Smart Network: Graph-Based Network Analysis, Event Detection, and Outage Simulation

A technical paper prepared for presentation at SCTE TechExpo24

Nivedhitha Sridhar
Machine Learning Engineer
Comcast
nivedhitha_sridhar@comcast.com

Bob Lutz
Senior Machine Learning Engineer
Comcast
bob_lutz@comcast.com

Ramya Narayanaswamy
Machine Learning Director
Comcast
ramya_narayanaswamy@comcast.com

Harshita Bhatt
Machine Learning Engineer
Comcast
harshita_bhatt@comcast.com

Sanket Walavalkar
VP Network Data Science
Comcast
Sanket_Walavalkar@comcast.com

Table of Contents

List of Figures

1. Introduction

As one of the largest broadband providers in the country, Comcast's network spans coast to coast, encompassing multiple layers that include the backbone, CRAN (converged regional area network), and access layer. Ensuring our customers receive the best possible experience is a challenging endeavor, particularly in achieving end-to-end visibility of our network during normal operations, maintenance (including preventative maintenance), and unforeseen events.

This paper focuses on a graph-based approach to monitor the core network, including the backbone and the CRANs, to improve customer experience. The backbone sites consist of three national data centers (NDCs) and three regional data centers (RDCs), and within these sites there are core routers, route reflectors, and dozens of peering edges that are interconnected. The backbone is integral to the IP infrastructure, connecting to 28 CRANs, where each CRAN features a pair of large aggregate routers (AR), Xfinity aggregation router (XAR), residential U routers (RUR), and commercial super U ring routers (SUR), where U refers to the shape of the surrounding topology. Depending on the needs, CRANs can extend several layers deep with multiple RURs and residential edge routers (RERs). The physical layer, also known as the optical/transport layer, connects these sites, while the network layer features logical connections. This complex mesh of devices, services, protocols, and connections, with built-in redundancies, makes event detection and localization particularly challenging.

Traditional tools and existing processes assist in identifying root causes and resolving issues, but are they the most effective? Given the vast volume of data generated across all systems and considering the network's topology, traditional tools are siloed and there are more efficient and rapid methods for detecting, correlating, and localizing problems.

The work in this paper is motivated by our analogous work in the access network, where we have employed graph algorithm-based approaches for clustering and localizing network elements for troubleshooting purposes. The growing demand for real-time event detection necessitates addressing issues even before customers notice a problem or experience service disruptions.

In this paper, we introduce the concept of utilizing the spatial and temporal aspects of events and topology and using graph algorithms aided by a depth-first search algorithm for traversing layers of our network topology to group and localize events. Using this approach, we can identify when events lead to partial outages, hazardous conditions (hazcon) where customers are still receiving service, versus complete outages (isolation). This methodology supports maintenance activities and addresses events unrelated to our network changes.

Our current work focuses on the logical layer, specifically Layer 3 (L3), the network layer. We also discuss how this approach can be enhanced by incorporating Layer 1 (L1), the optical transport layer, and real-time telemetry to extend from localization to root cause analysis (RCA) and expanded for predictive modeling. By utilizing these graph-based approaches, we can work towards representing the network through a digital twin, enabling us to simulate outages and effectively manage conflicts before and during scheduled maintenance, and aid with redundancy planning and optimization. The graph-based method detailed in this paper can be readily adapted to various topological architectures as our network evolves and enhances our overall network performance.

2. Background

Let us establish some concepts and terminology used throughout the paper.

2.1. Graphs 101

A *graph* is a data structure consisting of *vertices* (or *nodes*), which can be thought of as dots, sites, or other entities of interest; and *edges* (or *links*) connecting pairs of vertices. The nodes of a graph could represent, for example, railway stations, and the edges could represent railway tracks. The graphs in this paper are assumed to be *loopless*, i.e. no edge begins and ends at the same vertex. In a graph representing the core network logical topology, vertices will typically represent routers or sites, and edges will represent logical links (sometimes called *circuits*) between the vertices. That is, an edge between two devices indicates the ability of those devices to send and receive traffic to and from one another.

For a graph representing the physical topology, vertices will typically represent optical devices and edges will represent segments or bundles of fiber-optic cable connecting the devices. A *path* in a graph is a finite sequence $(v_0, v_1, \ldots, v_n)$ of vertices such that there is an edge between v_i and v_{i+1} for all i. Each circuit between logical devices relies on a prescribed sequence of physical fibers over which to actually pass traffic; rephrased in graph terms, each edge of the logical topology graph corresponds to a path in the physical topology graph.

The *connected components* of a graph are the sets of vertices reachable from one another by traversing edges. The connected components of a graph *partition* the vertices, i.e. each vertex belongs to exactly one connected component.

2.2. Network Events

We will be mainly interested in two types of "structural" events in the core network:

1. *Isolations* (or *segmentations*), in which one or more network devices lose connection from key touchpoints in the network (ARs in the CRAN or data centers on the backbone)
2. *Hazardous conditions* (or *hazcons*), in which the connection of one or more devices to the key touchpoints becomes single threaded, i.e. reliant on a single "link."

Within these two categories, events can be further refined by impact or severity. For example, is the event customer impacting, i.e. are any isolated devices responsible for serving traffic to customers? What services are potentially affected? Are any CDNs (content distribution networks) involved? We note that isolations are inherently a higher priority than hazcons, since isolations prevent the network from operating as intended, while hazcons only indicate a higher risk for isolation.

2.3. Data Sources

The logic for our analysis depends on data sources that we will now describe.

