Requirement Specification

Cloud Connect

12/08/2017

Project Sponsor: Tony Pallas

Faculty Mentor: Ana Paula Chaves Steinmacher

Team Members:

Parth Patel
Abraham Ramirez
Jacob Serafin
Steven Strickland

Accepted as baseline requirements for project:

Client Signature:________________________ Team Signature: ________________________

Date:__________________________________ Date:__________________________________

Table of Contents

Introduction

Within the hospitality industry, hotels are equipped with hardware on site to help produce an efficient and secure stay for customers. Devices on site proxy data to systems through the web where it can be used as a way to enhance the customer experience and keep track of payments due. Systems are also in place to organize reservation data to make sure customers get what they ask, and presumably pay for. Other such devices work in the same way, customized to each customer and keeps track of the impending bill that is left after a stay at a hotel.

Team CloudConnect has partnered with SkyTouch Technology and Choice Hotels to help provide a more modern solution to their current device proxy for on premises hotel technology. Choice Hotels is a chain of hotels with up to 6,500 hotels(globally) deployed globally. They are a billion dollar industry that operates, in aggregate, approximately 15.5 million available rooms each night. Each hotel is set up with a wide variety of sophisticated electronic systems. Some of these provide the security, conveniences, and luxuries for today's travelers. SkyTouch Technology is a Software as a Service (SaaS) company that provides an operating system for which they can move data from these electronic systems to and from each hotel.

The goal of our team is to improve the capability of SkyTouch's solution to access and manage the many varied hardware subsystems installed in today's modern hotels. Many of these hotels rely on standard RS232 serial port connections to send and receive information from these systems on site at hotels. Our team will create a solution to proxy serial and TCP/IP based protocols to Amazon Web Services(AWS). Our proxy application will replace custom software that is currently deployed on premises at each hotel and will need to support communication with thousands of devices housed at thousands of hotels.

Figure 1 is a modeled representation of SkyTouch's current solution to proxy devices from any given hotel to and from their own database located within their AWS. Within the "on-site" section to the left represents any given hotel. The devices run through their own systems and, currently, send serial data to a PC at the hotel that runs their custom made software: Java Enabled Device Interface (JEDI). From here, JEDI then runs a proxy from the hotel to a cloud based web space, this is provided by SkyTouch's Amazon Web Services account. All the data is translated and then sent back accordingly.

