March 14, 2019

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6

RE: ENSC 405W/440 Design Specification for the Distributed Computing Network by Oakion Systems

Dear Dr. Rawicz,

The attached document provides the design specification for the Distributed Computing Network being implemented by Oakion Systems described previously in our Requirements Document[1]. The definition of success for this project, will be to optimize the current Visual BACnet architecture at Optigo systems, a data analytics company for Building Automation, in a systematic method to help ease computational and bandwidth bottlenecks occuring on their current Visual BACnet SaaS product under high network traffic.

This document will outline the system architecture designs and implementation specifications in a modular format, complimenting our Microservice solution architecture layout. We will begin by breaking down these subsystems and providing background on why certain decisions are made in our system design. We will provide further details of how the implementation will occur, the coordination of the subsystems and the communication systems in place. Next, we will provide information on how the currently deployed embedded systems provided by Optigo Networks will be utilized and how that data will be displayed. Lastly, we will touch on how performance and optimization will be measured for the subsystems individually, and the system as a whole, as defined by our definition of success.

Oakion Systems is comprised of 5 talented senior engineering students: Justin Singh, Tony Tan, Shawn Wang, Swimm Chan, and Aaron Nguyen. These individuals come from a strong background in software development, problem solving and systems design. Oakion is confident that the task at hand will be delivered with success and confidence to solve Optigo's request.

Thank you for reviewing our design specification. If you have any inquiries, please do not hesitate to contact myself, Justin Singh by phone: 250-961-3527 or email: jksingh@sfu.ca

Sincerely,

Justin Singh.
Chief Executive Officer
Oakion Systems

Enclosed: The Design Specification for the **Distributed Computing Network**

DESIGN SPECIFICATION

Distributed Computing Network

flexible scalable robust

Project Members:	Swimm Chan
	Aaron Nguyen
	Justin Singh
	Tony Tan
	Shawn Wang
Contact Person:	Justin Singh
	jksingh@sfu.ca
	250-961-3527
Submitted to:	Craig Scratchley (ENSC 405W)
	Dr. Andrew Rawicz (ENSC 440)
	School of Engineering Science
	Simon Fraser University
Issue Date:	March 14, 2019

Abstract

This document outlines the design specification for the Distributed Computing Network project provided by Oakion Systems to Optigo Networks in accordance with ENSC 405W/440. The project comprises a complete software systems solution which will optimize Optigo's computational and bandwidth bottleneck issue that arise under high network traffic.

Oakion Systems' Distributed Computing Network leverages the existing embedded systems infrastructure deployed with Optigo Networks, to form a Cluster Computing Distributed System on LAN that provides a Microservice solution. Computing nodes (Capture Tools) produce packet data, perform pre-filtering, and strategically split the packet stream, before sending Kafka hosted through a cloud service provider. Kafka then publishes tasks to available subscriber nodes to complete analysis and aggregation. The end result is then sent to Optigo's server for visualization. This process solves the bottleneck issues that currently face the centralized client-server model.

The user interface for the Distributed Computing Network will consist of powering on the embedded Capture Tool using a on/off switch on the device, application performance monitor GUIs that will update the user on the performance of our middleware and Visual BACnet, providing visualization of the data analytics occurring and the health of the network that the Capture Tool is placed on.

Table of Contents

List of Tables

List of Figures

Glossary

ACID ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties of database transactions.

APM Application Performance Management monitors and manages performance and availability of software applications that strives to detect and diagnose complex application performance to maintain an expected level of service.[1]

AWS Amazon Web Service is a cloud computing platform built and hosted by Amazon.com to provide users with a pay-as-you-go scalable compute, storage and throughput computing model. [23]

BACnet Communications protocol for Building Automation and Control networks that leverage the ISO 16484-5 protocol.[1]

Cloud Storage Cloud computing model in which data is stored on remote servers accessed from the internet, or "cloud." It is maintained, operated and managed by a cloud storage service provider on a storage servers that are built on virtualization techniques.[9]

Docker A deployment tool that packages code and software including dependencies to allow applications to run reliably on one computing environment to another.[29]

DogStatsD An open-source lightweight metric aggregation service built in Go, extending the StatsD protocol. Part of the TICK Stack.
[https://docs.datadoghq.com/developers/dogstatsd/]

DSMS Data Stream Management System, a software that takes in data and converts various kinds of data into a single storage container, or aggregates diverse data into a consistent resource, such as a database. [1]

GUI Graphical User Interface is an interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances. This interface uses icons, menus and other visual indicator (graphics) representations to display information and related user controls, unlike text-based interfaces, where data and commands are in text.[26]

Grafana An open source metric analytics & visualization suite. It is most commonly used for visualizing time series data for infrastructure and application analytics but many use it in other domains including industrial sensors, home automation, weather, and process control.[27]

InfluxDB An open-source Time Series Database (TSDB) written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics. It provides SQL-like with built-in time-centric functions for querying. Part of the TICK Stack. [https://en.wikipedia.org/wiki/InfluxDB]

IoT Internet of Things, a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data through network without requiring human-to-human or human-to-computer interaction.[1]

Kafka A software that takes in data and converts various kinds of data into a single storage container, or aggregates diverse data into a consistent resource, such as a database. [7]

Microservice Architecture An Architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely coupled, independently deployable, organized around business capabilities. [11]

Nomad A deployment tool that allows users to manage the different modules in the Distributed Computing Network system.

PCAP Data Packet Capture consists of an application programming interface for capturing network traffic.

Telegraf An open-source plugin-driven server agent for collecting, processing, aggregating, and writing metrics. Part of the TICK Stack.
[https://www.influxdata.com/time-series-platform/telegraf/]

Visual BACnet is an advanced visualization tool for Building Automation System Service Providers. The powerful analytics engine quickly identifies common problems and anomalous behavior in the BACnet infrastructure. [10]

1 Introduction

Background

The prevalence of the IoT has presented an opportunity for Oakion Systems to manage the inundation of data that companies like Optigo face with. The amount of data that flows to Optigo exponentially increases as their client base grows. This growth will cause a computational and bandwidth bottleneck in the near future (~8GB/day currently). Without changing or improving their current architecture, Optigo will face issues with continuous analytics and performance.

Oakion Systems' Distributed Network solution focuses on a primarily software solution to the computational and bandwidth issues. More precisely, this solution is a Microservice Solution which provides scalability and robustness which are two core principles that can be scaled to the requirements of any customer's network. Also, it empowers them to maximize their software capabilities before justifying changes on the hardware level. Oakion Systems takes pride in following the current trends of large platform industries such as Amazon that have internally been evolving their applications toward Microservice architectures since 2015 [5].

This section will highlight the major advantages from a Microservice Software Architecture as follows:

1. **Strong Independent Modules**: Each module is development independent from each other, which becomes increasingly important as a company's development grows. Companies can form tight-knit teams around the ownership structure of a set of small services, which leads to clearer interfaces with stronger boundaries, not just between the teams, but as a consequence also between the software (sub) systems these teams are building[6].

2. **Independent Deployment***:* The system in natural allows a Continuous Delivery approach to software development[15]. This means that old architectural software which used to be released once a quarter or monthly, are now having release cycles several times per day. The main competitive advantage that results from the ability to deploy at a high frequency comes from the increased speed of responding to the market, or offering a new feature faster than your competitors[6].

3. **Increased Speed and Availability***:* Speed and availability of the overall system are improved, while at the same time cost, size, and errors due to software dependencies to the business are reduced. The much shortened feedback loop for code that is deployed

via small, isolated services, which means the overall rate of change to the system can be increased relatively safely[5].

1.1 Scope

This document outlines the design specifications for the Distributed Computing Network microservice solution provided by Oakion Systems. It will break down the design of the overall system and all the sub system modules. Proof of Concept Prototype will be emphasized throughout the document, however other design phases will be mentioned. **Appendix A** contains a Supporting Test Plan and **Appendix B** contains the User Interface and Appearance Appendices that provide further detail the usability of the product.

1.2 Intended Audience

This document is intended to serve as a technical reference for Oakion System Inc. employees, stakeholders, potential clients, Dr. Craig Scratchley, Dr. Andrew Rawciz and ENSC 405W/440 teaching assistants. Oakion Systems will refer to this document for design clarification throughout the implementation of all Prototypes.

1.3 Design Classification

Oakion Systems expected timeline for an individual design specification as follows:

1. Proof of Concept Prototype - Specification will be carried out by April 2019
2. Engineering Prototype - Specification will be carried out by August 2019
3. Final Prototype - Specification will be carried out past August 2019

To specify a requirement, the document will use the following scheme:

DES [Section].[Module Sub Section].[Requirement Number]-[Code Scheme]

Code Scheme	Description
PoC	Proof of Concept Prototype
EP	Engineering Prototype
FP	Final Prototype

NOTE: All specification dates may be subject to change.

Table 1.1 - Code Scheme

For example, the fourth specification in section 2.6 that will be included in the Engineering Prototype would correspond to

DES 2.6.4-EP Insert Specification Description Here

2 System Analysis

2.1 Proof of Concept System Overview

The Distributed Computing Network consists of five main modules:

Module 1: **Producer** - Collects original packets from smart devices
Module 2: **Kafka + Zookeeper (DSMS)** - Stores all data for centralized access
Module 3: **Consumer** - Processes related packet chunks
Module 4: **Reducer** - Combines the processed packet chunks through packet reducing
Module 5: **Grafana** - Monitors application metrics modularly for the entire system and visually displays these results

The PoC system overview of the Distributed Computing Network is shown in **Figure 2.1.**

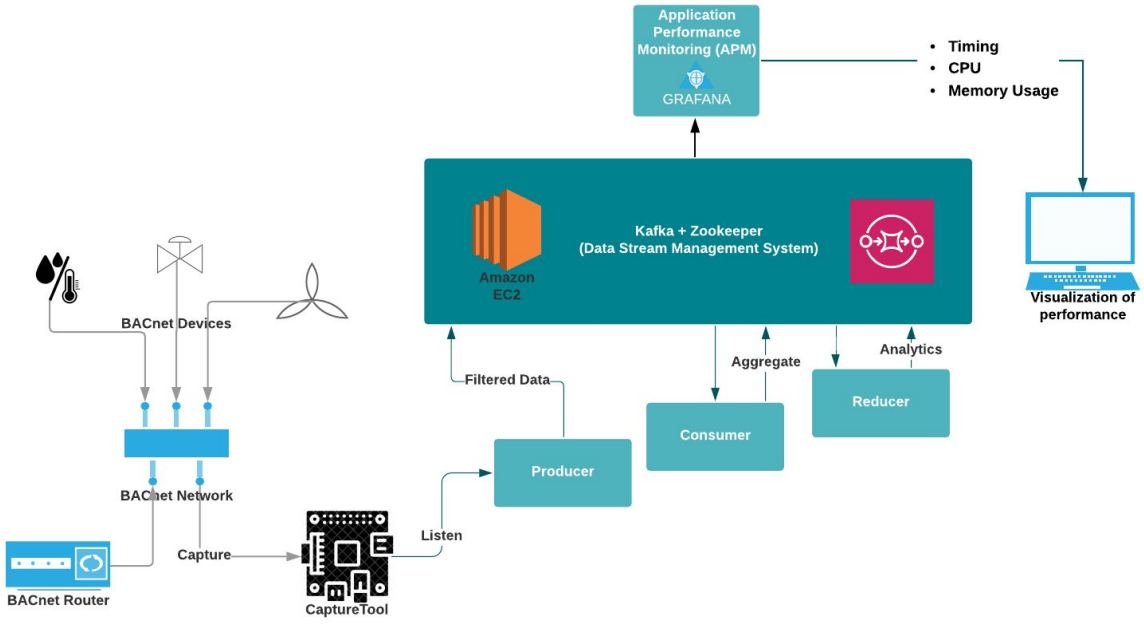


Figure 2.1 - PoC System Overview

All raw packets will initially go to the Producer on the Capture Tool from the building of smart devices. The raw packet data will be filtered separately and compressed on the Producer to reduce the data size before being sent to Kafka. Kafka, a centralized queue, will send and receive packet data from other modules. The Consumer gathers the necessary raw packet data and processes in parallel to result in more throughput to Kafka with minimal synchronization. Once all the packets have been processed, the Reducer module will then analyze these packets for specific network health checks and append the data into CSV formatted structure for future visualization on Visual BACnet. The Reducer is the last job completed before the analytics are sent back to the Optigo server displayed on the front-end, Visual BACnet. The performance of each module will be monitored through the application Grafana which will visualize the throughput to detect bottlenecks in our system.

2.2 Engineering Prototype System Overview

The Distributed Computing Network Engineering Prototype will add the following modules and functionality on top of the Proof of Concept prototype (**Section 2.1)**:

Module 6: **Splitter** - Integrated into the Producer module for symplicity, it strategically breaks up incoming packets through packet mapping to enable load balancing for the system
Module 7: **K2V Connector** - Inserts analytical data into Postgres database from Kafka and is integrated through REST API to allow Visual BACnet to pull this analytical data when ready
Module 8: **M2K Connector** - Notifies Visual BACnet when data for a certain PCAP is ready to be pulled
Module 9: **Kibana** - Monitors application logs for each module and visually displays this result

The Engineering Prototype system overview of the Distributed Computing Network is shown in **Figure 2.2**.