2.3.1. Syslog-Based Alarms

Devices in the network leverage syslog to send messages in real-time about events such as whether a device has been added or removed and if an interface has been removed, added, or updated. These logs function as the source of truth for the overall state of devices and interfaces in the network. The messages are streamed and aggregated into an alarm feed based on alarm types and additional stakeholder-defined rules.

2.3.2. Real-Time Logical Topology

The logical topology of the core network consists of devices and interfaces and is represented in a graph database. Enriching this topological data is a stream of events derived from syslog data. Examples of these events include "interface up" and "interface down". "Interface up" indicates that the connection between two devices is operational, while "interface down" indicates that the connection is non-operational. These events are used to determine the state of all vertices and edges in the graph, giving a "real-time" view of the logical topology of the network.

2.3.3. Fiber Data

This data maps out the transport devices and physical fiber that span the core network and indicates the path that traffic takes to get from the AR to each destination. This data source has the L1 mapping layer otherwise known as the physical links.

3. Event Detection & Conflict Management

A network operator's ability to detect and anticipate structural events in the core network can directly impact mean time to repair and customer experience for large groups of customers. In this section, we will describe how to pair a data representation of the logical topology of the core network with a syslog-based event stream to detect and manage isolations and hazcons, as defined in Section 2.1.

In the case of isolations, legacy approaches to event detection are highly multimodal, relying on screening device-level alarms (e.g. devices not responding to ping), manually interpreting or joining multiple datasets (e.g. syslogs and topological data), and communicating between experts in different areas (e.g. network engineers, field teams, and vendors). Even identifying the "blast radius" of affected devices and customers can take hours after an inciting event, during which the overall impact is unknown.

For hazcons, the situation is hazier. Because networks are typically provisioned to carry traffic over a single-threaded link if necessary, hazcons are usually invisible to network operators until an additional event causes an isolation. Sometimes an aggravating event, such as a fiber cut, is out of the control of the network operator. Often, however, the event is part of a foreseeable internal process, like network maintenance or change management. In the latter case, awareness of the initial hazcon could have prevented the isolation by deferring maintenance until the hazcon was cleared. Current troubleshooting processes are highly manual, involving traditional eyes-on-glass approach and using tools to localize the issue.

In this section, we will discuss proactive, automated approaches to event detection with two primary applications:

1. Real-time event detection, in which real-time device data is combined with the current topological state of the network to generate a consumable feed of adverse network events.
2. Change management, in which a proposed set of changes or maintenance are overlayed with existing tickets on the network topology to identify any conflicts and report them to the user.

We will discuss these problems in both the CRAN and backbone settings.

3.1. Real-Time Event Detection

The goal of this section is to automate the identification of network devices currently in an isolated or hazcon state and (in the backbone case) to aggregate these identifications into a coherent event report that can be used in downstream automation schemes or consumed directly by operations teams.

We rely on two sources of data to address the problem:

1. A view of the current network topology, such as a graph database, including all relevant network devices and all intended logical (layer 3) links or circuits between devices
2. An event stream consisting of "link up" and "link down" events, e.g. as derived from device syslog data

Together, these sources are used to build a real-time view of the network in which inoperative or “down” links are removed. This graph is built and analyzed once per minute to determine the state of every device in the core network. We will now describe the logic behind that analysis.

3.1.1. CRAN

Let us first focus on the CRAN context. In a typical CRAN configuration, traffic is passed to and from the backbone via ARs. All other CRAN devices rely on the ARs to send and receive traffic across the network. CDNs also connect through ARs and rely on them to deliver content to customers. Thus, a CRAN’s overall functionality is based on maintaining connections between ARs and other devices.

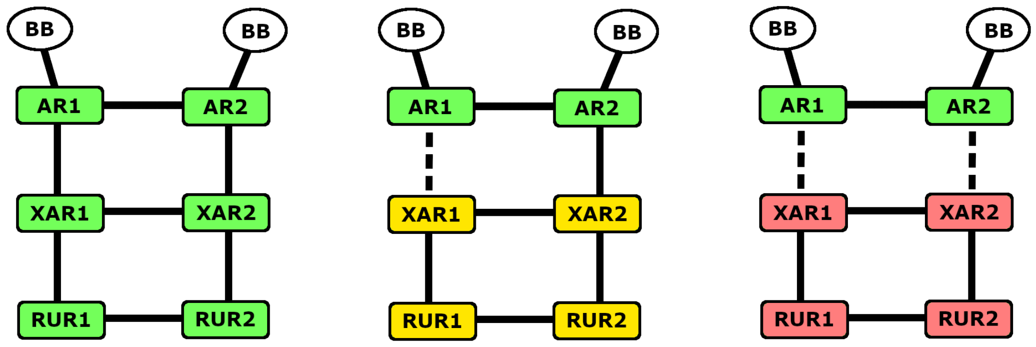

Figure 1 – A mockup CRAN with all devices fully connected (left); with some devices in hazcon (center); with some devices isolated (right)

Consider the simplified example CRAN illustrated in Figure 1. Here the ARs (AR1 and AR2) are connected to the backbone (BB). The other devices (XAR1, XAR2, RUR1 and RUR2) depend on the ARs to receive traffic from the backbone. In the left subfigure, with all devices green, the CRAN is pictured in its healthy state, with both XARs and RURs maintaining multiple connections to the ARs. In the center, the logical link between AR1 and XAR1 has gone down. This results in the connection of the non-AR devices to the ARs becoming single threaded over the link between AR2 and XAR2. The four yellow devices are therefore in hazcon. On the right, this previously single-threaded link has also gone down, isolating the four red devices.

