SCTE CABLE-TEC EXPO FAST FORWARD 2021
ATLANTA, GA
OCTOBER 11-14

2021 Fall Technical Forum
SCTE • NCTA • CABLELABS

Solving The Mysteries of the Distributed Access Architecture

A Technical Paper prepared for SCTE by

Matthew Stehman
Comcast
1800 Arch St, Philadelphia, PA
Matthew_Stehman@comcast.com

Ramya Narayanaswamy
Comcast
1800 Arch St, Philadelphia, PA
Ramya_Narayanaswamy@cable.comcast.com

Jude Ferreira
Comcast
1800 Arch St, Philadelphia, PA
Jude_Ferreira@comcast.com

Robert Gaydos
Comcast
1800 Arch St, Philadelphia, PA
Robert_Gaydos@comcast.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

List of Tables

Title **Page Number**

As the benefits of a distributed access architecture (DAA) continue to be seen from real world applications, Comcast is continuing to convert its analog cable modem termination systems (CMTSs) to virtualized CMTSs (vCMTS). The new technology of DAA not only allows for increased performance of the network with the switch to all-digital components, but also increased visibility with more sophisticated real-time telemetry. The DAA framework emits high fidelity telemetry data from many components, from the primary headend to the customer premises equipment (CPE). The scale of our DAA footprint is growing rapidly, and it is no longer feasible for humans to monitor all of the raw telemetry data and identify patterns of interest and issues. This paper introduces a computational framework and analysis methodology for automated monitoring and alerting for events of interest.

With analog CMTSs, telemetry data is acquired via polling MIBs from the CMTS OS, typically hourly and even down to five-minute intervals in some cases. Our vCMTS implementation has a dedicated telemetry core that constantly emits and writes all telemetry data at 15 second intervals to a time series database, so the real-time data can easily be acquired and analyzed by the DAA team. Each vCMTS captures thousands of telemetry streams, comprising over 1 billion samples per day from a single physical server, which houses many vCMTS cores. With this volume of data, we needed a telemetry analysis tool that could make sense of the data in its current form and continue scale up with DAA in the coming years. Comcast is currently developing a tool for this purpose, internally named “Sherlock.”

The name isn’t entirely coincidental. As the title of this paper implies, the very act of distributing an access architecture tends to uncover many infrastructural mysteries that could benefit from a sleuth. Some relate to the huge amount of data that flows in every 15 seconds from the thousands of broadband-foundational components within our physical infrastructure. The upstream signal path in particular is a trove of noise-related anomalies, as one example referenced within this paper illustrates. It represents an excellent network segment to expose to machine learning (ML) – which thrives on large amounts of data.

While the DAA telemetry covers most of the active components in the network, there are external tools and data that can significantly enhance the capabilities of Sherlock. As such, Sherlock interfaces with other systems such as: customer contact, automated support tickets, existing performance metric tools, etc. The core component of Sherlock is its ability to interface with a wide variety of data sources and create a single, time-aligned view of the entire system for analysis.

Once the single, time-aligned view of DAA is created, event identification and alerting can be implemented. Initially, logic-based event tagging is implemented based on common thresholds for events like partial service, plant-based noise as well as system statuses such as DAA cores offline, remote PHY device (RPD) reboots, and CPE connectivity. Once these events are determined, analytics can be performed to evaluate the frequency and severity of events. Using the event statistics, rankings are created to support the DAA team in prioritizing issues to address as well as keep track of persistent issues. The analysis and rankings are performed at different levels of aggregation: RPD, physical server, site and even division and national.

Future research utilizing the core functionality of the tool includes advanced ML techniques to find patterns/events outside of the standard events identified from traditional logic-based checks. This paper introduces several active research areas in ML in the DAA space. An overview of the architecture is introduced, and an example is discussed.

The development of this tool has had a large impact on the successful deployment of DAA in Comcast's footprint. A sample of example findings from Sherlock is discussed as well.

1. Overview of DAA Telemetry

Remote PHY (R-PHY) has taken hold as the technology of choice for deployment of DAA solutions. The concept of a distributed architecture decreases the amount of equipment that traditionally sits in a cable headend and then connects via hybrid fiber/coax (HFC) to neighborhoods and eventually to customer homes. DAA moves the PHY, or physical RF layer, closer to the user by deploying RPD-equipped R-PHY nodes that sit on the access edge of the network. DAA allows for higher speeds to the end-user because it uses digital fiber optics in place of legacy analog optics. Digital fiber links improve signal quality and support higher modulation orders. DAA also offers operational savings related to the cost of headend equipment, power and more, as small hub sites or curbside equipment act as the PHY layer of the network.

vCMTS technology enables us to shift to a DAA by disaggregating the CMTS. It also allows us to move to IP-based connectivity and converge voice and data services with video and other legacy services, with an added benefit of no longer needing to manage and maintain traditional, bulky CMTS gear.

The transition from legacy hardware to a distributed server-based architecture that can run external software applications allows Comcast extreme visibility into the DAA platform. The scalability and openness of DAA means the platform can now support applications such as real time telemetry streaming that would have been too demanding to run on legacy CMTSs. The DAA system is rich in telemetry, where individual components within the network transmit data as frequently as every 15 seconds to indicate the health of system/network. Mining DAA telemetry data to identify and detect issues in the network that could potentially lead to bad customer service is not only challenging but also involves combing through a lot of existing tools and data sets to build a smart access network.

1.1. Problem Statement

As we scale our DAA deployment to thousands of digital nodes and hundreds of vCMTSs and sites in the next few years, we anticipate operational challenges. Among them, monitoring and going through all telemetry data points to determine system health, and correlating the impact of one component in the architecture to the other key components, while identifying impairments proactively, before customers are affected.

Sherlock is a tool designed to address those challenges by looking at all the relevant metrics, creating a time-aligned view at the most granular level, scoring the health of the system, and identifying root cause of issues. It also proactively identifies anomalous patterns that lead to poor system performance. The goal of Sherlock is to analyze and identify patterns of impairments and rank them based on several criteria, that are outlined in Section 4.

1.2. DAA Topology and Telemetry

As mentioned previously, the DAA architecture offers rich telemetry, from the headend all the way to the CPE. A simplified representation of the telemetry coverage across the DAA topology is show in Figure 1. A variety of telemetry metrics are reported across the topology, ranging from system statuses to traffic and even HW/SW versions of the RPDs.

In our topology, the primary headend houses the core servers and provides statuses of the principal and auxiliary cores for global system health. The primary headend comprises multiple PPODs, or physical

point of deployments. A PPOD is a cluster of servers running the necessary services to operate the connected RPDs. The PPOD the connects to a set of DAAS (distributed access architecture switches) that transfer data to and from the RPDs. The primary core or GCPP, for Generic Control Protocol Principal, provides containerized services for automating deployments, managing applications, the initial authentication of the RPDs, and configuring RPD features and video services. The principal core does not provide any services (video or data).

The GCPP core performs the following three primary functions:

- Initial authentication of the RPD.
- Initial configuration of the RPD, including the list of cores to which it connects and the resources that those other cores will configure.
- Configuration of the multicast sources that the RPD uses to populate QAM video (broadcast and narrowcast) channels. The GCPP allows integrating videos on a standardized, single video platform.

