THE SELF-SUFFICIENT TICKETING RESPONSE SYSTEM FOR THE RAJANT SUPPORT NETWORK

A Thesis

Presented to

the Faculty of the College of Science

Morehead State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Maria E. Lemaster

April 26, 2019

ProQuest Number: 13864956

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 13864956

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Accepted by the faculty of the College of Science, Morehead State University, in partial fulfillment of the requirements for the Master of Science degree.

_____
Mr. Jeff Kruth
Director of Thesis

Master's Committee: _____, Chair
Mr. Jeff Kruth

_____
Dr. Ben Malphrus

_____
Dr. Charles Conner

_____
Date

THE SELF-SUFFICIENT TICKETING RESPONSE SYSTEM FOR THE RAJANT SUPPORT NETWORK

Maria E. Lemaster
Morehead State University, 2019

Director of Thesis: _____
Mr. Jeff Kruth

The proposed project is expected to be a self-sufficient response software program that will be client (company) and ticket-specific. This will involve a written Python code that will momentarily analyze the incoming ticket requests, identify whether or not the problem has previously occurred, and respond with appropriate instructions tailored to the customer's needs. The output files generated by this code will read through a Kinetic Mesh's® snapshot logs, identify inconsistencies within those logs and produce an appropriate response to the issues within a network. As of now, this is all a manual process.

Engineers look at the incoming tickets, analyze the problem, and then offer solutions. This proposal in turn, will provide the consumers using the ticketing system with support from Rajant Corporation's engineers on twenty-four hours a day, seven days a week basis with little to no hassle. Any difficulty that needs immediate attention can be resolved by the engineers, recorded in the ticketing system, and will be recognized by this proposed program if that

problem ever arises from a client again. This will also provide the customers with a maintenance report of their personal network. These ideas can take out the bulk of the previously tedious steps and make it much easier and much more organized to assist the client's needs.

This development plan was chosen because it was a part of a work assignment primarily dealing with Rajant's support ticketing system as previously defined. This project should take roughly 6 months to complete including documentation, production, and presentations. The assets required for this assignment will be a support software that can combine well with the ticketing website (Python being this source), report writing software called LaTeX, as well as the resources available in the Rajant Corporation Office at the Morehead, Kentucky location. The main project mentor will be one of Rajant's head software engineers, Marty Lamb; also, J. Kruth will be overseeing the progression of the project between now and April 2019.

Accepted by:

_____, Chair
Mr. Jeff Kruth

_____
Dr. Ben Malphrus

_____
Dr. Charles Conner

ACKNOWLEDGMENTS

I would like to acknowledge everyone who played a role in my academic accomplishments. First of all, my parents, who supported me with love and understanding. Without my family, I could never have attained this present level of achievement. Secondly, my committee members, each of whom has provided patient advice and guidance throughout the research process. Thank you all for your unwavering support. Thirdly, my coworkers within the support department and Rajant Corporation itself, who have administered persistence in this project and have contributed partnership throughout the entire process. Without you all this project would not have succeeded in the way it did.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

KEY WORDS

- Kinetic Mesh: Rajant Corporation's data networking ability.
- Bread Crumb: Rajant Corporation's mobile radios.
- BC|Commander: Rajant Corporation's constantly upgraded firmware that runs Bread Crumbs.
- BC|Assurance: The newest Rajant service that will provide network support to customers quickly and efficiently.
- Support: Assistance that is given to customers' problematic networks. Offered 24 hours a day, 7 days a week. (Currently a free service to customers)
- Ticket: The form as which the issues come into Rajant's support team. Customers send these via email or through a phone call.
- Zendesk: The current interface in which support tickets reach the support team.
- Odoo: The future interface for support tickets and other platforms within Rajant.
- LaTeX: A high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. This is also a free program.
- Inconsistencies: A setting in BC|Commander that shows errors in a particular network.
- Snapshot: A real-time capture of a customer's network. This can show any setting that's been applied to a network and is attached to a support ticket by the customer/reseller.
- Nodes: Components that make up a mobile network (in this case the radios).
- Peers: Paired nodes in a network.
- Networking: A system which distributes programming to multiple nodes simultaneously for the purpose of extending total coverage beyond the limits of a single broadcast signal.

- InstaMesh: A protocol that networks fixed, wireless, and mobile nodes together and will redirect traffic between the next best available points if any one peer is compromised or obstructed.
- Configurations: Within BC|Commander, the settings that are specific to each radio node.
- Recommendations: Support team’s suggestions in fixing a customer's' mobile network.

Literature Review

This section of the report contains the introduction and purpose of research for this project. Within this section a brief background of Rajant Corporation is provided. The next section includes a Significant obstacles section where the challenges of this project are addressed. Finally, a discussion of the work breakdown is established.

Introduction

This basis for this research originally stemmed from a project that needed attention from the engineers within Rajant Corporation. The need for an automated ticketing system is something that everyone wanted within the company. It had the potential to alleviate some of the demands coming from customers, as well as teach customers how to troubleshoot and look inwardly at their own network health. As the Rajant support group moves further along in this process, generating network health reports will teach customers a lot more about the products they've purchased and how they work together to create a wireless mesh network. There will be a greater need to purchase these reports as network health becomes more important to customers. How will this be achieved? It is my goal to not only to reveal that process within this report, but to help the support team develop the tools to break down barriers of networking issues within Rajant Corporation for current and future customers.

Rajant Corporation: A Brief Background

Rajant Corporation has been a top service provider for wireless mesh technology since its beginning October of 2001.[1] The founders Robert Schena and Paul Hellhake recognized the significant shortcomings in traditional wireless networks when it came to mobile voice and data networks used by first responders. This mesh network is more robust and allows these data networks to be fully mobile and can operate reliably in some of the most demanding

---

1. "Why Rajant?: Bringing to Life the Promise of Everywhere Productivity" in *About* (Rajant.com, 2019)

environments. Since 2001 Rajant Corporation's Kinetic Mesh® network has provided support to various mine sites, military surveillance, and many more businesses all over the globe. One of the numerous services Rajant Corporation offers is a ticketing system that connects the clients with the engineers directly. This ticketing system is a website-based program that can assist customers with questions about the Breadcrumb Mesh Network®. Recently, a support team of about five engineers has been assembled to assist with these incoming "tickets". This service will provide an immediate and personalized response to every consumer that has any questions or concerns and is already using Rajant Corporation's Kinetic Mesh Network®.

Purpose of the Research

Rajant Corporation is committed to achieve an increasing rise in customer satisfaction through improvements in the quality of its products, service and quality management system. The products include any platform that supports our firmware called BC|Commander. The radios, antennas, and other networking hardware are combined together to create a wireless mesh network; this is the basis for the entire company. What makes Rajant Corporation special is the constant upgraded versions of its firmware, BC|Commander. This firmware is what gives the radios the ability to mesh together (the Kinetic Mesh® network). "Using a combination of BreadCrumb® wireless network nodes and InstaMesh® networking software, Rajant Kinetic Mesh® networks employ any-node to any-node capabilities to continuously and instantaneously route data via the best available traffic path and frequency—for any number of nodes, all with extremely low overhead. Rajant BreadCrumbs can communicate with any Wi-Fi or Ethernet-connected device to deliver low-latency, high-throughput data, voice and video applications across the meshed, self-healing network.". [2]

---

2. "Rajant Innovation: Delivering on the Promise of Fully Mobile Network" in *Technology* (Rajant.com, 2019)

This concept is allows radio networks to work autonomously to provide the best connectivity across any organization's custom environment of fixed and mobile benefits, delivering powerful implementations in real-time. Through these concepts Rajant Corporation also provides total mobility, Kinetic Mesh® networks can act independently, with full routing abilities, making it a true peer-to-peer Mesh that is entirely mobile-permitted. Proven elasticity, with no single point of failure (due to rerouting options) and self-healing capabilities, Kinetic Mesh® networks ensure uptime of mission-critical implementations. The maximum bandwidth utilization of Kinetic Mesh® networks make use of all connected radios to route wired and wireless connections over the best accessible links. Traditionally mesh networks degrade as nodes are added, Kinetic Mesh® networks grow stronger with each additional node (nodes also self-configure). With a background in military and mining applications, Kinetic Mesh® networks are proven to resist the harshest conditions. They also support AES-CCMP (AES-Counter Mode CBC-MAC Protocol) and TKIP (Temporal Key Integrity Protocol) encryption with configurable per-hop, per-packet validation.