To detect isolations, we begin by computing the connected components of the real-time graph view, i.e. the sets of devices that are reachable from each other via operative or “up” links. In this language, a device is considered *isolated* if it belongs to a connected component that does not contain an AR. This means that there is no path consisting of “up” links between the given device and any AR. The main operation used is a *depth-first search* (DFS), in which a connected component is traversed one device at a time, starting from a device of interest. This gives us a “static” set of isolated devices.

Computing hazcons is slightly more involved. To determine whether devices are in hazcon, we remove edges from the graph one by one and re-compute the set of isolations to determine which devices were reliant on the removed edge. More specifically, we do the following for each edge in the graph:

1. Remove the edge in question
2. Compute the resulting set of isolated devices, as described above
3. Filter all “static” isolated devices from the set to obtain the set of devices that are single threaded over the removed edge
4. Replace the removed edge.

This procedure lets us identify all devices in a hazcon state, as well as the specific “risky” link over which their traffic is single-threaded.

Because the algorithm above scales quadratically with the number of edges in the graph, it might be unsuitable for real-time analysis of large CRANs. To solve this problem, we can decompose the CRAN graph into a sequence of nested subgraphs whose union is the entire CRAN. The logic above is performed on the smallest subgraph (typically consisting of the ARs and their immediate neighbors). We then compute for the next-largest subgraph:

1. All isolations and hazcons inherited from the previous subgraph
2. Any additional hazcons and isolations introduced by the current subgraph.

This inductive step is carried out on every subgraph in the sequence until the entire CRAN is accounted for. This approach can reduce the computational load significantly by restricting the maximum size of the graph considered in any single step.

3.1.2. Backbone

For event detection in the CRAN, the key “touchpoints” are the ARs. On the backbone, the main touchpoints are the data centers (DC), including NDCs and RDCs. Losing connection to a DC means losing services, content, and data hosted by the DC. In conjunction with CDNs that host and deliver much of the content from peers, DCs are responsible for the network operator’s mission-critical proprietary data.

There is a qualitative distinction between CRAN and backbone isolation events. Recall that a CRAN device becomes isolated only when it loses connection to *all* ARs simultaneously. In contrast, a backbone device is considered *segmented* when it loses connection to *any* DC. This is because ARs in the same CRAN perform the same role redundantly, but there is significantly less overlap among the roles of different DCs. Segmentation from a single DC could render certain services or data entirely inaccessible.

Because of this difference, a modified algorithm is needed to detect backbone segmentations. Here, the goal is to record segmented devices and hazcons *relative to* each DC. Thus, a device can be implicated in multiple segmentations at once or be single-threaded over a link relative to multiple DCs.

We start with the graph consisting of all DCs, backbone devices and ARs, with “down” links removed. We must perform a DFS starting at each DC until all DCs have been traversed, to record which sets of devices are reachable from each DC. This traversal must follow certain rules, however, so that it simulates the flow of backbone traffic. For example, traffic from DCs cannot pass into a CRAN and back out to the backbone. Thus, the DFS must not traverse from a CRAN device to a backbone device. This notion of reachability is not transitive, so the reachable devices are not connected components in the traditional sense; they might overlap partially or completely.

Using this modified DFS, the isolation and hazcon algorithms for the CRAN can be adapted to the backbone setting. For each pair consisting of one DC and one backbone device, we compute

1. Whether the device is segmented from the DC
2. A list of links over which the connection to the DC is single-threaded.

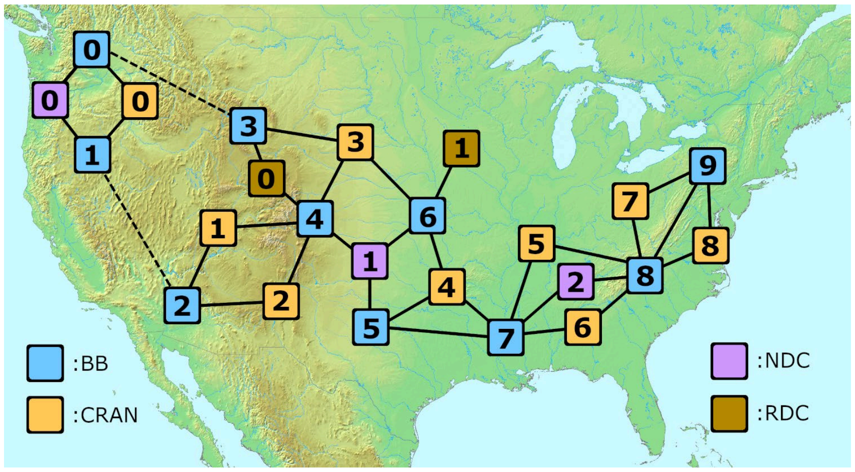

Figure 2 – High-level view of a mockup backbone topology including backbone sites, CRANs and DCs, with a segmented pocket in the northwest