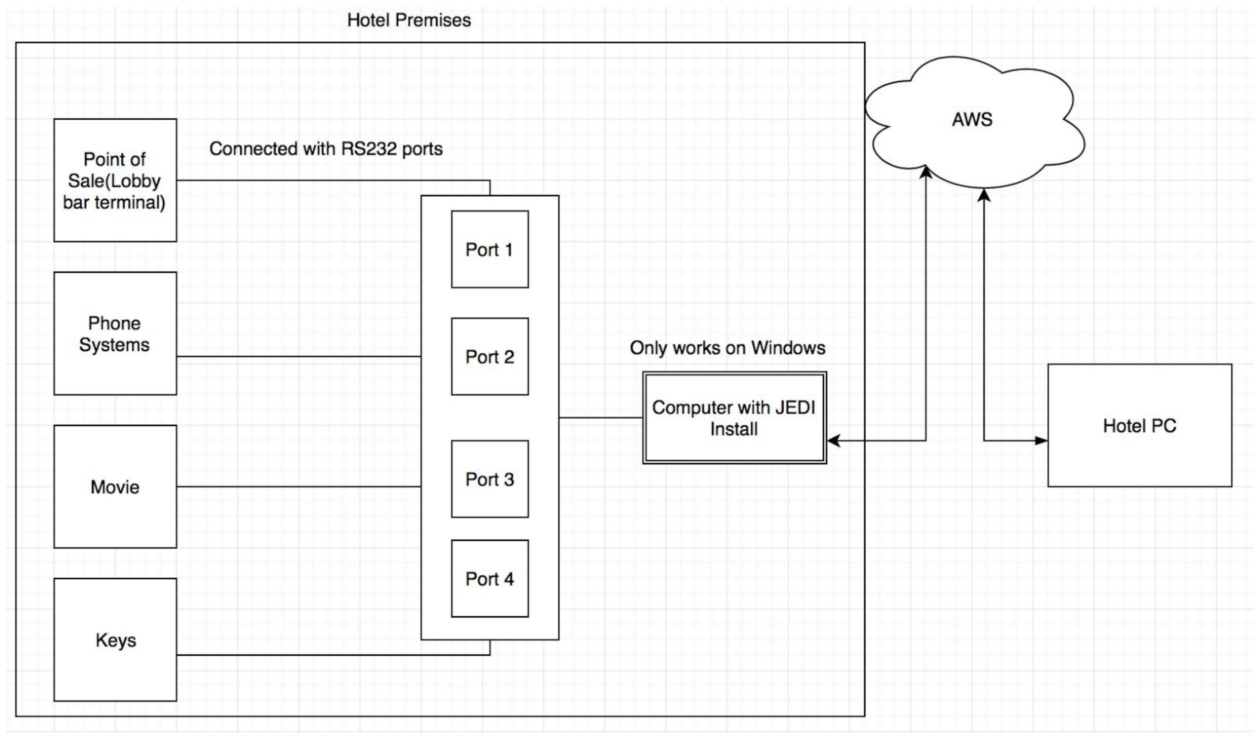

Figure 1

Overall this structure has been kept intact since 1998 when JEDI was first implemented into Choice Hotels' locations. In turn this means that all the devices being proxied are aged as well. To keep this structure at any given hotel it costs money to maintain and develop as time continues. SkyTouch has begun to move their systems into a cloud-based solution, but have yet to resolve the issue of how to produce a cloud based system which would not need JEDI at each location.

Our proposed solution is to eliminate the need for JEDI and install a terminal server that can communicate serial data through IP/TCP protocols to their database within AWS. A web application implemented within AWS would replace JEDI so that the only cost to install within a given hotel would be the account to access the virtual servers within AWS.

Problem Statement

Before we introduce the problem, we would like to explain how the current software works. SkyTouch offers different interface and Hotel OS. Point of sale(POS) is one of them. Point of sale is terminal that is used at lobby or a restaurant. For example, John Smith is staying at Quality Inn Flagstaff. He is in room 101. Now, Mr. Smith went to the lobby bar for a drink. Let's say that Mr. Smith likes to put his check on this room tab. Here is what happens behind the scene(follow along the figure 1): First, POS must be connected to a computer which has JEDI(Java Enabled Device Interface) installed via Legendary RS-232. POS will ask JEDI to look for Quality Inn Flagstaff. Once JEDI finds the hotel, it authenticates with AWS to get access to

Quality Inn Flagstaff Hotel OS to see if Mr. Smith is in 101 and has enough money to charge on the credit card he provided. Hotel OS communicates back to AWS to tell Quality Inn Flagstaff JEDI to tell POS whether it is okay to charge or not. This is how most of the interface work.

SkyTouch has to install and maintain the custom software, JEDI, for every hotel which takes a lot of time and labor. As mentioned in figure 1, a Four Port Pin box is very hard to configure because certain ports can be only connected with certain devices. This creates a problem because Hotel personnel are not usually aware of that. One of the other problems is that if JEDI goes down, all of the devices go down with it. Another major problem is that JEDI is very old, which is limiting SkyTouch from collaborating with newer hotel technologies.

Solution Vision

We will select a hardware device which will be connected to the devices. This hardware device then will connect to a router that can stream data into the AWS cloud based virtual server, where we will have a web application hosted that can communicate with a given hotel operating system (Figure 2).