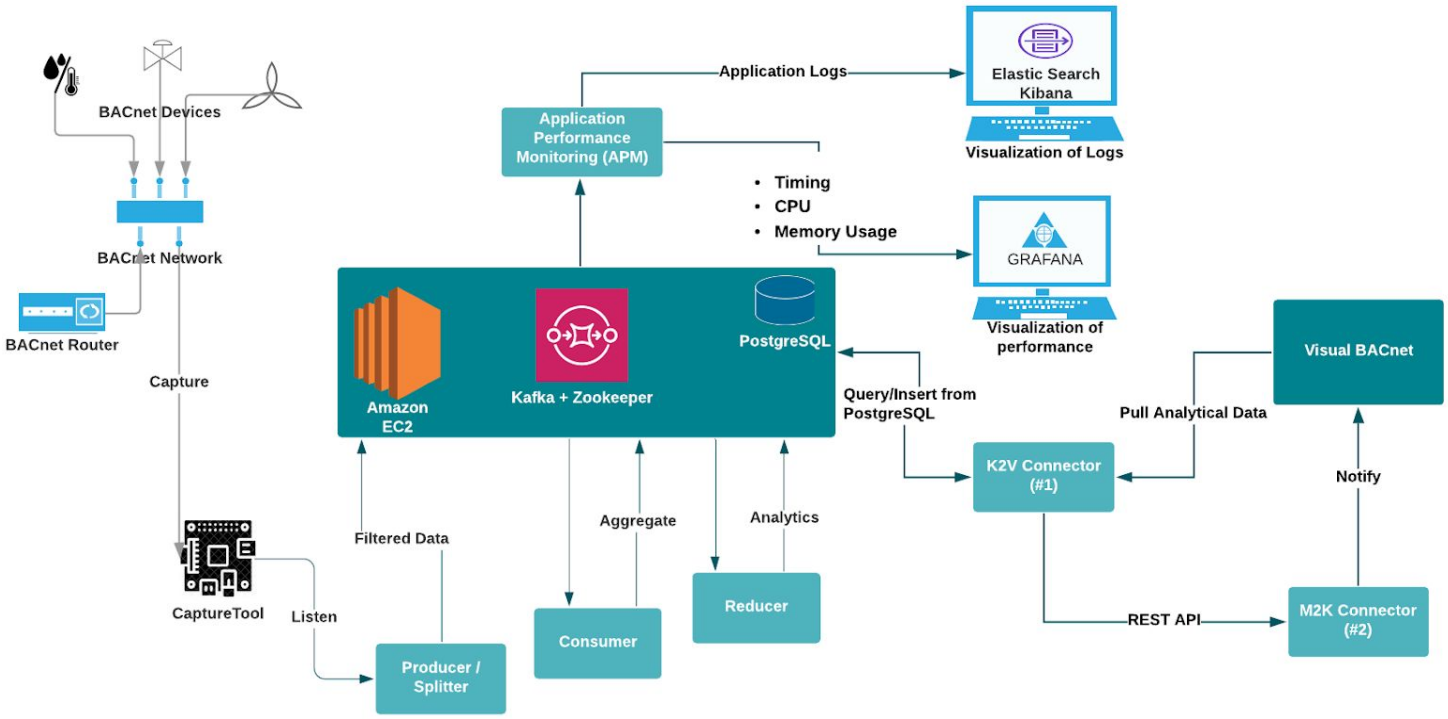


Figure 2.2 - Engineering Prototype System Overview

The majority of the Engineering Prototype will be focused on optimization of the PoC and integration with Optigo Network's Visual BACnet SaaS product. Regarding optimization, the Producer module now will not only filter and compress the raw packet data, but also systematically slice this data down to appropriate sized packet chunks (PCs) to reduce the amount of processing required for the Consumer and Reducer modules and enable the potential of load balancing (See **Section 7.2** for more details). Using application performance monitoring (APM) through Grafana and Kibana allows further improvement to fully optimize the microservice architecture as well as load balancing for the final product (See **Section 7.1** for more details).

Regarding integration, the K2V Connector is introduced to act as an intermediary, inserting the analytical data analyzed in our system into a PostgreSQL database on the AWS cloud and allowing Visual BACnet to pull this data when ready. The M2K Connector notfies Visual BACnet when PCAP file analytics are ready to be pulled, allowing encapsulation from our system in the event it is compromised in any way. (See **Section 6** for more details).

2.3 PCAP Life Cycle

The PCAP file which contains all the necessary BACnet information is passed along sequentially across the modules. The end-to-end life cycle is summarized in **Table 2.1**.

Steps	Source	Destination	Data Description
Step 1: The BACnet device sends its necessary status, information, and alerts through the BACnet network which ultimately lands on the Capture Tool. Note that the Capture Tool may obtain other protocols than BACnet protocol.	BACnet Device	Capture Tool	Data: PCAP Protocol Type: Many
Step 2: The Producer listens onto the same data on the Capture Tool	Capture Tool	Producer	Data: PCAP Protocol Type: Many
Step 3: The Producer decides to filter out all protocols except for BACnet protocol PCAP files, which is sent to Kafka.	Producer	Kafka	Data: PCAP Protocol Type: BACnet
Step 4: The Splitter receives the PCAP files from Kafka.	Kafka	Splitter	Data: PCAP Protocol Type: BACnet
Step 5: The Splitter decides to divide the PCAP files into Packet Chucks (PCs) based on a size, which is sent to Kafka.	Splitter	Kafka	Data: PCs Protocol Type: BACnet
Step 6: The Consumer receives the PCs from Kafka.	Kafka	Consumer	Data: PCs Protocol Type: BACnet
Step 7: The Consumer interprets the PC Headers & Data in a CSV format that is understable for analytics to be performed on, which is sent to Kafka.	Consumer	Kafka	Data: CSV formatted BACnet Data Protocol Type: BACnet
Step 8: The Reducer receives the CSV formatted BACnet data.	Kafka	Reducer	Data: CSV formatted BACnet Data Protocol Type: BACnet

Step 9: The Reducer analyzes all the data available for the amount of checks that it is designed to do. The analyzed data is sent to Kafka. More information on the checks can be seen in **Appendix A Table A.1.2.**	Reducer	Kafka	Data: Analyzed Network Data
Step 10: Kafka will be able to send the analyzed network data to the Connector module.	Kafka	Connector	Data: Analyzed Data
Step 11: K2V Connector will be able to place (write) the analyzed data into the Postgres database.	K2V Connector	Postgres Database	Data: Analyzed Data
Step 12: M2K Connector will notify Visual BACnet that PCAP file analytics are read to be pulled	M2K Connector	Visual BACnet	Data: ACK
Step 13: K2V Connector will also provide an API layer for Visual BACnet to query results from said database (read).	PostgreSQL Database	K2V Connector	Data: SQL parsed Analyzed Data
Step 14: Visual BACnet (Client) accesses the analyzed data through a series of queries to database.	K2V Connector	Visual BACnet	Data: SQL Commands

Table 2.1 - PCAP Roadmap

3 System High level specifications

DES 3.1-PoC	The system shall process all incoming BACnet data using Capture Tool Interface.
DES 3.2-PoC	The system performance must be measurable through combination of Grafana, InfluxDB and Telegraf (**Section 6.1**).
DES 3.3-PoC	The system shall perform 7 BACnet analytical checks.
DES 3.4-PoC	The deployment of the system must be done with Nomad and AWS.
DES 3.5-PoC	The system modules shall be written in GoLang.
DES 3.6-EP	All PCAP data shall be transferred using SSL/TLS throughout the system.
DES 3.7-EP	The system must be able to display all performing checks on Visual BACnet.

DES 3.8-EP	The system must be able to handle processing a single PCAP that is sized up to 1GB.
DES 3.9-EP	The system shall be to be integrated into the current Optigo System using self-defined APIs
DES 3.10.-EP	The system modules shall be able to join/leave the network for load-balancing
DES 3.11-EP	The system modules must have redundancy, if one module fails, a copy of the same module will take over
DES 3.12-EP	The system software shall be updated alongside Optigo's update cycle
DES 3.13-EP	The system modules must work in Linux containers such as Docker
DES 3.14-EP	The system application logs will be monitored by Kibana (**Section 6.2**)

Table 3.1 - High Level General System Specifications

4 Software Design Specifications

4.1 Producer Module

DES 4.1.1-EP	The Producer shall receive the PCAP from the Capture Tool.
DES 4.1.2-EP	The Producer shall upload the PCAP file to Kafka.
DES 4.1.3-EP	The Producer shall tag a reference ID of the original raw PCAP into Kafka
DES 4.1.4-EP	The Producer shall requeue the PCAP in the event Kafka is unavailable due to connection issues.
DES 4.1.5-EP	The Producer shall requeue the PCAP in the event of Kafka failures.
DES 4.1.6-EP	The Producer must stash unsent PCAPs locally until the connection is re-established between the Producer and Kafka.
DES 4.1.7-EP	The Producer shall delete oldest PCAPs if the local storage is full & the connection to Kafka is off.
DES 4.1.8-EP	The Producer must have a throttling mechanism if the queue is too busy/full.
DES 4.1.9-EP	The filtering shall remove all network data except BACnet data.

DES 4.1.10-EP	The filtering must support MS/TP, BACnet IP.

Table 4.1 - Producer Module Specifications

Proof of Concept

The Producer resides on the Capture Tool. It is the first module in our system. It receives raw packets (network traffic) intercepted from smart building device communications. As the system is only interested in BACnet data, the primary concern at the early stage is how to separate these BACnet data from the various other different network data that streams into the Capture Tool. This brings us to the filtering. The secondary concern, to be delivered by the time of Engineering Prototype will be Splitting PCAP files by row, in order to facilitate Consumer load-balancing.

4.1.1 Producer BACnet Filter

Filtering is a step that the Producer performs on raw, incoming PCAP data stream. The process may significantly reduce the workload of the subsequent processes and is thus crucial to system performance. In terms of bandwidth, it is likely that this process may help ease congestion in this aspect as well.

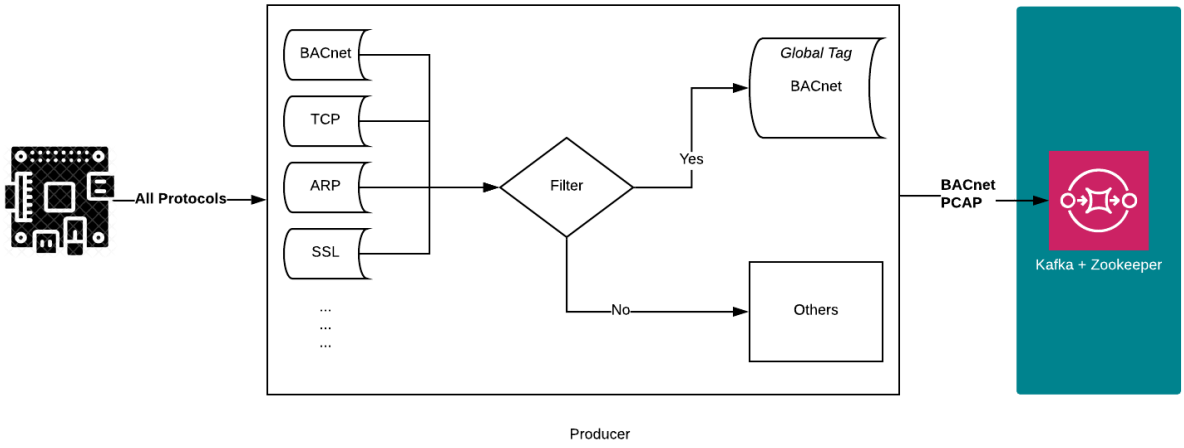


Figure 4.1 - Producer Proof of Concept

Figure 4.1 demonstrates the filter workflow. All network traffic is intercepted by the Capture Tool and streamed into our Producer module, consisting of numerous types of network data. The Producer performs a rudimentary inspection of each packet to determine its protocol type.

During this process, it determines the specific byte that delimits packets as well, i.e., parsing. This is convenient because it is necessary for our secondary feature of the Producer, the Splitter, to be able to split intelligently and not segment packet header or body.

The PCAP file format is defined as a standard for capturing packets of network data.

Figure 4.2 - PCAP Format

4.2 Kafka DSMS

DES 4.2.1-PoC	Kafka must run on a cloud server AWS S3.
REQ 4.2.2-PoC	There shall be methods to manage Kafka
REQ 4.2.3-EP	All modules shall only communicate through the queue on Kafka .
REQ 4.2.4-EP	Kafka shall be able to receive and handle requests from all other modules defined.
REQ 4.2.5-EP	Kafka must store data for at least 14 days.
REQ 4.2.6-EP	Kafka must garbage collect obsolete data.
REQ 4.2.7-EP	Kafka must be entirely transparent to the client.
REQ 4.2.8-EP	Kafka must be able to process data concurrently to achieve higher performance.

Table 4.2 - Kafka Module Specifications

4.2.1. Kafka Overview

Proof of Concept

Apache Kafka is a distributed streaming platform for the system modules, which is provided by Apache as open source. This allows our system to do 3 major tasks:

1. Publish and subscribe to streams of records, similar to a message queue
2. Store streams of records in a fault-tolerant durable way

3. Process streams or records as they occur

Kafka is used for the design because of its ability to build real-time streaming data pipelines that reliably get data between systems. More information about Kafka can be seen by the documentation provided by Apache [14].

By choosing cloud service providers, Kafka's performance is favourable due to the access, location, replication, and failure transparencies that it provides. **Figure 2.1 & 2.2** shows how Kafka will interact with the modules of the system.

Kafka is also part of the system design because the security it provides [14].

1. **Encryption of data in-flight using SSL / TLS**: This allows the data to be encrypted between the modules and Kafka.

2. **Authentication using SSL or SASL**: This allows the nodes to authenticate to the Kafka cluster, which verifies their identity. It's also a secure way to enable your clients to endorse an identity.

3. **Authorization using ACLs**: Once the clients are authenticated, the Kafka brokers can run them against access control lists (ACL) to determine whether or not a particular client would be authorised to write or read to some topic.

The basic architecture of Kafka is organized around a few key terms: Topics, Producers, Consumers, and Brokers. All Kafka messages are organized into Topics. If you wish to send a message you send it to a specific Topic and if you wish to read a message you read it from a specific Topic. A Consumer pulls messages off of a Kafka Topic while Producers push messages into a Kafka Topic. The data in a particular Topic may also be split, dividing a Topic into a number of Partitions. Partition allows for multiple Consumer groups to read from a topic in parallel, providing high message throughput[25].

4.3 SPLITTER

The reason for splitting PCAP file into chunks is simple and crucial - to divide a large PCAP file down to manageable sizes, in order to leverage the existing embedded infrastructure on site, namely, Capture Tools.

DES 4.3.1-EP	The Splitter shall be able to only receive raw PCAP from Kafka.
DES 4.3.2-EP	The Splitter shall be able split the PCAP file into PCs based on size.
DES 4.3.3-EP	The Splitter shall only split at the end of a header, preserving packet integrity
DES 4.3.4-EP	The Splitter can be operated under 2GB of RAM.

Table 4.3 - Splitter Module Specifications

Proof of Concept

Figure 4.3 below, shows the engineering prototype for the Splitter module. For the proof of concept, there will be no splitting of data as splitting mainly enables our system design to support load balancing in the Reducer and Consumer modules, please see **Section 6.2** for more information.

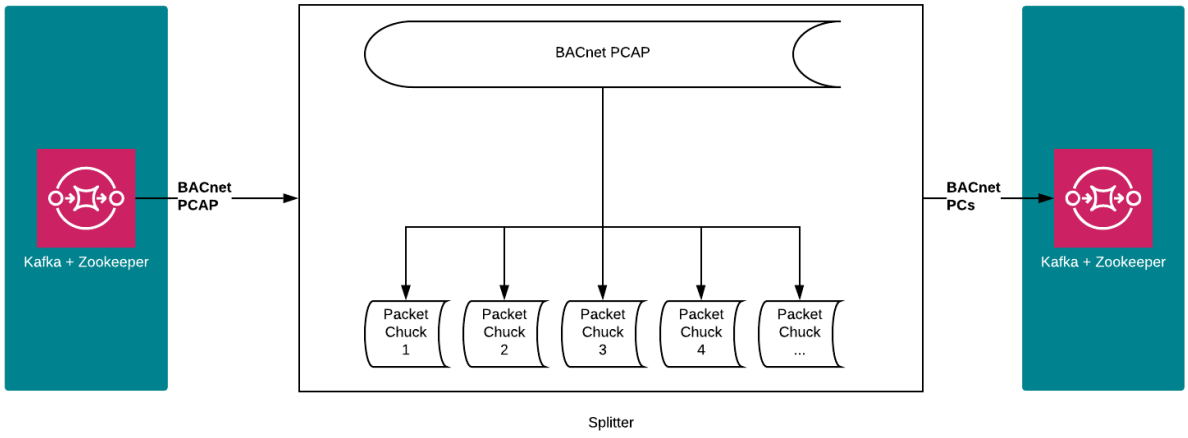


Figure 4.3 - Splitter Engineering Prototype

The Splitter is designed to split the PCAP file into a PCAP chunks which is sized based a parameter defined by the system. This is configured by the network administrator (user) based on network traffic conditions.