Significant Obstacles

This section of the report describes the significant obstacles and work breakdown involvedc with the project. There is a Knowledge base that reflected the obstacles within the first phase of the project. This was followed by a python programming section, where the obstacles occurred during the development phase of the project. The final obstacle, the BC|Assurance report generation and LaTeX files, occurred during the last phase of the project.

Knowledge Base

Establishing the process of assisting customers autonomously began with first building a baseline from previous issues that have been evaluated and solved. One of the lead software

engineers from Rajant Corporation has assisted tremendously with tickets in the past and has greatly helped the support group when solving tickets thus far. The process of building the baseline is a tedious process. First the tickets were sorted through and the most important and most helpful of them were recorded along with their solutions in a word document type folder that the entire support team could access. These issues ranged anywhere from country code accesses, to hardware issues, all the way to firmware update issues. There are so many factors to take into consideration that this will be the longest process of the entire project. Recording these issues and figuring out just how to solve them is paving the way for the next step: writing the commands for the automation process.

Python Programming

The programming side of this project has been split up among the whole group. The support group has been tasked to look through the monitoring program written by one of the head software engineers at the company. This goes through the command lines and reads breadcrumb statistics; it reveals everything needed to know about a specific network. So, in viewing this, the programs can reveal the existing problems within a network and give the engineers the knowledge of what to adjust or fix. Along with this and the knowledge base documentation the support team is able to write automations to sort through this code and its command lines controlling the snapshots of the networks sent in by various companies and customers.

BC|Assurance Report/LaTeX

As each support member has ran through their portion of the program and has completed their section of the report, the Python program will write out the responses to each issue and provide plots of the necessary section as needed using a software system called LaTeX. This is a

high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is also a free program that can be saved as virtually any file type (.pdf, .doc, etc.). The report was originally generated manually within Google Docs, this version allows everyone on the support team to edit the report in real time, while also commenting on each of each other's allotted sections. This however, was not produced from the Python Program that reads through the data that came from the individual snapshots.

The final report is generated first, from BC|Commander snapshots that are then read by the Python parsing program. As previously mentioned, this program reads the log files coming from BC|Commander, showing the inconsistent data within a specific network and creates various tables as well as suggestion sections that will fix these network bugs. An example report can be found on page 24 for this report.

The reports have had very positive feedback from the beta customers. The first beta customer was a company in Belgium located on a shipyard, their network faced some interference issues. This particular company saw the project in the earliest stages of development. It would usually take around a week and a half to generate a full report and network analysis, send the suggestions off for review, and finally edit and send the report to the customer. In return, they decided to drop our trial program for the time being. The second and final beta customer was a mining site in South America. This Mesh in particular is a lot larger than the original beta customer’s site, with almost two hundred Bread Crumbs to take into account. This has been a greater challenge for the support team, a larger network meant more room for error. By the time this company was picked up for the trial period however, the support team was more prepared. The Python parsing program is now up and running and with the help

of the entire support team, Rajant can now evaluate a network and produce a network health report through LaTeX within roughly one day.

Work Breakdown

Throughout the beginning of the semester a lot of research and planning went into this project on the Support Team's side. The engineers looked through old documentation and processes that Rajant Corporation had produced in the past to create and develop new systems completely from scratch. A Zendesk training was attended to learn more about the ticketing system in the early stages of the project. Weekly meetings with Rajant's CTO, Paul Hellhake, lead software engineer Marty Lamb, and the sales team in Malvern, PA are held to see exactly what they want out of this development. It came down to the Morehead support team creating a draft report of an example InstaMesh network. This report would be produced semi-automatically. Originally, the numbers and statistics of various network settings from the example mesh network were pulled from a log file or through looking at BC|Commander, and the support team evaluated the outcomes of this, made suggestions on how to alleviate the issues, and then listed them in the format of a report.

The draft report was originally set up into six sections and hand written using Google docs; the first being a BreadCrumb Mesh Synopsis (lists sort of general facts about the mesh), the second, an Instamesh and Networking section (where received packets, multi rates, and link health are tracked), the third, Configuration Issues (where evaluation of the settings that come up as inconsistent in the mesh are solved), the fourth is Flagged Events within the network (this is where the CPU usage and external device health statistics went), the fifth is a Dynamic Frequency Selection (DFS) (initial pulse reports, radar detection events, and downtime rates),

and the final section is a Report Conclusion (this is where the engineers gave an overall rating of the mesh network and left information for questions and feedback on the report).

The final report is now generated through the program 'LaTeX'. After each member of the report team inputs the correct suggestions and configurations, a final .json text file is created, parsed through by a "Snapshot Class" program, sent to LaTeX for data layout, and the report is generated into a readable document that is sent off for evaluation. After the document passes inspection and sanity checks, it is sent off to the customer. The billing aspect of this project is still in the development process but a sample budget has been attached on page 70 of this report.

Methodology

The methodology section of this report reviews the steps taken to ensure the success of the project. It explains in detail, the steps taken to create the final product; the network health report for the customers of Rajant Corporation. This explains in detail the process of the project including the Knowledge Base, Python Programs, as well as the produced BC|Assurance report.

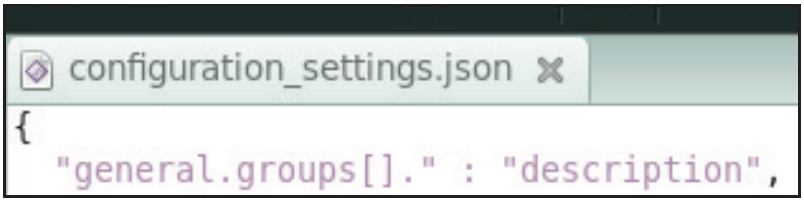
Knowledge Base

The Knowledge Base portion of this project was different for every engineer. The Knowledge Base is basically each possible configuration setting within a single Bread Crumb listed, and linked to its description. Written in the format of a .json text file (that is easily readable by programs such as Python), these different knowledge base categories come together to create a very vital portion of the automated report.

Generation of the ".json" File

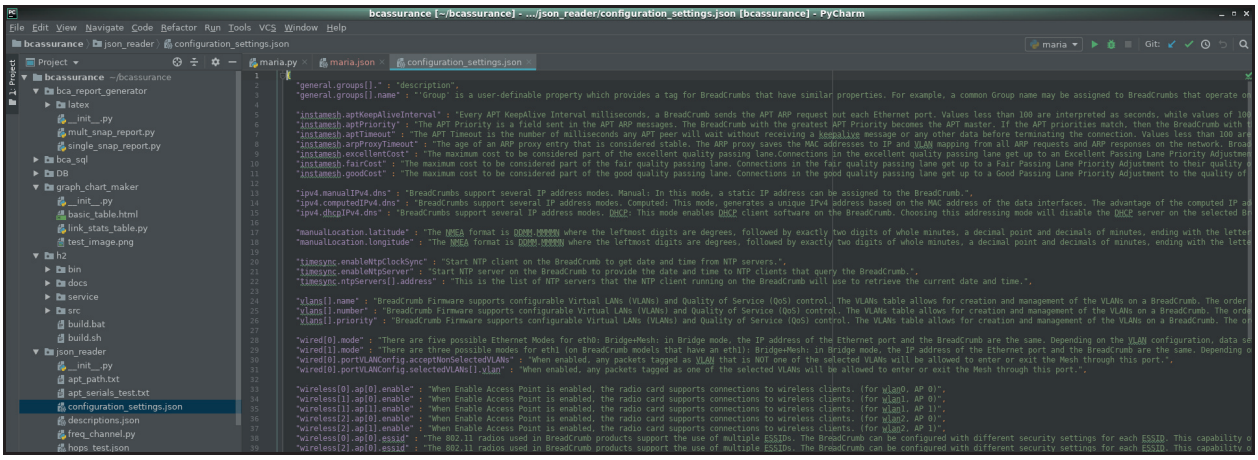
From BC|Commander the network settings are viewed. Within the network settings and Bread Crumb configurations the descriptions of each setting can be viewed. The descriptions are

the foundation for the Knowledge Base .json text files that eventually sort into the report that goes to the customer. A small example of the produced .json file is shown.[3] (see Figure 1).



[3] Figure 1: Screen capture of the .json file generated from BC|Commander descriptions. Created from screen capture of the python programming generator. [c. 2019]

This is the way the report was also formatted within Python. First, in the quotation marks goes by the name of the setting, then the colon to separate title from description, and lastly the description in the second set of quotation marks, followed by a comma. The next setting can then be entered. Python is a fairly easy coding mechanism to learn, it is also used by most of the Rajant Software Engineering staff and is therefore compatible with mostly every program the company uses. PyCharm is the version of Python that the Rajant - Morehead office used to generate any form of parsing from the data collected from mesh snapshots coming in from customers. PyCharm also creates legible text files that can be easily read into other software sets. The example of the Python program that generates .json files is shown below.[4] (See Figure 2).