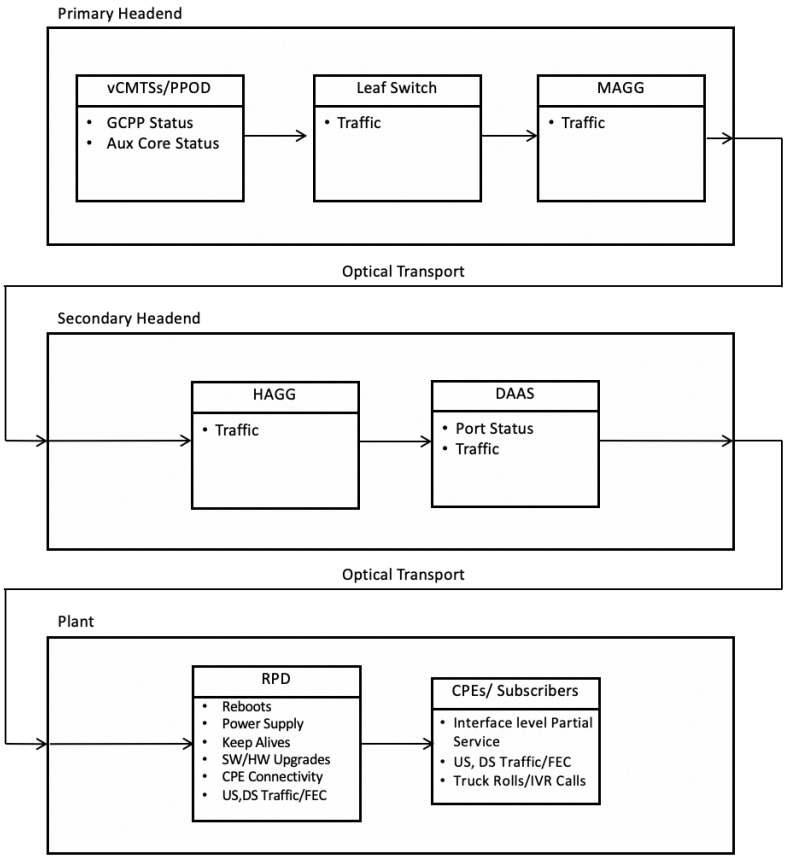

Figure 1 - DAA Topology Metrics Overview

The auxiliary core or GCP (generic control plane) is the second of the two main control planes within DAA architecture: The GCP, which sets up a control plane tunnel over a generic transport protocol such as TCP or UDP. GCP is used to program the R-PHY system upstream and downstream parameters from

the CMTS. It is also used to control the R-PHY system, thus if the AUX core is offline, no data can flow to/from the RPDs.

The secondary headend houses a variety of switches that prepare and breakout the signals prior to sending/receiving data from the RPDs. The main switch type of interest is the DAAS (distributed access architecture switch), which aggregates 10 Gbps connections to remote nodes. The DAAS switches report connection statuses which show the heath of the connection from the fiber port on the switch to the RPD.

In the plant region of the DAA topology, RPD metrics include RPD meta information, interface traffic and even CPE-level information. Having telemetry in different regions of the network topology allows for easier identification of where in the topology an issue may have originated. Table 1 includes more detail of the main metrics that are collected from the DAA network. These metrics were chosen through discussions with DAA experts as well as iterative exploratory data analyses during the initial development phases. These metrics, while not exhaustive, cover the key aspects of system/network health as well as customer experience. The list of metrics is continuing to grow as Sherlock is used in the development and deployment of DAA.

Table 1 - List of Metrics

Metric Type	Metric	Description
Platform Status	GCPP Status	GCPP Status captures the state of the Global Control Primary Plane and indicates if it is operational, offline or initializing
	Aux/GCP Core Status	The Aux Core Provides HSD services and the status indicates if it is online, offline or being configured
Network Status	Keep Alives	Keep Alives track the status of the TCP network connection between the principal cores and the switch interfaces
Hardware	RPD Reboots	Details about RPD reboots such as a reason for reboot, type and recovery time
	DAAS Port Status	DAAS port status captures the status of the DAAS switches located in the secondary headend
Traffic	Device, RPD, Routers-US and DS Traffic	Upstream and Downstream traffic is collected from various components from PPOD to CPE
Device Status	CPE Registration	Registration status captures the CPE devices attempts to pair with the vCMTS that must happened every 30s as dictated by the DOCSIS specifications
	US/DS Bonding Status	Bonding Status for each US/DS interface per CPE device is captured
FEC	Corrected and Uncorrected Codewords	CCW and UCCW are collected at device and interface levels. These can be an indicator of impairments within the plant/faulty modems that need to be proactively identified and addressed
Customer Contact	Truck Rolls, IVR Calls	Truck Rolls and IVR Calls capture customer contacts and are key metrics used within Comcast to measure operational efficiency

Originally, we were focused on finding metrics that correlated with trouble calls/truck rolls, since customer-facing technician appointments were thought to be a good indicator of system health issues.

However, it was found that trouble calls were highly sporadic, because of inherent human aspects, e.g., different tolerances to service interruptions and when calls happen relative to an issue. After exploring the customer-facing aspects, it was determined that Sherlock should focus on the engineering side of the DAA data and let the data drive the insights. Customer contact and technician visits are still evaluated, since those are valid indicators of issues, albeit not as straightforward to identify as data that comes directly from the system data itself.

1.3. Need for Data Aggregation

DAA telemetry has several dimensions, such as frequency of data, type of data (event-based and telemetry-based), the level at which telemetry is captured (device, RPD, PPOD, etc.), and traffic direction (downstream and upstream). This makes it challenging for a tool to thoroughly mine, to provide views at different levels in the network that help identify the health of the system, or identify areas of problem spots in the network.

In addition to the multi-dimensionality, data is transmitted and stored at different locations, which creates the need for a central data repository. As well, standardization of the data elements across different time intervals is needed, so as to have the data accessible for analysis and modeling while minimizing data transfer and storage costs.

To solve the problems stated in Section 2.1 as efficiently as possible, identification of the common components across DAA and outside data logs would enable the aggregation of information in either signal direction and still get the desired visibility of network health and customer experience. Considering the current DAA architecture, aggregating telemetry data points at the RPD level would enable us to focus on a specific RPD and its associated cable modems, or aggregate it to PPOD/site/vendor level. Aggregating at a device level, by contrast, would create millions of rows of data per metric, which would be computationally intensive and would not provide a view that would help operations or the DAA engineering team in understanding network/platform health. Aggregating data at a PPOD level would mask the issues encountered at an RPD level, given the mix of device types, software versions running on the RPD or vendor type.

2. Implementing Sherlock: a Big Data Analysis Architecture for DAA

As discussed, our DAA data is generated from many different components and are stored in a variety of different specialty database systems. The individual systems are customized specifically for the applications. While having compartmentalized data storage solutions for each type of data is simpler from a development and maintenance standpoint, it can make analysis tasks that require several data sources quite cumbersome.

To allow for efficient analysis of all relevant DAA data with minimal manual operations to join, clean and analyze the data, Sherlock was built using a big data analysis architecture. Sherlock has the ability to interface with a variety of existing cloud and on-premise data storage solutions (APIs, SQL databases, Prometheus, AWS), and combine all the relevant data for efficient analysis.