The Consumer module is the main beneficiary of this design. Through splitting a PCAP file, a Consumer instance is now tasked with only generating CSV data of PCAP chunks (more on this later in **Section 4.4** (Consumer Module), consisting of far few packets. By reducing the workload of a single instance of the Consumer, it can be ran on much more limited resources, carrying out the Micro-Service philosophy.

4.4 Consumer Module

DES 4.4.1-PoC	The Consumer shall be able to receive only required PCs from Kafka.
DES 4.4.2-PoC	The Consumer shall be able to format all PCs gathered from the Kafka to CSV format.
DES 4.4.3-PoC	The Consumer shall decode PCAP PCs using Wireshark.
DES 4.4.4-EP	The Consumer can be operated under 1GB of RAM.
DES 4.4.5-EP	The Consumer decoding task must finish under 10 minutes.

Table 4.4 - Consumer Module Specifications

Proof of Concept

The Consumer module decodes the packet chunks using Wireshark. The Consumer is designed to interpret all associated fields required for the analytics in a CSV format. The implementation is simple: Calling Wireshark specifies all the fields (column) and formatting parameters. However, which columns to extract becomes a bit more complex when considering load balancing to reduce computational time further, see **Section 7.2.3** for details. **Figure 4.4** shows the proof of concept for the Consumer module.

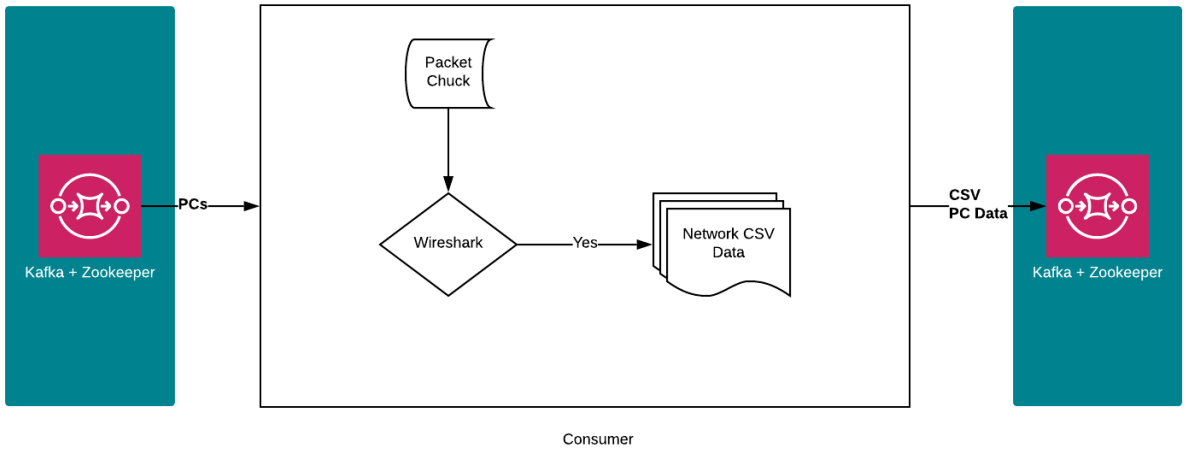


Figure 4.4 - Consumer Proof of Concept

Table 4.5 below, encapsulates a sampling of what fields are extracted using Wireshark:

Filter Reference	Field Name	Description
Frame		Display Filter Reference: Frame Protocol field name: frame
	frame.number	Frame Number
	frame.time_epoch	Epoch Time
	frame.len	Frame length on the wire
	frame.protocols	Protocols in frame
Malformed Packet		Display Filter Reference: Malformed Packet Protocol field name: _ws.malformed
	_ws.malformed.expert	Malformed Packet (Exception occurred)
Ethernet		Display Filter Reference: Ethernet Protocol field name: eth
	eth.src	Source
	eth.dst	Destination
	eth.fcs.status	FCS Status

Internet Protocol		Display Filter Reference: Internet Protocol Version 4 Protocol field name: ip
	ip.src	Source
	ip.dst	Destination
	ip.checksum.status	Header checksum status
Transmission Control Protocol		Display Filter Reference: Transmission Control Protocol Protocol field name: tcp
	tcp.srcport	Source Port
	tcp.checksum.status	Checksum Status
User Datagram Protocol		Display Filter Reference: User Datagram Protocol Protocol field name: udp
	udp.srcport	Source Port
	udp.dstport	Destination Port
	udp.checksum.status	Checksum Status
BACnet MS/TP		Display Filter Reference: BACnet MS/TP Protocol field name: mstp
	mstp.src	Source Address
	mstp.dst	Destination Address
	mstp.frame_type	Frame Type
	mstp.checksum.status	Checksum Status
BACnet Virtual Link Control		Display Filter Reference: BACnet Virtual Link Control Protocol field name: bvlc
	bvlc.function	Function
	bvlc.fwd_ip	IP
	bvlc.fwd_port	Port

Building Automation and Control Network NPDU		Display Filter Reference: Building Automation and Control Network NPDU Protocol field name: bacnet
	bacnet.mesgtyp	Network Layer Message Type
	bacnet.hopc	Hop Count
	bacnet.dnet	Destination Network Address
	bacnet.dlen	Destination MAC Layer Address Length
	bacnet.dadr_mstp	DADR
	bacnet.dadr_eth	Destination ISO 8802-3 MAC Address
	bacnet.dadr_tmp	Unknown Destination MAC
	bacnet.snet	Source Network Address
	bacnet.slen	Source MAC Layer Address Length
	bacnet.sadr_mstp	SADR
	bacnet.sadr_eth	SADR
	bacnet.sadr_tmp	Unknown Source MAC
	bacnet.rejectreason	Reject Reason
Building Automation and Control Network APDU		Display Filter Reference: Building Automation and Control Network APDU Protocol field name: bacapp
	bacapp.instance_number	Instance Number

NOTE: All Filters used may be subject to change.

Table 4.5 - List of Network Filters

4.5 Reducer Module

A Reducer instance is responsible for either the entirety of checks, or a subset of checks pertaining to a single PCAP file, **See Figure 4.5** below.

DES 4.5.1-PoC	In a single PCAP, a Reducer instance is responsible for all analytical checks.
DES 4.5.2-PoC	The Reducer shall be able to receive from Kafka.
DES 4.5.3-PoC	The Reducer shall contain a unified data structure on output data for the Consumer.
DES 4.5.4-EP	The Reducer shall be able to aggregate analytical data of every Consumer without missing data.
DES 4.5.5-EP	The Reducer shall be able to distinguish PCs that depend on each other.
DES 4.5.6-EP	The Reducer must be modular enough to add in more analytics/checks in the future
DES 4.5.7-EP	The Reducer can be operated under 4GB of available RAM.
DES 4.5.8-EP	Load Balancing on the Reducer must be present. See **Section 7.2.**

Table 4.6 - Reducer Module Specifications

Proof of Concept

For proof of concept, a single Reducer instance will be responsible for the entirety of checks pertaining to a single PCAP file. This means it will have O(n) performance, where n is the number of rows in a PCAP file. In this design, in the Reducer, there is an 'iterate' of sorts. This iterator is visible by all the checking methods. It starts from the first row, and is only incremented when all checking methods are finished processing this row. For trivial checks, this could mean only checking if a CRC checksum is incorrect. For more complex checks like Round Trip Time, this could mean checking if this is an appropriate datagram, and if so, calculating the time difference between this datagram and the previous occurrence.

When it is only responsible for a subset of checks pertaining to a single PCAP file, this becomes a bit more complex. Please see **section 7.2.4** for detailed analysis**.**

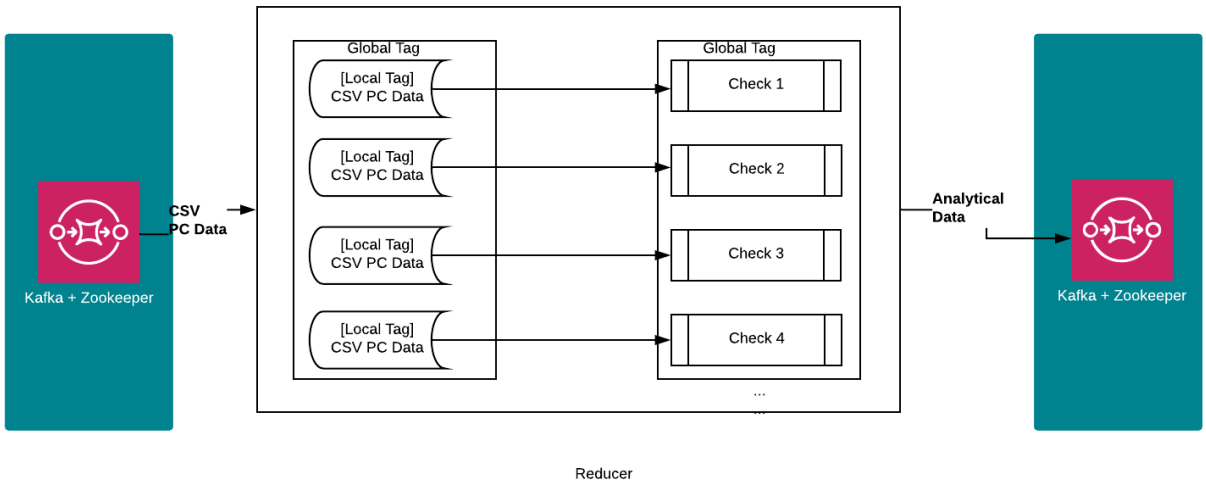


Figure 4.5 - Reducer Proof of Concept

5 Hardware Design Specifications

DES 5.1-PoC	The Capture Tool shall run a headless Linux OS.
DES 5.2-PoC	The Capture Tool shall be on a 32 or 64-bit ARM platform.
DES 5.3-PoC	The Capture Tool must have a minimum of 1GB memory
DES 5.4-PoC	The Capture Tool must have internet connection via wired ethernet.
DES 5.5-PoC	The Capture Tool internet connection must have a minimum of 5 mbit upload/download to maintain Oakion Systems softwares' usability
DES 5.6-EP	The Capture Tool OS maintains a boot time within 5 minutes.
DES 5.7-EP	The Capture Tool must support current Optigo software update cycles.
DES 5.8-EP	Current Optigo functionalities on the Capture Tool remains untouched after Oakion Systems softwares is deployed.

Table 5.1 - Capture Tool Specifications

Proof of Concept

Since the hardware design for this project pertains mostly toward the existing infrastructure currently deployed by Optigo Networks, this design will not be modified for ease of integration

for the customer and the cost/benefit of our design. The Capture tools are used to send and receive packet data from smart devices to the Distributed Computing Network. These Capture Tools are embedded devices enclosed in cases and deployed to customer sites to act as intermediaries between the smart devices and Optigo Networks. For our prototype and final design, we will be using a Capture Tool to provide the same functionality that it does in the field toward our network. In summary, the Capture Tool will match the same specifications as the current ones deployed by Optigo, for ease of integration as well as accuracy in the measurement of optimization our network provides in comparison to the current architecture at Optigo Networks.

6 Integration with Visual BACnet

Integration with Visual BACnet, as defined above in **Section 2.1 & 2.2** will be done for the Engineering prototype stage of design (EP) and will not be present in the Proof of Concept (PoC).

6.1 K2V Connector (#1)

DES 6.1.1-EP	The K2V Connector must receive data from Kafka
DES 6.1.2-EP	The K2V Connector must be insert into PostgreSQL database
DES 6.1.3-EP	The K2V Connector must have a REST API for reading data
DES 6.1.4-EP	The K2V Connector must have be designed to replace Aether (current architecture)
DES 6.1.5-EP	The K2V Connector must notify M2K Connector that the analytical data is processed

Table 6.1 - K2V Connector Specifications

KV2 Connector is primarily an intermediary tool between Visual BACnet and our Microservice system. The tool allows data to be stored into a PostgreSQL database located on the cloud and allows this data to be pulled correctly at the correct time when Visual BACnet is primed to receive, see **Figure 6.1**.

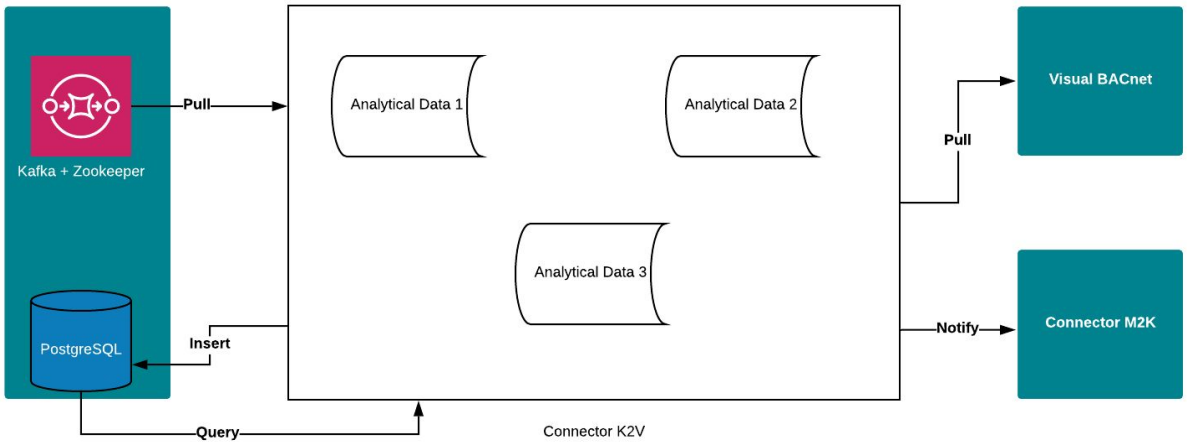


Figure 6.1 - K2V Connector Proof of Concept

6.2 M2K Connector (#2)

DES 6.2.1-EP	The M2K Connector must receive notifications from Connector 1 through REST API
DES 6.2.2-EP	The M2K Connector must send alerts to Visual BACnet that the PCAP file is complete.

Table 6.2 - M2K Connector Specifications

M2K Connector allows our system to notify Visual BACnet when analytics on a file are ready to be pulled to their frontend visualization system. The implementation will be written using REST API to connect between the KV2 connector and Visual BACnet, allowing for synchronization between our system and the customers, see **Figure 6.2**. M2K may seem redundant given KV2's existence in our system, but is necessary given that its deployment on Optigo's side allows their frontend to only pull when data is ready as well be informed in the

case that our system is offline.