3. Maria Lemaster, *Figure 1,* Screen capture from .json text file (Pluma: Text Editor v.1.20.4, 2019).
4. Maria Lemaster, *Figure 2,* Screen capture of printed code body (PyCharm: Community Edition, Python IDE for Professional Developers, 2019).

[4] Figue 2: Screen capture of the Python file that generates the .json files from the BC|Commander descriptions. Created from screen capture of the python program which generates the response file. [c. 2019]

Like the produced .json file, the code is first written with the quotation marks and in that is the name of the setting, then the colon to separate title from description, and lastly the description in the second set of quotation marks, followed by a comma. The next setting can then be entered. The Knowledge Base section is composed of each different section description that makes up the final report. The next step within this project consists of Python programming (titled bcanalzer) that reads the .json and log files produced from BC|Commander. This will then sort them into the .py files specific to each section of the report.

Python Programming

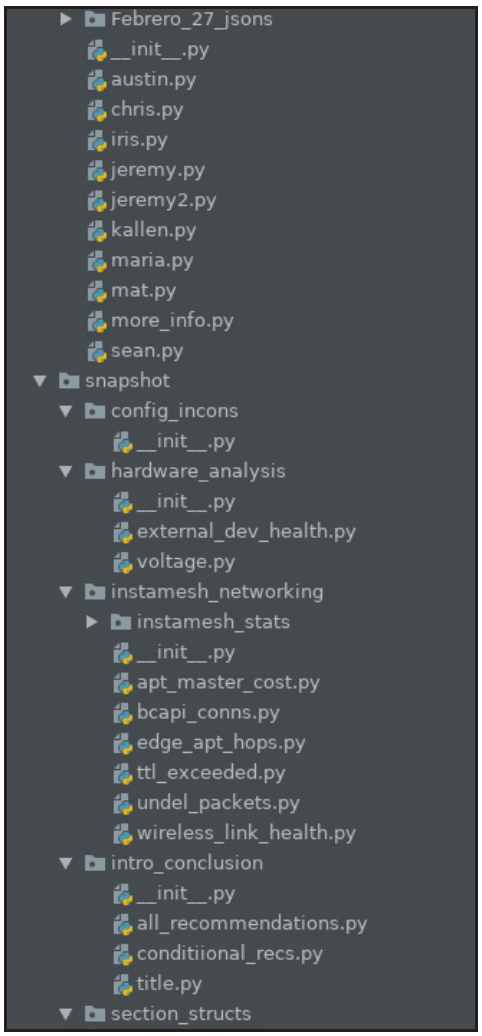
When the Knowledge Base was established, something was needed to parse through this data and sort it to the appropriate engineers for analysis. This program is where the suggestions and network analysis/health is documented for each InstaMesh snapshot Rajant received from it's customers using this service. These consist of various "if/then" statements which are native to most programming software. For example, if a certain statement is true, then this scenario will be put into play. This concept is unique to each new .mesh file that comes through the reporting process; at the same time, each of the engineer's section within the program has a different series of "if/then" statements that produce different things within the report. Each Python program is titled specific to that engineer and walks them through their section using a series of questions that are answered by the engineer and then re-coded into appropriate formats, legible by the reporting program process. A portion of the parsing program is shown below.[5] (See Figure 3).

```
import json
from terminaltables import AsciiTable
from shutil import copyfile
from snapshot import Snapshot
from snapshot.section_structs.helpers import get_bca_dir
from snapshot.section_structs import Recommendation
import os


class Index:
    def __init__(self):
        self.i = 0


class RecAdder:
    def __init__(self, snapshot, json_dir):
        self.snapshot = snapshot
        self._json_dir = json_dir
        with open(self._json_dir, 'r') as jfile:
            self.sec_rec_dict = json.load(jfile)
        self.section_list = list(self.sec_rec_dict.keys())

    def _backup(self):
        with open(self._json_dir, 'w') as jfile:
            json.dump(self.sec_rec_dict, jfile)

    def add_rec(self, recommendation, section):
        self.snapshot.init_dict[section].recommendation_list.append(recommendation)
        rec_dict = recommendation.create_dict()
        self.sec_rec_dict[section]["rec_list"].append(rec_dict)
        self._backup()

    def remove_recs(self, section):
        self.snapshot.init_dict[section].recommendation_list = []
        self.sec_rec_dict[section]["rec_list"] = []
        self._backup()
```

[5] Figure 3: Screen capture of the developed Python code that sorts the BC|Commander logs and pre-written Knowledge Base files into different sections. [c. 2019]

Once each of the eight sections (in total) are complete, each engineer has run through every possibility, and each necessary network health suggestion has been made, the program produces yet another .json text file where the information is stored. This file is composed of every suggestion that will make its way into the final product report. In the program PyCharm, these files get moved to their own folder that is produced at the time of the snapshot download. This folder can be seen in Figure 4 below.[6] (See Figure 4). This is where the data is parsed and sent off to each of the different engineers for suggestions, according .json file gets produced and the data is sent back out for combination schemes that will for the final report.

5. Maria Lemaster, *Figure 3,* Screen capture of code body (PyCharm: Community Edition, Python IDE for Professional Developers, 2019).



[6] Figure 4: Screen capture of the Python program list of sub categories within the snapshot class. [c. 2019]

Each of the eight different .py sections produced their own .json files as previously mentioned. These .json files are all saved in a folder within the master code that is titled differently according to the InstaMesh snapshot name. This can be seen below in Figure 5.[7] (See Figure 5).

6. Maria Lemaster, *Figure 4,* Screen capture of code sections (PyCharm: Community Edition, Python IDE for Professional Developers, 2019).

7. Maria Lemaster, *Figure 5,* Screen capture of .json files' home in Python code (PyCharm: Community Edition, Python IDE for Professional Developers, 2019).

[7] Figure 5: Screen capture of where the produced .json text files are stored after suggestions and recommendations are made for a specific InstaMesh snapshot. [c. 2019]

In this particular case, the snapshot that was sent to Rajant was called “Febrero_27”, therefor the Python code produced a “Febrero_27_jsons” where all of the 8 section .json texts can live and be combined into one text file. The Python program used by the support engineers has taken the most work and man hours to complete. This takes care of the bulk of the automation in return. This greatly cuts back on the time spent manually evaluating each and every snapshot. Before, engineers had to look deep into problems within BC|Commander, and sometimes team had to physically visit sites in order to find networking bugs. This program gives a complete network health scan, parses this data into a clean report, and even leaves room for professional engineers to make suggestions on fixing these network bugs. This has also proven to be a helpful resource to not just Rajant’s customers, but many other engineers throughout the company. This data has proven to have the ability to show Rajant employees networking data in real time, in remote locations. After the Python data is completed, it all be sent off through one last programming technique that will display it orderly into a LaTeX file.

BC|Assurance Report/LaTeX

This category of the report reviews the origin of the report document and the process involved with that. The next category discusses the LaTeX portion of the project, and finally the BC|Assurance report is discussed and a workflow diagram is displayed.

Original Google Docs Report

Originally the report and network analyses were done by hand, searching through BC|Commander for network inconsistencies. After around eighty man hours and split between eight engineers one report (including suggestions) over one InstaMesh snapshot was generated.

This was done so using a convenient tool through Google. Rajant's Corporate email addresses are Google based, giving each engineer access to the Google Drive.