Building a centralized framework to combine all the different data sources, however, is only half the battle. This task is even more challenging considering the growing scale of the DAA data streams. Consider: A typical RPD has thousands of metrics that are stored at 15 second intervals, and each PPOD can link hundreds of RPDs, so, in total, a single PPOD will generate billions of data points a day. Given that we are continuously deploying new vCMTSs, Sherlock needs to be able to scale with the growing DAA footprint and require minimal maintenance. It is easy to see why this operation must be automated,

since it is not feasible to expect a team to manually process data at this scale and frequency. This section discusses the requirements, implementation, and core features of Sherlock.

2.1. Requirements

Sherlock is meant to be used by the DAA deployment and operations teams to actively monitor and address any DAA deployment and operational issues. Thus, the tool must meet the following requirements:

- Full footprint coverage
- Scheduled analysis reports
- Ad hoc analysis abilities
- Fast computation

Those requirements ultimately allow for ML to discover hidden trends and patterns in the data. An overview of the requirements is presented below.

Since the vCMTS deployments are occurring nationwide, Sherlock must be able to analyze data at different levels of aggregation, from a single RPD to a headend and all the way up to the national level. The varying levels of analysis allow experts to not only understand issues with a single RPD but understand if that same issue exists elsewhere and to what extent.

Sherlock should be able to perform automated analyses and generate reports on a schedule so the deployment team can consistently monitor performance. The scheduled runs can be weekly or even daily if needed. The automated runs should produce a concise and consistent output to enable efficient tracking of performance metrics.

Even though scheduled analysis runs are great for consistent summaries of deployment statuses over a known time window, there will inevitably be ad hoc analysis tasks that require a specialized analysis and output. Therefore, Sherlock should also have a manual interface to easily interact with the core data structures and perform a specialized analysis if needed.

Finally, Sherlock must be computationally efficient when performing operations. There is no specific metric for this requirement, but the general motivation is that the computational framework should support the frequency of the scheduled runs in the above requirement. In addition, ad hoc analyses should be able to be completed in a reasonable time frame. That is, if it takes 10 hours to compile the data and prepare an analysis, the tool would not be useful. To allow for efficient interactions and analyses, computational operations should take only a few minutes in general, such that the analyst can stay engaged while working with the data.

2.2. Implementation

Given the requirements listed in Section 3.1, significant effort went into developing Sherlock's implementation, such that all requirements would be met, while still allowing future scaling as DAA continues to grow. The initial stage of development was to become familiar with the data sources, and in this stage, it became very apparent that an advanced solution would be required.

The first implementation attempt was to load the DAA telemetry data with basic Python packages directly via API requests to the DAA time series database. While this was quickest way to start accessing the data and start developing a plan for how the data should be compiled, cleaned, represented, and analyzed, it was not performant enough to meet the design requirements. Using this approach along with standard

Python parallel processing packages, it was taking hours to load a week's worth of data from just a handful of RPDs (representing a very small fraction of Comcast's RPD footprint). It became evident that a more computationally-efficient and robust solution would be required. At this stage, the first implementation was deliberately done at very small scale for exploratory analysis (for determining useful telemetry metrics and experimenting with different processing/visualization methodologies).

Once it was clear what data was important and how it was going to be analyzed, the team designed a production system to meet all the requirements. The diagram of the implementation is shown in Figure 2. The chosen implementation solution utilizes Amazon EC2 (Elastic Compute Cloud) computing resources, which can scale to the meet the needs of a specific task. The source code is written in Apache Spark, which is an open source distributed processing framework specifically for big data. Databricks is used as the resource management system that manages Spark sessions and coordinates the EC2 instances to complete the computation tasks. The raw data is processed and stored in Delta Lakes[1] that are optimized for efficient reading and writing of big data on the distributed Spark framework.

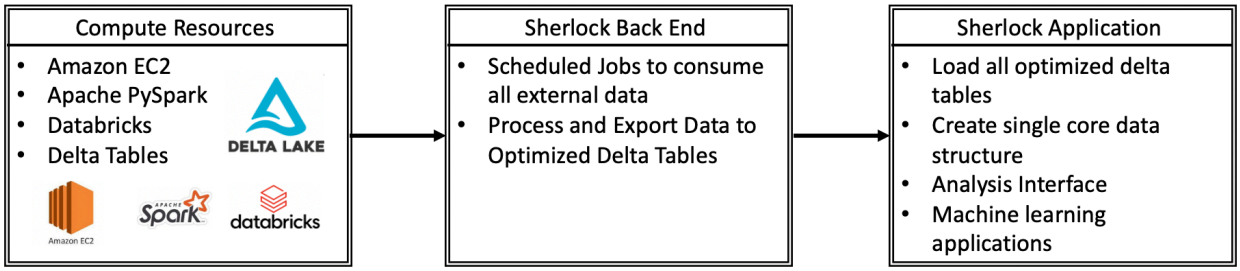

Figure 2 - Sherlock Implementation Diagram

Once the computing and storage frameworks were stood up for use, the data engineering team built the Sherlock backend. It consists of a pipeline of scheduled jobs that consistently read in the raw DAA data from the variety of sources previously discussed, to process them into efficient formats. The resulting data structures are saved to Delta Lakes for efficient access from the main Sherlock application. With the backend responsible for acquiring the data and pre-processing it, the main Sherlock application can then just reference these extremely efficient tables at analysis run time. The main application then uses this centralized data structure to perform a variety of analyses, which are discussed in later sections, as well as to provide the base data set for ML applications. An in-depth discussion about the main features/modules of the Sherlock application can be found in Section 3.3.

Once the end-to-end architecture was developed, the performance against two of the main requirements (full footprint coverage and fast computation) were evaluated. An analysis pipeline, including data loading, processing, event detection and plotting, was run for the entire DAA footprint (representing thousands of RPDs) for a one-week duration. This pipeline finished in less than one hour – the same amount of time it took to merely load 20 RPDs worth with standard Python processes. Similar analyses on subsets of the footprint take less than 20 minutes, and in the future the aim would be to build specialized pseudo-real-time operations that can be performed much quicker (such as real-time event detection with ML). These performance metrics surpass the design requirements discussed earlier. Scalability with the DAA framework also doesn't seem to be a concern at this point, either, due to the scalable and distributed computing framework provided by Databricks and Spark.

[1] Delta Lake "…is an open-source project that enables building what is called a Lakehouse architecture on top of existing storage systems such as S3, ADLS, GCS, and HDFS." For more information, see https://delta.io/

2.3. Features

Sherlock has five modules that handle different aspects of the analysis functionality: core, visualization, event classification, ranking and ML. The breakdown of module responsibilities is shown in Figure 3.

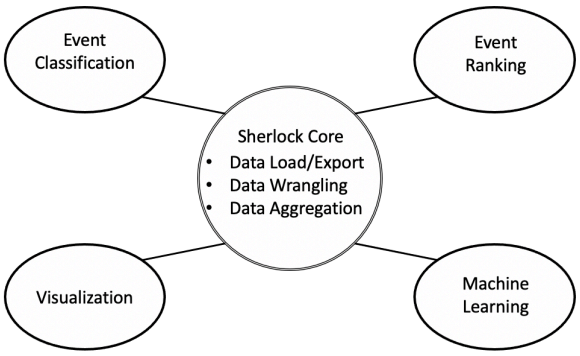

Figure 3 - Diagram of Sherlock Modules

The core module handles most of the data integration, such as loading the data from all the supported sources and converting to a time aligned view for each RPD. The core data structure in Sherlock is a Spark DataFrame with rows for each five-minute timestamp, for every RPD of interest, and columns for all the available metrics. Having the lowest level of aggregation at the RPD level was chosen since this allows for easy aggregation up in higher levels like sites, divisions, vendors, etc., while still being able to meaningfully aggregate CPE-level metrics (for example, upstream transmit and receive power). This core data structure is the foundation for the other modules.

