Massachusetts Institute of Technology — 6.033 Design Project

Gigabus:

Leveraging Technology to Improve the MBTA Bus System

Nicole Lu, Landon Carter, Brian Chen
nicolelu@mit.edu, lcarter@mit.edu, bpchen@mit.edu

Recitation: Peter Szolovits, TR1/2
Tutorial: Amy Carleton, F1

8 May 2017

1. Introduction

The Massachusetts Bay Transportation Authority (MBTA) bus system services millions of passengers in the Boston area every week. Bostonians count on MBTA buses every day to get them from home to work and everywhere in between. While bus riders can usually count on the buses to get them where they want to go, when they need to be there, the service is not perfect. A variety of events from increased demand to mechanical issues can cause hiccups in what is otherwise a well-oiled machine.

The size of the MBTA bus system, both in terms of number of buses and geographic area served, is one of the most valuable aspects of the system, enabling it to meet the needs of many area riders. Unfortunately, its size also makes it more challenging to ensure smooth operations. The age of the bus system poses yet another challenge to offering perfect service. However, the MBTA can better leverage its existing sensors and communication infrastructure to monitor and understand user experience (reliability and comfort) and inform future decisions about the system, in both the short and long term.

In this report, we present Gigabus, a computer system design and communication protocol to achieve this goal of leveraging existing infrastructure to improve MBTA service. In the Gigabus system, buses gather data from bus sensors and send it to the central servers with a prioritization system that maximizes utilization while guaranteeing a basic degree of data accuracy. The central servers use historical data and administrator input to dispatch buses to meet service targets. When service targets cannot be met due to heavy load or bus failures, Gigabus attempts to achieve a balanced compromise among bus routes. Finally, servers integrate with subscription and feedback mechanisms to keep passengers from all socioeconomic backgrounds updated and ensure their voices are heard.

1.1 Design Criteria

The MBTA offers bus transportation as a service to the citizens and visitors of Boston. As such, the MBTA has laid out specific criteria against which Gigabus can be evaluated. This section will introduce high-level goals for the system and define them.

- *Reliability*: Gigabus works to ensure MBTA riders can be confident that the bus will take them where they need to go and get them there on time. The system's reliability target is to have buses arrive at the origin, midpoint, or destination timepoints within three minutes of the scheduled arrival time at least 75% of the time.
- *Coverage/Accessibility*: Much of the design of Gigabus is inspired by making the bus system to be accessible to as many citizens as possible, especially those who rely on the MBTA's services. To this end, we want at least 75% of the general Boston metro population and at least 85% of low-income households to have at least one bus stop within .5 miles of where they live.
- *Comfort:* Gigabus helps ensure riding the bus is a pleasant experience. Thus, the maximum passenger-to-seat ratio should be less than or equal to 1.4, 96% of the time.

