Technological Feasibility Analysis

Cloud Connect

11/09/2017
Project Sponsor: Tony Pallas
Faculty Mentor: Ana Paula Chaves Steinmacher

Team Members:

Parth Patel
Abraham Ramirez
Jacob Serafin
Steven Strickland

Table of Contents

1 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 700,000 hotels deployed globally. They are a billion dollar industry that operate, 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 the cloud (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 Amazon Web Space. 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.

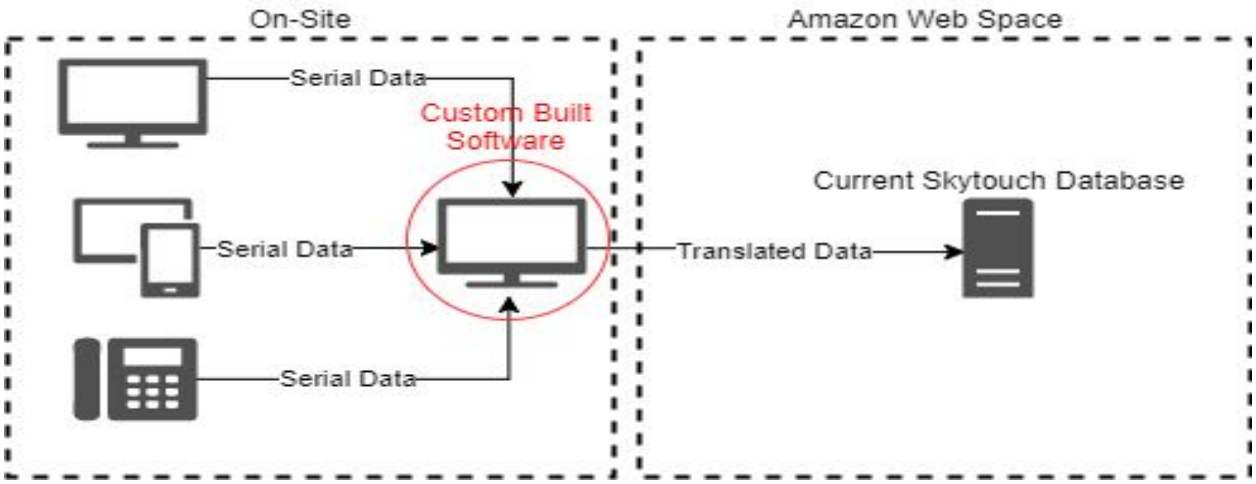

Figure 1

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.

1.1 Document Introduction

Our team will present upcoming challenges and how we plan to develop and overcome them. The challenges posed include handling a large amount of data, choosing the right device that can communicate with serial devices and convert to IP/TCP protocols, developing in a language that will give us all capabilities needed to seamlessly transition into SkyTouch's current systems, and creating an application with the correct functionality to seamlessly fit with SkyTouch's systems.

We will further describe our challenges within section 2, following up in section 3 with a solution to each of these subproblems. In section 3 we describe how each of our challenges can be solved, and what technologies exist currently that we can use develop a final solution. A listing of the benefits and hindrances of each of these available tools will help in configuring together our project.

2 Technological Challenges

2.1 Introduction

There are several major technological challenges that our team will have to overcome in order for us to solve our client's problem. These challenges include handling a huge amount of network data, picking the right programming language, picking a device to send the data through the internet, and implementing plugin architecture. This section introduces each of these challenges briefly.

2.2 Challenges

One of the biggest challenges on the four our application is the sheer volume of data that is needed to be collected and translated. Because Skytouch is an international business, they have hotels in multiple countries. Thousands of hotels each with multiple rooms, each room containing multiple devices that needs to send data to Sky Touch's online database sometimes all at once. The sheer volume of data that will need to pass through the system will prevent simple solutions. We need a way to handle the large amount of data quickly.

RS232 needs to be converted into TCP/IP in order to replace the RS232 devices. Thus we have to decide on what type of device we will be using and what approach is best for this project. It is up to us to decide what will be the best device to use and do various tests with them and see what fits best with the company and what works best. We have to keep in mind that there is a budget cost as well and have to stick with not so expensive devices that will work with their computers.

Another problem is choosing the right programming language. SkyTouch uses many different devices and technologies, all of which implemented using the different programming languages. We have to keep this in mind while we pick the language. We want to pick the right language that will work for most, if not all, of these devices and technologies.