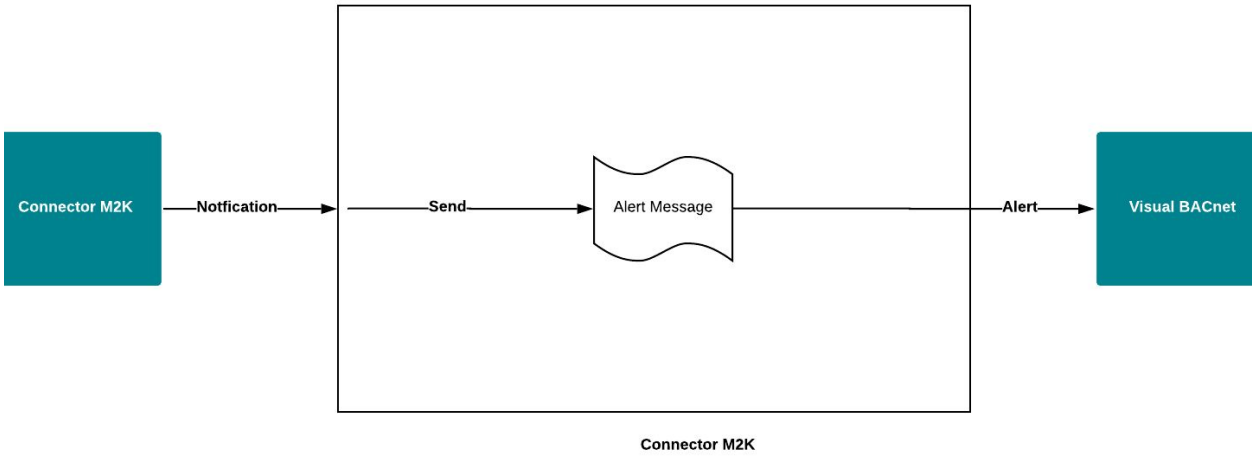


Figure 6.2 - M2K Connector Proof of Concept

6.3 PostgreSQL Database

DES 6.3.1-EP	The database must be able to receive data from K2V Connector
DES 6.3.2-EP	The database must store data in appropriate schema
DES 6.3.3-EP	Each check schema must have a primary key that links to the pcap schema

NOTE: The design of the database is subject to change

Table 6.3 - Database Design Specifications

Our proposed system must store the information analyzed in some organization manner for Visual BACnet to access. A relational database is the final component of our system and is located on the AWS cloud[22]:

1. **Self-describing nature of a database system**
    a. Contains the data & metadata of information which defines and describes the data and relationships between tables in the database.
2. **Data sharing**
    a. It allows for data sharing among employees and others who have access to the system
3. **Data Independence**

a. the data descriptions or metadata are separated from the system. This is possible because changes to the data structure are handled by the database management system and are not embedded in the system itself.

The PostgreSQL management system was chosen for the relational database for three main reasons[23]:

1. Open Source database management system
2. PostgreSQL is ACID compliant from ground up and ensures that all requirements are met
3. PostgreSQL is widely used in large systems where read and write speeds are crucial and data needs to validated.

Proof of Concept

Figure 6.3 shows a proof of concept on how the system will store analytical data once the network health checks have been completed by the Reducer module. As mentioned in **Section 6.1** and **Section 6.2**, the K2V Connector will insert and query from the database.

The database design has 2 major components:

- Each PCAP information will be stored in its own schema
  - Frame Number
  - Device Number
  - Vendor
  - Anything that is common between all checks

- Each check on the system will contain its own schema

This allows Visual BACnet to access required checks for any particular PCAP within the system. The combination between K2V Connector, M2K Connector and a PostgreSQL database fully supports our microservice solution.

Figure 6.3 - Database Proof of Concept

7 Overall Performance Specifications

7.1 Application Performance Monitoring

Application performance monitoring, as defined above in **Section 2.1 and 2.2** will be done using Grafana and Kibana, the latter of which will only be present during the Engineering prototype stage of design (EP) and will not be present in the Proof of Concept (PoC).

7.1.1 Grafana

DES 7.1.1-PoC	InfluxDB must store all measured metrics from modules
DES 7.1.2-PoC	Grafana must display the time of the life cycle of a PCAP for each module
DES 7.1.3-EP	Grafana must display the memory consumption for each module
DES 7.1.4-EP	Grafana must display the CPU consumption for each module

Table 7.1 - Grafana Specifications

In order to produce metrics with the least overhead, Oakion Systems chose Grafana, which is designed for analyzing and visualizing metrics such as system CPU, memory, disk and I/O utilization. Grafana is our metrics monitoring system because it is open source, and has the most appropriate visualization, dashboard creation, and customization available for our design.[28] It is feature-rich, easy to use, and very flexible. See **Section B.3** for visualization of Grafana.

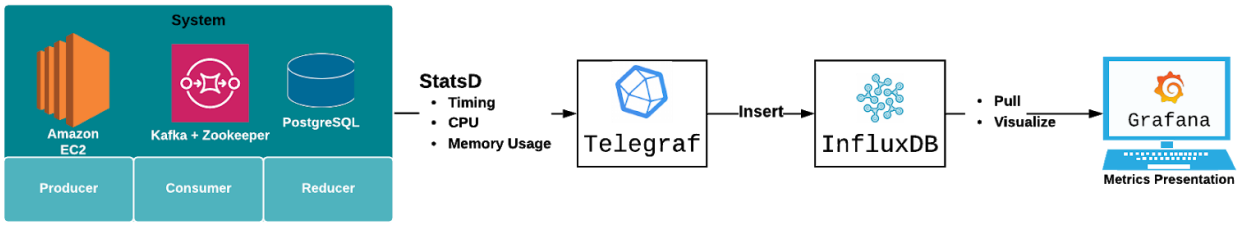


Figure 7.1 - Grafana Integration with our System[16]

Figure 7.1 encapsulates each module metrics such as CPU, RAM, load, and network traffic to Grafana.

1. Telegraf will collect all module time-series data (metrics) in real-time.
2. Telegraf will insert all necessary time-series data (metrics) into InfluxDB.
3. Grafana will pull all required information from InfluxDB to be displayed to the user

7.1.2 Kibana

DES 7.1.5-PoC	Kibana will display application logs for Kafka
DES 7.1.6-EP	Kibana will display application logs for all modules
DES 7.1.7-EP	BEATS must collect all application logs and metrics

DES 7.1.8-EP	Elasticsearch must store all application beat logs and beat metrics

Table 7.2 - Kibana Specifications

In order to properly display application logs, Oakion Systems chose Kibana, which is designed for analyzing and visualizing logs such as troubleshooting, forensics, and security. Our system depends on Kibana to produce a rich variety of visualization types, allowing us to create charts & tables, single metric specifications, time series and markdown visualizations, which are combined into dashboards. See **Section B.3** in **Appendix B** for visualization of Kibana.

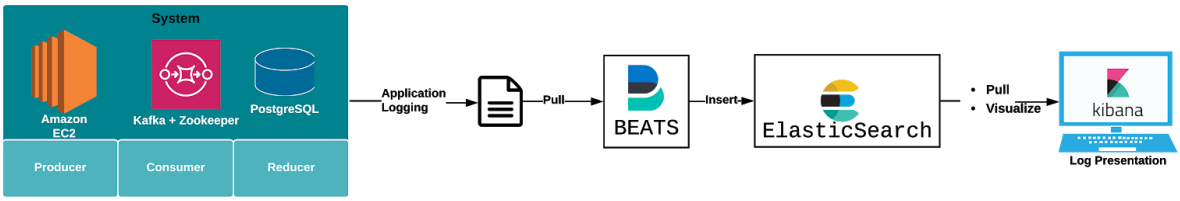


Figure 7.2 - Kibana Integration with our System[16]

As **Figure 7.2** illustrates, application logging generated by our system is monitored and pulled by BEATS. BEATS inserts new logging entries in ElasticSearch Time Series Database. The logging data is then pulled by Kibana for visualization.

7.2 Microservice Load balancing

One of the key features of our product is to have flexible scaling through automatic load balancing. During a surge of data influx, additional Consumer and Reducer instances can be launched easily to exploit parallelism. Therefor, this will require some changes to our system's internal tracking of data.

7.2.1 Efficiency analysis example

To balance workload amongst multiple Consumers and Reducers, we will segment the data set. Row splitting is mentioned previously in **Section 4.3** in detail. Let us now examine column splitting through an example.

Suppose for a total column set of $S \in \{1, 2\ldots n\}$, there exist checks A and B such that they both require the same column subset, $S1 \in \{1, 2, 3\}$, where $S1 \subset S$. Therefore, a Consumer may group columns 1, 2, and 3, publish data in S1 as a sub-topic, as opposed to publishing S as a whole.

On the receiving end, a Reducer instance is now ready to subscribe to the topic containing S1 instead of S. The remaining checks may be handled by another Reducer subscribed to S2 = {1, 4, 5… n} for example, or further divided.

Note that $S1 \cap S2 \neq \emptyset$. This is not only true because of the need for row ID in each individual column subset, but also because checks in different subsets may overlap in the columns it require. The implication of this is that column subsets should not be split too granularly. In the extreme case, if each Reducer only handles a single check, then for 35 checks our performance will degrade to O(35n), on par with iterating through the PCAP rows each time a check is performed, but also with additional I/O overhead due to the aforementioned overlapping.

If pre-filtering is done correctly, the PCAP is completely processed when $S1 \cup S2 \cup ... \cup Sn = S$, where n is the total number of column subsets. In practice, we will simply use the completion of all the checks as terminating condition. **Figure 7.3** illustrates column splitting in conjunction with row splitting in our system.

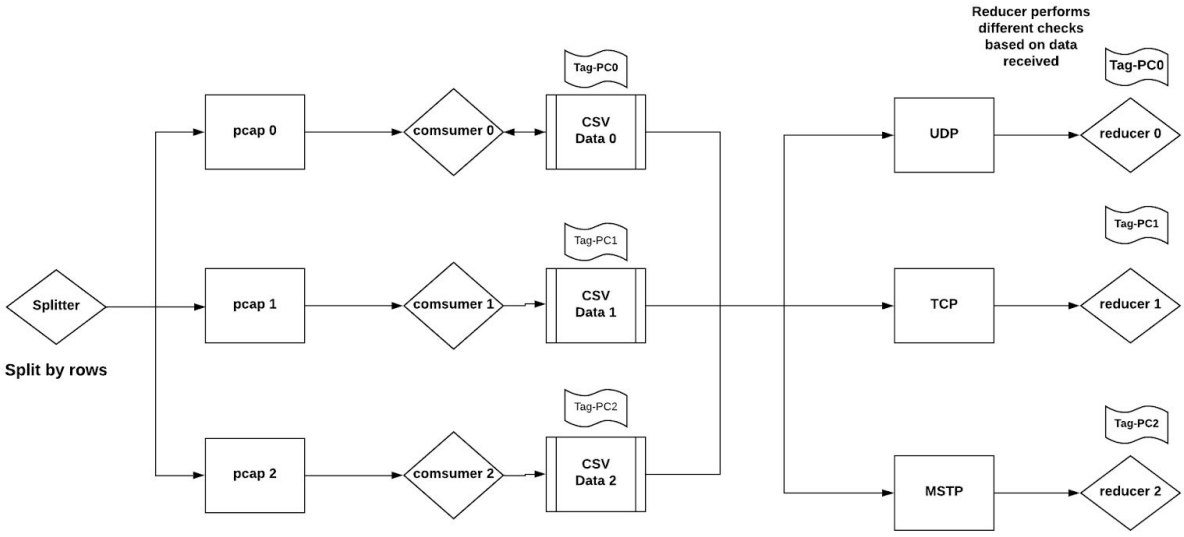


Figure 7.3 - Diagram of the combination of Consumer and Reducer load balancing processing and analyzing the PCAP in parallel

7.2.2. Dual Queue Design

A dual queue set up is now necessary to facilitate our synchronization needs in the new column splitting pattern: One queue is necessary for notifications and one for data. We will name them Job Queue and Work Queue, respectively. Recall in **Section 4.2.1** Kafka's Topic, Partition, and Messaging paradigm are introduced. They will now be defined and their roles explained for each of these three components in both queues.

The *Job Queue* serves as the coordinator of the system. The Topic serves as the unique identifier of a PCAP file being processed. The Partition on the other hand, defines the column subset, corresponding to some number of checks. The Message is flexible, it is assigned the total number of rows for checksum purposes. When a new PCAP file reaches our system, each of its unprocessed column subsets will be published to the Job Queue as a Partition. The idling Reducer can thus be notified and assigned a column subset to process. The actual data will however be received through the Work Queue.

The Work Queue will be responsible for delivering data to the correct Reducer instances. Thus each PCAP file will be allocated a Work Queue of its own. The Topic is defined to be the column set of a single PCAP. The Message will contain numerous data objects including row chunk ID, total chunk number, and the data chunks in the column subset. We leave this queue unpartitioned, as we do not need these messages to be consumed in parallel. In fact, only one Reducer should receive all the data pertaining to a Topic.

The Consumer instance publishes its column subset (this is detailed in **Section 7.2.3**) data to the Work Queue, and the subscribed Reducer instance will retrieve this data, until it obtains all the rows, to be able to perform analytical checks (this is detailed in **Section 7.2.4**).

In the case of a small PCAP file or load-balancing is simply turned off, the Job Queue becomes unpartitioned (only column subset includes all the columns), and the Work Queue only passes one message per Topic (the all column subset has a single row chunk, being all the rows in the PCAP file).

7.2.3 Consumer Load-balancing

DES 7.2.1-EP	The Consumer must split the column into subsets grouped based on the checks of that group.
DES 7.2.2-EP	The Consumer shall tag each column group for the Reducer, in order to uniquely identify the PCAP, PCAP chunk, and column subset
DES 7.2.3-EP	The order in which the Consumer sends data does not matter

Table 7.3 - Consumer Load-Balancing Specifications

Following the PCAP lifecycle of **Section 2.3**, after receiving each raw PCAP data, the Producer module will assign to it a global tag to uniquely identify this raw PCAP data in the event of multiple PCAP files existing in the pipeline. A new Topic is then added or resetted in the Job Queue, with a new pairing Work Queue spawned. Next, this data is split into packet chunks, with each chunk tagged as well, as seen in **Figure 7.4** below. As discussed in **Section 3.1.2**,

Consumer instances may now take advantage of this reduced workload in CSV generating, and concurrently process a single PCAP file.