In a severe event, the number of such pairs with segmentations or hazcons can be large. Within the graph, however, it is likely that the affected devices form distinct "pockets." Figure 2 shows an example of segmentations in a mockup backbone topology. Here, blue squares are backbone sites, orange squares are CRANs, purple squares are NDCs, and brown squares are RDCs. Each square contains potentially hundreds of devices. The link between backbone sites 0 and 3 is down, as is the link between backbone sites 1 and 2. This causes all devices in backbone sites 0 and 1 and all devices in CRAN 0 to become isolated from NDCs 1 and 3 and RDCs 0 and 1. Conversely, all devices in backbone sites 2–9 and CRANs 1–8 are segmented from NDC 0. Thus, there are many pairs of devices and DCs affected, but they all fall into two distinct pockets: one in the northwest and one comprising the rest of the network.

```
{
    "events":
    [
        {
            "crans": [
                "cran_0"
            ],
            "bb_sites": [
                "bb_0",
                "bb_1"
            ],
            "crans_partial": [],
            "bb_sites_partial": [],
            "ndcs": [
                "ndc_1",
                "ndc_2"
            ],
            "rdcs": [
                "rdc_0",
                "rdc_1"
            ]
        },
        . . .
        {
            "crans": [
                "cran_1",
                ...
                "cran_8"
            ],
            "bb_sites": [
                "bb_2",
                ...
                "bb_9"
            ],
            "crans_partial": [],
            "bb_sites_partial": [],
            "ndcs": [
                "ndc_0"
            ],
            "rdcs": []
        },
    ]
}
```

Figure 3 – Aggregated report of events in Figure 2: one event for the northwest and one for the rest of the network

To describe such an event succinctly, we aggregate the relevant device-DC pairs into a coherent "pocket-level" report. This aggregated report lists backbone sites or CRANs whose devices are all impacted in the same way: either segmented from the same set of DCs, or single-threaded to a set of DCs over the same set of links, or both. If some, but not all, devices within a CRAN or backbone site are affected, the impact can be listed as *partial*. For the example event in Figure 2, a JavaScript object notation (JSON) representation of the resulting report is shown in Figure 3. It contains two high-level events corresponding to the “pockets” identified above, instead of hundreds or thousands of device-level entries. This gives a view from which it is immediately clear whether an event is customer impacting (are any CRANs segmented from any DCs?) and what services are potentially affected (following from the list of relevant DCs). In an operations pipeline, these coherent event reports could more easily form the basis for tickets or jobs, whereas the per-device reports are often too numerous or lacking context.

3.2. Change Management

The logic from sections 3.1.1 and 3.1.2 can also be deployed in a change management or network maintenance application. In this context, an engineer or technician submits a ticket for proposed changes or maintenance to occur during a specific window of time. This ticket lists any devices or segments of fiber to be considered “down” during the maintenance window. The goal is to determine whether this ticket, when combined with all other tickets previously accepted for the same window of time, will cause any adverse events (isolations or hazcons), and to describe those events.

There are several factors that make this problem challenging when compared to real-time event detection. First, there is a predictive element to consider when building the graph used to simulate the tickets. If the window occurs one month in the future, for example, then the current real-time graph is not necessarily an appropriate basis for analysis. The states of the links are likely to change before the window occurs, and indeed even the structure of the network might change. Thus, rather than a representation of the network “as is,” it might be more appropriate to use a graph of the network “as designed,” or some combination of the two.

The second complicating factor is the need for pre-checks before the change or maintenance is rendered. Whichever graph is used to perform the analysis when submitting the ticket, it is likely that intermittent changes will have occurred in the network. Thus an ad hoc analysis, layering the ticket over the real-time event detection logic, is needed shortly before maintenance is carried out. Additionally, complicating these checks is the fact that the proposed work can occur outside the window on which the initial analysis was performed. For example, if work on an unrelated job takes longer than expected, the current change might be delayed or postponed.

The third challenge is user error. Ticket processing is ultimately reliant on details provided by the ticket submitter, including location, duration, and a comprehensive list of any devices, interfaces or infrastructure (such as fiber) affected. Even marginally incorrect details can completely invalidate any analysis performed to determine conflicts. The logic here does not provide safeguards against submission errors; if anything, this type of automation without proper oversight could weaken the checks and balances inherent in a more manual system involving multiple teams. Due diligence is therefore needed in confirming the details of every ticket with supporting documentation before allowing tickets to enter an automated change management system as described above.

3.3. Current State & Future Work

As of the writing of this paper, the CRAN event detection logic from Section 3.1.1 is running in a production environment. A typical CRAN can be analyzed in under 100 ms, likely much faster than a human performing the same analysis. The output is currently being evaluated against legacy syslog-based alarm systems (see Section 5) that report on a per-device basis, without reference to topology, as well as known cases of hazcons and isolations (e.g. those occurring during planned maintenance). Early indications are that the algorithm output, when viewed together with the logical topology as in Figure 1, provides an effective and accurate summary of significant network events as they unfold. With further validation, the goal will be to operationalize these summaries into workable tickets. When applied to cases of change management, the logic has detected conflicts accurately in the same ~100 ms runtime and is undergoing further testing and development to address the challenges listed in Section 3.2.

The event detection logic can be enhanced in several ways. The first enhancement centers on fiber. Hazcon is not a physical state as well as a logical state; indeed, a device can be single-threaded over a segment of fiber without being single-threaded in the logical view of the network. This happens, for example, whenever multiple logical links depend on a single fiber segment. The analysis described above will not necessarily detect a hazcon in this case, even though the device is a single fiber cut away from isolation or segmentation. Given a mapping whose keys are fiber links and whose values are the sequences of physical fiber segments taken by traffic passing through each logical link, the hazcon algorithm above can be easily adapted to detect physical hazcons. To do this, instead of removing and replacing each logical link in the algorithm, we remove and replace the set of all logical links dependent on each individual fiber segment.

The second enhancement focuses on CDNs. Much of the content delivered to customers comes not from DCs but from CDNs managed by external business partners. These CDNs are peered to CRANs, so that customers within those CRANs can receive content directly from the CDNs via their associated ARs. However, not every partner has a CDN peered with every CRAN. Hence some amount of CDN traffic is routed over the backbone and can even enter the backbone from a CRAN. In this way, CDN traffic follows different rules from DC traffic, and these rules are often influenced by policies or routers external to the operator's network. Because of this, the modified DFS described in the backbone traversal algorithm does not apply to CDNs. An enhanced traversal algorithm that simulates CDN traffic could be used instead of the modified DFS to include CDNs along with DCs in backbone event detection. All other logic would remain the same.

