Understanding GNN vs LLM based AIOPs for Autonomous Network Operations

CHINOG 13 CHICAGO, IL

27-28 May 2026

Dr. Deepak Kakadia, CEO Founder NetAI
d@netai.ai

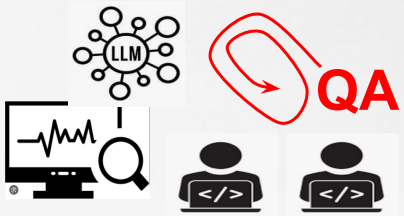
Agenda

- Why current AIOps still leaves teams doing manual RCA
- What is Missing in the existing stack
- AIOPS Architectural Patterns
- GNN Introduction
- Operational Use Case Study - POC MasOrange and Google Cloud
- Limitations and Scope
- Questions

Problem: Today's Workflow is **Reactive, Manual and Expensive**

Wait for Failure -> 4 Manual Touch Points -> Upset Customer !

GAP - Non Deterministic Root Cause



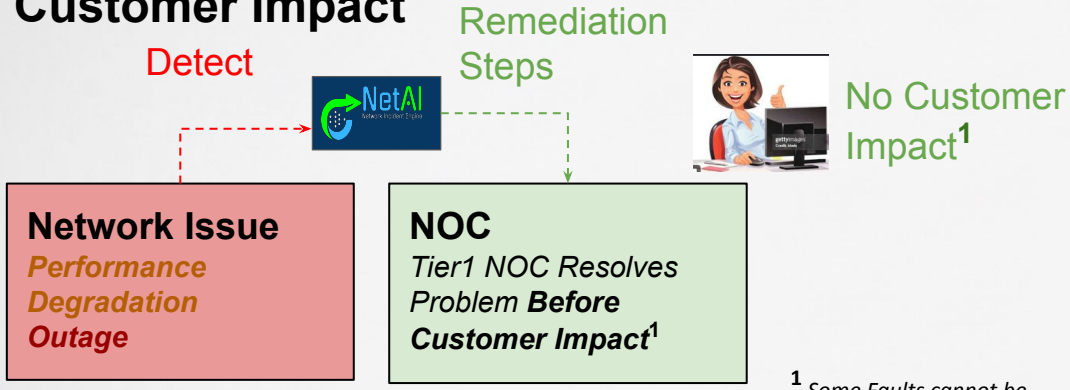
Network Issue	**Customer Impact**	**Customer Support**	**NOC**	**Network Engineer**
Performance Degradation Outage	*Customer Complaint to Support*	*Call Center Agent Escalates to NOC*	*Tier1 NOC Escalates to NetEng*	*Senior Network Engineers needed to ask Right Questions and* ***Check for Hallucinations.***

Hours to Days

GAP - Most Current AIOps Platforms Still Leave Root-Cause Validation to Operators.

Solution: Automated Workflow:Proactive, No Customer Impact

Proactively Detect and Guide Remediation with Fewer Touch Points and Lower Customer Impact



[1] *Some Faults cannot be predicted, Ex. Fiber Cut, In this case Customer Impact is Minimized, Not Avoided*

Seconds to Minutes

How ?

Graph Neural Networks (GNNs)

Provide Deterministic Answers:

- ➢ **Root Cause**
- ➢ **Associated Correlated Events**

Why ?

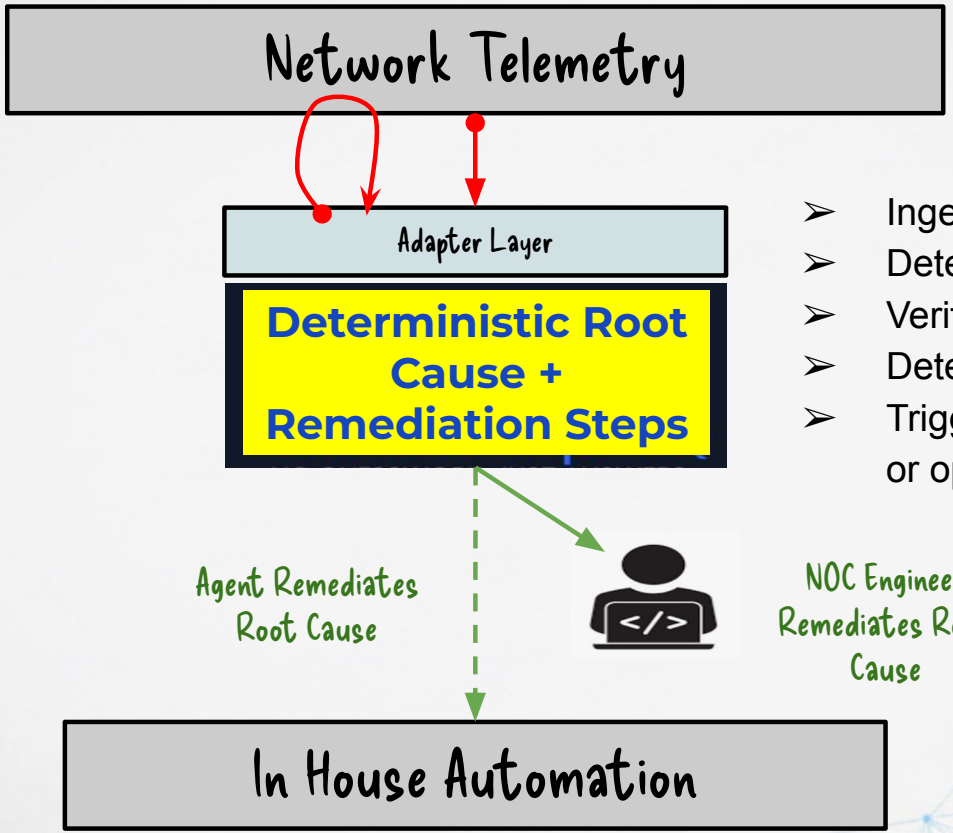
Better Customer Experience

➔ *Proactive*

Network Operations Efficiency

➔ *Faster Resolution*

The **Missing** Layer Between Observability and Automation



- Ingest telemetry from Current Tools
- Determine Root Cause
- Verify Root Cause
- Determine Remediation
- Trigger existing in-house automation or operator workflows

