SABR: Self-Adjusting Bus Routing

---

1 Introduction

The Massachusetts Bay Transportation Authority (MBTA) recently upgraded their existing infrastructure. They are considering design proposals for a real-time bus tracking system to help them track reliability, handle bus system failures, and maintain a target level of passenger comfort.

Self-Adjusting Bus Routing (SABR) effectively utilizes the MBTA infrastructure to provide reliable and comfortable transportation to the greater Boston area. SABR is divided into two main modules: data gathering and data analysis. The first module collects information from bus sensors and relays it to the MBTA servers. The second module (the MBTA servers) stores, analyzes, and makes decisions with the information. These modules work together to respond to failures automatically based on the current state of the bus system. In addition to these two main modules, the system proposes soliciting passenger feedback via an online form.

SABR ensures the MBTA is allocating its resources to appropriate places while maintaining a simple, reliable, and efficient design. The subsequent sections provide the framework of the system and detail the design tradeoffs made.

2 System Overview

SABR has two main modules: buses, which are outfitted with a few key components, and servers, which are kept in the MBTA warehouse. The two modules are able to communicate via a trunked radio network, which allows the system to collect data and respond to failures in real time. *Figure 1* gives a high-level overview of the design, showing the two modules and their key components.

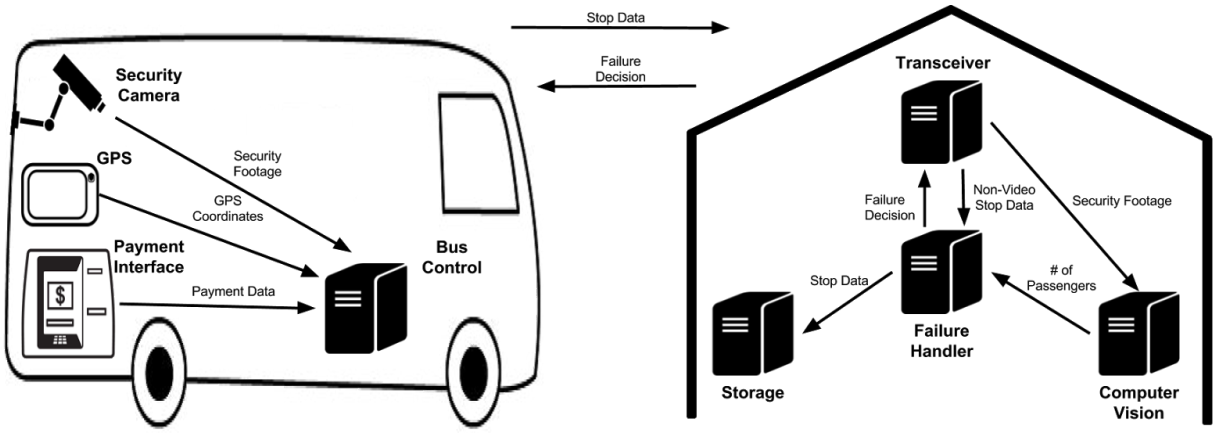

Figure 1. Overview of SABR. The bus control collects information from the bus modules and sends this information to the MBTA warehouse to be processed by the servers.

2.1 Buses

Each bus is equipped with a bus control, GPS, security camera, and payment interface. Before a bus leaves the warehouse for the day, the bus control is loaded with data regarding the stops the bus will traverse on its route. The bus control reads the GPS coordinates of the bus every second; when the bus reaches a stop, the bus control downloads security camera footage and recent payment information and sends this information over the trunked radio network to the MBTA servers.

2.2 Servers

Several servers are involved in the processing of data received from the buses. *Figure 1* shows the relations between these servers. Communications from the buses are received by the transceiver server, which is connected to the radio antenna. The transceiver server sends the camera footage to a server running a computer vision algorithm, which counts the number of passengers on the bus. This number, as well as the rest of the information received by the transceiver server, is forwarded to the failure handler. This server determines if a failure condition has been encountered and, if so, decides what to do and gives the transceiver server messages to relay to the buses. The failure handler also sends the stop data to a storage server for recordkeeping.

3 System Design

The two modules mentioned in the previous section provide the basis for the SABR system. The MBTA buses gather information throughout their routes and send this information to the MBTA servers in order to analyze it and improve the existing bus system.

3.1 MBTA Buses

The MBTA currently has 991 active buses and 45 reserve buses. The bus components utilized by SABR, as well as the data collected and sent from the buses, are enumerated in the sections below.

3.1.1 Bus Components

Each bus is outfitted with a small computer known as the bus control. The bus control can perform basic computation and store a reasonable amount of data. Its main task in the SABR system is to collect data from sensors installed on the bus and transmit said data to the MBTA servers via a trunked radio network. The sensors from which data are collected include the security camera, global positioning system (GPS), and payment interface.