We have to also keep in mind on how we will be processing data. Another challenge here is how we decide to organize the serial data over TCP/IP protocols. As a team, we must create a solution that replaces custom built software from a PC that interacts with the serial data. Ultimately we'll most likely be using AWS (Amazon web services) but we will keep an open mind and explore other cloud-server based solutions available which we will talk about in Section 3.4.

3 Technology Analysis

The following section is devoted to delivering deeper into each challenge and proposing several possible solutions to each problem related with our RS232 project. Ultimately we will decide on one solution and propose a way to proof that is satisfies the original challenge.

3.1 Volume of Data Challenge

The sheer volume of data that will flow through our system requires technology to deal with it. This data will be in the form of commands which can be payments, check ins, or other requests for services. The required solution will need to be able to ensure each and every bit of data will be handled quickly, which is a key requirement from our customer. As part of our project, we will be creating an application that collects all this data before translating it into meaningful commands and sending those commands to the current database.

Alternatives

Two different solutions can accomplish this. The first potential solution is our application using a loop and list structure to handle each bit of data in the order they came in, putting new data in a list to handle later. The second potential solution would be our application using a multi-threaded loop structure. This solution would have a loop create a new thread for each new bit of incoming data. This thread would handle that data properly then be closed when finished. These two alternatives are described in Table 1.

	Pros	Cons
Single-Threading	Simple, Easy to maintain	Would not be able to handle the data quickly
Multi-Threading	Can quickly handle all data	Complicated, requires higher-level knowledge to maintain and understand

Table 1

Chosen Approach

While the first potential solution is simpler, allowing for the application to be more easily maintained, the second potential solution allows for more timely handling of data which is a key requirement of the system. Using a multi-threaded application will ensure that every bit of data will be handled as soon as possible, effectively ensuring that paying customers trying to use these services get what they want quickly. While having our system be easier maintained would be nice, Skytouch have an educated team of engineers who are more than capable of

maintaining complex solutions such as multi-threading. The ability to handle all the data quickly is more important due to its direct relationship with our customer's customer.

Proofing Feasibility

Because the expected result of this solution is the ability to quickly handle large amounts of data, to test its feasibility we will create a small application that mirrors the structure described, then create a second small application that will send a bunch of data into the first application. We can then see how it handles this data to proof that it can handle it quickly.

3.2 Picking a Hardware Device

RS232 serial communication protocols have been around for many years and are very well established; in fact almost all computers have at least one RS232 port installed. The issue with RS232 communication devices is that they require separate cable connections for every device that you want to communicate with. If you want multiple devices connected to a single computer, you must install multiple RS232 ports in the PC in order to connect them. It is our job to solve this issue and choose a device that will allow multiple connections through a single cable.

Alternatives

We have various alternatives that we could implement to convert RS232 proxy protocol to TCP/IP network in order to minimize cables. We could use a hardware called Serial Device Server or a TCP-Com software to expose the serial ports on a PC to a TCP/IP network. Serial Device server work by having a RS232 serial port on one side and an Ethernet connector on the other, which basically establishes a network connection and then feeds any data that it receives through the RS232 port out over the network connection through a TCP/IP port and vice versa. The TCP-Software allows you to expose the serial ports on that PC to the networks, basically it does the same job as the hardware based serial device except that it is a software that can be run on a PC. While these alternatives work just fine they can be costly. A third approach we are considering is creating our own Hardware with the program installed to communicate RS232 devices with TCP/IP and avoid dealing with licensing issues and directly give the hardware to the hotels so they have direct access to the program and TCP ports. This hardware box will incl All three alternatives have their cons and pros which are mentioned on Table 2 below which we strongly considered before choosing an option.

	Pros	Cons
Serial Device Server	● Easy to use ● Converts RS232 easily ● Allows multiple devices to be used	● Can get bulky ● Costly
TCP-Com Software	● Can be run on PC ● Minimal Cables used	● Licensing issues can get costly
Create Hardware	● Hotels can directly connect it to their devices ● Secure	● Building the Hardware ● If there's an update need to locally retrieve hardware and update

Table 2

Chosen Approach

There are multiple approaches for our solution but we have decided to go with creating our own hardware. By creating the hardware with a pre-installed program in it, we avoid going through firewall issues allowing the program to communicate directly to the hotel PC and directly to the AWS cloud. Installing the devices will be simpler since we can directly implement them inside the hardware box as well as directly connect it to the hotel PC. The hardware device will have the new TCP/IP device and program inside. This will replace RS232 devices which will allow multiple devices to communicate directly aswell and have faster speeds with easier communication for the hotels.