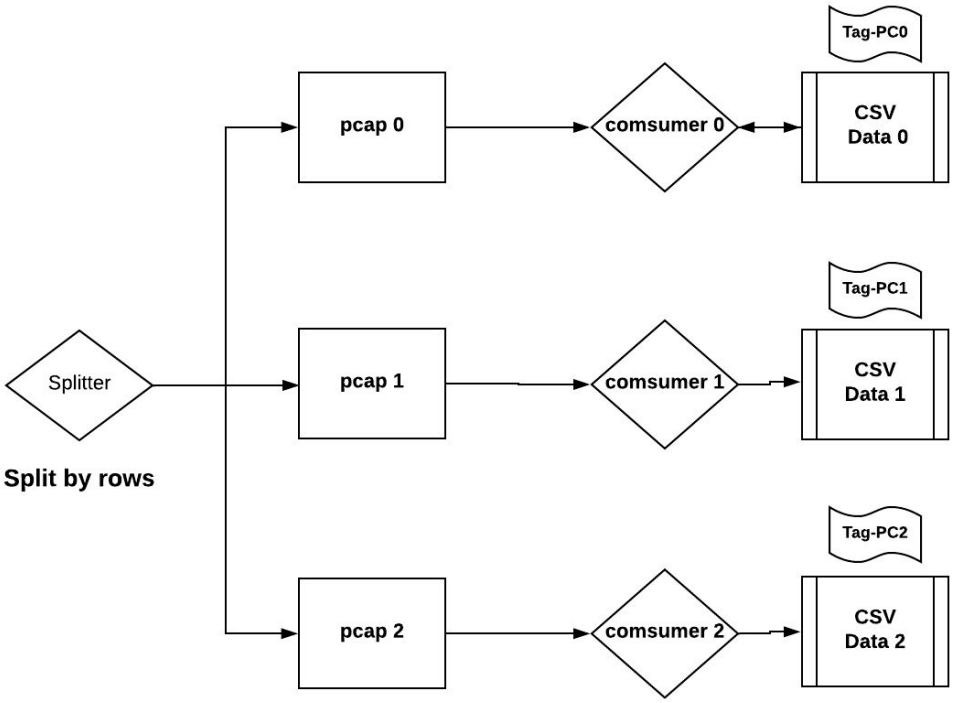


Figure 7.4 - Diagram of Consumer load balancing splitting by row, analyzing each chunk in parallel before placement in CSV

As discussed before in **Section 7.2.1**, since the column requirement for each check is predetermined, the Consumer further splits CSV data into columns subsets pertaining to one or more checks. The architecture will be smart enough to group which checks go with which column data for the Reducer to analyze, as demonstrated in **Figure 7.4** above. Each column subset that Consumer generates will be tagged with an identifier (topic), and published to the Work Queue of this PCAP file.

Thus this column tag, together with the aforementioned PCAP file tag and PCAP chunk tag, ensures that a Reducer instance will only process specific column data needed by the grouped checks, without processing any redundant data not needed by the column subset.

7.2.4 Reducer Load-balancing

DES 7.2.4-EP	Reducer will only listen on data of the specific PCAP topic, column subset Partition, and corresponding row chunks

DES 7.2.5-EP	Reducer will gather all row chunks in its lifecycle
DES 7.2.6-EP	Reducer will sort PCAP chunks using their PC tag, to recover the original CSV order.
DES 7.2.7-EP	Reducer will perform the checks corresponding to the Column set tag on the ordered CSV column subset.

Table 7.4 - Reducer Load-Balancing Specifications

The CSV data for a single PCAP file is now ready to be subdivided amongst multiple Reducer instances (see **Figure 7.5** below). An idling Reducer 0 for instance, will subscribe to a Topic, representing a PCAP file currently in the pipeline. Kafka will then assign it a Partition of this Topic - a column subset. Based on this subset, Reducer now knows to only process the checks pertaining to these columns - UDP for example.

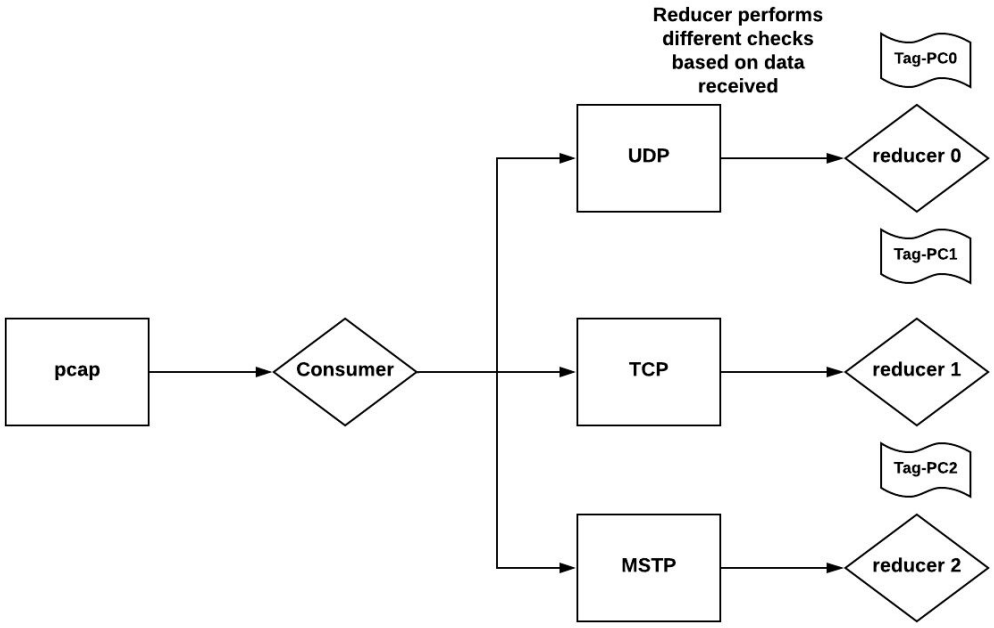


Figure 7.5 - Diagram of Reducer load balancing splitting by column analyzing each chunk in parallel before results stored in CSV

The Packet Chunk tagging mentioned earlier in **Section 7.2.3** becomes significant, as the first packet chunk of each column subset will mark the start of the Reducer's lifecycle. The Reducer will persist while listening for additional incoming Packet Chunks, until it receives all rows of the CSV column subset.

For trivial checks to verify the integrity of packets, such as CRC checksum calculation, the

checking may be performed while the wait for all Packet Chunks is underway. For complex checks that require global awareness, e.g., Round Trip Time, the Reducer will simply wait until a complete CSV column subset is received before performing the appropriate checks.

As a conclusion, each Reducer will be capable of either doing all the checks, or only a subset of checks pertaining to the column subset in load-balancing mode. Through this we allow multiple Reducer instances perform analytics in parallel on each PCAP file, achieving Load-Balancing. This analytical data is then saved in separate csv files per check and uploaded to kafka.

8 Conclusion

This document details the modular design process for Oakion Systems' Distributed Computing System. It encompasses the implementation of each individual subsystem along with the reasoning behind their chosen architectures and functionality. The communication between these subsystems is also justified, pipelining the data analysis process and creating the optimized middleware microservice system guaranteed by Oakion Systems. Performance specifications are also provided to support these claims. The following is a brief summary of each principal section defined throughout this document:

General Specifications

- The system will be scalable and able to process all incoming BACnet data then integrate with the current Optigo system.
- The system will not disclose any customer or commercial information beyond the scope of Optigo.

Software Specifications

- Producer Module: This module will receive raw PCAP data from the Capture Tool, filter the PCAP for data pertaining only to BACnet traffic, then send this filtered data to Kafka
- Splitter Module: This module will receive filtered PCAP data from Kafka, split the PCAP file into PCs based on size, before sending the PCs back to Kafka
- Consumer Module: This module will receive PC data from Kafka, decode the PCs using Wireshark and configure the data in a CSV format, before uploading the data back to Kafka.
- Reducer Module: This module will receive CSV formatted data from Kafka, process and analyze the data for the network health information, before sending this health report data back to Kafka

- Kafka: This module will act as the centralized queue for the system living on the cloud and receiving, storing and sending PCAP data throughout the system wherever applicable
- Grafana: Will visualize the data pertaining to analytics of each module for performance metrics
- Kibana: Will visualize the application logs pertaining to each module for internal and external auditing
- PostgreSQL DB: Will store analytical data (checks) pertaining to formly processed PCAP files
- K2V Connector: Will populate the Postgres DB with analytical data from Kafka and allow data to be pulled from Postgres from Visual BACnet
- M2V Connecter: Will notify Visual BACnet that data is ready to be pulled
- Visual BACnet: Will visualize the network health report data processed by the Distributed Computing Network system

Hardware Specifications

- Capture tool will match the same specifications as the current model deployed by Optigo and and communication to the DSMS will be encrypted

Overall Performance Specifications

- The Splitter module enables our system design to support load balancing in the Reducer and Consumer modules
- Load balancing for the system will be accomplished mainly on the Consumer and Reducer modules as this is where the bulk of computation resides

9 References

[1] S. Chan, A. Nguyen, J. Singh, T. Tan, and S. Wang, REQUIREMENT SPECIFICATION - *Distributed Computer Network.* Oakion Systems, 2019.

[2] "Solutions for Compute-Intensive Environments," *Distributed Computing and Grid Computing Solutions from Digipede*. [Online]. Available: http://www.digipede.net /solutions/distributed-computing.html. [Accessed: 04-Feb-2019].

[3] N. Peck, "Microservice Principles: Smart Endpoints and Dumb Pipes," *medium.com*, 01-Sep-2017. [Online]. Available: https://medium.com/@nathank peck/microservice-principles-smart-endpoints-and-dumb-pipes-5691d410700f. [Accessed: 02-Feb-2019].

[4] M. Rouse, "What is discoverability (in UX design)? - Definition from WhatIs.com," *WhatIs.com*. [Online]. Available: https://whatis.techtarget.com/definition/ discoverability-in-UX-design. [Accessed: 28-Feb-2019].

[5] Carneiro Jr, C., & Schmelmer, T. (2016). *Microservices From Day One: Build robust and scalable software from the start*. Berkeley, CA: Apress.

[6] M. Fowler, "Microservice Trade-Offs" *Martinfowler.com.* [Online]. Available: https:// martinfowler.com/articles/microservice-trade-offs.html. [Accessed: 03-Mar-2019].

[7] "Kafka, A distributed streaming platform," *Apache*. [Online]. Available: https://kafka.apache.org/intro. [Accessed: 04-Feb-2019]

[8] A. Kharenko, "Monolithic vs. Microservices Architecture – Microservices Practitioner Articles," *Microservices Practitioner Articles*, 09-Oct-2015. [Online]. Available: https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59. [Accessed: 06-Feb-2019].

[9] "Technology Dictionary Cloud Storage," Techopedia. [Online]. Available: https:// www.techopedia.com/definition/26535/cloud-storage. [Accessed: 04-Feb-2019].

[10] "Visual BACnet," *Optigo*. [Online]. Available: http://www.optigo.net/visual-bacnet. [Accessed: 03-Mar-2019].

[11] "Technology Dictionary Microservices," *Techopedia*. [Online]. Available: https://www.techopedia.com/definition/32503/microservices. [Accessed: 04-Feb-2019].

[12] "Mobile endpoint security: What enterprise infosec pros must know now", *IoT Agenda*, n.d. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. [Accessed: 06-Feb-2019]

[13] "Introduction to Apache Kafka Security," *Medium*. [Online]. Available: https://medium.com/@stephane.maarek/introduction-to-apache-kafka-security-c8951d410adf. [Accessed: 03-Mar-2019]

[14] IEEE Guide: Adoption of ISO/IEC TR 24748-3:2011, Systems and software engineering-Life cycle management-Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes)," in *IEEE Std 24748-3:2012* , vol., no., pp.1-130, 20 April 2012 doi: 10.1109/IEEESTD.2012.6189321

[15] M. Fowler, "Continuous Delivery" *Martinfowler.com.* [Online]. Available: https://martinfowler.com/bliki/ContinuousDelivery.html. [Accessed: 03-Mar-2019].

[16] "Open Source Time Series Platform," *InfluxData*. [Online]. Available: https://www.influxdata.com/time-series-platform/. [Accessed: 08-Mar-2019].

[17] M.Rouse, "Discoverability (in UX design)", *Whatis*. [Online]. Available: https://whatis.techtarget.com/definition/discoverability-in-UX-design [Accessed: 08-Mar-2019]

[18] IEEE Guide: Adoption of ISO/IEC 14756:1999, *Information technology* — Measurement and rating of performance of computer-based software systems, ISO/IEC JTC 1/SC 7, vol., no., pp.1-130, 3. [Accessed: 08-Mar-2019]

[19] IEEE Guide: Adoption of ISO 9241-161:2016, *Ergonomics of human-system interaction -- Part 161: Guidance on visual user-interface elements*, ISO/TC 159/SC 4, vol., no., pp.1-130, 14. [Accessed: 08-Mar-2019]

[20] "First Impressions", *Psychology Today*, n.d. [Online]. Available: https://www.psychology
today.com/ca/basics/first-impressions [Accessed: 13-Mar-2019]

[21] A.Watt, "Database Design - 2nd Edition" *BC Open Textbooks*. [Online]. Available: https://opentextbc.ca/dbdesign01 [Accessed: 13-Mar-2019]

[22] "PostgreSQL vs MySQL"., *2nd Quadrant*, n.d. [Online]. Available: https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql [Accessed: 13-Mar-2019]

[23] "What is Amazon Web Services (AWS)? - Definition from WhatIs.com," *SearchAWS*. [Online]. Available: https://searchaws.techtarget.com/definition/Amazon-Web-Services. [Accessed: 14-Mar-2019].

[24] K. Trapani, "What is Agile/Scrum," cPrime, 07-Jan-2019. [Online]. Available: https://www.cprime.com/resources/what-is-agile-what-is-scrum/. [Accessed: 14-Mar-2019].

[25] "Kafka in a Nutshell," *Kevin Sookocheff*. [Online]. Available: https://sookocheff.com/post/kafka/kafka-in-a-nutshell/. [Accessed: 13-Mar-2019]

[26] "Technology Dictionary Graphical User Interface (GUI)" *Techopedia*. [Online]. Available: https://www.techopedia.com/definition/5435/graphical-user-interface-gui. [Accessed: 13-Mar-2019].

[27] "Grafana Docs" *Grafana Labs*. [Online]. Available: http://docs.grafana.org/v4.3/. [Accessed: 08-Mar-2019].

[28] "Prometheus vs. Grafana vs. Graphite - A Feature Comparison", *Loom Systems.* [Online]. Available:https://www.loomsystems.com/blog/single-post/2017/06/07/prometheus-vs-grafana-vs-graphite-a-feature-comparison.[Accessed: 13-Mar-2019].

[29] "Enterprise Application Container Platform," *Docker*. [Online]. Available: https://www.docker.com/. [Accessed: 15-Mar-2019].

Appendix A: Supporting Test Plans

It is noted that the following supporting test plans encompass the testing of components and functionality initially present in the Engineering prototype (EP) stage of design. For this reasoning, validation of expected output is only certified for the EP and final design stages of this product.