The security camera is able to record footage, which can then be used to extract information regarding the number of people on the bus. SABR utilizes the security camera, rather than the beam sensor, due to the accuracy of the computer vision algorithm used by the MBTA. Additionally, if the beam sensor is used unaccompanied, it is impossible to detect passenger movement through the back door. The drawback of using the security camera footage is that more data must be sent over the network; however, this did not prove to be an issue when designing the system. Calculations and arguments regarding this design choice will be covered in the following sections.

The GPS sensor updates the current longitude and latitude coordinates of the bus once every second, with an error of ± 5 meters (m). Each coordinate provided by the sensor is 64 bits. The bus control reads from this sensor every second in order to receive the most up to date information regarding the location of the bus.

The Payment Interface collects data regarding the payments made on a bus, such as the time of transaction, price of transaction, and, if applicable, a Charlie card/ticket ID. If the transaction occurs with cash, the information collected is 64 bits. However, if the transaction occurs with a ticket or Charlie card, the information collected is an additional 64 bits (total of 128 bits).

At the beginning of the day the bus control is loaded with bus stop IDs, stop names, and GPS coordinates that make up a route. This information is at most approximately 2 KB, and its layout is described in *Table 1*. The bus operator interface will display this route information to the operator and notify him or her of any changes made to the bus's route by the MBTA servers.

Bus Stop ID (64 bits)	Bus Stop Name (512 bits)	Longitude (32 bits)	Latitude (32 bits)
0x0000000000000000	Massachusetts Ave @ Newbury St	0x4398FE9E	0x390D8EBA
0x0000000000000001	Massachusetts Ave @ Massachusetts Ave Station	0x4398FF00	0x390D8EBB
...	...	...	...
0xn	Dudley Station	0x439949D1	0x390D8F45

Table 1: Representation of a route stored by the bus control. The maximum number of stops in a route is approximately 30 and each stop consists of approximately 640 bits of data. Therefore, the total data stored will not exceed 2 KB.

3.1.2 Data Collection

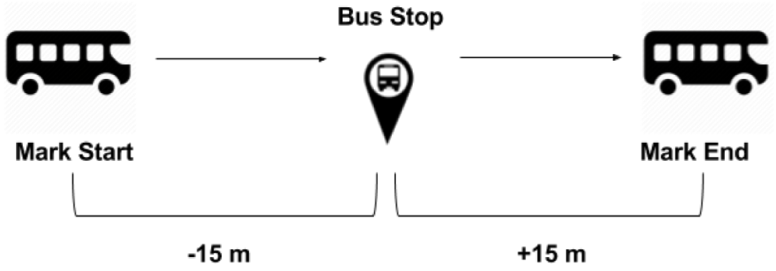

Figure 2: The start of a bus stop is defined as 15 m before the bus stop itself. Similarly, the end of a bus stop is defined as 15 m after the bus stop.

While a bus drives on a route, the bus control receives updated location data from the GPS every second. It compares its current location with the location of the next stop on the stored route. When the distance between the current location and the location of the next stop is calculated to be 15 m or less, the bus control marks this as the beginning of the stop. The bus control then marks the end of a stop when the location of the bus is at least 15 m away from the stop location, as depicted in *Figure 2*.

The ±15 m window around the bus stop was selected to ensure that the GPS updates the current location at least once within the window. Thus, the bus control is still able to send data about the stop regardless of whether or not the bus actually stops at or drives past the stop. In order for the GPS sensor to not update within this window, the bus would need to be traveling approximately 30 meters/second or 67 miles/hour, which is highly unlikely.

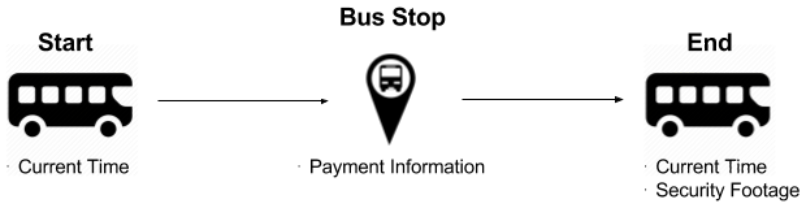

Figure 3: The flow diagram above displays the three positions within a bus stop and enumerates data collected at each position.

When the bus marks the start of a bus stop, the bus control stores the current time (32-bit timestamp) as the start time of the stop. At the stop, the bus control will store all data obtained via the payment interface. The bus control then marks and stores the end time of the stop after the bus travels 15 m or farther from the stop. Furthermore, after reaching this location, the bus control stores 5 frames of security footage from the security camera. *Figure 3* depicts the collection of data at each point. All collected data are then sent to the MBTA servers via the trunked radio network.