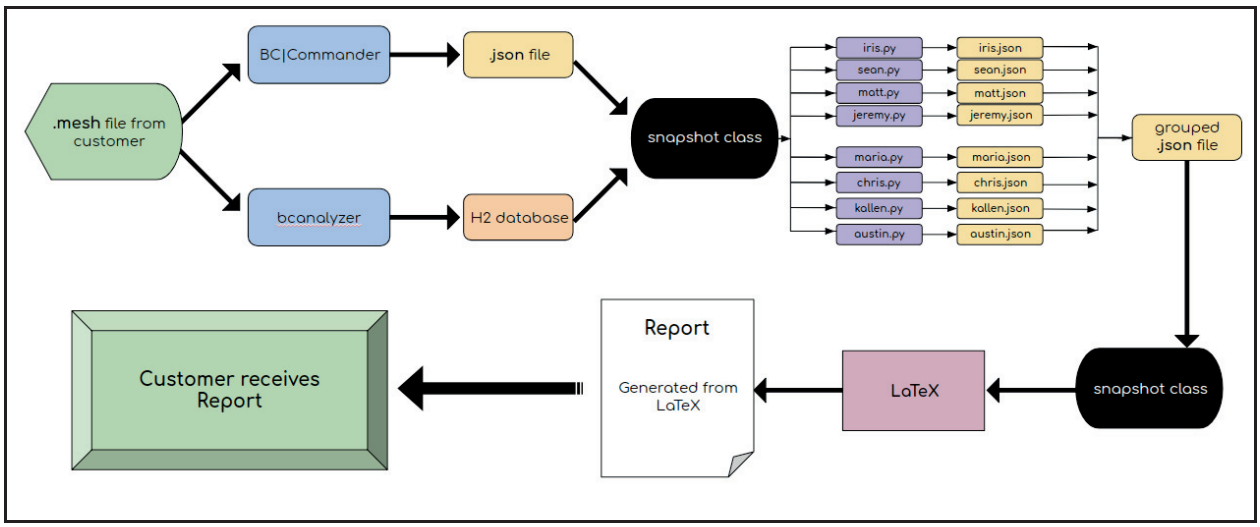
This is a place where, documents, presentations, even excel sheets can be stored an edited. Another good thing about the Google drive is that, once everyone within a defined group has been granted access to a specific file, it can be edited in real time by anyone on the team. Commenting sections were a large help within this phase of the project; it was the only fail checks the team had at the time. The template shown in the next section is a format the support team carries into the production of the LaTeX file later on and is a direct example of what the LaTeX file will look like once printed.

LaTeX

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is a qualified standard for the communication and publication of scientific documents. It is also available as free software. LaTeX allows the .json text files to be sorted into a reporting format after running through the final cycle of the snapshot class code.

BC|Assurance Final Report

The final report is complete once the files have all gone through the LaTeX program and a document is produced in a readable format. The six-month coding obstacles that the support team has faced has led to the production of the original Google docs report automatically. It has the potential to be in the customer's hands in a fraction of the time that it took to manually evaluate the networking issues. Below is a flow chart of BC|Assurance and how each aspect plays into the production of the network health file.[8] (See Figure 6 below).



[8] Figure 6: Flowchart showing the network health report's generation journey from start to finish. Created using Microsoft PowerPoint software. [c.2019]

The LaTeX file can only be generated from an incoming InstaMesh snapshot, therefore the example report is only a template for what a full report would look like. This was done to keep the confidentiality of the customers' networks safe. The template can be found in the Appendix of the report and explains in depth what each section entailes.[9] (See Appedix). The BC|Assurance report template is the basis of the entire project. What is produced within this document gives insight to the entire network health, while also leaving room for suggestions on how to improve it.

Timeline Breakdown

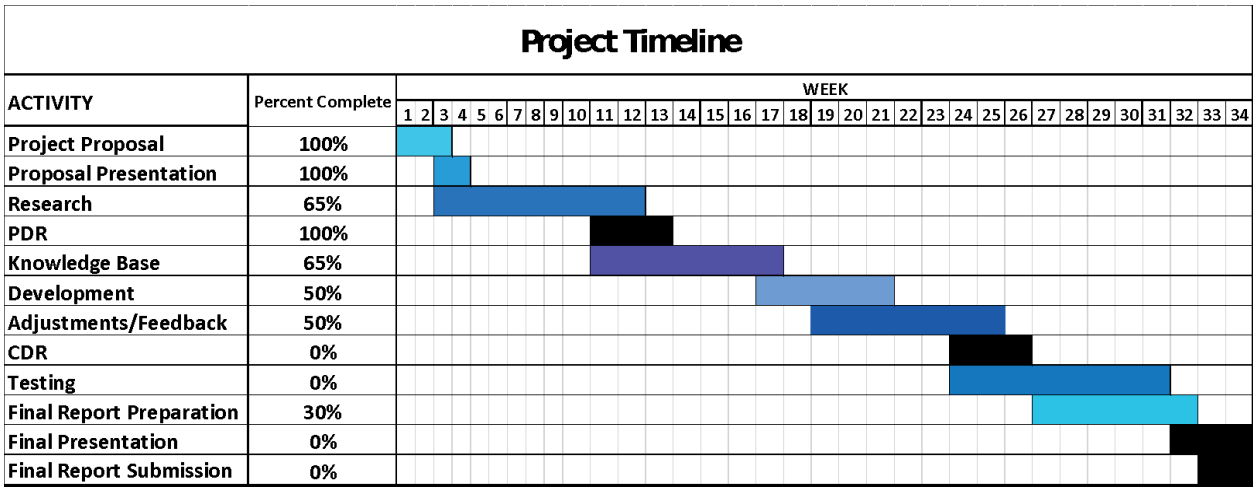
Below are the comparisons of the original timeline versus the updated timeline. This shows how the workflow developed over time and how it was originally planned, versus how the project turned out. Figure 7 and Figure 8 show these below.

Original Project Timeline

8. Maria Lemaster, *Figure 6,* Chart showing project flow (Microsoft PowerPoint v. 2011, 2019).

9. Maria Lemaster, Iris Hezeltine, Sean Buckler, Mathew Hardin, Kallen Schmidt, Jeremy Rice, Chris Graves, and Austin Clark, *BC|Assurance Google Docs Report* in Appendix. (Google Docs Editing Software, 2019).

The original timeline was laid out week by week of how the project was projected to go.[9] (See Figure 7). First was the proposal phase and the presentation and approval section of the project. Research was projected to take up a lot of the first semester of thesis. The Knowledge Base had a timeframe of only three weeks, which was very off target. Instead of taking three weeks to complete, this phase took more like three months to finish. Development was another section that took longer than projected. Adjustments are still being made, and the support team continues testing the full product.



[10] Figure 7: The original project timeline including percent completeness. A very rough draft of how the project was expected to go, this was the timeline used during the first semester of work. Created using Microsoft Excel. [c. 2018]

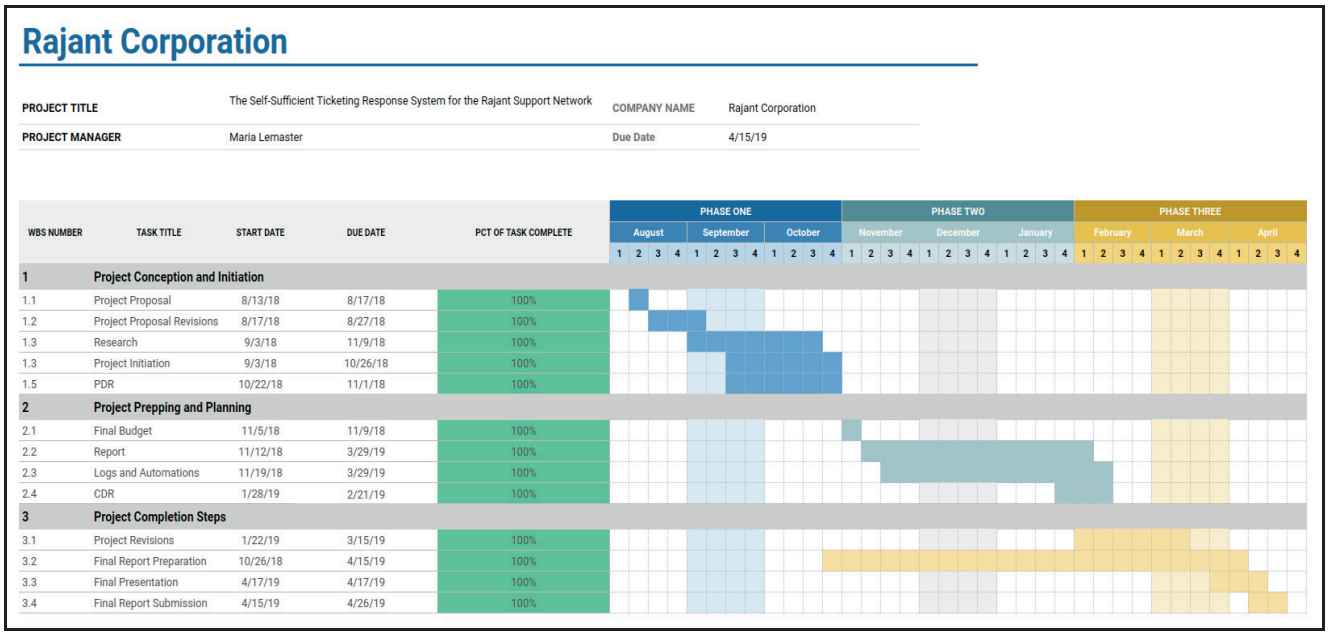
Final Project Timeline

The final project timeline is split into different phases of the project.[10] (See Figure 8). This allowed for more organization of the project steps and ensured a release date for Rajant support engineers.