Once the single time aligned view is created, the event classification module identifies events of interest. This module has a fully configurable pipeline that runs through a variety of event detection algorithms and combines the results into a summary table with the event type, start/end times and any other useful metadata about the event. The specifics of the event classification pipeline and logic are discussed in detail in Section 4.1. At this stage, any events identified for individual RPDs are available for use by other modules to support in-depth analyses.

The ranking module uses the events identified from the event classification module to rank the RPDs/sites, from worst to best, and prioritize any issues on the network. The ranking is performed at the RPD level and can thus be aggregated up to other levels as desired. Weights are assigned to each event based on several factors, and the final RPD ranking is the weighted sum over the events for each RPD. The ranking algorithm is discussed in further detail in Section 4.2.

The RPD/site-level rankings are then used to filter down the raw data to regions where there are a lot of interesting characteristics requiring investigation. The visualization module is then used to create charts that highlight the specific areas where the issues occurred. Currently, Sherlock generates plot files on request, given that the project is still in development at this writing (summer 2021). However, once the concept views are finalized, a dashboarding solution with all the views will be stood up to allow for easier access to plots. The two main plot types are RPD timeline and site timeline. An example RPD timeline plot is shown in Figure 6.

The RPD timeline views combine a wide variety of data. Although these plots are currently stand-alone files, they are fully interactive HTML plots. Even though Sherlock is in the development stage and production dashboards are not implemented yet, users of the output significantly benefit from the ability

to zoom/pan on events of interest. Having this ability allows for more productive interactions with the visualizations and serves as a proof of concept (PoC) for production visualization, to gather feedback for the production views. Figure 6a contains the AUX and GCPP core statuses. Figure 6b shows the battery level for the backup power supplies feeding the RPD. Figure 6c displays the CPE status information including counts of devices online/offline, and in different partial service states across the RPD. Figure 6d shows US/DS traffic in the form of total octets transferred, as well as US forward error correction (FEC) percentages (including unerrored codewords [UECWs], corrected codewords [CCWs] and uncorrected codewords [UCCWs]) across all interfaces on the RPD. Figure 6e contains all event-based information, including customer calls, automated alerts, technician repair tickets, and RPD SW/HW updates. Figure 6f contains the DAAS switch status between the RPD and vCMTS core.

Sherlock is the first tool in Comcast to make all this data readily available and digestible in a concise visualization. It is easy to see how powerful this timeline view is, as it allows clear visibility into events across the entire architecture, from vCMTS statuses in the headend all the way to customer experience and contact.

While the RPD view is very useful for deep-diving into specific events affecting areas of the DAA network, it doesn't easily allow for accessing the scale of the event. For example, power outages would likely affect multiple RPDs at time, whereas other issues, like noise ingress, are likely to be very localized. To help visualize and assess the scale of the events across the footprint for a given time window, Sherlock produces event heatmap plots, as shown in Figure 4. This view aggregates the individual RPD levels to the site level. The time dimension is hourly and the heatmap shows the count of RPDs that experienced a specific event in each hour. This type of view allows for a very quick review and determination of how widespread specific issues are across a given aggregation level.

The final module in Sherlock is the ML module, which can utilize the results of the other modules. The ML portion of Sherlock is talked about in detail in Section 6. The main goal of this module is to use all the core data generated through Sherlock's operations as training data for ML algorithms. Given the obvious richness in the DAA data set, there is a clear benefit to applying novel ML applications to mine the dataset and uncover complex patterns. The immediate use cases are:

1. Finding similar events via clustering.
2. Using pattern recognition to discover complex patterns and relationships across the vast dimensions of the data set.
3. Prediction of future issues based on current data.

These applications, if successful, have the potential to completely transform the DAA space. More discussion on the ML aspect of Sherlock is presented in Section 6.

3. DAA Event Classification and Rankings

As part of building a reliable and robust access network to deliver fast speeds to customers, we need to ensure our plant, network and platform health are constantly monitored to proactively detect and mitigate issues and reduce impact to customers.

There are cable industry standards and specifications which are widely used within Comcast to characterize the health of the HFC plant, CMTS and the connection to cable modems. Some of the more common metrics that are tracked are signal-to-noise ratio (SNR), modulation error ratio (MER), transmit power, receive power, and FEC, to and from CMTS. Each of these metrics has acceptable ranges. When telemetry data point goes above or below those ranges, the variance and the duration could indicate

different issues within the network. Comcast tooling currently classifies and captures those anomalies and alerts are sent out. Sherlock leverages those existing alerts to measure DAA plant health.

As stated previously, it becomes an operational challenge to mine through all the data using Grafana or existing dashboards, to pinpoint where and when things go wrong. Thus, Sherlock implements a scalable event detection pipeline to automatically detect events of interest. Sherlock can then leverage the detection of events to rank sites, PPODs and RPDs, based on the occurrence of said events, and help prioritize teams to address issues.

3.1. Event Classification

Sherlock makes use of the centralized core data structure with the telemetry metrics discussed in Section 2.2 to implement an automated event detection pipeline at several levels of aggregation. The lowest level of aggregation is currently the RPD level, while events can also be detected at PPOD level and above. The individual event criteria are specified as objects in the pipeline, and then the pipeline executes each object on the raw data to detect events.

Currently, the events are logical/threshold-based, because that is a great starting point to easily identify any known types of events. The types of events include offline status, anomalous trends in each metric, RPD reboots, SW/HW upgrades and even customer contact. While a single event only looks at specific metrics, the pipeline groups multiple events into a single event to infer when a more complex event is happening. This is especially useful in cases of known maintenance, such has RPD SW/HW upgrades, since these events will undoubtably cause outages and anomalous traffic patterns. This ability allows us to connect any maintenance/upgrade events to corresponding outages so they can be scored differently than unplanned outages.

The event pipeline is typically run on a weekly basis, which allows for analytics and tracking to be performed to document the type and frequency of events across the footprint. The event findings are summarized and automatically distributed to the DAA team to evaluate.

A brief description of the currently implemented events for the RPD level are presented in Table 2. As we continue to add new telemetry metrics and develop the existing list of events, this area will be constantly fine-tuned.

Once these events are identified in the RPD data, they can be aggregated up to higher levels to determine whether events are local, or more widespread. As previously mentioned, Figure 4 displays a visualization known as an event heatmap, which shows the number of RPDs in each site exhibiting a given event at a given time throughout a week. In this example, most events are a spread across a few RPDs, however, there are pockets of “no event found” flags that occur daily at the same time. These specific events were determined to be nominal nightly CPE reboots where the DAA system is healthy, but most CPE devices are offline performing scheduled updates. These types of dense views provide an extremely useful view of the network at a glance and easily display any major issues on the network that would require further investigation.