3.1.3 Data Transmission

The information sent to the MBTA servers over the trunked radio network at each bus stop contains the data collected from the stop and additional metadata. The packet sent is described in the table below:

Data	Size
src addr	48 bits
dst addr	48 bits
stop id	64 bits
start time	64 bits
end time	64 bits
payment information (each)	128 bits
security footage (5 frames)	1.2 MB (0.24 MB/frame)

Table 2: Description of the data items contained in the packet that is sent to the MBTA servers after each bus stop.

The payment information is encrypted with AES encryption on the bus control before it is sent over the trunked radio network. AES encryption increases the size of the payment information by a few bytes but ensures all sensitive customer information is protected.

Calculations regarding the trunked radio network are completed in *Section 4.1*. These values demonstrate that the network does not experience an excessive load and SABR continues to function well even in worst-case scenarios.

3.2 MBTA Servers

SABR utilizes ten MBTA servers: one transceiver, three computer vision, one failure handler, three storage, and two replica servers. *Figure 4* shows the relations between these components.

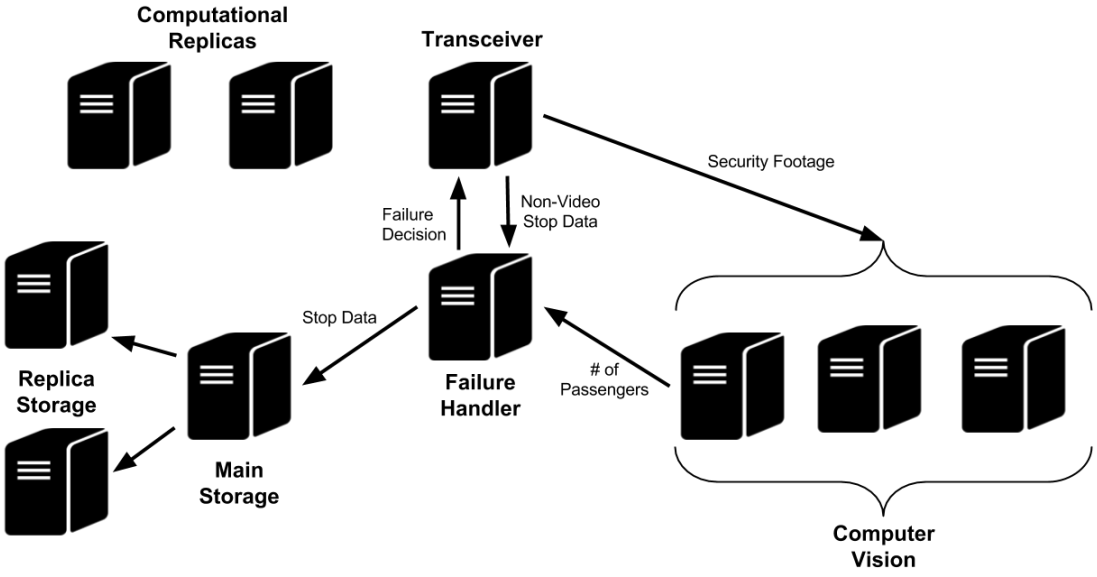

Figure 4: Data Flow Diagram for the MBTA Servers. Information collected by the bus control is sent into the transceiver, which then dispatches this information to be analyzed and stored by the other servers.

3.2.1 Transceiver

The main task of the transceiver server is to transmit and receive data to and from the MBTA buses. Upon receiving a transmission, it sends the security footage to one of the three computer vision servers and the remainder of the data to the failure handler server. The transceiver does not keep any of the data it receives from buses. Additionally, the transceiver is notified if a bus

needs to be re-routed or dispatched from the reserve fleet. It then transmits the new route to the specified bus.

3.2.2 Computer Vision (CV)

Due to the computational intensity of the CV algorithm, three machines are allocated to this part of the system (see calculation in *Section 4.1.1.3*). The transceiver selects which CV server to send security footage to in a round robin fashion (e.g. if the transceiver previously sent footage to CV server #1, it would next choose to send footage to CV server #2). After receiving 5 frames of security footage, the CV server runs the algorithm on the footage and sends the results (i.e. passenger count) to the failure handler server.

3.2.3 Failure Handler

This machine receives data sent from a bus stop and decides if a route is suffering from high demand. When the failure handler detects high demand or is notified of high demand by a system administrator, it utilizes the real-time tracking information in order to provide a decision for rerouting active buses and/or pulling from the reserve bus fleet. *Section 3.3* describes how this decision is made. This decision is then sent to the transceiver server for dissemination to the buses. Furthermore, after processing the data it receives, this server sends all data to the main storage server to be persisted.