4. Event Grouping and Correlation

Over the course of a day, thousands of alerts are generated across the network, indicating state changes, maintenance upgrades, reboots, power outages, etc. The objective of this set of algorithms is to group and summarize these alerts based on the graph topology to provide a concise localization of the source(s) of the problem and the blast radius.

These alarms may indicate issues at different levels – a device might be down, it may be single threaded, one or more of its neighbor relationships may be down, etc. As in the case of event aggregation, a lot of steps towards event aggregation and understanding the causes of an event involve manual efforts.

In this section, we will go over the automated approaches to group alerts with two use cases:

1. Event localization, in which the summarized group of events is combined with the topological layout of the network to identify one or more origination points or loci of the event
2. Adverse fiber event detection, where we layer in the fiber-level mapping to compute whether any fibers or transport nodes are involved, and if so, pinpointing those.

4.1. Graph Representation of Network

We take a graph-based approach to determine the configuration of the devices within the geographical sites that comprise each CRAN.

There can be any number of configurations of L3 links within a CRAN. They all start at the ARs, which act as the AR and conduit for traffic to get from one site to another within a CRAN. Some topologies are more complex – a geographically spread out CRAN might have a remote AR (RAR) which performs some of the functionality of the AR. In other cases, there might be SRs (spur router) which have no direct logical connection to the AR and instead connect to one or more intermediate sites, creating additional dependencies. Similarly, there are several conditions that determine the fiber path that traffic takes to get from the ARs to the destinations on the L1 graph. The first step towards analyzing the network as a graph is to compute these nuances.

4.1.1. L3 Graph

To compute a topological hierarchy of the graph, while also accounting for further dependencies, such as the spur site (J1 in Figure 4 below) being dependent on its predecessors (C2), some steps are taken on the core network graph of L3 links:

- Since the ARs connects every site in a CRAN, we disconnect the devices from the AR and compute the connected components
- Once we know the pockets of devices that are interconnected, we add back the connections to the ARs, and exclude the edges between any devices that exist on the same level. This creates a tree graph rooted at the ARs.
- Computing the path from the root to the leaf of each tree gives us the overall hierarchy of the whole component and the dependencies for each level in the tree.

4.1.2. L1 Graph

The paths that traffic can take from the site to/from the AR are predetermined based on a set of standards, including but not limited to:

- Every site, excluding spur sites, must have at least one logical connection to both ARs in that market
- To the furthest extent possible, the two paths must be entirely independent of each other, to ensure redundancy in case of outages or maintenance on the other side. If independent routing is impossible due to geographical restrictions (as is the case with the section of the network in Figure 8 involving Y2 – X1 – Y1), other measures are taken to facilitate redundancy, for instance, through a combination of overhead and underground cables.

4.2. Sample Data

To illustrate the methods and graph algorithms used in event grouping/localization, we use an anonymized version of a subset of sites within one CRAN.

Each node on this graph represents a site, which is made up of devices that serve the logical network links as well as the physical transport links. Figure 4 shows the layout of the L3 graph; B0 and D1 are the ARs, while each node in the middle is a site composed of a combination of residential and commercial super U-ring routers (RURs and SURs, among others), switches, and connections to the access network. The last node is J1, which is the spur site, and is dependent on the node C2 to reach the ARs.

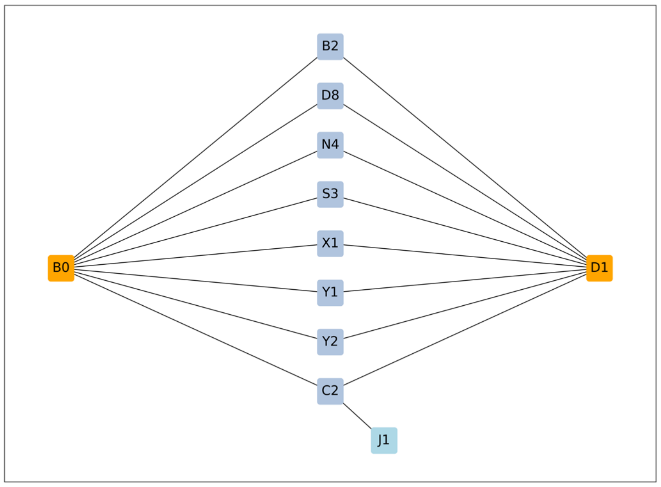

Figure 4 – Mockup of the L3 logical links within a CRAN with ARs on either side and one spur site (J1)

Each of these logical links has a fiber connection to the ARs, which may be composed of one or more hops along transport nodes. This makes up the L1 graph. In the L1 graph, as shown in Figure 5, each node on the graph represents the transport device – these devices make up the an optical transport network (OTN). For instance, the node S3 connects to the two ARs by taking the following paths, highlighted in different colors in the figure:

B0 ↔ S3: B0 – B2 – Y2 – S3

D1 ↔ S3: D1 – D8 – N4 – S3

i.e. traffic from B0 to S3 traverses the edges from B0 – B2, B2 – Y2, etc.