Table 2 - Description of event classifications

Event Name	Event Description
auxOfflineEvent	Aux core is offline
DAASOfflineEvent	DAAS port is offline
datagapEvent	Telemetry drop outs
gcppOfflineEvent	GCPP core is offline
keepAlivesEvent	RPD keep alive counter is non zero
noEventFound	No other event is flagged
rpdBatteryDrainEvent	RPD is on backup battery power
rpdHealthyEvent	RPD is online with expected CPE connectivity
rpdIopEvent	Automated ticket assigned to RPD
rpdIvrEvent	Customer support call
rpdMacUpdateEvent	RPD hardware was changed
rpdRebootEvent	RPD rebooted
rpdSWUpdateEvent	RPD software was changed
rpdTcEvent	Technician dispatched
usFecEvent	US FEC UCCW exceeded threshold
usOctetsEvent	US RPD traffic out of family
usPartialServiceEvent	Number of CPEs in US partial service above threshold
usPerfEvent	OpTek US System Alert

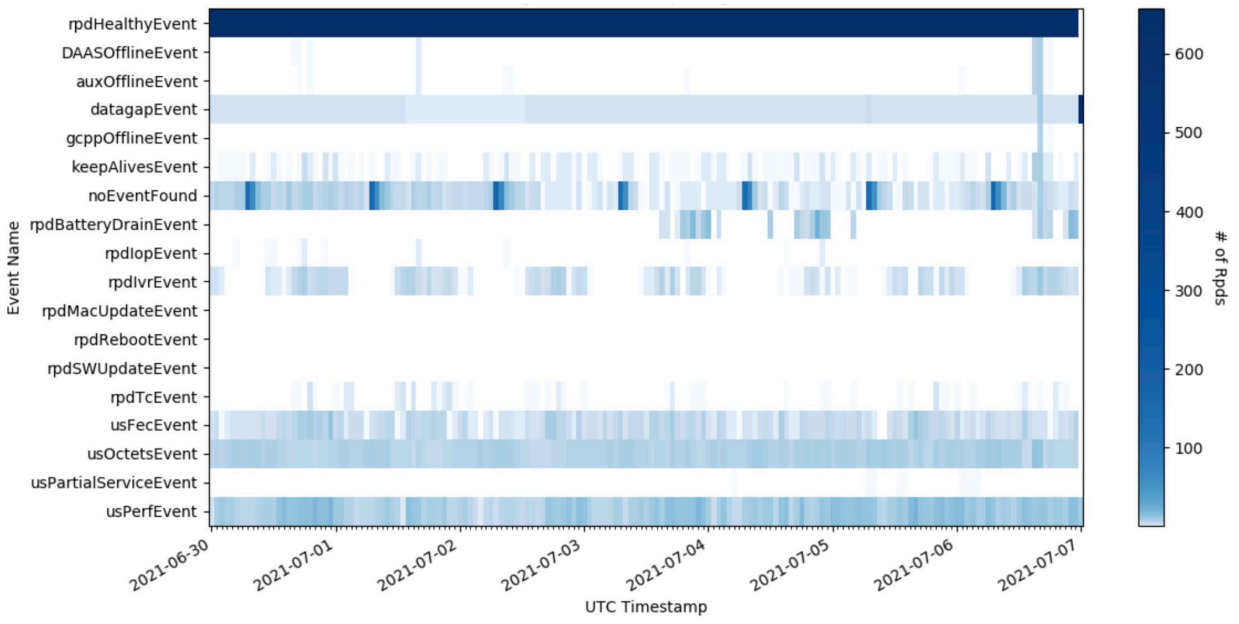

Figure 4 - Event Heatmap for a Single Site Over a Week

3.2. Ranking Methodology

Based on the events mentioned in the previous section, Sherlock ranks sites, PPODs or RPDs with events that are impacting customers for any given duration the most, and assigns them the highest rank. Ranking is aimed to identify sites, PPODs or RPDs with widespread events or events that occur frequently, causing service disruption to customers.

Individual events are assigned weights based on severity, duration, and customer impact. The final ranking is thus the weighted sum of the event coefficients to determine the most impaired sites/PPODs/RPDs. There are some nuances that go into ranking, where events that occur during a nightly maintenance window are ranked lower than events during non-maintenance windows for the same duration and frequency. Additionally, most of the events that occur immediately after RPD HW/SW changes are assigned a lower rank or aren't counted, as the entire system resets to clear current configurations and make the assigned changes. While customer contact events are not directly used in the weighted sum, because of the reasons discussed in Section 2.2, they can be used to break ties when two items have the same score.

Figure 5 shows the workflow that takes the telemetry data through event classification and ultimately to the final rankings. Once the rankings are complete, they are sent to the DAA team for evaluation. Section 4.3 discusses how the rankings are used.

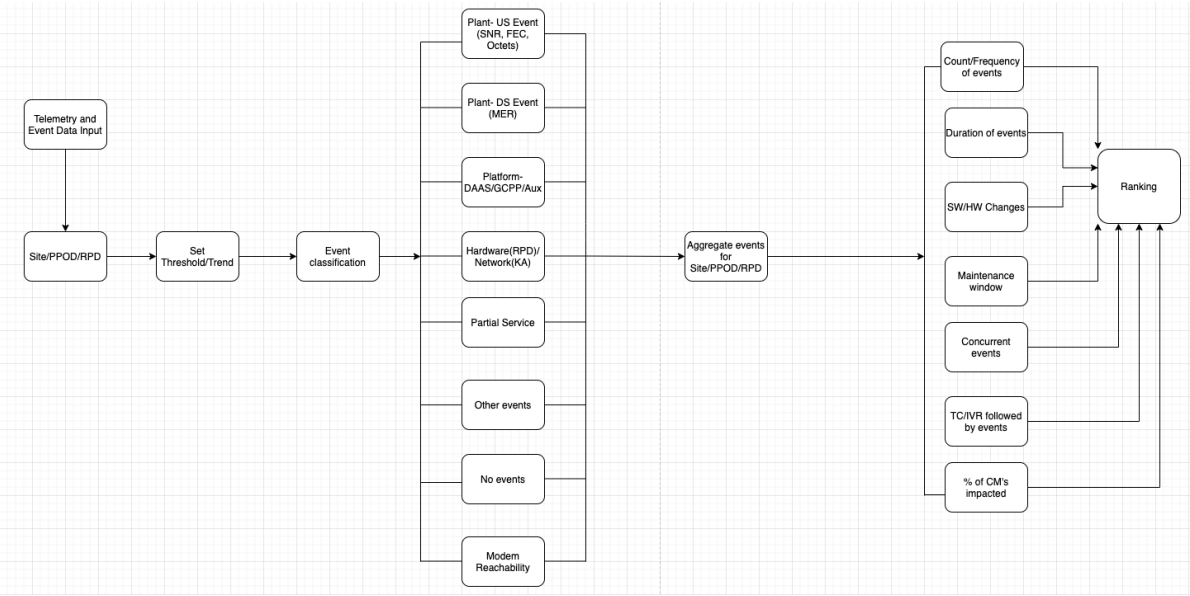

Figure 5 - Sherlock Workflow for Event Classification and Ranking

3.3. Ranking Usage

Once Sherlock generates weekly rankings for the entire footprint, or ad hoc rankings based on business needs, reports and views are stored within AWS. Reports capture highest ranked sites and RPDs based on events listed in Section 4.1, and list all the different events that were captured for that time period. This report serves as a starting point for engineering and operations teams to pinpoint any potential issues within the vCMTS architecture.