3.2.4 Storage

Dataset	Size (MB)
Census data	600
Bus metadata	10
Route and schedule data	100
Alternative stop data	100
Operator data	10

Table 3: The table above enumerates the datasets stored by the MBTA and their corresponding sizes.

date	route_id	bus_id	stop_id	start_time	end_time	payment_info	num_passengers

Table 4: Together, the items listed above comprise a record in the database for each packet received.

The storage servers maintain a number of datasets required by the MBTA. Descriptions of these datasets and their sizes are given in *Table 3*. Additionally, one of these servers receives data from the failure handler regarding bus stops and persists the data for analytics or quality

assurance by the MBTA. Each stop entry in the database will contain all the fields enumerated in *Table 4*. For increased reliability, the stop entry is then pushed to the two replica storage servers. These replicas, along with the main storage server, are configured to provide strong consistency guarantees. Given that these servers possess 10 TB of storage, SABR will be able to store all data indefinitely (see *Section 4.1.4* for calculation).

3.2.5 Handling Server Failure

SABR makes the assumption that the transceiver server cannot fail. In order to determine if another server has failed, the transceiver sends out periodic heartbeat messages to all the servers and waits for responses. If a server is not responding to heartbeat messages the transceiver will mark that server as temporarily down. The transceiver will continue to send heartbeat messages to the crashed server to detect when the server recovers. *Figure 5* depicts this process.

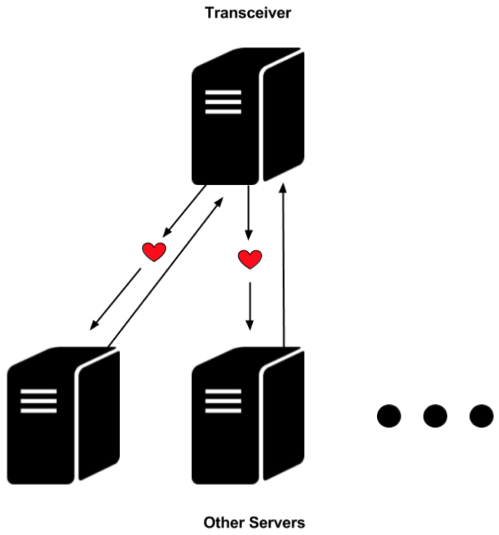

Figure 5: The transceiver server sends out periodic heartbeat messages to all other servers and waits for replies. If the transceiver does not receive a response from a server, it marks that server as down.

There are three failure scenarios after a crashed server is identified:

1. The server is computational (i.e. CV and failure handler servers)
2. The server is a replica
3. The server is a storage server

3.2.3.1 Computational Failure

In the event that a computational server fails, the system employs one of the two replica servers as a replacement. The replica servers contain the functionality of all the computational servers and therefore will readily be able to replace any that fail.

3.2.3.2 Replica Failure

In the event a replica server fails, it will not be able to take over the functionality of any computational servers until it recovers.

3.2.3.3 Main Storage Failure

In the event of a main storage server failure, the system will employ one of the two replica storage servers as a temporary replacement for the main storage server. Secondly, the replica storage server will maintain a log of all new persistent data it receives and later use this log to guarantee consistency between itself and the original main storage server.

3.3 Failure Handling

The design of SABR allows the system to respond dynamically to external stressors. There are three common problems that a bus service should be able to respond to: high demand, the need to create an additional route, and route disruption.

3.3.1 High Demand

Some routes may become overloaded with passengers depending on the time of day or due to special events. SABR aims to place buses where they are needed most.

3.3.1.1 Definition

A route is in high demand when the average passenger-to-seat ratio (PSR_{avg}) across all buses on the route is greater than $k_{comfort}$ ($k_{comfort}$ = 1.4 for current MBTA goals). Note that this method of measuring demand means that an individual bus can have a PSR greater than $k_{comfort}$ on a route under normal demand. This is intentional and ensures that resources are only allocated to routes that are under consistently high demand.

3.3.1.2 Detection

The system allows for real-time tracking of the number of people on each bus, and the database stores the maximum seating occupancy of each bus. These two facts make it simple to calculate PSR_{avg} for any given route.

3.3.1.3 Response

SABR can respond to high demand by sending additional buses to the route. To decide if it should and how to do so, it performs some calculations and then, potentially, follows a priority ordering to determine which bus(es) to pull onto the route under high demand.

3.3.1.3.1 Calculations

PSR_{avg} is calculated by summing the number of people currently on all buses on a route and dividing by the total number of seats across all buses on the route.

