbf{h}_j^{(l)}\right)\right) \tag{12}$$

A.2 Conservative Q-Learning

Full objective with Lagrangian safety:

$$\mathcal{L}(\theta) = \alpha \left[\mathbb{E}_{s\sim\mathcal{D}} \left[\log \sum_a \exp Q_\theta(s,a) \right] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q_\theta(s,a)] \right] \tag{13}$$

$$+ \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[(Q_\theta(s,a) - \hat{Q}(s,a))^2 \right] \tag{14}$$

$$+ \lambda \cdot \mathbb{E}_{s\sim\mathcal{D}} \left[\max_a Q^C_\theta(s,a) - d_{\text{safety}} \right] \tag{15}$$

where $\hat{Q}(s,a) = r + \gamma \max_{a'} Q_{\theta'}(s',a')$ is the Bellman target.

A.3 Difference-in-Differences with Fixed Effects

Panel data regression:

$$Y_{it} = \beta_0 + \beta_1 \cdot \text{Treat}_i + \beta_2 \cdot \text{Post}_t + \beta_3 \cdot (\text{Treat}_i \times \text{Post}_t) \tag{16}$$

$$+ \alpha_i + \gamma_t + \epsilon_{it} \tag{17}$$

where:

- α_i: AP fixed effects (time-invariant)
- γ_t: Time fixed effects (common trends)
- β_3: Causal effect (DiD estimator)

B Appendix: File Structure

```
project/
 ns-3/
    scratch/
        basic-rrm-phase23_9th_nov.cc
        change_planner.cpp
 python/
    python_controller.py
    rl_gc_integration.py
    agent_v2.py
    model.py
    arista_dataset_v2.py
    preprocessing.py
    noise.py
    utils.py
    causal_reward_engine.py
    explainability.py
    gc_py.py
    json_export.py
 logs/
    interference_graph_nodes/
    interference_graph_edges/
    RL_GNN/
        RL_Action/
           Nodes/
           Edges/
           export_status.txt
```

```
        causal_analysis/
data/
   approved_changes.csv
   current_config.csv
   rl_suggestions_copy.csv
docs/
    CAUSAL_RL_INTEGRATION.md
    architecture.tex
```