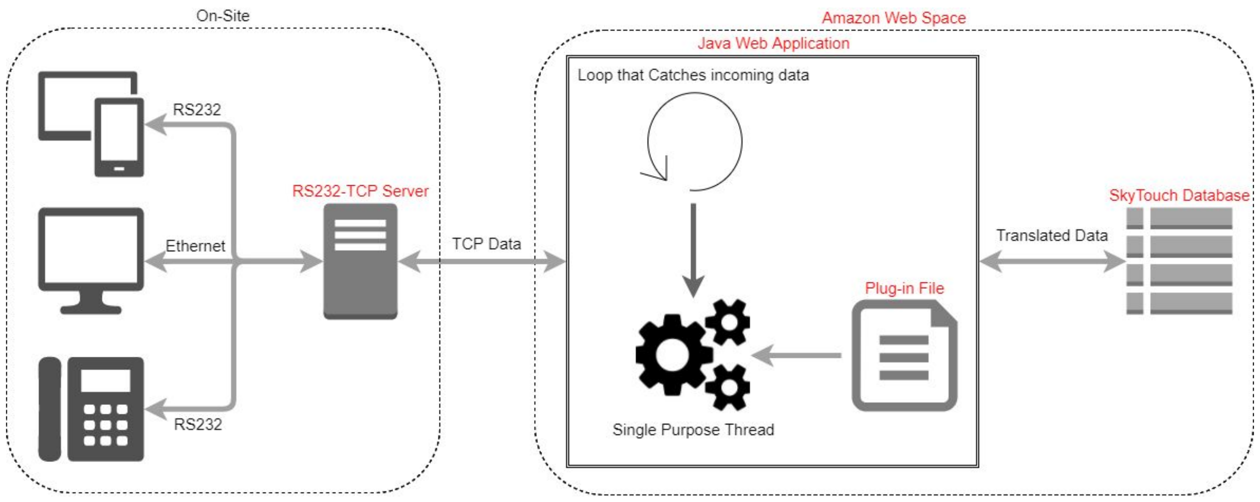

Figure 2

Key Features:

- TCP protocol to communicate with AWS
- Java Web Application in AWS that will communicate 2-ways with on premises devices
- Hardware devices will be able to support other third party interfaces
- Eliminating the need for JEDI configuration within each hotel

The selected hardware device will have to be configured to be able to stream data to the correct IP address and port number to allow it to connect to the AWS virtual server. Once a user pushes a transaction, this information gets sent as a log that will be sent from the hardware device to a router that streams the data in the form of TCP to our web application. This log will go to SkyTouch to be handled accordingly. There will need to be a confirmation, which is why our solution will work backwards, sending data through TCP from SkyTouch back to the system that originally sent the transaction.

Our system will change the way SkyTouch Currently works in back end. There is not going to be much change on front-end current system. There might be an extra login authentication that front end user will have to go through because we will have to include an extra layer of security for our Java application since it will be in cloud. With this implementation, maintenance can be easier and any updates to this application can be made at SkyTouch headquarters.

In short, our solution plan is to replace JEDI. This means SkyTouch will not have to customize hotel configuration that will save a lot of time because JEDI is complicated and it takes time to build that customization for each hotel. It will be replaced by a hardware device that will connect to cloud directly. Our solution can potentially bring more partners with newer technology for hotels with very outdated systems.

Project Requirements

The following sections describe three different forms of requirements that our solution will satisfy. The first set are Functional Requirements; requirements that will be able to be empirically tested to ensure satisfaction of our client. The second set are Performance Requirements; requirements that dictate how our solution must solve the client's problem. The last set described are Environmental Requirements; requirements that are given to us by real life practicalities. All together, these requirements describe in detail the functionality that our team must satisfy in our solution.

For the following sections detailing the various requirements, TCP Data refers to data wrapped in TCP formating, which means the original data is contained inside of formating data following TCP(Transmission Control Protocol) standards. While the original data itself can be almost anything in any format due to the nature of 3rd party software and hardware, the formatting of TCP data is standardized and is handled using a Hardware RS232-TCP Server at each hotel. TCP Data formatting is huge and very complex, however, modern programming languages do all the work, all that is required is the original data to send and the destination IP address. Below is an example of one of the many different forms of the original 3rd party data before being converted into TCP data and then being sent to our application:

010000000000000000 2 314 0 429 429 3 330123012 0 26 2.12 2.00 0.00 0.00 0.00 0.00 0.00 0.00 0.12 0.00 0.00 0.00 0.00 2138

The original data string contains the room number, device code, and several amounts in the record, the first of which is the total of all the charges (food, beverage, tax, etc). In this example the room number is 314, the total charge is $2.12, and the device code is 3.

Once the TCP data is received by our application, it will need to be translated back into the original data, then used along with Skytouch API to update or adject data into SkyTouch's Database. Any data that then is returned by Skytouch Database will then be converted back into the original format and sent back to the original device.

Functional Requirements

Our solution will be able to do the following:

FR1: Java Application

FR1.1: Communication

- Will be able to accept data through TCP.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.
- Will be able to send data through TCP.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.