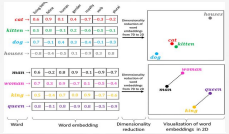
AIOPS Achitectures - Whats the Main Difference?

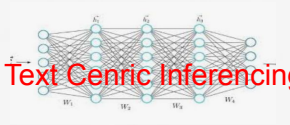
LLM based workflows require Human QA on hundreds of Alarms

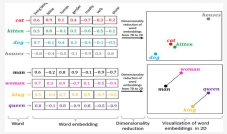
Network

Convert Issues to Words

LLM - Word Representations



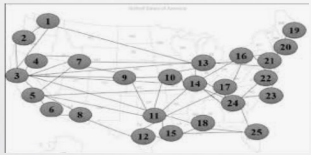


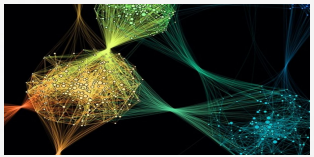


Non Deterministic Answers

GNN based workflows process Burst of Alarms→ Reduce to Root Alarms

GNN - Direct Network Representation





Deterministic Answers

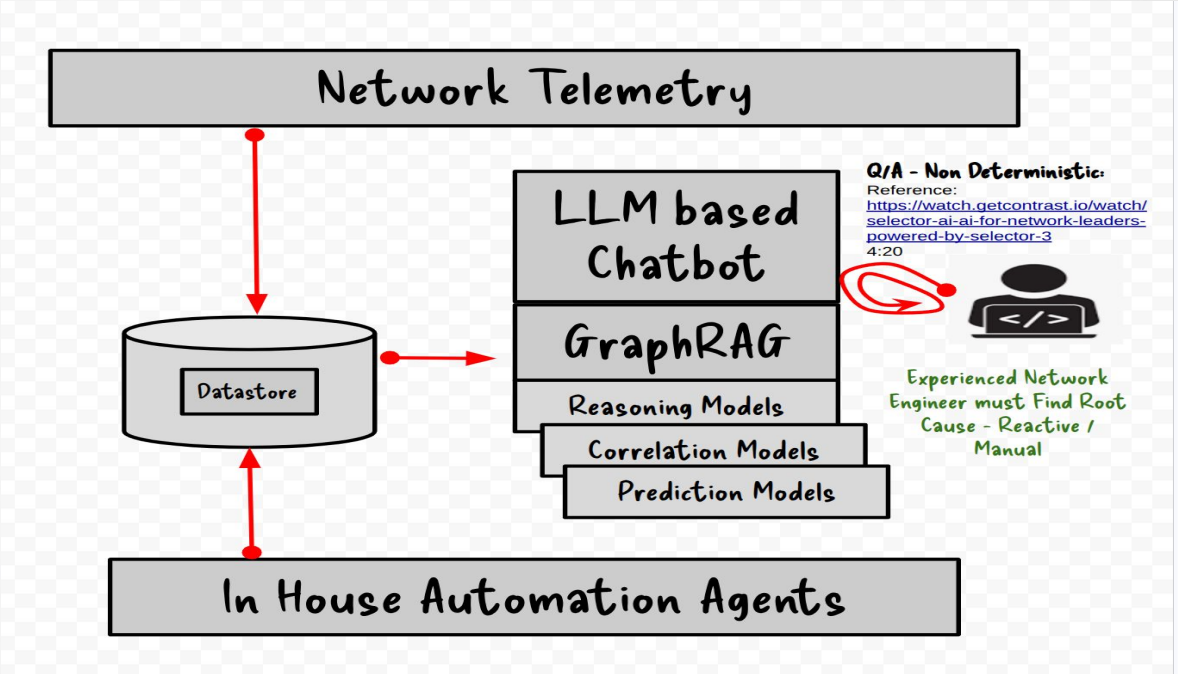
Network Aware Causal Reasoning

AIOPS Systems Architecture - **Non Deterministic**

Non Deterministic Approach requires Experienced Network Engineer to ask Correct Questions

1. **Reactive**
2. **Starts with All Alarms**
3. **Manual**
4. **Requires Experienced Network Engineer to ask Right Questions due to Non Deterministic Answers - "..***Its up to the Engineer to go on the right Path.***."**