A.1 Functional Tests

The purpose of functional tests is to determine the basic functionality of each individual module in the Distributed Computing Network. The modules are designed to have the capability to work individually without depending on other modules. The feature of this functionality allows the modules to be more robust, and less prone to error during integration. The following modules compromise the Functional Test Suite:

1. The Producer
2. The Splitter
3. The Consumer
4. The Reducer
5. The Connector #1 (K2V Connector)
6. The Connector #2 (M2K Connector)

Oakion Systems and those interested will be able to validate all Functional Test cases through **Table A.1.1**.

Notice that Kafka has not been tested for functionally as it is not created by Oakion Systems. Kafka will be tested with integration testing.

Functional Test Suite					
Module	**Function**	**Description**	**Input/Expected Outcome**	**Validation**	**Comments**
Producer	1. Receive Binary PCAP	The Producer can receive a binary PCAP file.	***Input***: Binary PCAP ***Expected output***: The same Binary PCAP	**Date:** **Initials:** **Pass/Fail**	
	2. Send Binary PCAP	The Producer can send a binary	***Input***: Binary PCAP	**Date:**	

		PCAP file.	***Expected output***: Binary PCAP in folder	**Initials:** **Pass/Fail**	
Splitter	1. Receive Binary PCAP	The Splitter can receive a binary PCAP file.	***Input***: Binary PCAP ***Expected output***: The same Binary PCAP	**Date:** **Initials:** **Pass/Fail**	
	2. Split Binary PCAP file	The Splitter can split the binary file by size into smaller Packet Chucks.	***Input***: Binary PCAP ***Expected Output***: Smaller PCAPs	**Date:** **Initials:** **Pass/Fail**	
	3. Send Binary PCs	The Producer can send a binary Packet Chunk.	***Input***: Packet Chunk ***Expected output***: Binary Packet Chuck in folder	**Date:** **Initials:** **Pass/Fail**	
Consumer	1. Receive Binary PCs	The Consumer can receive a binary PCs file.	***Input***: Binary PCs ***Expected output***: The same Binary PCs	**Date:** **Initials:** **Pass/Fail**	
	2. Convert Binary PCAP to CSV Format Data	The Consumer can convert PCAP binary into CSV Formatted Data.	***Input:*** The PC file ***Expected output***: A file that contains the CSV.	**Date:** **Initials:** **Pass/Fail**	
	3. Send CSV Formatted Data	The Consumer can send a CSV formatted Data	***Input***: CSV formatted Data ***Expected output***: CSV formatted Data into a folder	**Date:** **Initials:** **Pass/Fail**	

Reducer	1. Receive CSV Formatted Data	The Reducer can receive CSV Formatted Data	***Input***: CSV formatted Data ***Expected output***: The same CSV formatted Data	**Date:** **Initials:** **Pass/Fail**	
	2. Perform Checks	This will be covered in **Table A.1.2**			
	3. Send Analytical Data	The Reducer can send Analytical Data	***Input***: Analytical Data ***Expected Output***: The same Analytical Data stored in the database	**Date:** **Initials:** **Pass/Fail**	
K2V Connector	1. Insert the database with Analytical Data	The K2V Connector is required to insert the database with all types of Analytical Data	***Input***: Analytical Data ***Expected output*:** Populated Database of Analytical Data	**Date:** **Initials:** **Pass/Fail**	
	2. Query the Database for Analytical Data	The K2V Connector is required to query the database for all types of Analytical Data	***Input***: Populated Database ***Expected output*:** All data from the database	**Date:** **Initials:** **Pass/Fail**	
	3. Send Notifications to M2K Connector	K2V Connector is required to send notifications	***Input***: Notification Message ***Expected Output***: Send notification	**Date:** **Initials:** **Pass/Fail**	
M2K	1. Receive	M2K Connector is	***Input***: Notification	**Date:**	

Connector	notifications	required to receive notifications	Message ***Expected Output***: Message awareness	**Initials:** **Pass/Fail**	
	2. Send notifications	M2K Connector is required to send notifications	***Input***: Notification Message ***Expected Output***: Send notification	**Date:** **Initials:** **Pass/Fail**	

Table A.1.1 - Functional Test Suite

Each analytical check is systematically designed to encapsulate the logic inside the Reducer module. Therefore, Oakion Systems' Quality Assurance team can test the functionality of these checks independently of the system. Oakion Systems can then compare these results with Optigo's current Visual BACnet System for verification. **Table A.1.2** discloses the current checks implemented and being tested in the system.

Note that **Table A.1.2** is subject to change based on time constraints and limited resources.

BACnet Analytical Checks		
Check	**Description**	**Validation**
Checksum Error	**Indicates:** The percentage of packets with checksum errors in the capture. ***Reasons*** 1. Packet is malformed (it's gibberish). 2. On MS/TP, it typically means the packet was clobbered by the network (poor wiring). 3. On IP, it can be bad wiring or electrical influence on the wire. 4. Not a BACnet router problem, it's a physical router problem. The actual router itself is failing, or perhaps you have loose cable, or power fluctuations.	**Date:** **Initials:** **Pass/Fail** **Comments:**
Detect		**Date:**

Circular Networks	**Indicates :** The number of routers that are potentially in a circular network in the BACnet system. ***Reasons*** 1. Circular networks happen when you have two or more routes to the same controller. 2. Look at the Hop Count value of every packet. Would need to be BACnet IP. 3. If the Hop Count drops below 10, it gets flagged as a Circular Network. 4. Will always also trigger the Low Hop Count check. 5. Could happen with BACnet MS/TP, but it is rarer. 6. Typically, the routes are BACnet/IP and BACnet/Ethernet and both are communicating on both networks.	**Initials**: **Pass/Fail** **Comments:**
Duplicate Device ID	**Indicates**: The number of device ids with more than one SNET-SADR pair, which is an indication of device ID conflict in an internetwork. ***Reasons*** 1. More than one device on the same BACnet Network with the same BACnet Device Instance (aka Device ID). 2. Does not distinguish between different UDP ports, so if they are on different ports, it could be a false fail (have to drill down into frame info). 3. Still recommended to give them all unique IDs in case you need to change your applications or reconfigure the site in the future.	**Date**: **Initials**: **Pass/Fail** **Comments:**
Duplicate Source IP Address	**Indicates**: The number of SNET-SADR pairs with more than one device id, which is an indication of SADR conflicts in a network. ***Reasons***	**Date**: **Initials**:

	1. Similar to Duplicate Device ID, but instead looks at the SNET and SADR. 2. This fails if more than one device sends an I-Am with the same Source Network and Source Address. 3. In Duplicate Device ID, the DIP switch settings between two devices are different, but the Device IDs are the same. In Duplicate Source Address, you have different Device IDs, but the DIP switch settings are the same. 4. If they are on the same network and have the same MAC, this will fail (these are used to derive the Source Address). 5. More likely to occur in MS/TP, unlikely in IP and Ethernet (possible, but very unlikely).	**Pass/Fail** **Comments:**
Low Hop Count	**Indicates**: The number of devices sending packets with low hop count values. ***Reasons*** 1. Look at the Hop Count, more to identify incorrectly initialized Hop Count values. 2. Hop Count allows the packet to travel through as many routers as the Hop Count value. 3. Device should be configured to send a first packet with 255 hops. Each time it passes through a router, one "hop" is docked. It rarely goes below 250. 4. Some manufacturers configure using only four bits, so it starts at 15. This violates the BACnet standard. 5. Circular Network will always trigger a Low Hop Count. 6. Low Hop Count is always a warning, it never fails.	**Date**: **Initials**: **Pass/Fail** **Comments:**
Token Round Trip Time	**Indicates**: The average round-trip time for the token throughout the capture. ***Reasons***	**Date**: **Initials**:

	1. Amount of time it takes for a token to complete one full trip. 2. Shows the path the token took. 3. Time is measured from when a master receives the token to when it receives it again. 4. Average of all the token round trips for every master in the capture. 5. Long average round-trip time shows problems that may be consistent every trip. 6. Fails if Average Token Round-Trip time is more than 2000ms (2 seconds). 7. Warning if Average Token Round-Trip time is more than 85ms (0.085 seconds).	**Pass/Fail** **Comments:**
Unresponsive Router	**Indicates**: The number of BACnet networks that have at least one Who-Is-Router-To-Network unconfirmed service request associated but do not have any corresponding I-Am-Router-To-Network unconfirmed service requests. ***Reasons*** 1. Similar to unresponsive devices, but the router that is responsible for a network doesn't reply with I-Am router to network (with network specified). 2. Does not take into account Global Who-Is Router to Network (there is no network specified, so can't tell if one is missing).	**Date**: **Initials**: **Pass/Fail** **Comments:**

Table A.1.2 - BACnet Analytical Checks

A.2 Integration Tests

The purpose of Integration Test is to determine if a system can work in parts and expose faults in the interaction between the modules. The test suite shown in **Table A.2.1** is done based on the module and the communication it performs between its neighbours.

For example, earlier in **Figure 3.3**, for the Splitter, it receives PCAP data from Kafka and sends out PCs to Kafka.

Integration Test Suite		
Module Integration	**Description**	**Validation**
Capture Tool → *Producer* → Kafka	**Test** the Producer if it can receive PCAP from the Capture Tool and send the same PCAP to Kafka. ***Input:*** PCAP from Capture Tool ***Expected Output***: PCAP to Kafka	**Date**: **Initials**: **Pass/Fail** **Comments:**
Kafka → *Splitter* → Kafka	**Test** the Splitter if it can receive PCAP from Kafka and send PCs to Kafka, ***Input:*** PCAP from Kafka ***Expected Output***: PCs to Kafka	**Date**: **Initials**: **Pass/Fail** **Comments:**
Kafka → *Consumer* → Kafka	**Test** the Consumer can receive PCs from Kafka and send the CSV formatted data to Kafka. ***Input:*** PCs from Kafka ***Expected Output***: CSV Formatted Data to Kafka	**Date**: **Initials**: **Pass/Fail** **Comments:**

Kafka → *Reducer* → Kafka	**Test** the Reducer if it can receive CSV formatted data from Kafka and send Analytical Data to Kafka. ***Input:*** CSV Formatted Data from Kafka ***Expected Output***: Analytical Data to Kafka	**Date:** **Initials:** **Pass/Fail** **Comments:**
Kafka → *K2V Connector* → Database	**Test** if the K2V Connector can receive Analytical Data from Kafka and send Analytical Data to the Database. ***Input:*** Analytical Data from Kafka ***Expected Output***: Populated Database	**Date:** **Initials:** **Pass/Fail** **Comments:**
Database → *K2V Connector* → M2K Connector	**Test** if the K2V Connector can send notifications to M2K Connector. ***Input***: Notification message ***Expected Output***: Sent notification message	**Date:** **Initials:** **Pass/Fail** **Comments:**
K2V Connector → *M2K Connector*→ Visual BACnet	**Test** if the M2K Connector can receive notifications from K2V Connector send notifications to Visual BACnet ***Input***: Notification message ***Expected Output***: Sent notification message	**Date:** **Initials:** **Pass/Fail** **Comments:**

Table A.2.1 - Integration Tests

A.3 System Tests

The purpose of System Test is to evaluate a system's compliance with many PCAP files, End-To-End and requirements. The Distributed Computing Network should also report the performance of the modules through Grafana. The inputs of the system will be controlled through Oakion Systems where PCAP files are chosen to test the system for robustness. The test suite shown in **Table A.3.1** shows a summary of tests being performed.

System Test Suite		
System Test	**Description**	**Validation**
End - End System - Small sized PCAP	All modules are required to work together from the beginning (Capture Tool) to the end (Visual BACnet), for a small (under 1GB) PCAP file. ***Input***: PCAP File 1 under 1GB ***Expected Output***: Analytical Data on Visual BACnet	**Date**: **Initials**: **Pass/Fail** **Comments:**
End - End System - Max sized PCAP	All modules are required to work together from the beginning (Capture Tool) to the end (Visual BACnet), for the max sized PCAP file (1GB) ***Input***: PCAP File at 1GB ***Expected Output***: Analytical Data on Visual BACnet	**Date**: **Initials**: **Pass/Fail** **Comments:**
End - End System - Over Max size PCAP	The System shall not be able to take PCAP files over 1GB. ***Input***: PCAP File over 1GB ***Expected Output:*** Error	**Date**: **Initials**: **Pass/Fail** **Comments:**

Performance Measurements	The system should display all performance and timings through Grafana. ***Input***: Run System ***Expected Output:*** 1. Memory over time 2. Time to process & analyze a PCAP	**Date**: **Initials**: **Pass/Fail** **Comments:**

Table A.3.1 - System Tests

A.4 Acceptance Tests

The purpose of Acceptance Test is to evaluate the system's compliance with Optigo's requirements and assess whether the system is acceptable for delivery. The Test Suite has been determined and documented in Oakion Systems' Requirement Documents.[1]

Appendix B: User Interface Design

B.1 Introduction

Oakion Systems endeavours to provide a feasible and affordable solution to manage the inundation of data that companies like Optigo face. Our User Interface (UI) will be simple and straightforward, allowing users to easily integrate it into their system. In the Engineering Prototype (EP), we are building on top of the embedded system device that is already deployed on the client side, which is Optigo.

The Distributed Computing Network created by Oakion Systems will contain the following core UI components:

1. **Management**
   The management UI will be used for deployment, scaling, and controlling the amount of modules running on resources.

   a. **Deployment: Nomad**
      A deployment tool that allows users to manage the different modules in the Distributed Computing Network system, allowing the user to handle hundreds of devices in one unified headquarter.

2. **Resources**
   Resources is the physical hardware that is running Oakion Systems' Distributed Computing Network.

   a. **Capture Tool - Embedded Systems**
      Deploying modules on the Capture Tool allows us to take advantage of unused computational power from these embedded devices, while still keeping the initial specification of Optigo's capture tool design. Modules will be deployed to Capture Tools under Nomad's control.

   b. **Amazon Web Service (AWS) - Virtual Machine**
      In alternative to Capture Tool, we will leverage AWS for computation. AWS will handle heavy workloads that requires large amounts of CPU and memory, while the Capture Tool will focus on lower computation jobs that are less resource intensive. Modules will be deployed to AWS under Nomad's control.

3. **Application Performance Management (APM)**
   APM is the interface where Optigo' users can check the status of the Distributed Computing Network, such as the health of the system, how fast the system is running, whether there is enough resources allocated, if more machines are required, and errors.

   a. **Kibana**
      During data processing, the web GUI (Kibana), a log management tool, provides immediate updates on the different modules and captures the logs.

   b. **Grafana**
      The web GUI provides performance measurements to users such as timing analysis and the amount of data that has been processed.

4. **Visual BACnet**
   The web application provided by Optigo which presents BACnet analytical checks to clients. Oakion Systems' Distributed Computing Network will compute and provide the analytical data for Visual BACnet through APIs.

B.1.1 PURPOSE

This appendix aims to provide an overview of the main UI features of the Distributed Computing Network for Oakion System Inc. employees, stakeholders and potential clients.