The first phase is the "Project Conception and Initiation Phase" of the project where the support team planned the startup of this project and decided where to begin. This section includes the proposal, research, and initiation subcategories, as well as the Preliminary Design

10. Maria Lemaster, *Figure 7,* Spreadsheet showing project completion from August 2018-January 2019 (Microsoft Excel v. 2011, 2019).

Review. The second portion of this project was the "Project Prepping and Planning" phase. This is where the Knowledge Base, development, and all Python programming was accounted for and the time it took to complete them. The Critical Design Review also occurred during this phase of the project. The final phase of the project is titled "Project Completion Steps". These are the final steps that were taken in order to ensure the success of this project as a whole for the Rajant Support team. Within this section lies the project revisions or testing, the final report preparation, and the final thesis presentation.



Rajant Corporation

PROJECT TITLE	The Self-Sufficient Ticketing Response System for the Rajant Support Network	COMPANY NAME	Rajant Corporation
PROJECT MANAGER	Maria Lemaster	Due Date	4/15/19

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	PCT OF TASK COMPLETE
1	**Project Conception and Initiation**			
1.1	Project Proposal	8/13/18	8/17/18	100%
1.2	Project Proposal Revisions	8/17/18	8/27/18	100%
1.3	Research	9/3/18	11/9/18	100%
1.3	Project Initiation	9/3/18	10/26/18	100%
1.5	PDR	10/22/18	11/1/18	100%
2	**Project Prepping and Planning**			
2.1	Final Budget	11/5/18	11/9/18	100%
2.2	Report	11/12/18	3/29/19	100%
2.3	Logs and Automations	11/19/18	3/29/19	100%
2.4	CDR	1/28/19	2/21/19	100%
3	**Project Completion Steps**			
3.1	Project Revisions	1/22/19	3/15/19	100%
3.2	Final Report Preparation	10/26/18	4/15/19	100%
3.3	Final Presentation	4/17/19	4/17/19	100%
3.4	Final Report Submission	4/15/19	4/26/19	100%

PHASE ONE: August, September, October | PHASE TWO: November, December, January | PHASE THREE: February, March, April

[11] Figure 8: Phases 1, 2, and 3 are outlined in three different colors to show the different steps. There is a percent of completeness portion of the figure showing progression of the project. Created using a project timeline template in Microsoft Excel. [c. 2019]

Cost Estimates

Below is the example budget that was created to represent the cost this project would create for the company.[12] (See Figure 9). The final estimated price turned out being over $3

11. Maria Lemaster, *Figure 8,* Spreadsheet Template of sample timeline (Microsoft Excel v. 2011, Accessed 2019).

12. Jeff Kruth, *Figure 9,* Spreadsheet Template of sample budget (Microsoft Excel v. 2011, Accessed 2019).

million. The budget was a specific part of the design that was used to teach cost scenarios throughout a real-world project and the financial tolls it can cost a company.

Cost Estimates Example

The cost estimates are a rough study of how much this project would have cost considering software, hardware, man hours by engineers, overhead burdens, facilities overhead, and other company budget schemes. The actual budget for this project is not listed these are merely educated guesses at what prices would seem to fit for this case. Pricing is something that is still in discussion between the support engineering team and the CTO of Rajant Corporation. Figure 9 is shown below demonstrating this example.

Project Budget Sheet
Lead Engineer: Maria Lemaster
Project: The Self-Sufficient Ticketing Response System for the Rajant Support Network
Start Date: August 13, 2018
Due Date: April 2018

	Labor		Materials				
Tasks:	**HR**	**$/HR**	**UNITS**	**$/UNITS**	**TRAVEL**	**EQUP/ SPACE**	**BALANCE**
Project							
Lead Engineer	2080	$65					$135,200
Ticketing System:					$4,000	$4,000	(total Materials,
Software			10	$650			travel, equip./
Hardware			5	$3,000			space)
etc.			10	$0			$29,500
Direct Costs:							
Zendesk			10	$600			$600
Python			10	$0			$0
Engineers	2080	$65	10	$65			$1,352,000
Indirect Costs:							
Overhead Burdens			4	$25,000			
General and Administrative			1	$20,000			
Facilities Overhead			15	$100,000			
Cost of Money			10	$5,000			1670000
SUBTOTAL							$3,187,300

[12] Figure 9: Example budget spreadsheet for the Self-Sufficient Ticketing Response System for the Rajant Support Network. Created using Microsoft Excel. [Accessed 2018].

Results

The results section of this report states the progression over time, the conclusion, and the success/failure of this project for Rajant Corporation as a whole. This also coincides with the success/failure of the project that was sought out in August of 2018 at the beginning of the thesis course. The engineers worked over the course of eight months to complete this task.

Progression of Project Over Time

This project originally started out as an idea that would alleviate some of the incoming tickets that the support team were facing on a weekly basis. Twenty-four hour/ seven-day a week shifts split evenly amongst eight systems engineers was getting tough between other projects and being on call all the time. The CTO of Rajant corporation was on board with this plan, even though it was a bit vague as to what exactly be accomplished from this.

The beginnings of this project started off slow with many ideas being thrown around in different directions. Finally it was settled upon a sort of automated network report with suggestions that would give network improvements to Rajant customers. Rajant also wanted to switch from charging its customers one way, into charging for this service; so the success of this project would greatly benefit the company. The deadline of April 15th, 2019 was assigned about a month into the project, while weekly meetings were keeping the team on target.

Success/Failure of Project

The success of this project was necessary once the deadline was set. Even though there were changes along the way, the project came together in the end. The testing phase is still underway. The support team is testing the Python programs with different mesh networks to ensure the production of quality reports for customers. Rajant Corporation is committed to achieve increasing customer satisfaction through improvements in the quality of its products and services. This is exactly what this project accomplished, it provides network analysis to customers, which in turn, provides customer fulfilment. Rajant Corporation will be using this product within the company starting this year.

APPENDIX

Lemaster, Maria, Iris Hezeltine, Sean Buckler, Mathew Hardin, Kallen Schmidtt, Jeremy Rice, Chris Graves, and Austin Clark, *BC|Assurance Google Docs Report,* [c. 2019], Appendix. Pages 29-63.

BC|Assurance

Location:

Latest Snapshot:

Report Date:

Rajant Corporation
200 Chesterfield Pkwy
Malvern, PA 19355
Tel: (484) 595-0233
Fax: (484) 595-0244

Recommendation Summary

The following table totals the accumulated recommendations throughout the different sections of the report. Recommendations are given different categories based on severity. In addition, Assurance recommendations are broken down into normal recommendations that should always be observed, and 'conditional' suggestions that should be looked into if a given circumstance is met.

Cumulative Recommendation Summary	
Recommendation Category	**Number of Recommendations**
Total Critical Recommendations	
Total Important Recommendations	
Total No Recommendations	

Recommendations Explained

A **Critical Recommendation** is given when the analysis of a section reveals something that critically affects the mesh. If these situations are not remedied the mesh is in danger of downtime/outages.

An **Important Recommendation** is given when the analysis of a section reveals something that adversely affects the mesh. We want to ensure that BreadCrumbs are working optimally within your network design. Ignoring these errors could result in network unreliability, performance loss, or overworked BreadCrumbs.

A **No Recommendation** is given when the analysis on a section does not warrant any recommendations. In almost all cases, this is a good thing to see! As our system develops you may see new recommendations in place of 'No Recommendation' when we notice something adversely affecting the mesh that we had not noted before.

Recommendations

The following recommendations are based on a detailed analysis of the mesh network. Explanations for recommendations are given in their corresponding sections which are listed below.

Reminder: A **Critical Recommendation** is given when the analysis of a section reveals something that critically affects the mesh. If these situations are not remedied the mesh is in danger of downtime/outages.

Critical Recommendations:

- **Recommendation:** Short description. See "link" for more information.

Important Recommendations:

- **Recommendation:** Short description See "link" for more information.

Conditional Recommendations

The following are recommendations only to be implemented if the given conditions are met:

Important Recommendations:

- **Recommendation:** Short description See "link" for more information.

Additional Information

To ensure an accurate analysis, please provide the following additional information to: bcassurance@rajant.com

- **Please provide a list of BreadCrumbs under special circumstances and the circumstances for each.** Special circumstances may require BreadCrumbs to be removed from some parts of the analysis for accuracy.
- **Please provide a list of BreadCrumbs that routinely power off.** Any BreadCrumbs that routinely power off affect uptime statistics. See "Snapshot Information: Uptime" for more information.
- **Please provide a list of mobile BreadCrumbs.** Any BreadCrumb frequently leaving/entering the network can interfere with analysis by skewing data or creating unnecessary recommendations that would not apply to mobile BreadCrumbs.