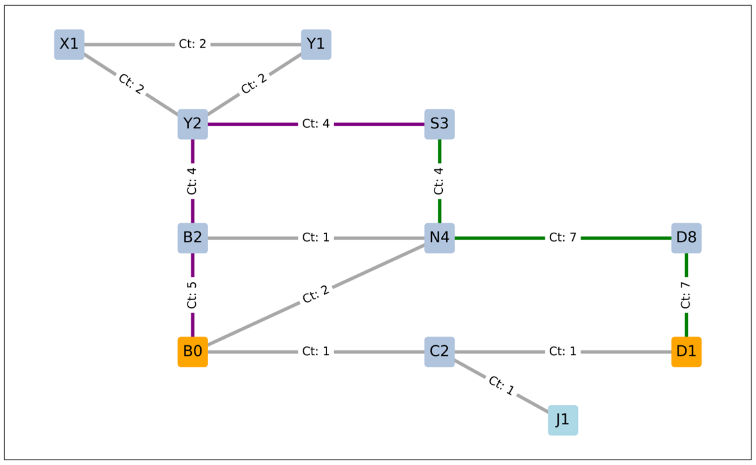

Figure 5 – Mockup of the L1 (physical) links within the CRAN. Edge labels represent the number of site pairs that depend on each link.

4.3. Event Grouping

The last dataset for this process is the feed from the routers' syslogs, which create an alarm every time there is a state change in a router's operational status or any adjacency relationship state changes. This feed is first processed to account for repeated alarms, flapping between UP and DOWN states, and out-of-order events. Next, alarms involving devices that are not service impacting are filtered out. These alarms are then grouped by chronological and regional proximity, such that events within a certain geographical range, whether connected component or CRAN, and within a certain time period of each other, are considered independent sequences of events.

Figure 6 represents the conditions that the event summary is passed through as part of the localization process. If, within the time window, 1 or 2 connected components are down, that is then tested for node isolations and a modified version of the lowest common ancestor algorithm is applied. If the outage summary spans several connected components, it is tested for fiber events on a CRAN level.

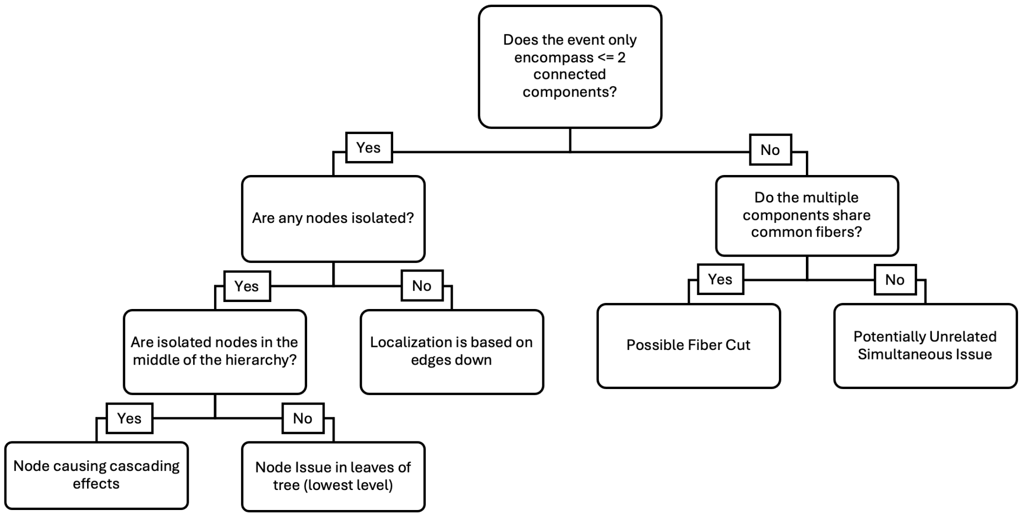
Does the event only encompass <= 2 connected components?
Yes
No
Are any nodes isolated?
Do the multiple components share common fibers?
Yes
No
Yes
No
Are isolated nodes in the middle of the hierarchy?
Localization is based on edges down
Possible Fiber Cut
Potentially Unrelated Simultaneous Issue
Yes
No
Node causing cascading effects
Node Issue in leaves of tree (lowest level)

Figure 6 – Decision Tree of which methodology is used based on event summary

4.3.1. Component-Level Events

Within a component, we can build out a hierarchy, since several levels of aggregators and functionality exist for different purposes. From the event summary, we attempt to perform localization to identify the level of the hierarchy where we first see an event.

This is where the hierarchical view of the component is used. Knowing each device's dependencies, we can scan the tree layer by layer to identify the common element in a given component. This can be either an edge or a node, and the algorithm will return both.

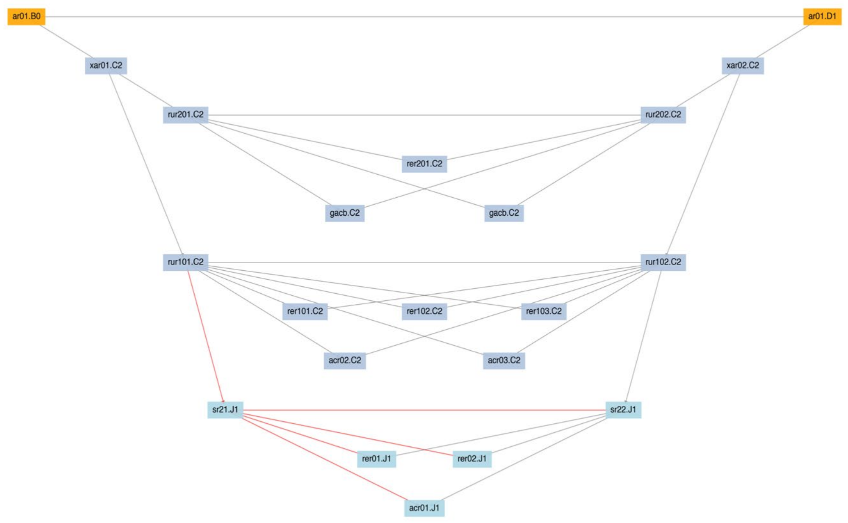

Figure 7 – A mockup event of an event in the L3 graph involving the spur site (J1).