Using the ranking report, the heat map views at the site and PPOD are used to quickly see if the issue is widespread or isolated, and if there are specific events that occurred more frequently than the other events. The next layer analyzes the time series view at the RPD level and compares all the metrics that feed into Sherlock to ascertain end-to-end system health. RPD-level views also provide an events summary, with event duration and type, with the functionality to zoom into those events. Each event is assigned a unique ID, which helps in further analysis. This way we can quickly identify events at various levels in the network, and interdependencies between events as well as impact on customers. Ranking and views generated by Sherlock also help in flagging changes made to the system, such as hardware or software upgrades that caused a specific set of events and customer impairments.

Sherlock reports provide week-over-week trending, which helps in highlighting chronic vs. transient issues. Since homes-passed-per-RPD are relatively small in DAA, compared to analog nodes, some of our existing tools can prioritize the number of customers affected by events that would otherwise under-rank DAA issues. As such, Sherlock reports are specifically designed to better understand the DAA network and highlight problem spots in digital nodes, regardless of the number of homes passed.

Sherlock has an integration point with our internal messaging service where reports and visualizations are posted. Plans are underway to migrated to a web-based UI to provide enhanced analysis functionalities to DAA teams.

4. Practical Use Cases and Example Findings

In this section we illustrate the power of Sherlock with examples of specific events and views that highlight the health of the system.

4.1. Noise/Ingress

Table 3 and Figure 6 indicate an upstream noise event which is seen in the form of elevated UCCW. During the upstream noise event, plant and hardware elements (Figure 6a, b, c, e and f) appear to be functioning normally, whereas Figure 6d shows a drop in CCW and a corresponding increase in UCCW. In a DOCSIS plant, transient noise is a normal upstream event and the impact to customer service is minimal. Several tools already exist to sends alert on such events. Table 3 shows a sample of events determined by Sherlock during a portion of this noise ingress example. Most notably, Sherlock identifies usFecEvents as well as usPerfEvents (OpTek) events, indicating that our existing upstream performance monitoring tools are catching these events as well. The exact details of the OpTek events could be overlayed to determine more specific information, such as number of interfaces affected, etc.

Table 3 - Sample Event List for a Single RPD During US Noise Event

Event Type	Event Start Time (UTC)	Event End Time (UTC)
usFecEvent	6/25/21 9:35	6/25/21 9:35
usPerfEvent	6/25/21 16:35	6/25/21 22:35
usPerfEvent	6/25/21 22:50	6/26/21 2:15
usFecEvent	6/25/21 23:50	6/25/21 23:50
usFecEvent	6/26/21 1:05	6/26/21 2:00
usPerfEvent	6/26/21 2:35	6/26/21 3:10
usPerfEvent	6/26/21 4:40	6/27/21 0:35

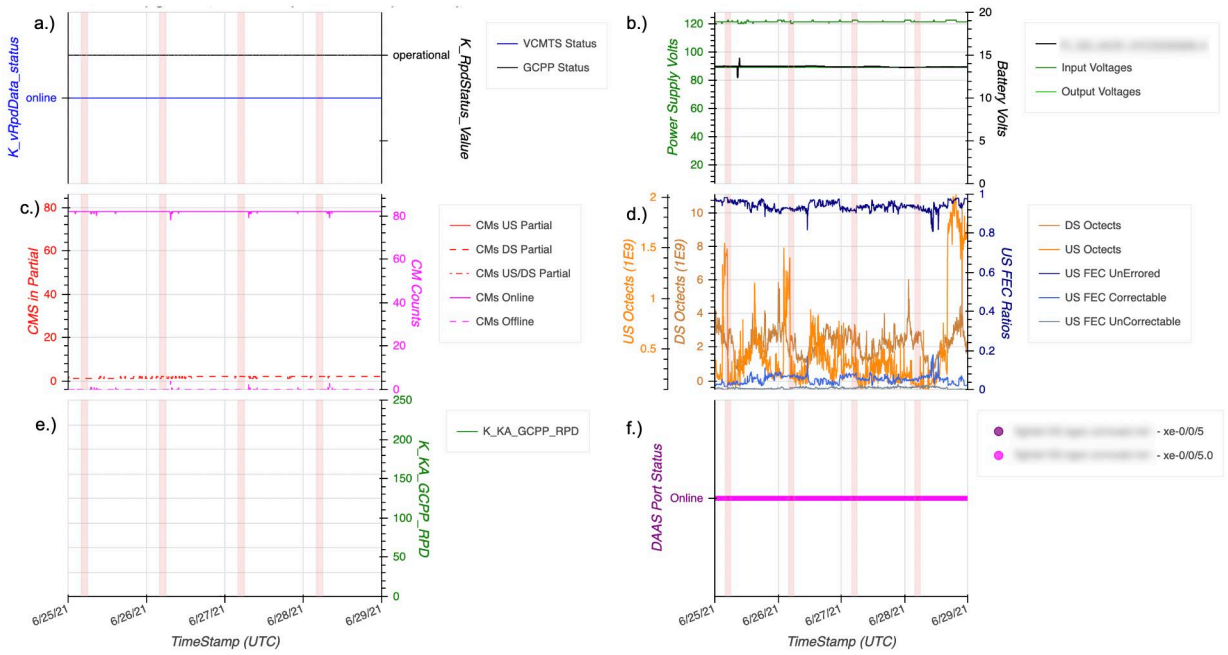

Figure 6 - Time Series View of RPD Noise/Ingress Event: a.) vCMTS/GCPP Statuses, b.) RPD Power Supply, c.) CPE Counts, d.) Traffic and US FEC e.) System Events (Customer Calls, Automated Alerts, RPD SW/HW Changes and f.) RPD Switch Status

4.2. SW/HW Upgrades

Table 4 and Figure 7 show a planned software update. This event is marked in Figure 7e with the initiation of the event as the RPD reboot/reset. Then corresponding dynamics across the other telemetry metrics are shown in Figure 7a, c, d and f. When the RPD software is updated, the AUX core goes offline, the CMs go offline (with some partial service along the way), traffic dips below nominal levels and the DAAS port also goes offline. Having the context of a software upgrade is key, since under other circumstances these types of system responses would be not ideal. However, software upgrades are performed during maintenance windows to minimize customer impact.

A sample of the event list is given in Table 4 showing a subset of the events identified in this time frame. It is worth noting that since the RPD reports its SW version at any given time, software upgrades like these can be determined directly from the data without relying on a ticketing system or external dependencies.