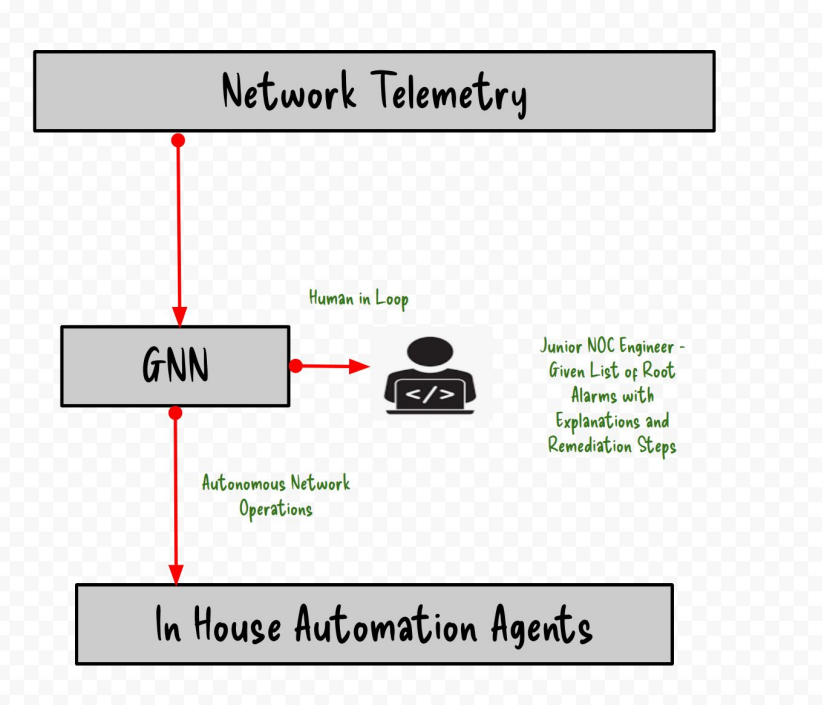
See Reference: 4:20
https://watch.getcontrast.io/watch/selector-ai-ai-for-network-leaders-powered-by-selector-3



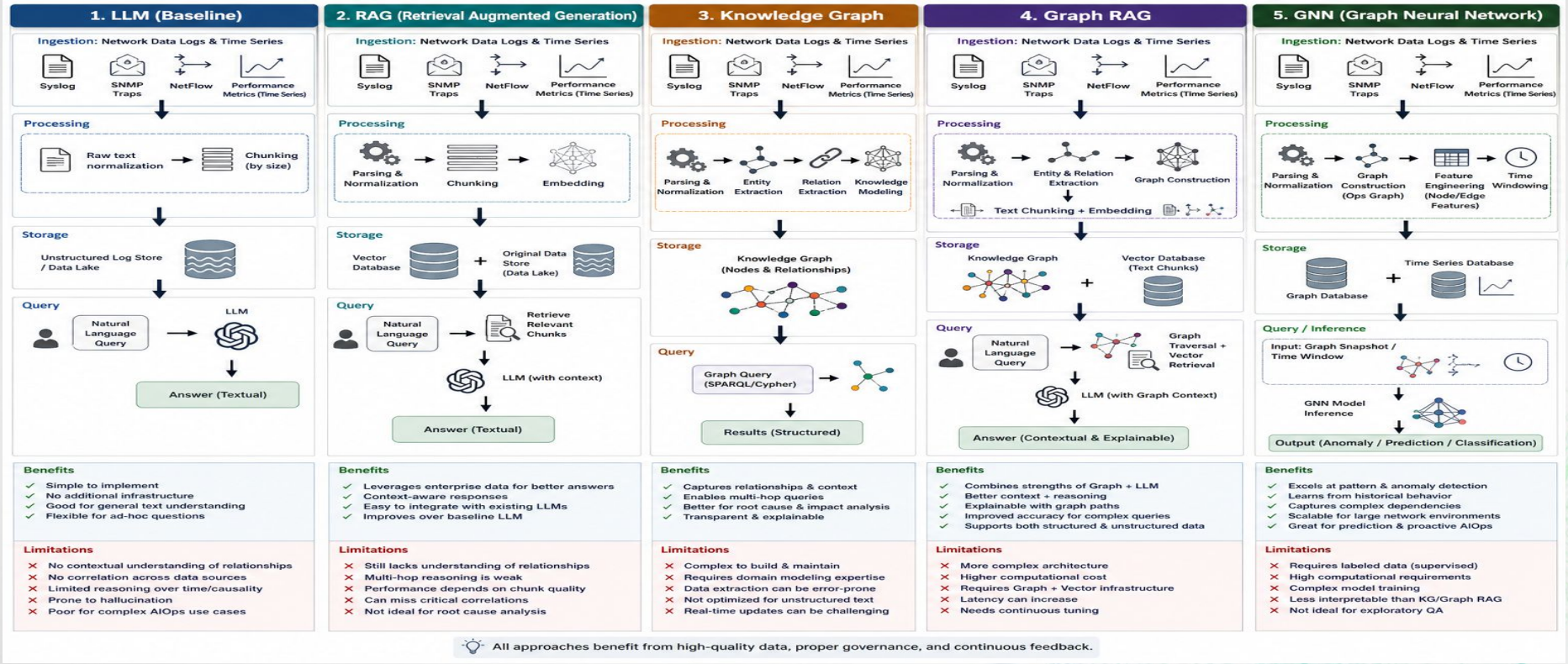
AIOPS System Architecture - **Deterministic**

Deterministic Approach enables Automation

1. **Ingest from Existing Observability Platforms or Direct**
2. **Root Cause -> Remediation**
3. **Trigger Existing In House Automation Agents**
4. **No Rip-and-Replace**



AIOPS Architectural Patterns - Reference



GNN Introduction

What is a Graph Neural Network?

- *Class of Neural Networks that operate on Graph Data*

Why does it matter in Network AIOPS?