Snapshot Synopsis

This section summarizes standard snapshot information and statistics.

- **Total recommendations:**
- **Total critical:**
- **Total important:**
- **Total no recommendations:**

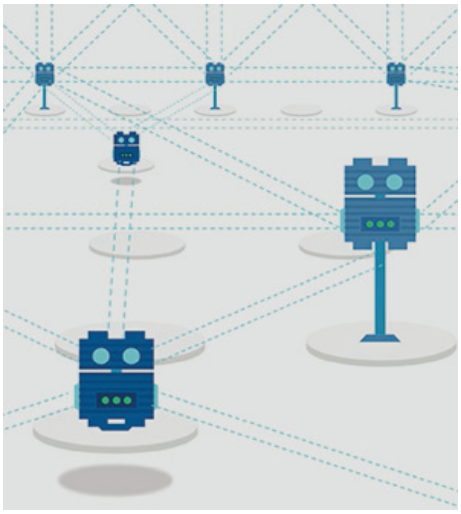
Snapshot Information

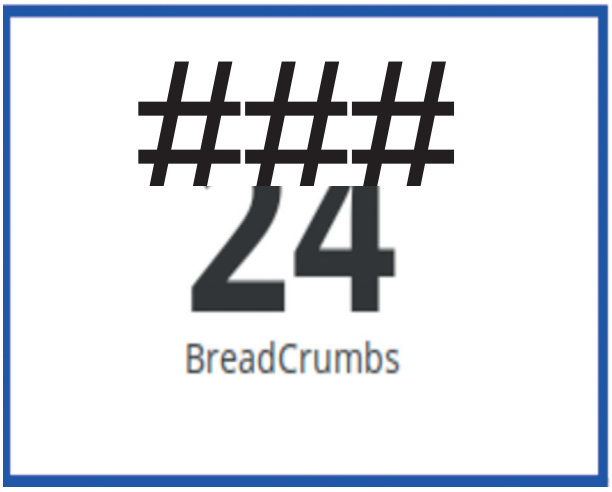
Local Date and Time the Snapshot Was Taken:

Location: **Latest Snapshot**:

APT Master: (APT Master is denoted as **LXx-xxxxx-xxxxx** in charts.)

Total Reporting BreadCrumbs





Important Recommendation: (Placeholder)

Serials of BreadCrumbs without logfile data or not captured in Snapshot:

Uptime

This statistic gives insight into how long the BreadCrumbs have been operating since their last full reboot.

The average time of all BreadCrumbs on the network since their individual last full reboot is approximately **(d/h/m/s).** The maximum is roughly **(d/h/m/s)** by **(Serial #)**. The minimum uptime is about **(d/h/m/s)** by **(Serial #)**.

It appears that the lowest uptime BreadCrumb has been up significantly less than the average on the mesh. This can skew statistics and can be a symptom of various issues. However, there are many cases where this is normal and to be expected.

Serial	Name	Uptime (d/h/m/s)

Important Recommendation: (Placeholder)

CPU Usage

CPU process usage, denoted as a percentage of a processor, should always be lower than 90%. Any BreadCrumbs reporting such high CPU utilization should be monitored. Units that still report such high percentages on the next report could indicate a BreadCrumb in need of diagnostic support.

Note: The percentage(s) of processes used to take the snapshot itself are not taken into account.

No recommendation: (Placeholder)

Frequency Usage

Frequency usage indicates which radios and which frequencies are being used most. This will allow for proper planning for future expansion, as well as indicate which RF channels the BreadCrumbs are favoring with our costing algorithm. The following is a table summarizing frequency usage.

Frequency	Channel	Total Destination BreadCrumbs Reached Through this Frequency over all BreadCrumbs	Total Number of BreadCrumb Connections

The x.xGHz frequency was being utilized on most units. The majority of BreadCrumbs (Placeholder).

Important Recommendation: To use Rajant radios to the fullest, enabling mesh in radio configuration settings, and turning radio cards on would be recommended unless specific reasons are present in the network requiring them to be in the existing states.

MAC Stats

Individual MAC address stats are listed here, along with their originating BreadCrumbs. These statistics show which devices on the network are generating the most packets to the APT Master and which devices have the most packets sent to them from the APT master. Items are normalized on a per second basis where applicable, based on their individual uptimes. The nodes with the lowest Uptime values are noted in the "Uptime" section.

APT Master: Serial # (APT Master is denoted as **(Serial #)** in charts.)

Total MACs on APT Master: (Placeholder)

APT Master's MAC Garbage Collection Threshold: (Placeholder)

Important Recommendation : (Placeholder)

Sent/Received

This table indicates what MACs are sending the most packets through the APT Master, denoted as BreadCrumb the MAC is attached to (wired device, wireless client, BreadCrumb itself, etc.) and MAC address sending the packets.

Sent		
Source BreadCrumb	Device MAC	Packets Per Second

This table indicates the MAC addresses that are most often the destinations for packets received by the APT Master, denoted as BreadCrumb and MAC address to receive the packets.

Received		
Destination BreadCrumb	Device MAC	Packets Per Second

The entities that are causing the most sent and received traffic through the APT are as follows from greatest to least:**(Placeholder).**

Important Recommendation: (Placeholder)

Multicast

This table indicates what devices are most often the source for multicast received by the APT Master, denoted as the BreadCrumb and MAC address that is sending the multicast.

Multicast		
Originating BreadCrumb	Device MAC	Packets Per Second

(Analysis Placeholder)

Important Recommendation: (Placeholder)

Floods / Source Floods

This section indicates the MAC addresses most often the originators of packets being dropped by the APT Master because they exceed the maximum flood limit, denoted as the BreadCrumb that was originally sending the packet(s) and MAC address originating the packet(s). **Source for floods dropped based on BreadCrumb sending the multicast:**

Source for Floods Dropped by Master		
Originating BreadCrumb	Originating MAC Address	Packets Per Second

(Analysis Placeholder)

Source for floods dropped based on MAC sending the multicast:

Source for Floods Dropped by Per-Source Limit on APT Master		
Originating BreadCrumb	MAC Address	Packets Per Second

(Analysis Placeholder)

Important Recommendation: If the network's applications are running smoothly, no changes would be needed. If there are issues, flood limiting per source and total may need to be adjusted.

Sources/Destinations for Discoveries

This table indicates those devices most often attempting to discover routes throughout the network, denoted as the BreadCrumb and MAC address that is trying to discover routes.

Source for Discoveries Processed by APT Master		
Originating BreadCrumb	Device MAC	Packets Per Second

This table indicates the MAC addresses that are most sought after destinations of packets that go through the APT Master, denoted as the BreadCrumb and MAC address that is the destination trying to be discovered.

Destination for Discoveries Processed by Master		
Destination BreadCrumb	Device MAC	Packets Per Second

(Analysis Placeholder)

Important Recommendation: (Analysis Placeholder)

Destinations/Sources for Undeliverables Received by Master

This table indicates the MAC addresses that are most often the destination for undeliverables received by the APT Master, denoted by BreadCrumb and MAC address that is the destination for the undeliverable.

Destination for Undeliverables Received by APT Master		
Destination BreadCrumb	Device MAC	Packets Per Second

This table indicates the most common sources for undeliverables received by the APT Master, denoted as the BreadCrumb and MAC address of the source of the undeliverables.

Sources for Undeliverables Received by APT Master		
Originating BreadCrumb	Device MAC	Packets Per Second

(Analysis Placeholder)

Important Recommendation: (Analysis Placeholder)

Destinations/Sources for Undeliverable Transmit Failures on APT Master

This table indicates the MAC addresses that are most often the destination for undeliverable transmit failures on the APT Master, denoted by BreadCrumb and MAC address that is the destination for the undeliverable.

Destinations for Undeliverable Transmit Failures		
Originating BreadCrumb	Device MAC	Packets Per Second

This table indicates the most often sources for undeliverable transmit failures on APT Master, denoted as the BreadCrumb and MAC address of the source of the undeliverables.

Sources for Undeliverables Transmit Failures on APT Master		
Originating BreadCrumb	Device MAC	Packets Per Second

(Analysis Placeholder)

No Recommendation: (Analysis Placeholder)

MegaBits (Mbps) per Second

Bits/sec indicates the total amount of data traversing individual BreadCrumbs, which details information on how much throughput each BreadCrumb is providing as a whole, instead of per radio. The following graph shows the top 5 bit rates for receiving and transmitting sides of BreadCrumbs for wired and wireless interfaces.