Table 4 - Sample Event List for a Single RPD During SW Update

Event Type	Event Start Time (UTC)	Event End Time (UTC)
auxOfflineEvent	6/17/21 14:50	6/17/21 14:50
usOctetsEvent	6/17/21 14:50	6/17/21 14:50
rpdRebootEvent	6/17/21 14:50	6/17/21 15:45
rpdIopEvent	6/17/21 14:50	6/17/21 14:50
keepAlivesEvent	6/17/21 14:50	6/17/21 14:50
DAASOfflineEvent	6/17/21 14:50	6/17/21 14:50
datagapEvent	6/17/21 14:50	6/17/21 14:50
rpdSWUpdateEvent	6/17/21 14:55	6/17/21 14:55

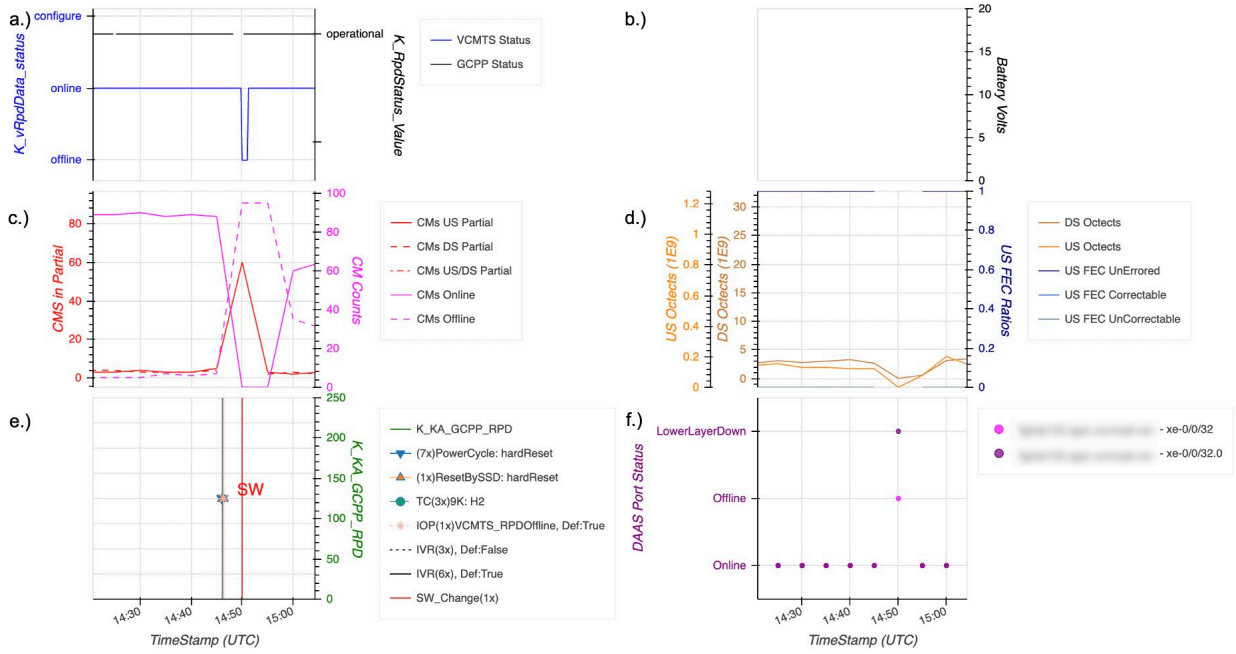

Figure 7 - Time Series View of RPD HW/SW Event: a.) vCMTS/GCPP Statuses, b.) RPD Power Supply, c.) CPE Counts, d.) Traffic and US FEC e.) System Events (Customer Calls, Automated Alerts, RPD SW/HW Changes and f.) RPD Switch Status

5. Machine Learning Applications

As discussed earlier, the data aggregations and processing done by Sherlock represent an ideal setup for ML applications. The DAA data spans many dimensions, and ML/data mining techniques should be used to extract as much useful information as possible to optimize deployments and, ultimately, customer experiences. While the Sherlock ML module is nascent, three immediate use cases are currently being explored.

Clustering: The first application of ML is clustering. Clustering attempts to find smaller groups of similar samples in the raw data. This is especially useful when trying to determine if certain populations of data are more impacted than others, as well as understanding if there are subgroups of data inside major groups. An example of how this could be applied in the DAA space would be taking an RPD event classified via the methods in Section 4.1 and attempting to see if there are sub-populations of RPDs that experienced a given event for different reasons to help triage the event. Specifically, if partial service events are identified, clustering would be able to determine if some RPDs have CPEs in partial service mode because of noise ingress, platform-related issues, configuration issues, scheduled maintenance or even CPE-specific issues. This information could then be used to address the root cause of the individualized partial service issues. A simplified example of this is shown in Section 6.1.

Pattern Recognition: The next application of ML is advanced pattern recognition. As discussed earlier and as per industry standards, events on the network are typically identified via logic-based threshold exceedances, where an event is identified when a certain metric exceeds a predetermined value. While this is useful is many cases, it is limited when it comes to multi-dimensional events with complex relations, because completing a comprehensive detection algorithm with nested if/then logic becomes very cumbersome and hard to maintain. This is where ML shines: If example patterns in the data can be labeled by experts, models could be trained to find the important relationships across many different metrics to identify more complicated patterns than traditional logic-based approaches. An example use case of this in DAA could be identifying RPD backup battery degradation by looking at current and voltage drain during power outages. This application is in development.

Prediction: The third initial application of ML in DAA is the prediction of future issues given real-time data. At Comcast, customer experience is paramount and the ability to forecast and address issues before customers are aware of them is groundbreaking. Given the expansive coverage and real time nature of DAA telemetry, it is possible to use ML methods to find leading indicators of customer impacting events that are classified by the methods discussed earlier. An example of this for DAA could be forecasting when core server load will be too high, to the point of potentially shutting down.
This can be proactively addressed to obviate an outage. This application is also in development.

5.1. Clustering Example

This section presents a real-world use case for clustering DAA data. The example used here is trying to identify clusters of issues that cause partial service events at the RPD level. Using the Sherlock event classification module, partial service events were identified across all RPDs, where an event is classified as 25% or more of CPEs are in US partial service mode for at least 15 consecutive minutes.

Since the Sherlock data is very high dimensional and time-based, the first action is to perform a dimensionality reduction, to help the model focus on important features of the data. Several methods are possible here: traditional feature extraction/engineering, principal component analysis and even auto-encoding neural networks. This compresses the data into a smaller feature domain for the model to learn patterns. The architecture for clustering is shown in Figure 8.

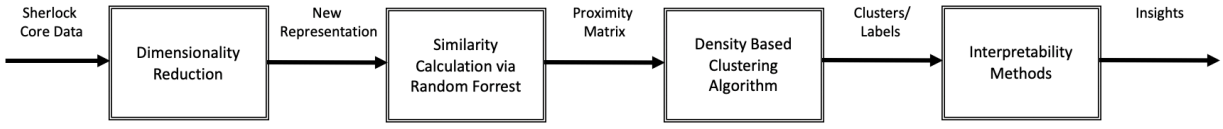

Figure 8 - Clustering Architecture

Clustering algorithms require a distance/proximity matrix that contains the distance between every pair of points in the sample data. This is often an area that requires a significant amount of tuning, because the choice of distance metric used has a huge impact on results. In most cases, Euclidean distance is used. However, in cases where the data set contains mixed continuous and categorical data, Euclidean distance hardly makes sense and typically custom distance metrics are derived for the specific problem at hand. Developing custom distance metrics can be extremely time consuming. For this reason, Sherlock is using a relatively novel distance calculation that relies on a type of ML method called "random forests," which can handle continuous and categorical data simultaneously with minimal pre-processing. The random forest is run on the data with training mode off, which essentially splits the data based on inherent similarity (typically entropy or Gini index.) The result can be turned into a proximity matrix as the number of times pairs of samples ended up in the same leaf node across all the trees in the forest.