- *Networks map directly to Input Graph Data that can be understood by GNN*

GNN Introduction

Idea: Map Network to Node and Link Embeddings(Vectors)

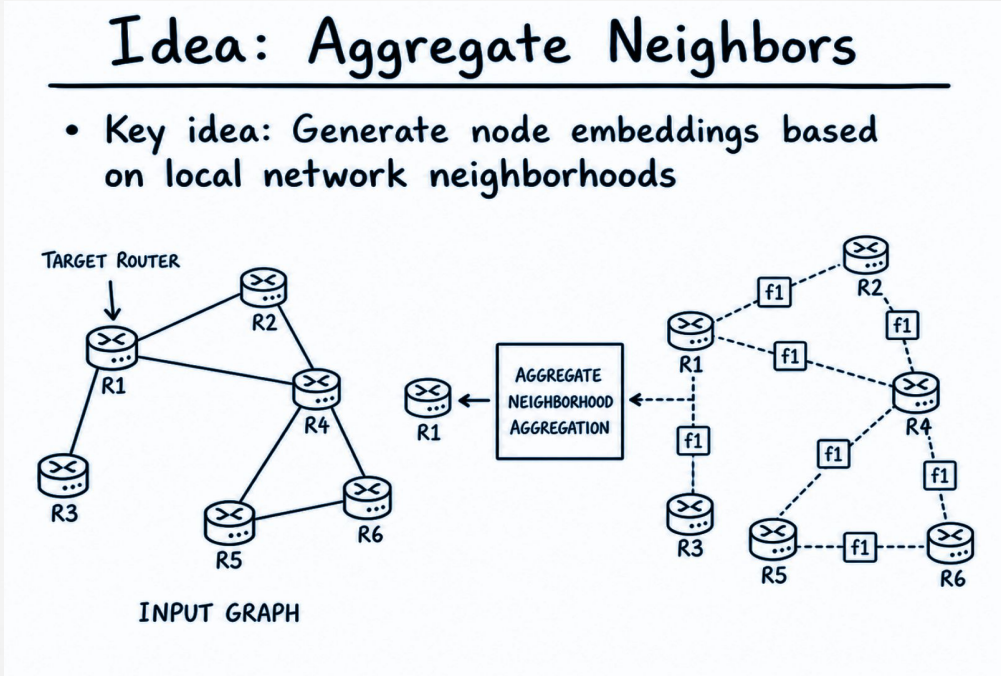
Direct Representation of:

- Network Elements
- Relationships

Maps to

Graph Data

- can be processed by AI



GNN Introduction to basic graph computations

Main Idea:

- ➢ Data Maps to Model - Direct Representation of Real World Network
- ➢ Model Captures:
  - ○ Relationships between Routers and Links
  - ○ Fault Propagation Patterns
- ➢ Learns -> Becomes Smarter Over Time for that Particular Customer

GNN message passing equations

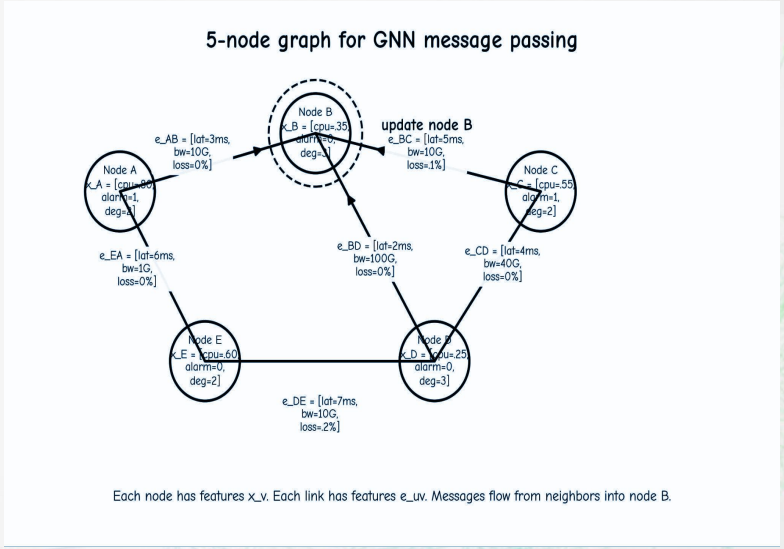
Generic message passing:

$$m_v(k+1) = AGG(\{ phi(h_u(k), e_{uv}) : u \text{ in } N(v) \})$$
$$h_v(k+1) = UPDATE(h_v(k), m_v(k+1))$$

For the diagram, if we update node B:

$$m_B(k+1) = AGG(\{ phi(h_A(k), e_{AB}), phi(h_C(k), e_{BC}), phi(h_D(k), e_{BD}) \})$$
$$h_B(k+1) = UPDATE(h_B(k), m_B(k+1))$$

Here h_A, h_C, h_D come from the node features,
and e_AB, e_BC, e_BD come from the labeled link features.



GNN Introduction - basic computations simplified

Target Router = v, Neighbor Routers = u, x = Router Features, h = latent state, B_v,W= learned Weights

What is **f**?