B.1.2 SCOPE

The Proof of Concept (PoC) and Engineering Prototype (EP) iterations will be prioritized in this appendix, as the Distributed Computing Network is still in preliminary design and follows the agile development methodology.[24] Future iterations of this document will be updated and versioning will reflect this.

B.2 USER ANALYSIS

Oakion Systems' solutions are marketed toward companies that deal with the transfer of network data, whether through deployed infrastructure, the cloud or a combination of both. These companies seek optimization of their current systems and how to deal with their data, no matter their size (ie: Optigo Networks). Our primary intended users will have a software and networking background, knowledge of embedded design, communication protocol and system design and integration; conceivably a companies IT Systems or Development Operations team.

Given this target user's background, our UI design will be simple and straightforward, utilizing the interfaces that Grafana, Kibana and Nomad provide to visualize information pertaining to optimization and application monitoring of the companies data.

B.3 Technical Analysis

Oakion Systems' Distributed Computing Network UI design is inspired by Don Norman's "Seven Elements of UI Interaction", which are the following principles: discoverability, feedback, conceptual models, affordances, signifiers, mappings, and constraints.

B.3.1 Discoverability

Discoverability, the extent of ease with which a user can find all the UI elements the first time encountering with the product.[17] Oakion Systems aims to provide high discoverability by providing a clean interface with only essential UI elements. In other words, minimizing the amount of UI elements required.

Steps to start the Distributed Computing Network:

1. Allocate machine on AWS (see **Figure B.1** below)
2. Start DSMS (Kafka) service
3. Deploy modules on Nomad
4. Monitor system health and metrics from Grafana and Kibana

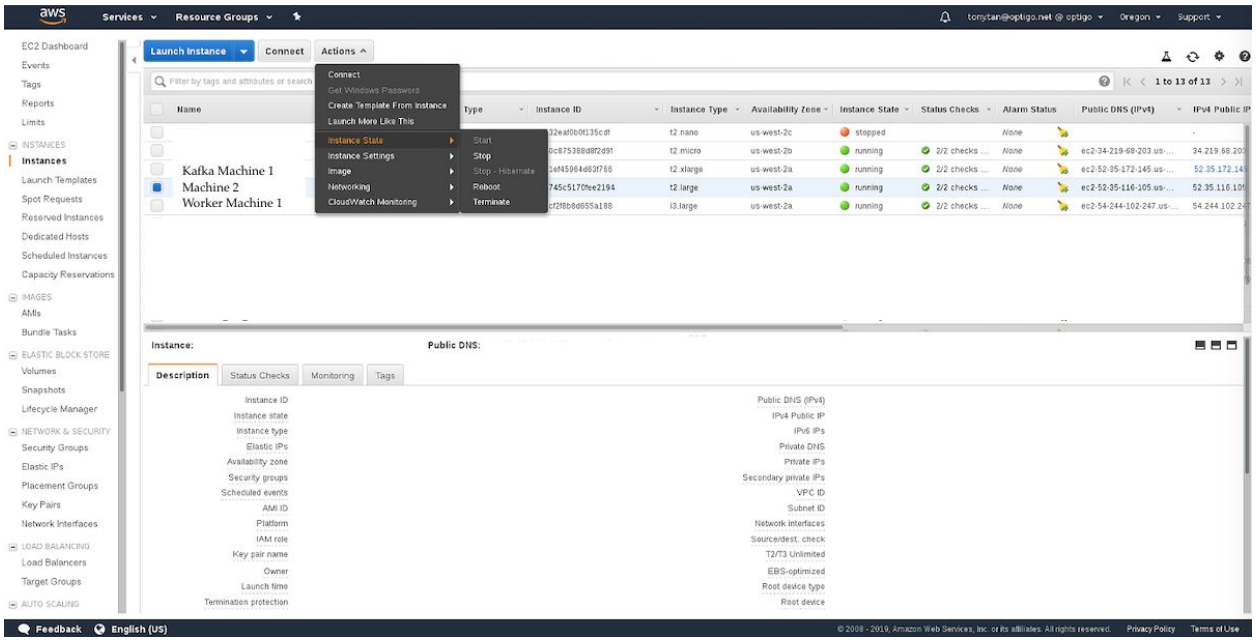


Figure B.1 - AWS Interface

B.3.2 FEEDBACK

Feedback, the information that is sent back to the user of what has been done. Our Distributed Computing Network will have a yellow LED light on the Capture Tool indicating that the Capture Tool is powered on and the system is connected to the network, as shown in **Figure B.2**.

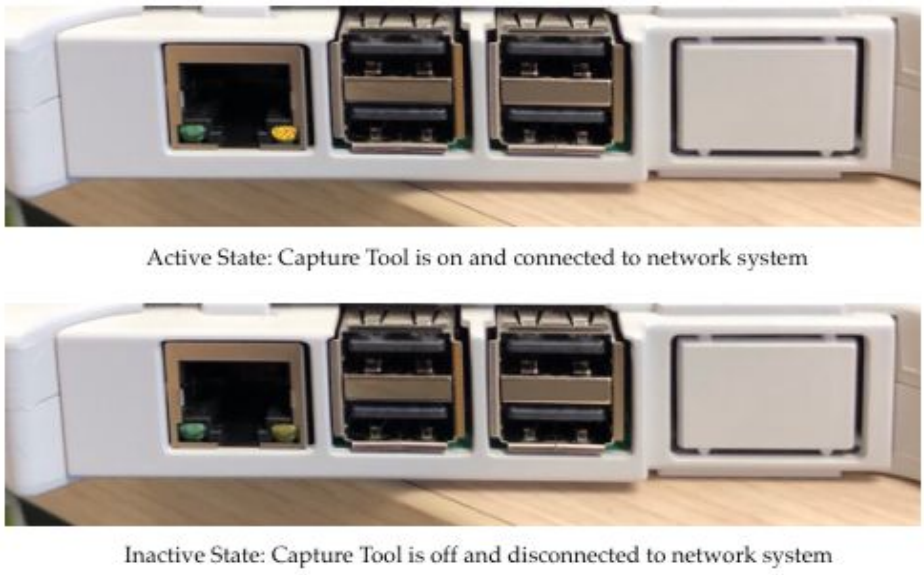


Figure B.2 - LED indicating active and inactive state of Capture Tool

After the analytics have been processed, timing performance and memory usage will be uploaded to Grafana providing immediate feedback for users as shown in **Figure B.3** below.

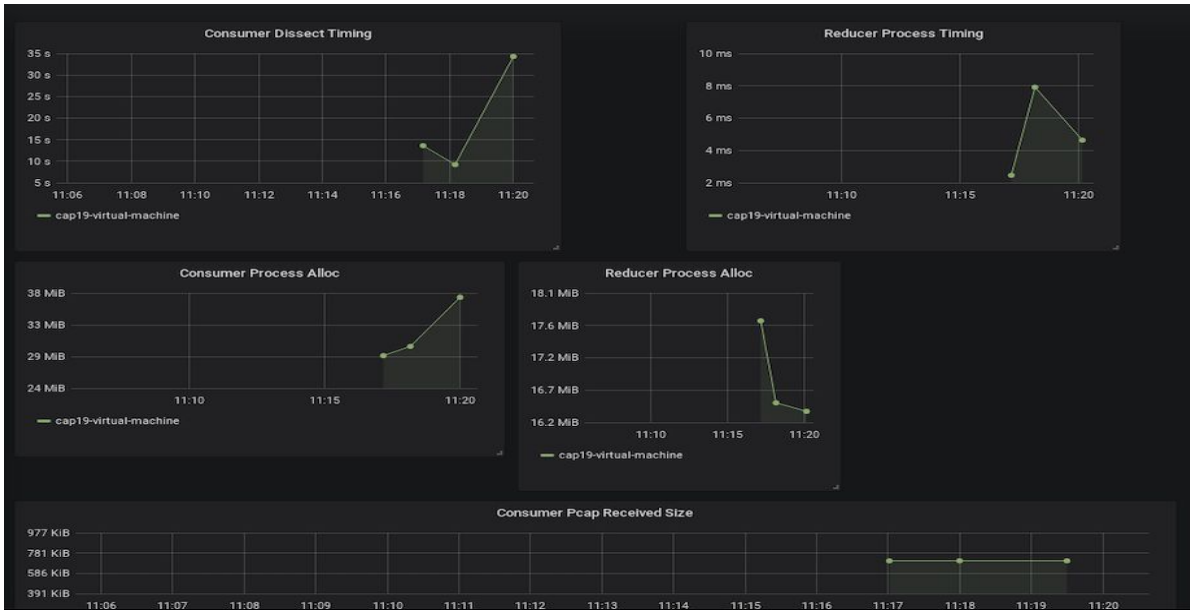
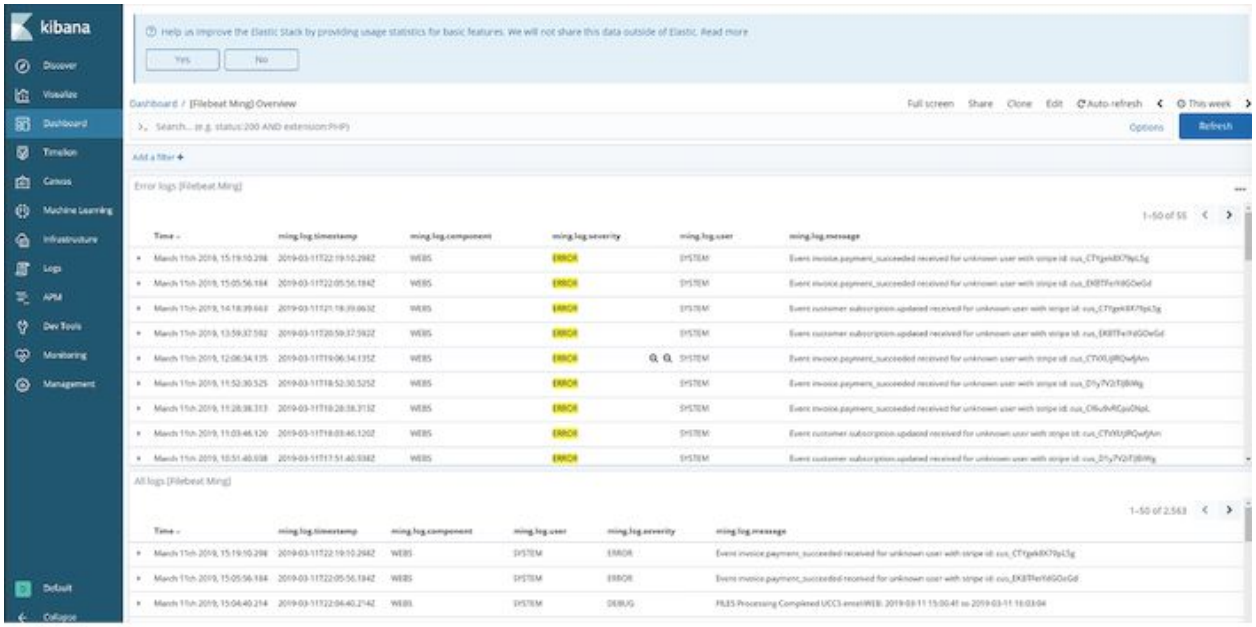
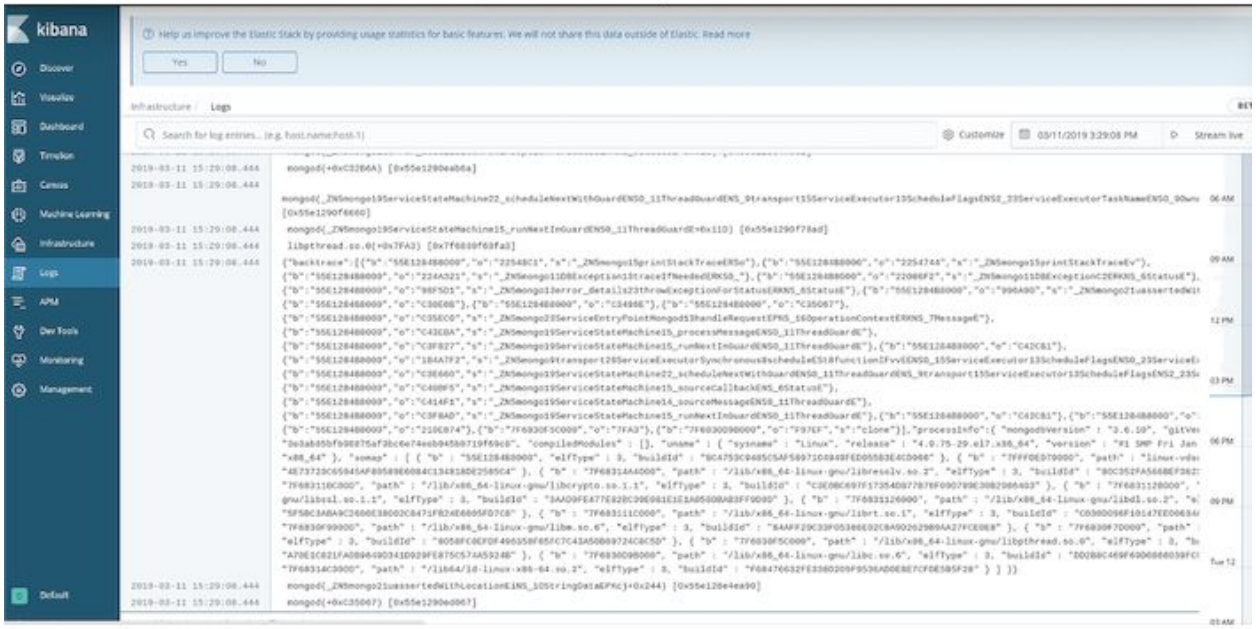


Figure B.3 - Grafana Web Interface

Kibana provides logs and the health of our Distributed Network System, as shown in **Figure B.4** below.



(a) List of error logs on Kibana



(b) Log on a specific error

Figure B.4 - Kibana Web Interface

B.3.3 CONCEPTUAL MODELS

Conceptual model, a representation of a system where the composition of concept is intuitive for stakeholders to understand the Distributed Computing Network.