Figure 7 is a simulated example of one such event based on a real incident – the event was limited to one component, i.e. the site C2 and the spur site J1. The device SR21 in J1 was entirely isolated, leading to some hazcon events for the devices below it in the hierarchy.

4.3.2. Adverse Fiber Events

Fiber events may occur due to a variety of reasons, such as street accidents, power outages, construction, etc. Thus, it is imperative to quickly identify the locus of the fiber event to diagnose the root cause, compute the blast radius, and deploy fixes.

The methodology of determining whether an event summary denotes a fiber cut involves:

- To consider the scope of the event and account for the fact that L3 routers to L1 devices have a many to one relationship, we aggregate the events and topology data one level higher, this time based on site-headend site pairs.
- Counting the number of L3 site pairs that depend on a specific fiber segment.
- Calculating what percentage of L3 site pairs per fiber segment are down
- The center of the fiber cut is derived as the segment(s) with 100% of their dependent L3 site pairs down. In cases where multiple segments have 100% of their dependent L3 site pairs down, a confidence score is assigned to each based on the total number of links that fiber segment carries. For example, of two fiber segments, if one fiber has 2/2 of its links down, while the other has 11/11 links down, the latter is given a higher confidence score and is considered the locus of the fiber cut.

For instance, assuming the event summary lists the following L3 links as down:

D1 – Y1, B0 – S3, D1 – Y2, D1 – X1

The algorithm calculates the number of outages on each edge and based on the confidence score, and giving a higher weight to fiber segments that carry more dependent L3 links, it determines the locus of the outage, as shown below.

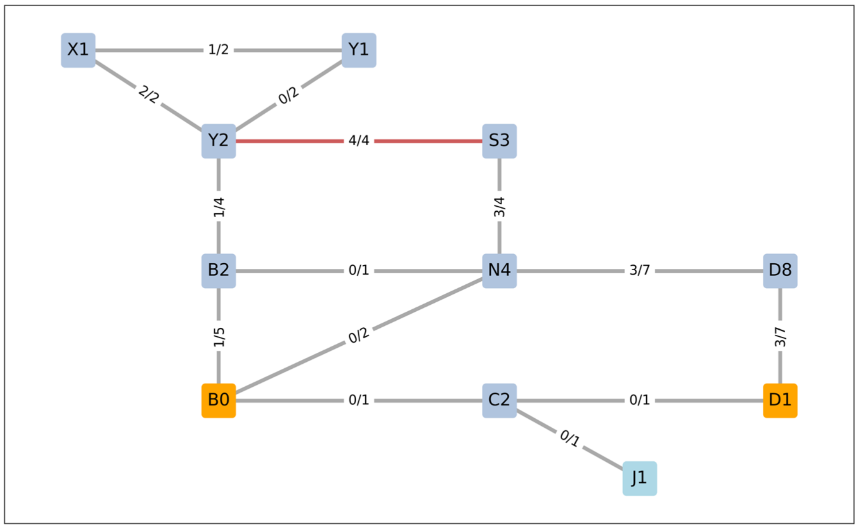

Figure 8 – Simulated Example of a fiber cut event with the damaged fiber highlighted in red. Edge labels represent the number of site pairs down on the numerator and total site pairs on the denominator.

As per Figure 8, the graph with current outages can be fed into a modified version of the conflict management algorithm, which would flag which sites are currently in hazcon because of the fiber cut, and ensure that maintenance upgrades, etc. are not performed on the endangered links. In this example, Y2, Y1, and X1 are in hazcon since they are only being served traffic from one side (B2 ↔ Y2). A hypothetical loss of service in the link from B2 ↔ Y2 would lead to complete outages in Y1, X1, and Y1.

4.4. Enhancements

Another approach towards detecting fiber cuts would be to use optical time dimension reflectometer (OTDR) shots. The OTDR fiber monitor tests light levels, among other variables. And if there is a fiber cut, it would either return no result or an anomalous result, i.e. the distance measured by the shot would be significantly shorter than the established baseline. This would help confirm the computed fiber cut and provide more granular information on its location.

Shared risk link groups (SRLGs) are sets of two or more failure points where simultaneous outages, would cause a hard down outage somewhere in the network. Outage simulation would facilitate the ability to compute these combinations by removing pairs and sets of edges and determining which ones cause outages.

5. Data Quality & Validation Process

To ensure confidence in our analyses and models, we must trust the data we use, which includes topology and events. This trust is crucial for accurately analyzing, grouping, and correlating events and gaining a comprehensive and understandable view of the network. Therefore, we emphasize data quality, focusing on the completeness and accuracy of data from various sources. This emphasis also helps us understand any underlying assumptions necessary to support our models.

5.1. Challenges

While utilizing multiple data sources provides a more comprehensive view of the network, juggling multiple sources comes with challenges. These challenges include:

1. Understanding the data.
2. Determining the "whys" for different processing methods (uncovering underlying post processing extract, transform, and load (ETL) and assumptions).
3. Stitching together data sources to get a complete, accurate, and actionable view of the core network.