Initial 0-th layer embeddings are equal to node features

$$h_v^0 = x_v$$

embedding of v at layer k

$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \forall k \in \{0, \ldots, K-1\}$$

Total number of layers

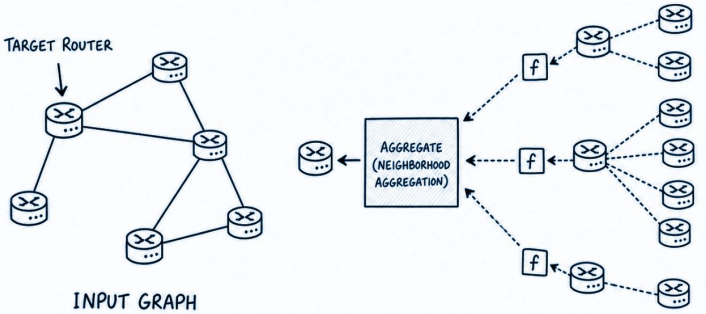
Average of neighbor's previous layer embeddings

$$z_v = h_v^{(K)}$$

Embedding after K layers of neighborhood aggregation

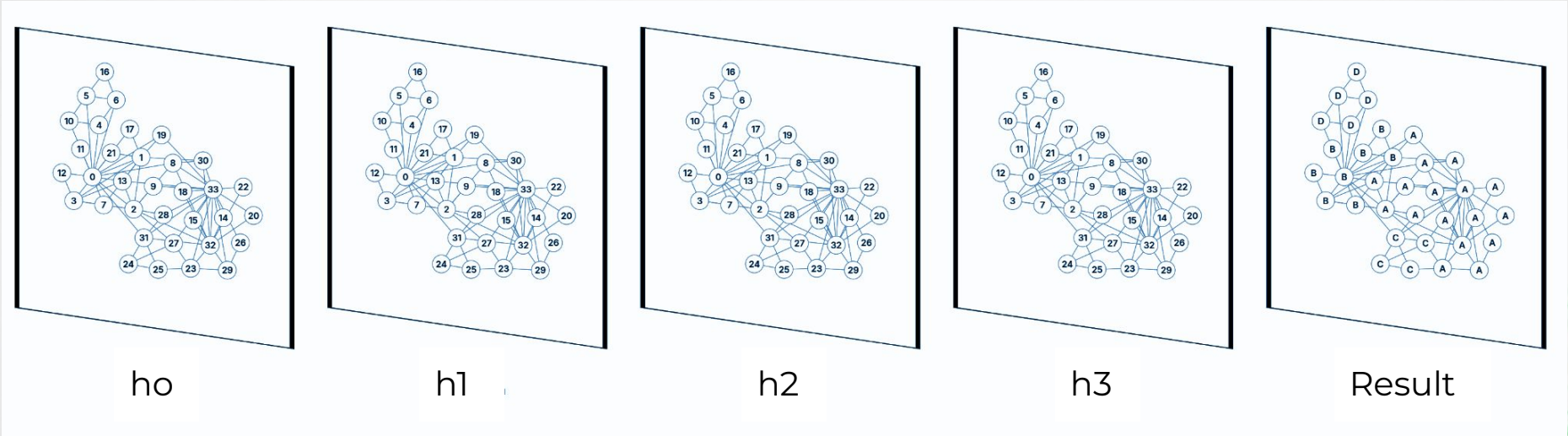
Non-linearity (e.g., ReLU)

Notice summation is a permutation invariant function.