Proofing Feasibility

Once we create our hardware device we will have to run some tests to ensure it works efficiently. To do this we will set up a connection with our device and a PC and create a simulator that we will be receiving from our client. This will allow us to do dummy tests data and see if the device is communicating correctly and converting the data into TCP/IP and uploading to the cloud. We will be testing various devices and data transfers to ensure all the data is successfully being converted to TCP/IP and being stored accordingly.

3.3 Picking a Programming Language

Our client provides services to about 700,000 different property. They used several different interface and devices within each hotel. Different devices has different requirements as

well as different configuration. There are some devices which are also very outdated which only transfer data over RS-232 ports. We would have to pick a language that will be able to transfer data from this outdated devices.

Alternatives

Since we are trying to feed information to our application which is going to act as a middleman for data to be transformed over to Amazon Web Services, we could use different programming languages utilize socket programming. As described in Table 3, some of our options are choosing C++, Python or Java. After doing some research on C++, we found out that C++ can get very tedious to work with especially when it comes to socket programming. C++ has some capability to work with byte arrays, but Python and Java has more functionality with byte arrays. As mentioned in Table 3, that Python and/or Java would be our best option for this projects because it will save us a lot of time.

Programming Language	Pros	Cons
Java	● Works with AWS ● Flexible with socket programing ● Easier with Object Oriented Programing	● Debugging Enterprise level Java code
C++	● Works with Object orientated programming ● Is very powerful, and can be used to create just about any program, including low-level system programs	● Difficult language, often distracted by peculiarity of C++ language instead of solving the real issue ● Lacks the ability to define completely custom operators. ● Does not work with AWS
Python	● Python is easy to learn for even a novice developer. ● Supports multiple systems and platforms	● Has limitations with database access ● Does not work with AWS

Table 3

Chosen Approach

While both Python and Java work as solutions, we are going to be using Java. All of the team members are more familiar and more experienced with Java than Python. Since our client is very familiar with technology, we were told that majority of their devices are implemented with Java. While there might be some third party interface or device that company uses that is not Java based, for most part, they are all java based.

Another reason we chose Java is because we are going to be feeding information to Amazon Web Services. AWS heavily relies on Java. While AWS could work with Python, our research indicates that Python can cause glitches occasionally on AWS.

Proofing Feasibility

We will create dummy data to simulate the various devices which we will run through our software application which will be coded in Java. Our software device will detect this dummy data that we created and it will transfer over to the AWS which is heavily based in Java. Once we succeeded with the dummy data we will be choosing one of client device to do the same test.

3.4 Plugin Architecture

The challenge posed here will link our given solution to handle Choice Hotels' data and how we decide to organize the serial data over TCP/IP protocols. Respectively, it does not matter how we decide to do the latter since we would still need a way to implement Java in a way that allows our transport protocols to be received by SkyTouch's Amazon Web Service (AWS) cloud storage. Our team must create a solution that replaces custom built software from a PC that interacts with the serial data.

Alternatives

There are many other cloud computing services, but for this scope AWS is a given solution requirement since SkyTouch has their Java environment setup with AWS services through their paid account. From a networking standpoint, our alternatives come out to three major forms of communication: Port Forwarding, Cloud Connection using IO Software, Mobile Serial Adapter. Each alternative has some software development to implement into our previously chosen devices and are under the assumption that we have a given "device" chosen by our client that best suites their cost benefit needs. The pros and cons of each form of plug-in are as follow:

Chosen Approach

Our approach here is to stick with the most efficient cost-benefit solution that tailors to our clients needs. Creating a custom IO software that can take in serial data and send this data

over the internet using transport protocols is the most efficient use of our clients resources. It is the most secure, and in this scope, security is a big factor, considering Choice Hotels manages 700,000 hotels daily. The best solution we can produce would involve efficiency and reduce the amount of costs needed. With a custom built Web Application that utilizes the cloud based server given by SkyTouch's AWS account, we can proxy serial data through TCP/IP protocols and reduce the need for storage at each on site hotel.

Proofing Feasibility

With further development, we will want to see the active "JEDI" application in use by our client, SkyTouch, and emulate our Web App to configure to their protocols. The biggest area of development would be in socket programming. We would need to configure the given serial data and create our Web App not only to receive, but to send this data over transport protocols. We will also take a look at the AWS "Device Proxy" documentation as this may give insight on how SkyTouch will be storing and using the data within their cloud based server.