Wireless Mbps: Calculated by the summation of each wireless interface's MegaBits transmitted/received per BreadCrumb over the BreadCrumb's uptime.

Serial	BreadCrumb Name	Received Mbps

Serial	BreadCrumb Name	Transmitted Mbps

Wired Mbps: Calculated by the summation of each wired interface's MegaBits transmitted/received per BreadCrumb over the BreadCrumb's uptime.

Serial	BreadCrumb Name	Received Mbps

Serial	BreadCrumb Name	Transmitted Mbps

No Recommendation: (Analysis Placeholder)

Radio Busyness

This indicates how busy each radio is when considering noise and actual packet traffic. When noise numbers reach above 10% or busy time is above 40%, this is considered an above average amount of traffic or noise indicating possible issues in the mesh or environment around that node. Transmitting, Noise, and Receiving Time all fall beneath Busy Time. They can help troubleshoot where the BreadCrumb(s) is having the most trouble whether it is receiving, transmitting, or noise side of the node.

Serial Number	Channel Number	Noise Time (%)

Serial Number	Channel Number	Busy Time (%)

Serial Number	Channel Number	Receive Time (%)

Serial Number	Channel Number	Transmit Time(%)

(Analysis Placeholder)

Critical Recommendation: (Analysis Placeholder)

Kernel Only Events

These events happen on a low system level that handles the actual operation of the radios and other technology within BreadCrumbs.

- **Baseband Hang:**

  Indicates RF or hardware problems when present

  No Recommendation: (Analysis Placeholder)

- **Baseband Watchdog:**

  Indicates RF or hardware problems when present

  No Recommendation: (Analysis Placeholder)

- **Fatal Hardware Error:**

  Indicates significant RF issue or hardware problem when present

  No Recommendation: (Analysis Placeholder)

- **Transmitting Hardware (TX HW) Error:**

  Indicates high congestion or noise when present

No Recommendation: (Analysis Placeholder)

- **Phase Locked Loop (PLL) RX Hang:**

  A Sign of a hardware issue when present

  No Recommendation: (Analysis Placeholder)

- **MCI Reset:**

  If this is present, there is a hardware issue.

  No Recommendation: (Analysis Placeholder)

- **Transmitting (TX) Path Hang:**

  Indicates high congestion or noise when present

Serial Number	BreadCrumb Name	Major Failing Radio Interface/MAC	Errors since the Last Full Reboot	TX Path Hang Resets/Second

There seems to be evidence of high congestion or noise seen in multiple sections.

Symptom/Effects: (Analysis Placeholder)

- **MAC Hang:**

  Indicates high congestion or noise when present

Serial Number	BreadCrumb Name	Major Failing Radio Interface/MAC	Resets Since Full Reboot

(Analysis Placeholder)

Symptom/Effects: (Analysis Placeholder)

- **Stuck Beacon:**

  Indicates high congestion or noise, can interfere with BreadCrumb discovery

Serial Number	BreadCrumb Name	Major Failing Radio Interface/MAC	Errors since Full Reboot	Stuck Beacon Resets/Second

(Analysis Placeholder).

Symptom/Effects: (Analysis Placeholder)

Instamesh & Networking

Assurance Recommendations:

- **Total recommendations:**
- **Total critical:**
- **Total important:**
- **Total no recommendations:**

This section briefly discusses Instamesh protocol statistics, network transmission rates, and other core topics related to networking performance of every Rajant mesh.

Top Ten Instamesh Stats

The following tables list the top 10 BreadCrumb serial numbers for Instamesh statistics, normalized per second. Those are **Packets Received, Packets Sent, Packets Multicast, Floods Dropped, Source Floods Dropped, ARP Request Received, ARP Request Sent, ND Request Received, ND Request Sent, Undeliverables Received, Undeliverable Transmit Failures, & Discoveries Sourced.** If different statistics are needed on future reports please contact the BC|Assurance team.

Packets Received/Sent

Sent and received packet rates are good indicators of how much traffic is passing through individual nodes and can indicate if the mesh requires more coverage in different locations.

Packets Received		
Serial	BreadCrumb Name	Per Second

Packets Sent		
Serial	BreadCrumb Name	Per Second

No Recommendation: (Analysis Placeholder)

Packets Multicast

Multicast rates indicate how much of the network bandwidth is being used to re-transmit packets from one location. Multicasts are messages being sent to every node who can hear and accept it. If there is an abnormally high multicast rate for a particular node, the network can be affected drastically.

Packets Multicast		
Serial	BreadCrumb Name	Per Second

(Analysis Placeholder)

Important Recommendation: (Analysis Placeholder)

Floods Dropped

This indicates when a packet that is from a multicast exceeds the maximum flood limit setting. This can be empty due to the flood limit setting not being set or a lack of multicast. Overall, this is a good indication of if the network is working properly.

Floods Dropped		
Serial	BreadCrumb Name	Per second

(Analysis Placeholder)

Important Recommendation: (Analysis Placeholder)

Src Floods Dropped

Source Flood Drops indicate where the dropped flood packets are originally sent from.

Source Floods Dropped		
Serial	BreadCrumb Name	Per second

No Recommendation: (Analysis Placeholder)

ARP Requests/ ARP Sent

ARP is the networking protocol that maps IPv4 addresses to MAC addresses.

ARP requests are essentially a “where are you?” request. If there are too many ARP requests on the network, this could indicate poor connectivity.

ARPs sent are responses to ARP requests. If there are too many ARPs on the network, this could indicate poor connectivity as well.

ARP Request Received		
Serial	BreadCrumb Name	Per Second

ARP Responses Sent		
Serial	BreadCrumb Name	Per Second

No Recommendation: (Analysis Placeholder)

ND Requests Received / ND Requests Sent

ND-Neighbor Discoveries: IPv6 version of ARP (see above). The translate IPv6 addresses into Ethernet MAC addresses. Too many ND requests on the network could indicate poor connectivity.

ND Requests Received		
Serial	BreadCrumb Name	Per Second

ND Requests Sent		
Serial	BreadCrumb Name	Per Second

No Recommendation: (Analysis Placeholder)

Undeliverables Received / Undeliverables Transmit Failures

Denotes dropped packets, which usually indicate problems between radio links, such as building obstruction or other interference. These failures could also be the result of BreadCrumbs leaving the network within a highly mobile environment, in which some undeliverables are to be expected.

Undeliverables Received			
Serial	BreadCrumb Name	Per Second	Percentage of Total Packet Rate

Undeliverable Transmit Failures			
Serial	BreadCrumb Name	Per Second	Percentage of Total Packet Rate

Important Recommendation: (Analysis Placeholder)

Discoveries Sourced

This denotes the rate of the BreadCrumb listed looking for a route to a destination. When the rates are high, it means the BreadCrumb is having trouble reaching required destinations. If nodes have high mobility, this could be affected.

Discoveries Sourced		
Serial	BreadCrumb Name	Per Second

This value should be less than 1 per second depending on the network situation. In the current situation even with the highly mobile nodes the network is dropping far to many macs per second.

Important Recommendation: (Analysis Placeholder)

Edge to APT Wireless Hops

Edge to APT wireless hops can be defined as the largest number of hops any edge node must make to an APT master or slave. With fewer hops comes a better connection.

Important Recommendation: (Analysis Placeholder)

APT Master Cost (To and From)

The APT costs both to and from the master are indicators of network health as all traffic will need to pass through the APT Master to leave or enter the Rajant Mesh. Undiscovered BreadCrumbs are not represented here, as their cost should not be compared to currently tracked nodes. Listed here are the 10 known connected BreadCrumbs with the highest cost from the APT Master and the 10 known BreadCrumbs with the highest cost to the APT Master:

Serial	Name	Cost FROM Master

Serial	Name	Cost TO Master

Important Recommendation: (Analysis Placeholder)

Undeliverable Packets

Denotes packets that were returned to the sending BreadCrumb, which usually indicate problems between radio links.

Sending Address (SA): Source MAC address of ethernet packet.

Destination Address (DA): Destination MAC address of ethernet packet.

Transmitting Address (TA): Transmitting MAC address for the radio sending the packet. This is shown in the peers and details panes.

Receiving Address (RA): Receiving MAC address for the radio receiving the packet. This is shown in the peers and details panes.