We wish to calculate PSR_{avg}^{+n} and PSR_{avg}^{-n}, the predicted PSR_{avg} of a route after adding or removing n buses, respectively. This can be done simply by repeating the PSR_{avg} calculation with an updated number of seats.

A secondary metric T is also used, which is the average amount of time in minutes between buses departing on the route. The system seeks to keep T under some target value T_{target} for all routes (T_{target} = 20 for current MBTA goals). Since we do not expect high demand to significantly affect T, the system does not ever attempt to add buses to decrease it; however, it tries to avoid increasing T above T_{target} when pulling a bus from another route.

T is calculated by dividing the time taken to complete a loop of the route by the number of buses on the route. This assumes that buses travel at the same speed and are evenly spaced, which we consider to be a reasonable approximation since the MBTA should be attempting to do this to maximize usage of the bus network.

Finally, as with PSR_{avg}, we wish to be able to calculate T^{+n} and T^{-n}, which is done by repeating the above calculation with a new number of buses.

3.3.1.3.2 Pull Priority

We define an additional PSR_{avg} threshold of k_{full}. This value represents the point at which buses approach crunch capacity, preventing customers from boarding. Then the system uses the following priority ordering to decide how to respond to high demand.

1. Pull a bus from reserve
2. Pull a bus from a route with $PSR_{avg}^{-1} < k_{comfort}$
3. Pull a bus from a route with $PSR_{avg}^{-1} < k_{full}$ but only to assist a route with $PSR_{avg} > k_{full}$

A bus is always pulled from the highest level of priority available. Within a level, the system pulls a bus from the nearest viable route, with nearness evaluated as the minimum distance from the origin or midpoint of the route from which it is pulling to any point along the destination route. This is repeated n times until $PSR_{avg}^{+n} < k_{comfort}$ or there are no more buses that can be pulled.

Additionally, a bus is never pulled from a route with $T^{-1} > T_{target}$. This guarantees that the MBTA metric for service frequency is always met.

3.3.2 Additional Routes

Under some circumstances, the MBTA is required to create additional bus routes. The MBTA knows what these routes are, and we assume they have a conservative ridership estimate for each one.

3.3.2.1 Response

A new route can be viewed as simply an extension of the high demand problem. The system starts with a ridership estimate and zero buses on the route, putting the route under initially infinite demand. The system will then immediately detect high demand and use the bus pulling procedure to create a projected $PSR_{avg} < k_{comfort}$ for the new route. After buses are dispatched, the system adjusts to the actual demand for the route.

3.3.3 Route Unavailability

Temporary issues, such as construction or police activity, can unexpectedly block routes. The goal in these situations is to minimize the number of passengers unable to reach their destination. Re-routing in SABR follows a three-step process:

1. The re-routing algorithm discussed in *Section 3.3.3.1* generates a potential route
2. The route goes through a human review process
3. The updated route is transmitted to the appropriate buses

3.3.3.1 Re-Routing Algorithm

The goal of the re-routing algorithm is to minimize deviation from the original route.

We prioritize preserving stops above everything else—even if that leads to possibly longer routes—because changing stops inevitably causes some people to miss the bus. Thus, every accessible stop in the original route is preserved in the new route.

The algorithm replaces inaccessible stops by alternate stops that are close to them. We expect that the affected stops typically form connected intervals on the route; the strategy described below is applied individually to each interval.

The total cost of an alternate interval of stops is defined as:

$$C = D^2 + \lambda_1 \sum_{0 \le i < |stops|} dist(stops_i,\ nearest(stops_i, newstops))^2 + \lambda_2 |newstops|^2 \qquad (1)$$

In *Equation 1* above, *D* is the length of the new route, *stops* is the set of affected stops, *newstops* is a subset of alternate stops, and λ_1 and λ_2 are adjustable parameters. *dist(x,y)* is the distance between points *x* and *y*, and *nearest(x, A)* is the nearest point in array *A* to *x*.

The algorithm selects *newstops* from the set of alternate stops in order to minimize *C*. In the expression for *C*, the first term preserves a short route length, the middle sum ensures there is an alternate stop close to every original stop, and the third term maintains a low number of stops. All terms are squared in order to penalize routes with small terms on average but a few large ones. λ_1 and λ_2 control the relative priorities of the terms. These parameters are selected once, at the installation of the system, based on experiments performed by the system administrators.

Only alternate stops within 1 mile of an original stop are considered. If no such alternate stops exist, the buses are sent to the next accessible original stop.

Figure 6 shows a possible result of the algorithm.