FR1.2: Translation

- Will be able to translate incoming TCP data into Skytouch Database commands using Plugin Architecture in **FR1.3**.
    - Skytouch Database commands are specific commands that interact with the Skytouch Database and are specified in the Skytouch Database's API.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.
- Will be able to translate incoming TCP data from the Skytouch Database to TCP data for the hotel's device specified in that incoming TCP data using Plugin Architecture in FR1.3.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.

FR1.4: Plugin Architecture

- Will be able to determine what type data is contained inside incoming TCP data.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.
- Will be able to use Plugin files to translate TCP data in FR1.2.
    - TCP data is data formatted according to TCP(Transmission Control Protocol) standards.
    - Plugin files are xml files that can created by SkyTouch's employees.
    - Each Plugin file describes how to translate a single 3rd party device protocol into Skytouch Database commands. Also, they describe how to translate from TCP data from Skytouch Database into the original 3rd party device protocol.

With the implementation of the processes detailed in **FR1**, our product will be able to automate these requirements for several cases that exist within the systems of a given hotel. Such cases include, but are not limited to:

1. Point of Sale (POS): data from an on-site device that posts food and beverage charges
2. Telephone Activation (PBX): data pertaining to telephone status, whether the room has the phone line on or not, and guest information to show during a call to a front desk.

Performance Requirements

Our solution will be able to do the following:

PR1: Java Application

PR1.1: Will be able to handle at least four concurrent sets of incoming TCP data using multithreading.

- TCP data is data formatted according to TCP(Transmission Control Protocol) standards.
- Multithreading refers to the program creating a new “thread” (process) that is separate of the actual application that will then handle the incoming TCP data. Once that data is handled, it will then close, allowing a new thread to be created for different sets of TCP data.

PR2: Modification

PR2.1: Will not need to be re-created to support additional third-party device protocols.

- Re-created refers to the process of re-designing and re-programming a new solution.

Environmental Requirements

Our solution is reliant on the following:

ER1: Hardware Device

ER1.1: Our solution requires a Hardware RS232-TCP Server at each hotel that will send each device at the hotel’s data to our solution.

ER1.2: Our solution requires the previously described RS232-TCP Server to have a working internet connection to our application.

ER2: Skytouch Database

ER2.1: Our solution requires a working internet connection between our application and the Skytouch Database.

ER2.2: Our solution requires the Skytouch Database to give valid (according to the SKytouch API) TCP data to our application.

ER3: Plugin Files

ER3.1: Our solution requires a valid Plugin File to be able to be used by our application for the aforementioned functional requirements..

- This Plugin File must be specific for the type of data needing to be translated.

Potential Risks

SkyTouch deals with thousands of transactions by the hour, all across the globe. If one of their systems would stop suddenly working it could throw a domino effect and affect a lot of other systems and this is not an option they would be happy with. Our project definitely has risks involved and some could affect the customer and some the company.

There are a few risks that could happen with this project. One of the risks involved with this project would be getting a piece of data not being transmitted by the TCP device itself. That being a money request, room request or even someone's information. We have to ensure when we code our application to report and store it in the cloud correctly by testing the application before we even implement it and ensure all TCP devices work accordingly. The risk of getting a malfunctioning device could cause some serious damage. Perhaps a big company that booked a conference room for a certain date is coming during the weekend and the information typed in the front desk using our device malfunctions, then this could cause some serious unexpected issues. The chances of this happening are hopefully very slim. We'll be coding this application to store a lot of device requests into the cloud for storage. It is up to us to test the device thoroughly and carefully to ensure every data transmission goes up smoothly and no errors occur. Testing this will take time but it's vital for the company to have their information stored correctly.

Another risk that could happen in this project is a device hardware malfunction. The risk involved here is the device could be configured incorrectly, meaning all those transactions being sent will not be logged and in turn can lead to services being unpaid. Technology fails now and then so this is a potential risk we might not be able to avoid. However, this is why it is imperative that SkyTouch and we select a device that has all the capabilities needed and can be configured easily such that the risk of an incorrect IP address does not occur

Potential risks are always something to keep in mind in case things go wrong. There is however a way to plan these risks by having a back-up plan. As I said in our risk report we will have to test things thoroughly and efficiently to ensure transmissions are fast and accurate. We'll also ensure the hardware installed is functional and reliable to ensure our client is happy and customers get settled in quickly. A lot of testing will be required and do our best to minimize these potential risks.