Top 10 BreadCrumbs with Highest Undeliverable Packet Errors per Second				
Serial	BreadCrumb Name	Error Count	Log Time (Seconds)	Errors per Second

Most often occurring RA, TA, DA, SA (individually):

Receiving Address	Average Errors per Second

Transmitting Address	Average Errors per Second

Destination Address	Average Errors per Second

Sending Address	Average Errors per Second

Most often occurring RA/TA, DA/SA, RA/TA/DA/SA pairings:

Receiving Address	Transmitting Address	Average Errors per Second

Destination Address	Sending Address	Average Errors per Second

RA	TA	DA	SA	Average Errors per Second

Important Recommendation: (Analysis Placeholder)

TTL Exceeded

TTL Exceeded - "Time to Live Exceeded" indicates dropped packets because they lived too long on the network. This could be from:

1. Congestion that is slowing the network such that packets are staying inside radio queues too long.
2. A very busy network as packets arrive faster than they can be sent by the network.
3. A busy BreadCrumb with crypto turned on where packets arrive faster than they can be processed (encrypted) by the BreadCrumb. In this case, adding a dedicated SlipStream unit into the mesh network may help with the encryption process.

Correct selection and placement of infrastructure can mitigate heavy workloads on a single unit and increases possible data throughput (speed).

Sending Address (SA): Source MAC address of ethernet packet.

Destination Address (DA): Destination MAC address of ethernet packet.

Transmitting Address (TA): Transmitting MAC address for the radio sending the packet. This is shown in the peers and details panes.

Receiving Address (RA): Receiving MAC address for the radio receiving the packet. This is shown in the peers and details panes.

Top 10 BreadCrumbs with Highest TTL Exceeded Errors per Second				
Serial	BreadCrumb Name	Error Count	Log Time (Seconds)	Errors per Second

Most often occurring RA, TA, DA, SA (individually):

Receiving Address	Average Errors per Second

Transmitting Address	Average Errors per Second

Destination Address	Average Errors per Second

Sending Address	Average Errors per Second

Most often occurring RA/TA, DA/SA, RA/TA/DA/SA pairings:

Receiving Address	Transmitting Address	Average Errors per Second

Destination Address	Sending Address	Average Errors per Second

RA	TA	DA	SA	Average Errors per Second

Important Recommendation:(Analysis Placeholder)

BCAPI Connections

This value denotes the number of BCAPI connections running on each BreadCrumb. These can originate from the radios themselves, Rajant software in use on the mesh, and even custom made applications utilizing BCAPI. If there is a large number of 'stale' connections left running on a BreadCrumb it's performance can suffer. Any BreadCrumbs with more BCAPI connections than the expected number, the number of BC|Commanders or BC|Connectors running on the network, should be restarted to clear the 'stale' connections.

Serial	BCAPI Conn.

Important Recommendation: (Analysis Placeholder)

Wireless Link Health

This segment details the number of BreadCrumbs with great wireless connections. A great link is a link in which both cost and SNR values are good. A BreadCrumb has a great connection if it has at least 3 great links, has a good connection if it has 1-2 great links, and has a poor connection if it has no great links.

The Top 5 Best BreadCrumbs are chosen based on ratio of great links to all links. The higher the ratio, the better the wireless link health.

The Worst 5 Good BreadCrumbs are chosen based on ratio of great links to all links. The lower the ratio, the worse the wireless link health.

The Worst 5 Poor BreadCrumbs are based on total links. The more links it has, the worse the wireless link health (more links means higher probability that it should not be a poor connection).

Note: Slipstream units are ***not*** reflected in this chart because they have no wireless capabilities.

Great Connections:		**Good Connections:**		**Poor Connections:**	
Best 5	Great/Total	Worst 5	Great/Total	Worst 5	Great/Total

The highlighted BreadCrumbs in the good and worst sections are nodes that are not identified as mobile and should have more great links when connecting to the rest of the mesh compared to mobile nodes.

All Poor Connection BreadCrumbs:

Configuration Inconsistencies

Assurance Recommendations:

- **Total recommendations:**
- **Total critical:**
- **Total important:**
- **Total no recommendations:**

As networks grow in size and complexity, so does the task of maintaining appropriate settings across infrastructure. Addressing these inconsistencies can improve the network's usability.

Listed here are all of the inconsistencies within the mesh. Some of them are actionable, while others are fine or even suggested to be inconsistent. Please see the "Recommendations" section for a list of actionable items.

Instamesh Inconsistencies

This is where the inconsistencies can be found for the Instamesh configurations.

- Settings Placeholder

No Recommendation: (Analysis Placeholder)

VLAN Inconsistencies

This is where the inconsistencies can be found for the VLAN configurations.

- Settings Placeholder

No Recommendation: (Analysis Placeholder)

TimeSync Inconsistencies

This is where the inconsistencies can be found for the TimeSync configurations.

- Settings placeholder

Important Recommendation: (Analysis Placeholder)

Wired Inconsistencies

This is where the inconsistencies can be found for the Wired configurations.

- Settings placeholder

No Recommendation: (Analysis Placeholder)

Wireless Inconsistencies

This is where the inconsistencies can be found for the Wireless configurations.

- Setting Placeholder

Important Recommendation: (Analysis Placeholder)

Hardware Analysis

- **Total recommendations:**
- **Total critical:**
- **Total important:**
- **Total no recommendations:**

This section details hardware analysis of BreadCrumbs and external devices connected them.

Voltage

This section details the maximum and minimum voltage a BreadCrumb handled since the last snapshot capture. This is used to track unsupported voltage levels which could indicate power surges or the use of unsupported power adapters. These events can permanently damage infrastructure and possibly void warranty/support for those BreadCrumbs affected by non-Rajant Approved POEs.

Lowest Voltage	
Serial	Min Voltage

Highest Voltage	
Serial	Max Voltage

Critical recommendation: (Analysis Placeholder)

External Device Health

We monitor devices connected to our BreadCrumbs by number of link changes from boot as well as ensuring the device is successfully negotiating speeds with our ethernet ports.

Link state changes are often a sign that there is an issue with the ethernet cable used to power the BreadCrumb, or the POE powering the device.

Any radio with more than three link state changes is usually a sign of issues. The physical setups of the below devices should be investigated.

Ethernet Link State Changes

Serial	BreadCrumb Name	Total Link State Changes since Boot	Estimated Changes Per Day[1]	Wired Link That's Changing

Critical Recommendation: (Analysis Placeholder)

Site Specifics

Assurance Recommendations:

- **Total recommendations:N/A**
- **Total critical:N/A**
- **Total important: N/A**
- **Total no recommendations:N/A**

The site's **Assurance Profile** is being built upon as snapshots are uploaded and as additional information is provided about the site. This is used to create a personalized analysis not only for the Site Specifics section, but also throughout the entire report.

Questions & Feedback

Rajant strives to personalize our service for each individual BC|Assurance site by building an individualized **Assurance Profile**. If there are any questions regarding this report, topics that should be added, or topics that should be removed from future reports, please do not hesitate to contact Rajant Support via **support@rajant.com** to help us continue to provide an accurate preventative service.

As always, thank you for being a Rajant customer.

Regards,

Rajant BC|Assurance Team

[1]This is estimated from the number of link state changes per second, converted to per day.

REFERENCES

Lemaster, Maria. *Figure 1.* Screen capture from .json text file. (Pluma: Text Editor v.1.20.4). (2019).

Lemaster, Maria. *Figure 2.* Screen capture of printed code body. (PyCharm: Community Edition, Python IDE for Professional Developers). (2019).

Lemaster, Maria. *Figure 3.* Screen capture of code body. (PyCharm: Community Edition, Python IDE for Professional Developers). (2019).

Lemaster, Maria. *Figure 4*. Screen capture of code sections. (PyCharm: Community Edition, Python IDE for Professional Developers). (2019).

Lemaster, Maria. *Figure 5.* Screen capture of .json files' home in Python code (PyCharm: Community Edition, Python IDE for Professional Developers). (2019).

Lemaster, Maria. *Figure 6.* Chart showing project flow (Microsoft PowerPoint v. 2011. (2019).

Lemaster, Maria. *Figure 7.* Spreadsheet showing project completion from August 2018-January 2019 (Microsoft Excel v. 2011). (2019).

Lemaster, Maria. *Figure 8*. Spreadsheet template showing project completion from August 2018-May 2019 (Microsoft Excel v. 2011). (2019).

Kruth, Jeff. *Figure 9.* Budget Template Spreadsheet in Microsoft Excel v. 2011. (Accessed 2019).

"Rajant Innovation: Delivering on the Promise of Fully Mobile Network" in *Technology* Rajant Corporation. Accessed April 15, 2019. https://www.rajant.com/technology/.

"Why Rajant?: Bringing to Life the Promise of Everywhere Productivity" in *About*. Rajant Corporation. Accessed April 9, 2019. https://rajant.com/about/.