The proximity matrix is then passed to a clustering algorithm to attempt to find clusters. Since the underlying structure of the data is not known a priori, representation-based clustering algorithms such as K Means are likely not a good fit. Instead, density-based methods such as density-based spatial clustering of applications with noise (DBSCAN) are utilized, since they make no assumptions about the shape/structure of the clusters. DBSCAN is also a good choice, since it does not require the desired number of clusters to be specified and instead attempts to identify the ideal number of clusters as well as any outliers. Once the clusters are identified, interpretability techniques should be employed to identify what clusters represent in the real world.

Once the cluster labels for each sample are determined, the data can then be passed to an interpretable ML classification algorithm. Essentially, the raw data and the corresponding cluster labels are used to train a classification model. In this case, the model is a single decision tree, to determine the path a sample takes to its classification target. Once the model is trained, it can be investigated to understand if the cluster labels have any real-world meaning.

In the example of US partial service clustering, the event pipeline discussed in Section 4.1 was run on the full footprint of DAA for three weeks, during which 897 partial service events were identified. The features for the clustering algorithm are a wide data table with Boolean flags for other events that occurred in proximity to the USPartialServiceEvents. In this example, the only other events considered were usFecEvents and rpdRebootEvents (to simplify the analysis); all events would be considered in a full analysis. The clustering algorithm was able to identify four clusters. Those four clusters were then passed as labels in addition to data as training samples to a decision tree. The resulting decision tree is shown in Figure 9.

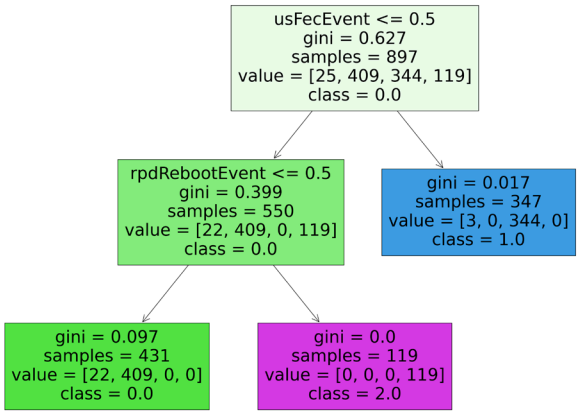

Figure 9 - Interpreting Cluster Results via a Decision Tree

From Figure 9, one can see the learned set of rules the tree uses to assign the cluster labels to a given US partial service sample. A tabular summary of the tree paths for each class is given in Table 5. From here it is easy to see that the decision tree can learn the cluster meanings relatively well. It is worth mentioning that the cluster label -1 is provided from the DBSCAN algorithm as outlier points that don't fit in to particular clusters, thus the decision tree has no path to correctly label those samples.

Table 5 - Decision Tree Paths for Clusters

Cluster Label	Number of Samples	Decision Tree Path	Decision Tree Label Accuracy
0	409	(usFecEvent <= 0.5) and (rpdRebootEvent <= 0.5)	95%
1	344	(usFecEvent > 0.5)	99%
2	119	(usFecEvent <= 0.5) and (rpdRebootEvent > 0.5)	100%
-1	25	N/A	N/A

The results from the above clustering example can be used to identify why RPDs experience widespread partial service events and lead to further mitigation-related enhancements to try in the future. In this case, partial service seems to be driven by usFecEvents and rpdRebootEvents. Further work could be done to understand the samples that had no usFecEvent and no rpdRebootEvent. While the results from this analysis are not too surprising, the architecture is a springboard for correlating different types of events and trying to identify where to dig deeper in understanding non-trivial issues. It is easy to see how this type of analysis could be expanding to more complex issues like understanding sporadic auxOfflineEvents, provided the correct data was fed to the ML architecture.

6. Conclusion

When Comcast began deploying DAA, the need for an automated big data analysis framework was immediately apparent. The DAA framework enables extremely rich telemetry with high frequency sampling rates, making two things true: 1) manual data analysis was infeasible, and 2) exposing the large amounts of data that is perfect for ML-based analysis. Our solution, internally called Sherlock, combines high fidelity data from a variety of sources across our physical infrastructure into a single centralized data structure that can be easily accessed for an assortment of analyses. Creating a centralized data structure with relevant DAA data proved to be instrumental in providing actionable insights from Sherlock analyses.

Sherlock was implemented using state-of-the-art technology that will allow for future scaling as we continue to grow our DAA footprint. Sherlock's core data structure allows for a multitude of analysis implications including event detection, event ranking, visualization, as well as ML advancements. These features allow us to identify and prioritize system issues at a glance, whereas such analyses were previously much more involved and required many manual operations. These analyses are currently being used by our internal teams to monitor DAA deployments and overall system stability.

While Sherlock is a relatively new tool, it is already starting to expand with applications to enhance the power of the insights provided to the DAA teams. As part of this work, we are exploring ML applications to find important trends in the complex DAA data. The immediate ML applications include clustering, pattern recognition and future event prediction. We are also working to integrate Sherlock into our expanding network topology graph, which will also open up new possibilities for advanced insights on the DAA network. The goal was and is to build a platform that can identify issues and recommend preventive maintenance before customers are impacted. Sherlock has proven to be extremely powerful in

its present state and is expected to become even more useful as the next generations of features are developed.

Abbreviations

API	application programming interface
AUX	auxiliary
AWS	Amazon Web Services
CCW	corrected codewords
CM	cable modem
CMTS	cable modem termination system
CPE	customer premise equipment
DAA	distributed access architecture
DAAS	distributed access architecture switch
DBSCAN	density-based spatial clustering of applications with noise
DOCSIS	Data-Over-Cable Service Specifications
DS	downstream
EC2	[Amazon] Elastic Compute Cloud
FEC	forward error correction
GCP	generic control plane
GCPP	Generic Control Protocol Principal
HAGG	headend aggregation switch
HFC	hybrid fiber/coax
HTML	hypertext markup language
HW	hardware
ID	1) identification; 2) identifier
IP	Internet Protocol
MER	modulation error ratio
MIB	management information base
ML	machine learning
OS	operating system
PHY	physical layer
PoC	proof of concept
PPOD	physical point of deployment
QAM	quadrature amplitude modulation
RPD	remote PHY device
R-PHY	remote PHY
SCTE	Society of Cable Telecommunications Engineers
SNR	signal-to-noise ratio
SQL	structured query language
SW	software
TCP	Transmission Control Protocol
UCCW	uncorrected codewords
UDP	User Datagram Protocol
UECW	unerrored codewords
UI	user interface
US	upstream
vCMTS	virtualized cable modem termination system

Bibliography & References

https://www.cisco.com/c/en/us/td/docs/cable/remote-phy-devices/rpdsw51/b_rphy_system_startup_config_5_x/gcpp_support_for_remote_phy.pdf

https://www.cisco.com/c/en/us/td/docs/cable/remote-phy-devices/configuration/guide/b_rphy_management_8_x/rpd_reset_8x.pdf

http://mibs.cablelabs.com/MIBs/DOCSIS/

https://www.nctatechnicalpapers.com/Paper/2018/2018-node-provisioning-and-management-in-daa