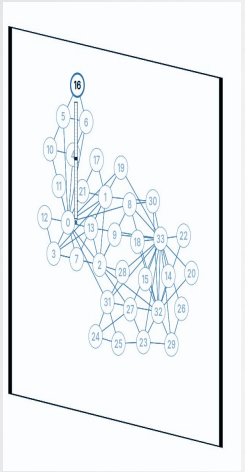
GNN Introduction - Visual explanation

Sequence of Iterations on computing a Node Latent State - 4

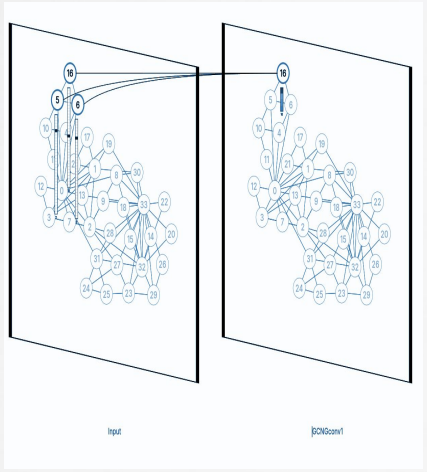


GNN Introduction - Computation of Target Node

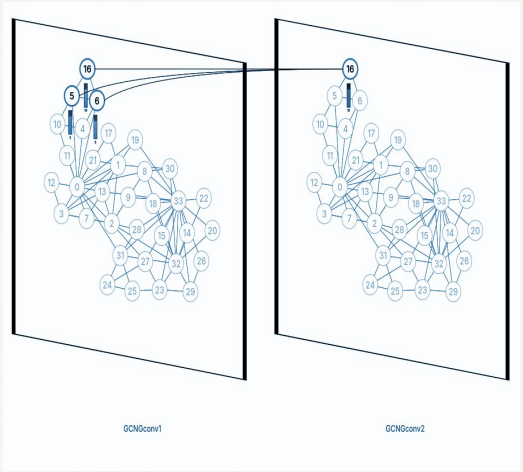
Steps



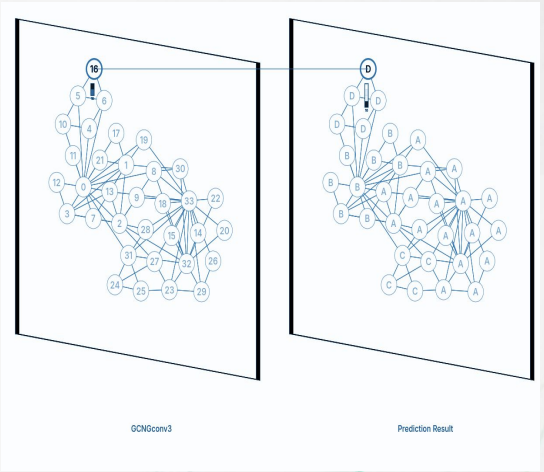
Compute h0
- Router
Feature Vector



Compute h1



Compute h2

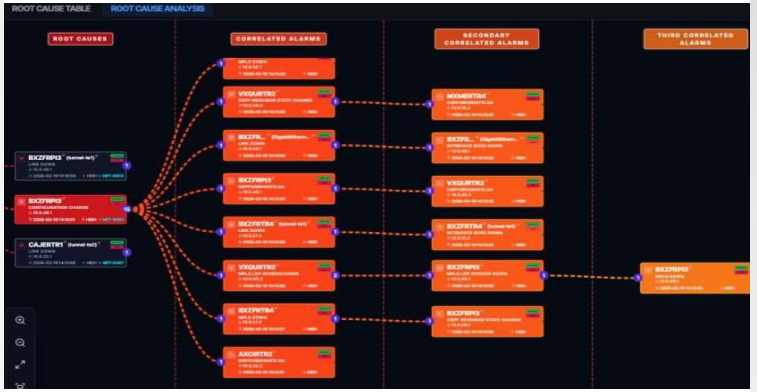


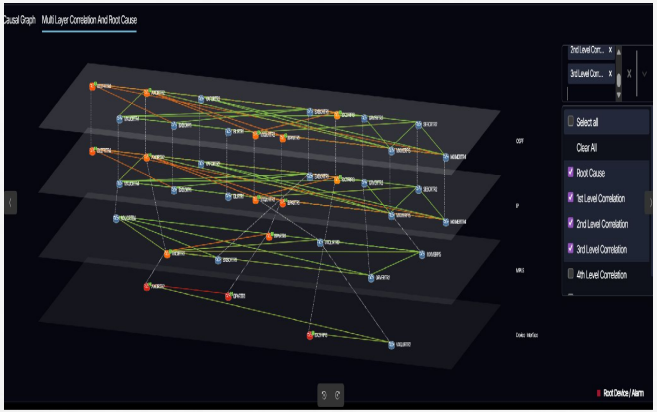
Compute h3, then
mapping h3 to class range
= Result

Case Study - large European Mobile Operator

Mobile Operator was able to prove Autonomous Network Operation using GNN - MWC 2026, Blogs

- **Incident**: Engineer made Configuration Change Error
- **Output**: Root Cause + Correlated Alarms, Explanations, Remediation
- **Operator Action:** Verified Explanations, Followed Remediation Steps
- **Business Result:** Reduced diagnosis from hours of manual triage to a validated root-cause workflow in minutes.





Incident Summary

On February 25 2026 an OSPF-origination issue was detected on POVIRTR2 ("OSPFOriginateLSA"). Within seconds several other routers reported "ospfVirtIfStateChange" events while still showing full adjacencies. No downstream routing or reachability problems were recorded – there are no impact alarms.

1) Root Cause Event

Device: **POVIRTR2**

Alarm: OSPFOriginateLSA

Detail excerpt: *"Issues with LSA origination…the router has not originated router-LSTAs with maximum metric."*

This alarm marks the first observable problem – the OSPF process failed to originate LSAs correctly.

2) Direct Correlated Alarms

Device	Alarm	Detail quote
OXDIRTR1	ospfVirtIfStateChange	"…stable adjacency … no counters indicating failures…"
POVIRTR2	ospfVirtIfStateChange	"Neighbor 10.80.10.13 … in FULL/BDR state … no explicit errors…"
BXZFRTR4	ospfVirtIfStateChange	"Neighbor 10.80.10.4 … in FULL/BDR state … no explicit errors…"

All three correlate back to the same root UUID (d88671d5…). They all report virtual-interface state changes occurring immediately after the LSA origination failure.

3) Secondary Impact Alarms

None – the incident payload contains zero impact entries.

4) Time-Ordered Sequence & Causal Chain

```
09:05:25  POVIRTR2  →  OSPFOriginateLSA            ← root
                  │
                  ├──→  OXDIRTR1  ospfVirtIfStateChange  ← correlates
                  │
                  └──→  POVIRTR2  ospfVirtIfStateChange  ← correlates
                                │
                                └──→  BXZFRTR4  ospfVirtIfStateChange  ← correlates
```

At 09:05:25 POVIRTR2's OSPF process could not generate LSAs.

Four seconds later OXDIRTR1 logged a virtual-if state change even though its neighbor remained FULL/BDR.

One second later POVIRTR2 itself logged another virt-if state change.

A minute later BXZFRTR4 also logged such a change.

Each subsequent alarm references the original LSA failure through the link relations supplied.

5) Tree Diagram

```
Root: OSPFOriginateLSA (POVIRTR2)
├─ OXDIRTR1 : ospfVirtIfStateChange
└─ POVIRTR2 : ospfVirtIfStateChange
     └─ BXZFRTR4 : ospfVirtIfStateChange
```

Shared impact nodes do not exist because no impact alarms appear.

6) Remediation Steps

Immediate Triage

Verify system clocks on POVIRTR2, OXDIRTR1, BXZFRTR4 – small timestamp gaps can mislead correlation.

Confirm OSPF process health on POVIRTR2 (`show processes ospf, debug ospf packet`) to see why LSAs weren't generated.