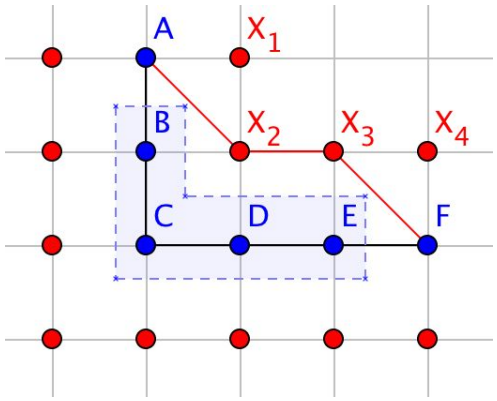

Figure 6. Blue dots are original stops, red dots are alternate stops. A-B-C-D-E-F is the original route, and B, C, D, and E are affected stations. The new route is A-X_2-X_3-F. Note that the number of stations can change.

The algorithm runs on the MBTA servers. To guarantee a timely response, it returns the best result found in 1 minute of execution. Because of the typically small number of affected stops and the threshold of 1 mile for alternate stops, we expect a simple implementation based on brute-force and heuristics to perform well in the allotted time.

3.3.3.2 Route Review and Publishing

The resulting route is displayed to a system administrator who reviews it and has the right to adjust it.

The route is then sent by the transceiver server to all buses on the affected route. The operators on the route announce the changes, and the modifications are also noted on the MBTA website. If possible, MBTA employees are sent to put up signs at the original stops informing people of the new route.

The number of buses on the changed route is regulated automatically through the high demand response algorithm described in *Section 3.3.1.3*.

3.4 Customer Feedback

We encourage the MBTA to collect feedback through an application that does not communicate with the bus control. In particular, we feel that a mobile-friendly web application on the MBTA website would best serve this purpose. This would provide passengers with a structured way to give feedback without requiring them to install any special application.

4. Evaluation

In this section, we evaluate SABR on the requirements and use cases specified by the MBTA. This includes an analysis of the performance, fault-tolerance, and simplicity of the system. In addition, we evaluate the system in two other design areas that are not explicitly required by the MBTA: security and scalability.

4.1 Metrics

In the following subsections, we calculate multiple metrics in order to prove that SABR meets the requirements of the MBTA under the given hardware limitations. We also use these metrics to justify some design decisions.

4.1.1 Data communication throughput

The throughput capacities of the different parts of the system need to match. Therefore, we need to show that each module is able to process the data it receives at an appropriate rate.

4.1.1.1 Bus-side

When a bus is not in a station, it only reads data from the GPS sensor, at a rate of 64 bits/second. This rate is very small compared to the capabilities of processors today; thus, the bus control will have no problems with it.

When a bus is at a station, it needs to store and process several pieces of data until the bus leaves the station and the transmission happens. These are enumerated in *Table 1*. The only variable size is in the payment information; assuming a worst-case scenario in which 77 passengers enter a bus at a station, which is the crush capacity of the largest bus operated by the MBTA, the payment data takes 1.25 KB. In that case, the size of the packet adds up to less than 1.21 MB.

We assume that the bus control is able to process this amount of data well and that it has enough memory to store it. After the bus control transmits the information, it discards it, so there is no overlapping data between stations.

4.1.1.2 Bus-server transmission

The radio network has a throughput of 16 Mbit/second, and it has 10 frequencies through which the buses can send data in parallel. In a worst-case scenario where 1200 buses send packets at the same time, it takes less than 73 seconds for all the data to arrive at the transceiver. This duration is calculated by multiplying the packet size by the number of buses, dividing by the number of frequencies, and dividing again by the throughput. Note that we assumed that the time to get assigned a frequency is negligible if a frequency is available.

Based on observations of the MBTA bus routes, we assume that it takes on average around 100 seconds for a bus to go between two consecutive stations. Therefore, even in the worst-case when all buses arrive in a station at the same time, all data can reach the transceiver server before the the next wave of transmissions begins.

The previous calculation shows that sending all the security camera footage at every stop functions within the network limits. Therefore, a higher accuracy is achieved without major complications.

4.1.1.3 Server-side

Finally, we need to show that the servers are able to process the data at the rate it comes in. The maximum throughput to the transceiver is 160 Mbit/second, which is calculated as the network throughput multiplied by the number of frequencies. The internal network has a throughput of 1 Gbit/second, which is sufficient to move the data around.

Most servers perform simple operations such as transmitting, sorting, and storing data. We assume that these operations can be performed at a rate that matches the inbound rate of 160 Mbit/second. However, there are two modules that perform more complex operations: the computer vision module and the failure handler module. The latter runs only when there is a case of high demand, which we assume is infrequent, so it does not affect the general throughput.