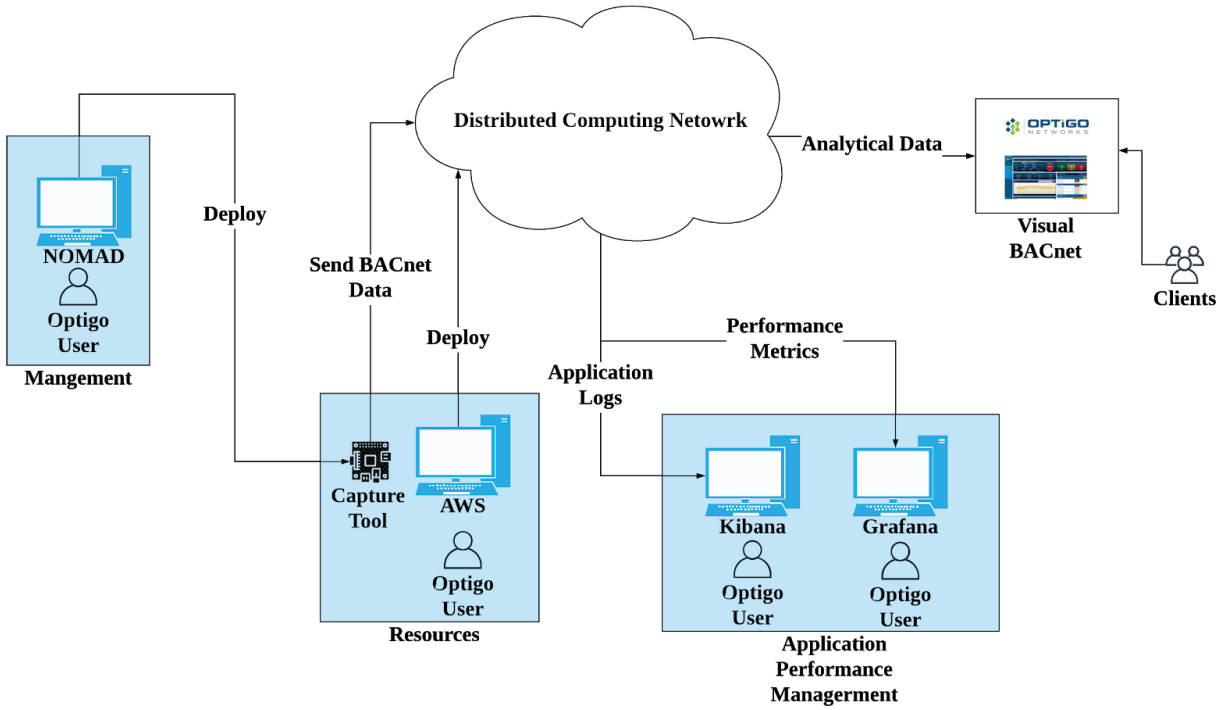


Figure B.5 - Conceptual Model of the Distributed Computing Network

B.3.4 AFFORDANCES

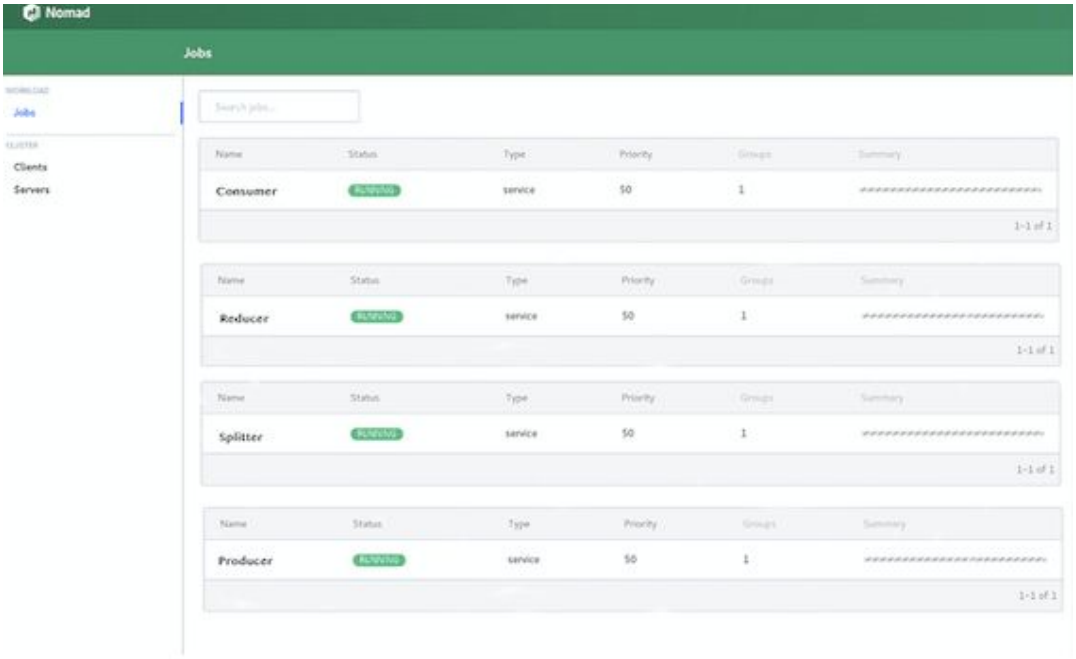
Affordances, attributes that allow users know how it's used. Oakion Systems focuses mainly on the perceived affordances with the following key attributes:

1. **Limited Interaction with Capture Tool**
   The whole application will be launched with one script, simplifying the setup process and affording users (network administrators) the ability to process data with one click.

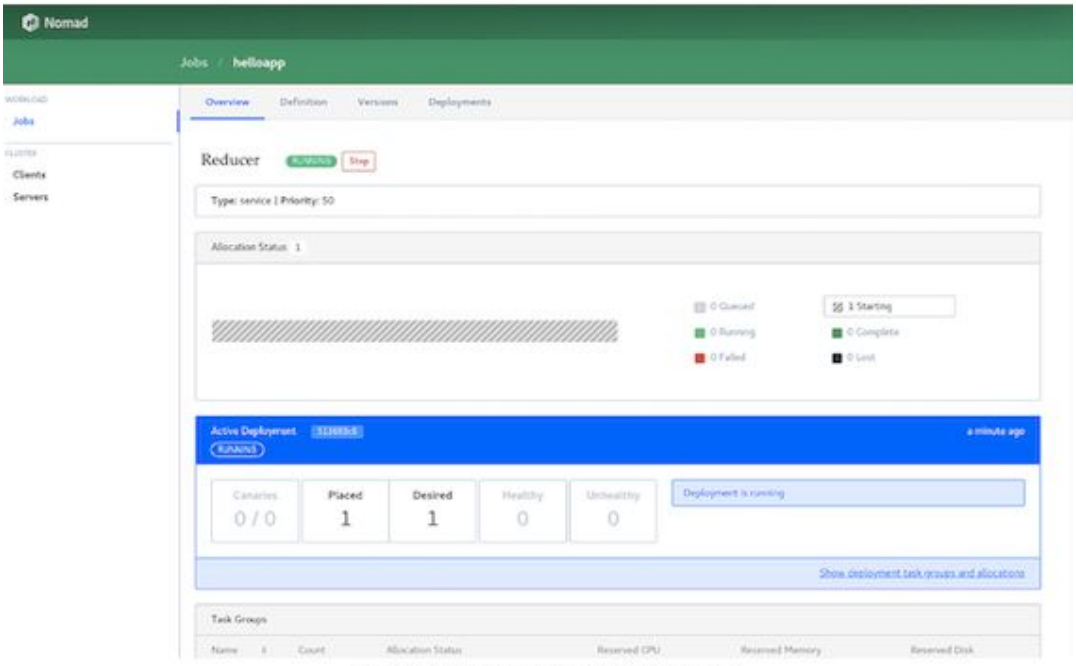
2. **Charts on Grafana**
   The timing performance and memory usage will be properly displayed and labelled, affording users to quickly identify the results.

3. **Modules on Nomad**

The modules are displayed separately on Nomad, affording the user to quickly locate and manage the different modules, see **Figure B.6** below.



(a) All modules displayed



(b) Management for one module

Figure B.6 - Nomad Web Interface

B.3.5 SIGNIFIERS

Signifiers, signs to indicate affordances to users. Integrated in Grafana are signifiers such as "Refresh every 5s" and "Save" (see **Figure B.7** below). Oakion Systems' services include backend integration that utilize the signifers seen on Grafana so that users can easily navigate the interface.

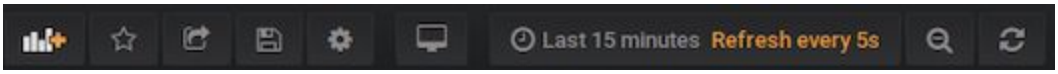


Figure B.7 - Signifers integrated in Grafana

In addition to the LED light (**Figure B.2**), Oakion Systems will label the operative features on the Capture Tool, such as the network and power slot.

B.3.6 MAPPINGS

Mapping, the relationship between controls and their results. The layouts for Nomad, Kibana, AWS and Grafana follow the same convention as other modern performance monitoring websites, making them user friendly and straightforward for users that are not familiar with them.

B.3.7 CONSTRAINTS

Constraints, restricting the possible actions that can be performed to minimize error. The Capture Tool does not contain any buttons besides the power switch, forcing the user to manage the modules on Nomad and AWS. Currently, Oakion Systems' Distributed Computing Network only supports BACnet protocol traffic, making the timing and performance analytics integrated on Grafana's backend solely supporting this protocol for now.

B.4 ENGINEERING STANDARDS

B.4.1 ISO 9241-161 ERGONOMICS OF HUMAN-SYSTEM INTERACTION -- PART 161: GUIDANCE ON VISUAL USER-INTERFACE ELEMENTS

This ISO gives a guideline on visual user-interface elements presented by software and provides requirements and recommendations on when and how to use them. It is intended for use by those planning and managing platform specific aspects of user interface screen design. It also

provides guidance for human factors/ergonomics and usability professionals involved in human-centred design. **Figure B.8** summarizes how Oakion Systems will follow ISO 9241

1. Interactive Properties - How a user interacts with User Interface
2. Informative Properties - What a user views from the User Interface
3. Decorative Properties - How aesthetically pleasing the User Interface is

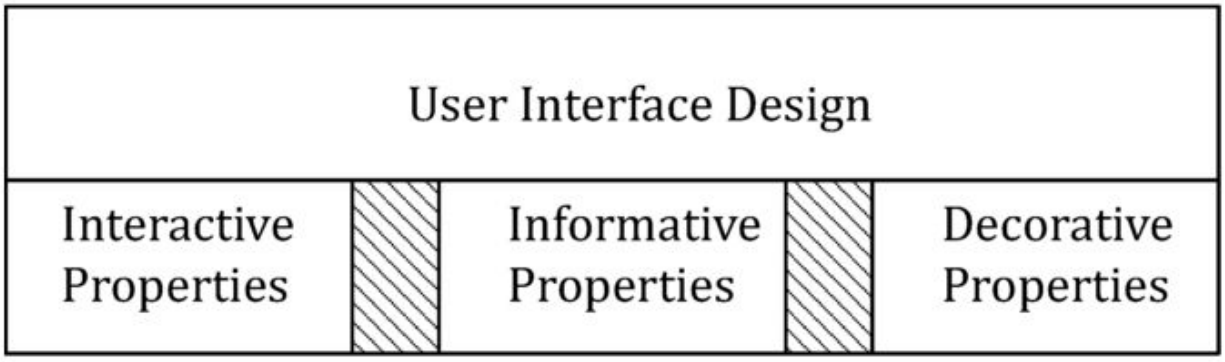


Figure B.8 - User Interface Design

B.4.2 ISO 14756 Measurement and Rating of Performance of Computer-Based Software Systems

This ISO defines how user oriented performance of computer-based software systems may be measured and rated. Software System is data processing system as it is seen by its users, e.g. by users at various terminals, or as it is seen by operational users and business users at the data processing center.

This ISO outlines the main aspects of user oriented performance terms and specifies a method of measuring and rating these performance values which is described as:

1. Execution Time
2. Throughput
3. Timeliness

Oakion Systems will follow as closely as possible with:

- Section 2 Principles of measurement and rating
- Section 3 Detailed procedure for measure and rating

B.5 Usability Testing

Performing usability test allows designers to check that clients can use the product and that they like it. The goal is to improve usability and diagnose real usability problems with real users. Oakion Systems will perform usability testing in two different ways: analytically and empirically.

B.5.1 ANALYTICAL

In the analytical testing stage, Oakion Systems will perform a series of tests to identify any overlooked errors or flaws that could be present in our UI. After all of the new issues have been discovered, reviewed, and documented, we will discuss potential solutions to improve our design and enhance the user experience.

Logs and auditing

1. Events of failures and errors on modules are clearly identified via Kibana
2. Audit logs are available for trace backs on system activities
3. Custom log queries can be made to form trendlines, identifying periodic events

Reboot of System

1. User can reboot the modules and services via Nomad
2. The system will be rebooted within 5 minutes
3. Kibana will identify a reboot event

Grafana

1. Timing performance is uploaded and displayed on Grafana within 60 seconds after the PCAP file has finished processing
2. Graphs are labelled appropriately
3. Graphs can be moved and resized according to user's preference
4. Additional graphs can be added in real time
5. Graphs will help identify resource consumption and bottlenecks

Resource Pool

1. All the modules that are deployed on the Capture Tool and AWS are displayed
2. When one of the modules go offline (i.e. user has powered off the Capture Tool), Nomad and AWS console can detect the error
3. Resources(Capture Tool, AWS) will report to Nomad, identifying themselves as available resources for deployment

B.5.2 Empirical

In the empirical testing stage, Oakion Systems will reach out to networking companies that deal with data processing as we aim for our product to be deployed to other potential companies besides Optigo Networks. These potential clients will be asked to perform a series of simple tasks regarding our Distributed Computing Network functionality.

Task 1
Power on the Capture Tool and connect the network cable

Questions
1. Was the switch visible and easy to locate?
2. Were you able to recognize the LED is turned on?
3. Was there sufficient feedback indicating that the Capture Tool was powered on?

Task 2
Managing the system through AWS console

Questions
1. Was the system reboot process intuitive?
2. Was the rebooting time acceptable?
3. Is allocating additional AWS resources intuitive?

Task 3
Readability of data generated on Grafana

Questions
1. Were the results displayed on monitor easy to read?
2. Were there sufficient information displayed?
3. Were the results labelled correctly?

Task 4
Error recovering through Kibana

Questions
1. Was it easy to locate the log containing the error?
2. Was the error message(s) clear and precise?
3. Was the time it took for error message(s) to appear acceptable?

Oakion Systems will make adjustments or modifications to our Distributed Computing Network after reviewing the feedback received to increase usability and satisfaction of our clients.

B.6. SUMMARY

First impressions are shown to be hard to change as they form a mental image in our heads[20]. The UI design of a product is very crucial as it gives a first impression to users and/or potential clients. Providing a design that is easy to use and learn will allow clientele to have an enjoyable experience with the product.

For Oakion Systems' Distributed Computing Network proof of concept iteration, the UI design will consist of the Capture Tool, AWS (**Figure B.1**), and Grafana. Users can deploy our system to Capture Tool and AWS, and visualize the performance measurements on Grafana as shown in **Figure B.3** after a PCAP file is uploaded and analyzed.

The Engineering Prototype of the Distributed Computing Network will incorporate Kibana, a log management tool, and Nomad into the system. Users can manage the modules that are deployed on the Capture Tool through Nomad (**Figure B.6**) and visualize the logs for each module as shown in **Figure B.4**.

Oakion Systems aims to achieve a minimalistic UI design for our Distributed Computing Network solution. Based on the feedback from our clientele, we will continue to make improvements to our UI design to provide an efficient and effective to use product.