Check memory/disk usage on POVIRTR2; resource exhaustion often stops LSA origination.

Likely Fix Actions

Restart the OSPF process on POVIRTR2 (`clear ip ospf process <process-id>`).

If restart fails, reload the entire device or upgrade firmware where known bugs affect LSA generation.

Ensure correct OSPF area configurations (area IDs, authentication) so LSAs aren't silently dropped.

Validation Checks

After restarting, run `show ip ospf database router` on POVIRTR2 to verify new LSAs appear.

On neighboring routers (OXRIDTR1, BXZFRTR4), check `show ip ospf neighbor` – they should now display proper metrics and no state-change alerts.

Monitor for any recurrence of `ospfVirtIfStateChange`; absence confirms resolution.

No anomalies were flagged in the payload, so data quality is considered acceptable.

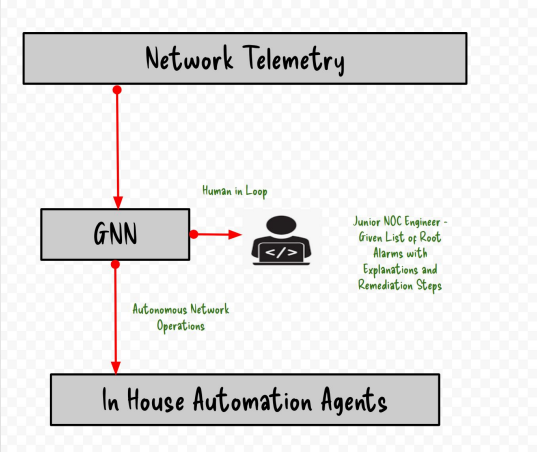
Reference:

https://cloud.google.com/blog/topics/telecommunications/autonomous-networks-at-mwc-2026

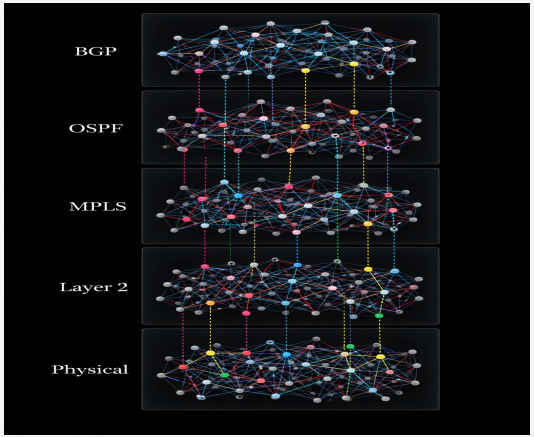
How It Works

1. Ingest Telemetry, Alarms, Topology, Logs, Configurations from Existing Repositories or Directly
2. Build Topology aware Causal Reasoning across Network
3. Output Root Cause, Associated Correlated Alarms, Blast Radius and Remediation Guidance
4. Feed Remediation to NOC Engineer or Automation

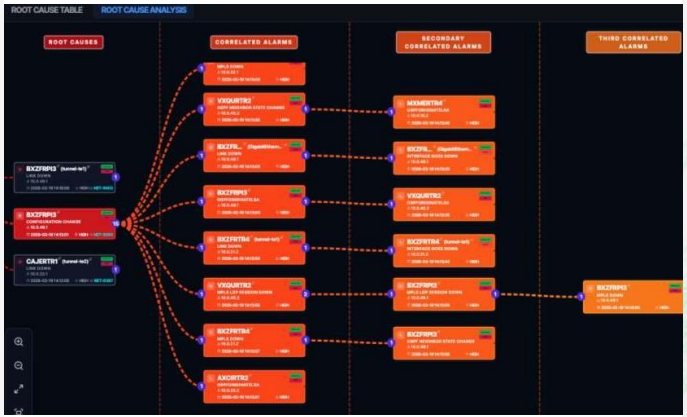
Pipeline - Autonomous Ready



GNN



Root Cause and Associated Correlated Alarms



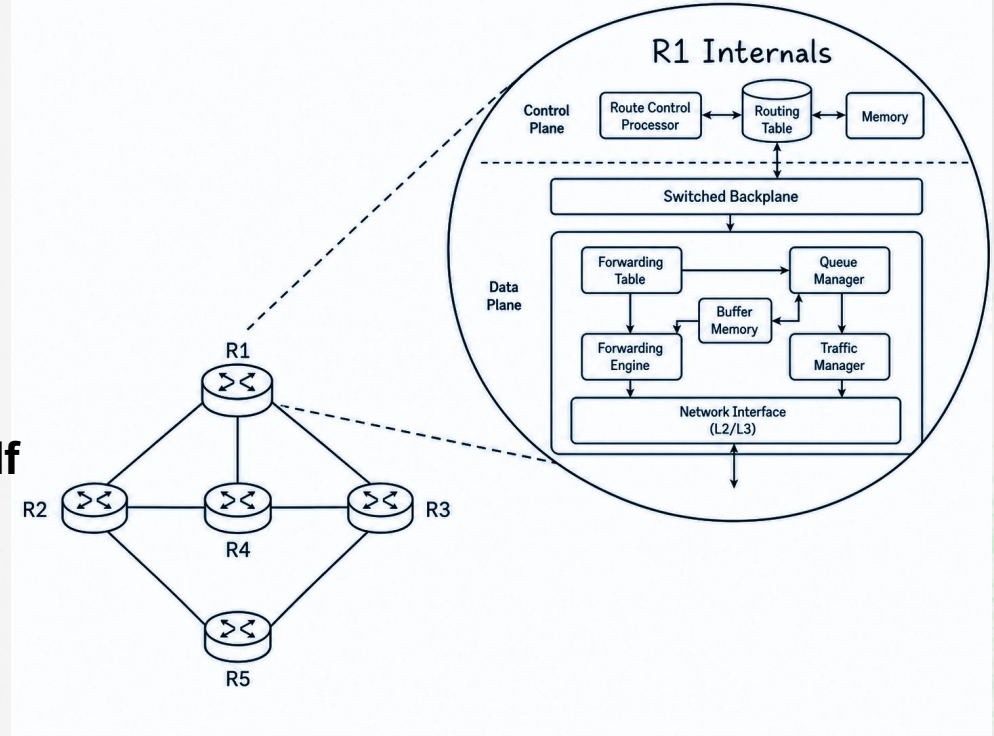
Limitations and Scope - Level of Abstraction that Matters