5.2. Data Processing

As previously mentioned, our event detection and conflict management algorithms depend upon real-time topology data. To ensure the completeness of the detected events, we compare them to the underlying data, i.e. syslog-based alarms. However, this raw data is unstructured and often too chatty to be managed and analyzed efficiently without additional processing, including filtering and aggregation. Different processing methods allow this data to be consumed and then enhanced for different applications to provide insights regarding the range of events that a device can go through. At the simplest level, different data sources ingest refined syslog-based alarms and process them differently to meet the unique needs of different teams, products, and stakeholders.

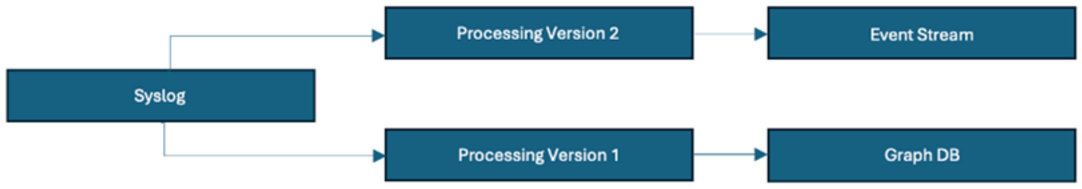

Figure 9 – Data processing and flow

For example, the real-time logical topology takes syslog-based alarms and processes it into data that is leveraged by the real-time event detection algorithm. The graph-algorithm described in section 3.1 takes input criteria based on CRAN/Backbone and leverages the graph database to output a status for every device in the network. These statuses indicate whether a device is in hazcon or isolation (see section 1 and 2) and offers a high-level overview of network health. Here, we can leverage data validation to verify that the statuses and messages provided by various data sources are in alignment. In general, the purpose of data quality measures is to better understand the data coming from different sources and transform them into quality insights to leverage for event detection, correlation, and conflict management pipelines.

5.2.3. Data Validation Methodology

Our data validation methodology for syslog-based alarms and the real-time logical topology is broken into five steps:

1. *Data cleaning* – The data sources are cleaned to make sure times, dates, device names, and interfaces are aligned to ensure an accurate comparison.
2. *Matched events* - We overlay all the data sources for a sample set of CRANs alongside any rules and filtering applied ad-hoc to see which events are missing and the severity of those missing events.
3. *Scores* - Percentages are then calculated to see how many events matchup between data sources.
4. *Additional checks* – Any discrepancies between the two data sources are checked against a third data source - an event stream that aggregates the syslog-based data by applying pre-defined rules

which in turn generates alarms. The alarms indicate when a device undergoes an event causing it to be in a suboptimal state.

5. *Automation* – An automated process allows these discrepancies to be monitored daily and stored as a delta table that converts into a csv that can be shared as an automated message to stakeholders. This automation process allows us to create a foundation to level-set and share information regarding the data to make sure all involved partiers can benefit from our learnings.

With this method, we can quantify and identify data gaps and knowledge gaps. For example, if a device is in hazcon, indicated by the real-time logical topology, we can cross-reference it with the data from the event stream and syslog-based alarms to understand any discrepancies and improve our confidence of device events. Reasons for these discrepancies between the data sources include filtering certain devices and interfaces, removing non-customer impacting devices, and alarm suppression.

These data sources help monitor the health of the core network, while outputting events and alarms differently. It is crucial to enhance our understanding of how these data sources generate and process events, especially given how instrumental they are in conflict management and event grouping.

6. Conclusion

In this paper, we introduce the concept of using graph algorithms, data, and topology for event detection, categorization, and localization both at the time of an event and proactively as part of change management. This approach provides comprehensive visibility of the entire core network, including CRAN, backbone, and peering edges, and can be invoked in real-time when an event occurs, eliminating operational silos and minimizing manual input. This significantly aids in our journey towards automating the management and troubleshooting of our core network, ensuring we offer the best customer experience.

The current localization approach can be further enhanced by overlaying OTDR to improve accuracy in detecting and localizing adverse fiber events. Additionally, event detection logic can be enhanced by integrating CDN traffic flow rules for every CRAN, allowing for better representation of hazcon and isolations, and identifying which traffic types will be most impacted.

This approach is easily expandable for representing the network through a digital twin, enabling us to simulate outages to study network behavior, model optimal paths, plan for redundancy, and detect shared risk link groups (SRLG). This critical information is essential for maintenance, fiber footprint expansion, and virtualization efforts. Crucially, this graph-based approach is designed to be durable and future-proof, ensuring it can effectively adapt and accommodate a variety of topological architectures in the network and respond to technological advancements in the industry.

7. Acknowledgments

The authors thank the Comcast Ocular, Smart Network Platform and Core Network Engineering teams for technical and organizational contributions to the work in this paper.

Abbreviations

AR	aggregation router
BB	backbone
CDN	content distribution network
CRAN	converged regional area network
DC	data center
DFS	depth-first search
ETL	extract, transform, and load
hazcon	hazardous condition
JSON	JavaScript object notation
L1	Layer 1
L3	Layer 3
NDC	national data center
OTDR	optical time domain reflectometer
OTN	optical transport network
RAR	remote AR
RCA	root cause analysis
RDC	regional data center
RER	residential edge router
RUR	residential U-ring router
SR	spur router
SRLG	shared risk link group
SUR	super U-ring router
XAR	Xfinity aggregation router

Bibliography & References

[1] Photon Avatars in the Comcast Cosmos: An End-to-End View of Comcast Core, Metro and Access Networks, Venk Mutalik, Steve Ruppa, Fred Bartholf, Bob Gaydos, Steve Surdam, Amarildo Vieira, Dan Rice, SCTE 2022