Project Plan

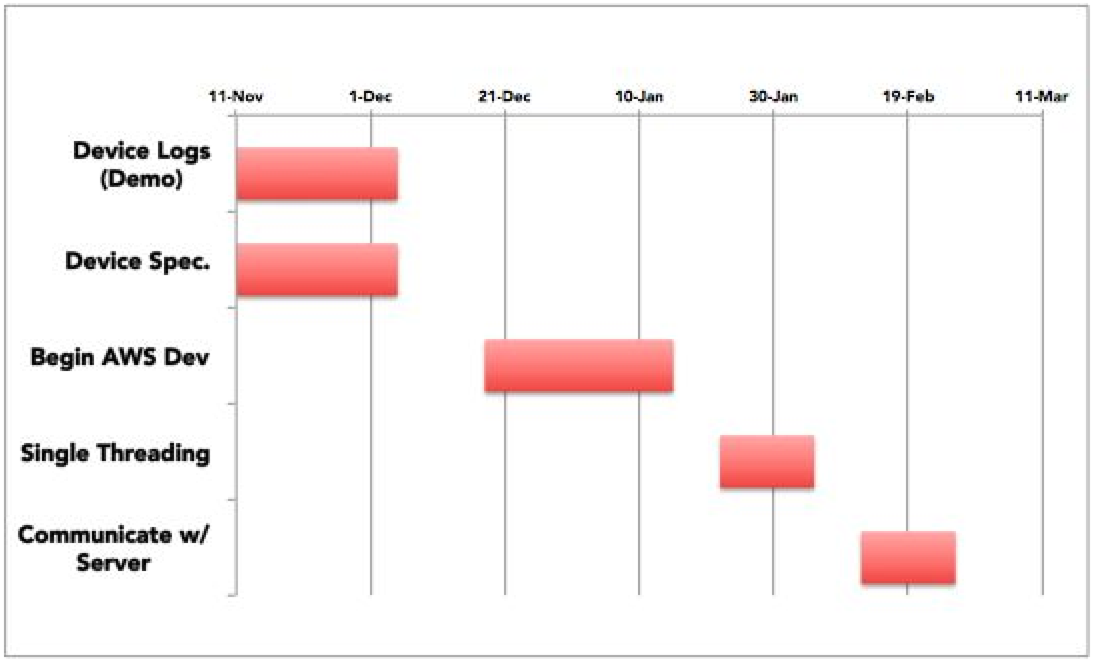

Figure:3

Our execution plan as shown in Figure 3 is as follows. We will begin by transmitting data through TCP from one host to another to ensure data transfer works correctly and is accurately stored to have a better understanding how to correctly save logs from a hotel. This will take place between end of November month and first week of December. The device specifications is mostly complete as we know what we will be using, this is just a matter of getting the materials in order to get a prototype hardware device. During December it is important for us to get better acquainted with AWS, starting with a basic application that can accept information and store it.

By the time the next semester starts, AWS training shouldn't be a problem. Starting with single threading, we want our web application to be able to take in a string with a transaction log and can properly send this to SkyTouch to be saved. We will be testing this with dummy information just to ensure everything is being accordingly set to their destination before sending the data to the cloud. Our final steps with the project will take place in February, where we implement multi-threading so speed can be optimized.

Finally, to complete our project we want to test it live with the actual terminal server chosen. This way we can ensure once we connect to SkyTouch's database that we are ready to transmit transaction logs accurately. We plan to leave some time before the final deadline so we can adjust any errors and clean up our code.

Conclusion

In conclusion, we are creating a solution to proxy serial and TCP/IP based protocols to the AWS which will eliminate the need for a custom application for each hotel. This will be a hardware device installed from a box that will be implemented at each hotel. We will have to thoroughly test each application that we will be creating starting with device logs using dummy data. From Device logs we will move into AWS development and how we will be transferring data into the cloud. After AWS development we will begin multi-threading implementation that will receive the requests from clients and sort them by their appropriate category such a POS (point of sale) purchase, hotel check in or a long distance call. After all is said and done our final testing implementation will begin which will be communicating all of these applications with the terminal server and make sure they are fully functional. All these tests are split up into dates which was shown in figure 3 which we have to strictly stay with in order to meet the deadlines. With our project progressing we are positive we are improving our customer needs by making their communications quicker and being able to support more devices to communicate via TCP.