Suitable Class of Problems - NetOps

- ❏ Burst of Alarms - Finds Independent Root Causes and Associated Correlated Alarms
- ❏ Multi Layer, understands NS EW operational network relationships
- ❏ Intermittent Problems, Anomalies that are materially significant and known to cause outages

Unsuitable Class of Problems - What If

- ❏ Q/A Path Analysis - Forward Networks
- ❏ Text Q/A - Ideal for LLM
- ❏ Packet

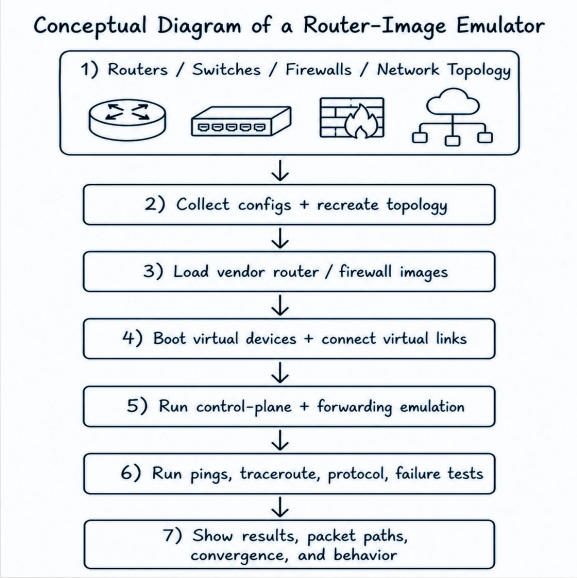


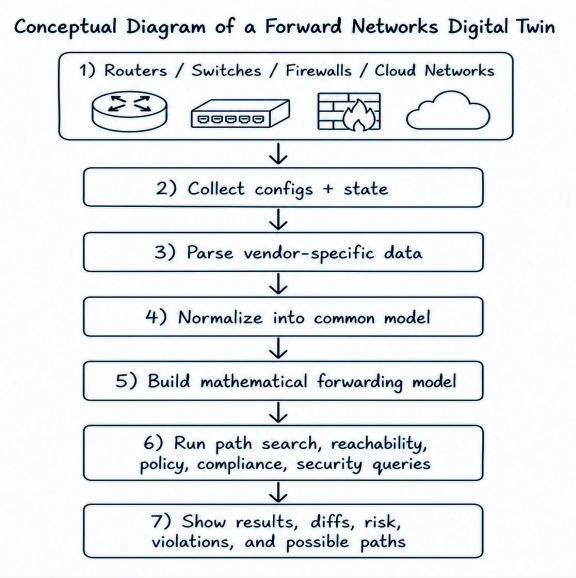
Limitations and Scope - Digital Twin - Usually GNS3

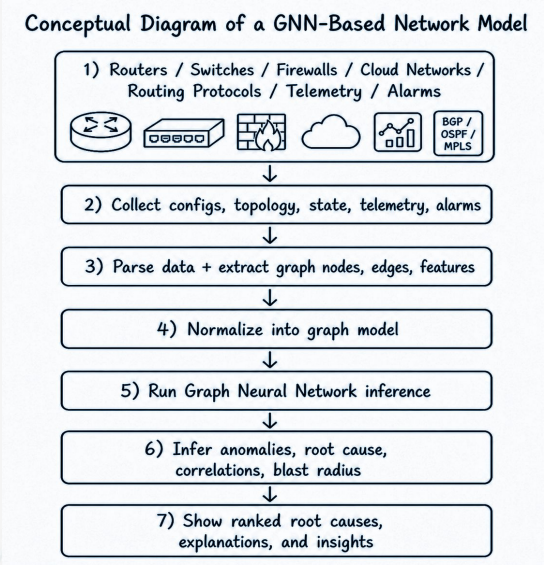
Router Emulator: GNS3
Device Behaviour

Forwarding Plane Mathematical Model
Packet Reachability Behaviour

A GNN emulates the
Failure-Propagation behavior of the network







Limitations and Scope

Suitable Class of Problems - Optimized for Network Operations

- Deterministic Root Cause and Causal Hierarchy
- Ensemble of Node, Edge, SubGraph, Multi Layers
- Why?
  - ❏ **GNNs** learn from Structure and Fault Propagation Patterns
  - ❏ GNN improves with customer-specific network history over time

Unsuitable Class of Problems

- Text Simple QA - Ideal for LLM
- Path Analysis - Ideal for Forwarding Plane emulators based on Mathematical Device Models
- Device internal behaviour - Device emulators

NetAI

Thank you