	Pros	Cons
Port Forwarding	● Easy way to send communications from terminal server to cloud using transport protocols on Internet ● Would use cheapest form of serial device converter ● Software would only have to focus on TCP/IP protocols forwarding from location to AWS storage.	● Not secure ● Need to configure each hotel location and have access to router ● Cost ● Would require on site implementation
IO Software	● Most Secure ● Would make use of a “multithreaded” application ● Can give AWS access direct from Hotel ● Secure connection to AWS from Hotel ● Build upon already set Java environment at SkyTouch: J.E.D.I.	● Still needs a “controller” ● Would need to decide if to create on PC or through different device ● Recycling of SkyTouch’s current solution.
Mobile Adaptor	● Can be operated by any employee on site ● Inexpensive way to expand current Java environment ● Application would be available for any new location	● Still have to spend on a mobile device and adaptor for each given hotel ● Cost dependent on device choice ● Security relied upon quality of application ● Range; may need multiple mobile devices to support bigger hotels with more on site serial devices

Table 4

4 Technological Integration

4.1 System Architecture

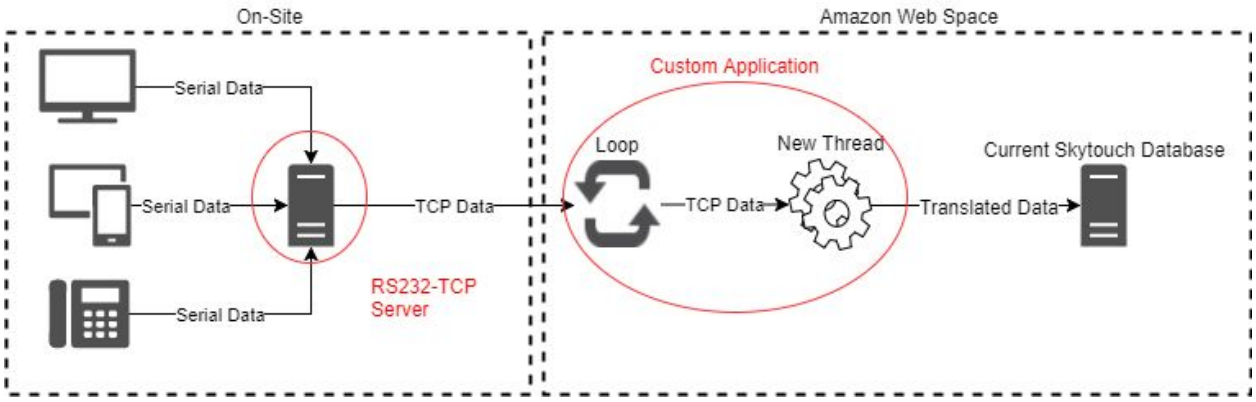

Figure 2

4.2 Integration

Each hotel will have one hardware device installed at their location, that will be supported with TCP/IP communication. This hardware device will be installed directly to the hotel PC allowing direct communication to the AWS cloud. This device will act as a RS232-TCP Server and will wrap the data from the device and send it as TCP data to our application in AWS as seen in Figure 2. This application will handle each bit of TCP data by constantly looping, grabbing each bit of incoming data and creating a new thread that will specifically deal with that TCP data. That thread will translate that TCP data and use the current Skytouch Database's API to send the translated data into the current Skytouch Database. Both applications will use Java as their programming language.

5 Conclusion

In conclusion, we are creating solution to proxy serial and TCP/IP based protocols to the AWS which will eliminating the need for a custom application for each hotel. This technology will save company millions of dollar in the long run because once we implement this in SkyTouch we will not have to custom While we create this technology, we will be facing threading issues, choosing a programming language and choosing the right hardware device, but we are confident that we will overcome this challenges by applying the solution we came up with section 3.

Tech Challenge	Proposed Solution	Confidence Level
Picking a Programming Language	Since we must use AWS, we have to pick Java	100%
Large amounts of Data	Use Multi-threading to handle	85%
Picking a device	Create Hardware Device	80%
Plugin architecture	Custom IO Software	80%

Table 5

As mentioned in table 5 , we feel that we have good plan for our technical issues. We are also prepared for any unexpected issues or changes that may occur. Our plan is to take this process day by day. Also, once we look it Java Enabled Device Interface(JEDI) at Sky Touch's location, it will give us boost to our project as well. We are confident that we can resolve these challenges with our proposed solutions. Resolving these issues will allow us to achieve our main goal which is to provide a better, faster and cheaper solution to Sky Touch.