The computer vision module runs the algorithm for every packet received. If buses need 100 seconds to go between stations and if there are 1200 buses that each send 5 footage frames, the input rate is 60 frames/second. We assume that the computer vision servers have a processing speed of 0.05 seconds/frame (i.e. 0.25 seconds to process the 5 frames); so the processing rate is 20 frames/seconds. Therefore, we need three servers running in parallel in order to match the throughput in this worst-case scenario.

4.1.2 Frequency Assignment Delay

It is important to know how long a bus may wait to get assigned a frequency. If all buses try to send data at the same time, based on the calculations in *Section 4.1.1.2*, it can take up to 73 seconds for all buses to transmit. The last bus to send needs to wait for all but 9 buses in front

of it to go (due to the 10 frequencies); this is upper bounded by the 73 seconds above. A slightly tighter bound could be calculated, but this suffices for the calculations where the metric is used.

4.1.3 Collection-to-Storage Delay

The duration between a bus arrival at a stop and the final storage of the data at the warehouse is composed of the time it takes to get assigned a frequency, the time to transmit the data, and the time to move the data from the transceiver to the storage.

Section 4.1.2 shows that it takes at most 73 seconds to get assigned a frequency and *Section 4.1.1.2* shows that a bus needs to send up to 1.21 MB at 16 Mbit/second. In addition, the latency of the network is 0.01 seconds. At the warehouse, the camera footage is processed in 0.25 seconds, and the other time spent between servers is negligible. Adding these, we conclude that the maximum transmission time is upper bounded by 74 seconds.

For permanent storage intentions, it does not matter how late the data arrives, as long as it is within reasonable limits (e.g. the same day). For failure detection intentions, this duration will lead to a reasonable response time, described in detail in *Section 4.1.5*. To achieve faster transmission times, the amount of data sent would have to decrease, which would lead to a lower accuracy.

4.1.4 Data Storage

The servers store all data sent from the buses, with the exception of the security camera footage, which is reduced to a 16-bit integer representing the number of people in the bus. To get an estimate for the storage needed, we assume for now that data are stored with no additional overhead.

The bus data from a packet, without the camera footage and the payment information, are 36 bytes. For the payment information, the assumption in *Section 4.1.1* of having 77 passengers at every stop leads to an overestimate. Instead, we know that 446,700 people use the MBTA buses on average every day. By assuming each of them uses the bus two times, and by allowing a little extra margin, we estimate that less than 1,000,000 payments are registered every day. This translates to 16 MB of payment data per day.

Assuming that 1200 buses run 24/7 and arrive at a new station every 100 seconds, and using the payment information estimate above, the system would have to store less than 55 MB per day. To store all the data for 100 years, only 2 TB of storage are needed, which is much less than the capacity of a storage server. Storing a database requires additional overhead, but even if the size is doubled or tripled, the data can still be stored for a century. We assume that in 100 years there will be more performant methods of storing data, so thinking about longer periods of time is not necessary.

4.1.5 Failure Response Time

As argued in *Section 4.1.3*, it can take 74 seconds for the data from a bus to be processed on the servers; this is also the time it may take to detect high demand. If a route needs to be adjusted, the re-routing algorithm takes up to 1 minute, and we estimate that the systems administrator needs up to 2 additional minutes to validate the change. Therefore, less than 4.5 minutes are needed to send a command to a bus.

If a bus is sent from the warehouse, the time until it arrives at a route depends on how far the chosen route is from the warehouse location. Assuming the warehouse is located in a central location, we estimate that it would take 10 minutes in the average case and 25 minutes in the worst-case to drive to a route. Adding the detection time to this, the average response time is 15 minutes and the worst-case response time is 30 minutes.

If the bus is sent from another route, the times can be both smaller and larger. In the best-case, a bus is taken off of a route that intersects the current route, and the shift is instantaneous. In the average case, we expect that a bus would need around 20 minutes to get to a route, and in the worst-case, a bus may need to travel 50 minutes between two stations (corresponding to the distance between Chestnut Ave., Burlington station and Holbrook/Randolph station, which is a diameter of the MBTA network). Adding the detection time to this, the best-case response is around 5 minutes, the average-case response is 25 minutes, and the worst-case response is 55 minutes.

Note that excessive traffic could lengthen the time almost indefinitely in all cases.

4.1.6 Data accuracy

All data are expected to be transmitted correctly. The route ID, stop ID, timestamps, and the payment data are completely accurate. One problem might appear if two consecutive bus stops are very close to each other on a route (e.g. less than 40 m apart); this does not happen in the MBTA bus routes.

The security camera footage can be used to achieve a 95-98% accuracy for the number of people in the bus. Because high demand is triggered only if the average passenger-to-seat ratio on a route is too high, possible small errors of the computer vision algorithm are unlikely to drive the average by a significant amount. In a worst-case scenario, the error rate is 5%. We further assume that each error has magnitude ±15, in buses with capacity 40. Note that the expected number of errors on a route with x buses is $0.05x$ in one round of transmission. In the buses that have an error, the PSR varies by ±0.375. This would shift the average passenger-to-seat ratio by ±0.01875. Because the MBTA threshold is two orders of magnitude higher, such a variation is unlikely to have significant impact on high demand detection.

4.2 Security

Security is not a major design goal of SABR. However, the design minimizes security risks wherever this does not get in the way of other goals. In particular, payment information is sensitive, so this information is encrypted before it is sent over the radio network. An adversary listening on the radio network can intercept the camera footage, but this is not worth the computational cost of encryption, especially given that buses are open to the public.

4.3 Use Cases

This section contains an evaluation of SABR on the use cases mentioned in the specifications of the system.

4.3.1 Normal

Under normal operation, there are no failures and all buses work optimally. Any minor issues such as small bursts of passengers on a route are resolved by adding idle buses.

4.3.2 High Demand

Under high demand, all idle buses are utilized. In addition, buses from less crowded routes are moved to the busy routes. If it is possible to satisfy every route with a reasonable PSR, then SABR converges toward that. Otherwise, if demand is extremely high across the city, some routes may be served poorly.

4.3.3 Construction

In case of construction, every affected route adjusts to avoid the construction region. The system then redistributes buses to keep a low PSR. All routes should still be functional in most cases. In a scenario where a significant area becomes unreachable, SABR may not be able to adjust the route in a way that makes sense, but the requirement that alternates routes be approved by a systems administrator allows a human to step in and make a rational decision.

4.3.4 Historical Data

The information from the buses is transmitted and stored, regardless of route failures. The MBTA should be able to reliably reconstruct bus history by querying the storage server.

4.4 Fleet Monitoring

The MBTA would like to be able to use the data collected by SABR to evaluate their system for themselves. This section explains how SABR allows the MBTA to do so.

4.4.1 Frequency of service

The MBTA wants buses to be departing on each route at least once every twenty minutes on average. To check if this target is being met, one can check all of the departure timestamps for a given stop on a given route on a given day and calculate the average of the time spans between them.

4.4.2 Coverage

One can accesses the census data and compare it with the route data to check how many citizens in the metro area live within .5 miles of a bus stop.

4.4.3 Reliability

To check how many buses arrive at their origin, midpoint, and destination timepoints within three minutes of their scheduled arrival time, one can check the arrival timestamps and compare them with the times in the schedule data.

4.4.4 Comfort

To calculate the maximum passenger-to-seat ratio for a given ride, one can search the records of the specific bus that performs the ride. These records contain the number of passengers on the bus at each stop. These numbers can be divided by the number of seats in the bus, obtained from the bus metadata, to determine the passenger to seat ratios.

4.4.5 Total Load

Calculating the number of unique people using the bus system is a bit trickier. One can sum the number of passengers who paid to enter the bus on the stops that occurred within a time interval of interest, but this would not account for people who ride the bus multiple times. One would need to estimate the number of times the average passenger rides the bus within a time span to estimate the number of unique passengers. Looking at riders with Charlie Card IDs could potentially provide some insight. To determine how many transit-dependent passengers are there, one can include only the stops that are in low income areas as identified by the census data.

4.4.6 Value to Network

The number of transferring passengers can be found by checking the payment information, which notes whenever a transfer takes place. The payment information is sent with stop data, so one can easily filter for transfers that take place within a certain region or time period.

4.5 Scalability

If SABR is to be extended to larger cities or areas, some modifications are necessary. The two main bottlenecks of the system are the radio network throughput and the computer vision servers. If more buses are added, data would be unable to be processed at a desirable rate.

With the current network throughput of 16 Mbit/second, the maximum number of buses supported is 1650. To go over that limit, the MBTA would have to either have more radio frequencies or faster transmission methods.

With three computer vision servers, the maximum number of buses supported is only 1200. However, this limit is easily extensible: the MBTA can simply buy more computer vision servers. More specifically, one server would be needed for every additional 400 buses in order for the processing rate to match the inbound rate of data. Because computer vision is a field that evolves quickly, faster algorithms could also become available in the near future.

5 Conclusion

SABR allows efficient monitoring of the bus fleet and reliable handling of failures. It does not require additional investments, and it performs well on all of the use cases required by the MBTA. Therefore, we believe that it can serve the city of Boston well and improve the satisfaction of the MBTA customers.