2. System Overview

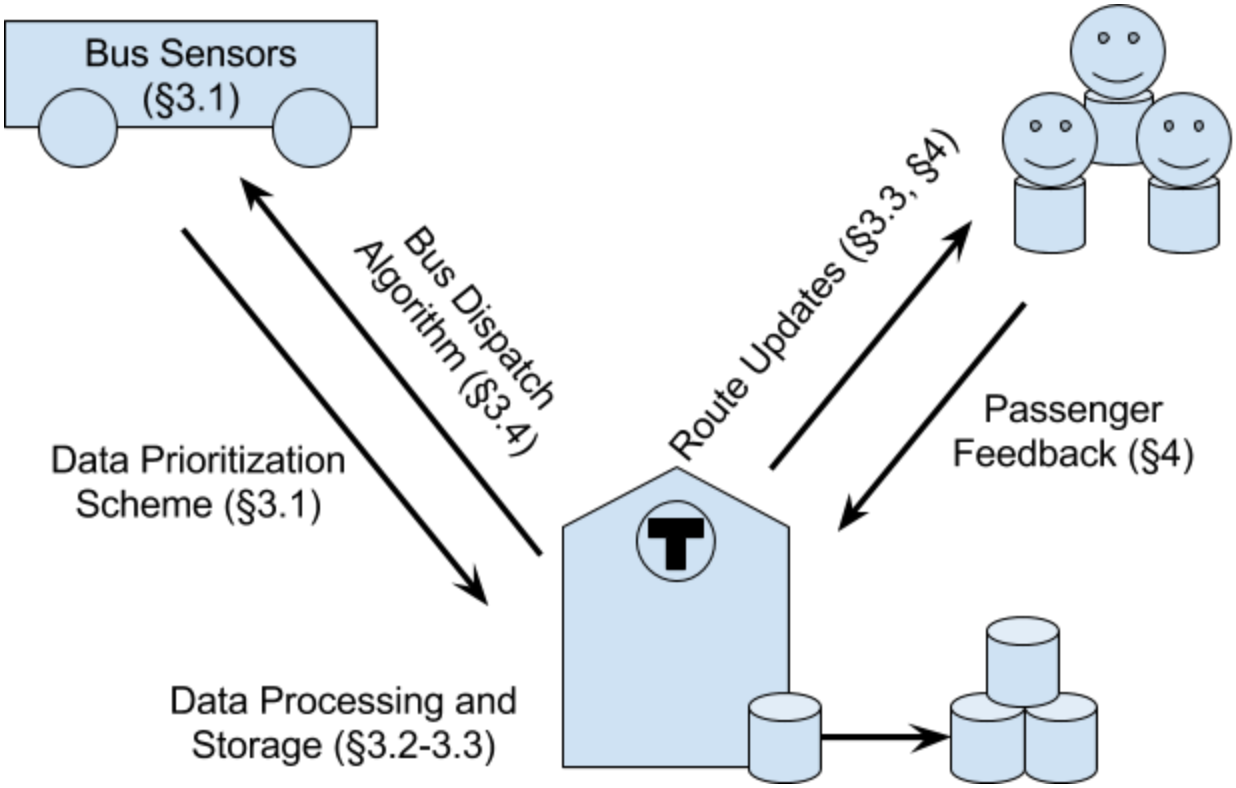

Figure 1 - Gigabus system overview and report outline.

Each bus in the system is equipped with a suite of 5 sensors, which provide the primary source of data that enters the Gigabus system. This data can be used to monitor the time buses take to traverse routes and the number of passengers on buses. However, there is not enough bandwidth to continually send all collected data to the central servers, so Gigabus implements a prioritization scheme to send low-bandwidth data that can be used to compute all necessary information with a good degree of accuracy. The low-bandwidth data is complemented when possible by high-bandwidth image data, which can provide more accurate passenger counts for more accurate historical data and a finer level of control over Gigabus' day-to-day operation. We will discuss the data sources and prioritization scheme in Section 3.1, as well as elaborate on how the prioritization scheme enables us to reliably meet service targets.

In Section 3.2, we will elaborate on the role of the central servers in data processing. The central servers are responsible for receiving and aggregating the data, as well as dispatching buses based on data to meet service targets, as described in Sections 3.3 and 3.4. To be resilient to server failures, Gigabus replicates historical data across multiple servers via a distributed database system. During normal operations, the system will dispatch buses at regular intervals along routes, but when the system predicts from historical data or learns from administrator intervention of traffic or high-demand scenarios, it will dispatch buses using an algorithm that first tries to meet all service targets using idle buses, and, failing that, makes balanced concessions across all routes to come as close to the service target as possible.

An aspect of our system that cannot be overlooked is Gigabus' interaction with passengers. Using existing technology, Gigabus can ensure a more convenient and reliable experience for riders by publishing bus

schedules and live status updates. We can further utilize common technologies to solicit and collect feedback from passengers and allow them to subscribe to certain route updates. This will be further discussed and elaborated on in Section 4. Finally, Gigabus' evaluation metrics and methods for tuning are discussed in Section 5.

3. System Design

3.1 Data Sources and Prioritization Scheme

There is a suite of 5 sensors present on each bus in the MBTA fleet: a GPS sensor, payment interface, operator interface, beam-break counting sensor, and security cameras. These data sources can be roughly sorted into 2 categories — *"low-bandwidth"* data sources and *"high-bandwidth"* data sources. The only high-bandwidth data source is images from the security camera; all other sensor data falls into the low-bandwidth data sources category.

It is feasible for our system to run entirely on the low-bandwidth data — the only extra information that the security cameras provide are more accurate passenger counts. However, this additional accuracy is only marginal — beam sensors can already provide 85–90% accuracy, while security cameras increase this to 95–98% accuracy. It is because the system can function entirely off of low-bandwidth data that we prioritize it. This is equivalent to prioritizing stability and robustness over performance.

Specifically, our prioritization scheme assigns a send priority to each piece of data the bus controller has not yet sent. In each timestep, the central servers publish a bandwidth utilization metric, and the bus controller decides based on this and the send priority of each piece of data, whether to send the data or wait. Send priority is increased exponentially over time for low-bandwidth data in order to assure prompt delivery. On the other hand, send priority is increased linearly over time for high-bandwidth data, and only the most recent pack of 5 images is sent, while older images are discarded. Gigabus' priority scheme and extrapolated bandwidth requirements are laid out in Table 1. Note that the total aggregate bandwidth is 160 Mbps.

Table 1 - Data transmission frequency targets and extrapolated aggregate bandwidth requirements.

Data Type (ordered by priority)	**Desired Frequency (seconds)**	**Data size (bits)**	**Extrapolated bandwidth requirement (Mbps)**
Voice and operator reports	-	-	-
Payment/transfer logs	180	129 × (num passengers) + 128	0.0082
GPS	10	64 + 128	0.0199
Beam Sensor	180	32 + 128	0.0092
Image	60	9600000 + 128	165.8

To provide a few details on how the numbers in Table 1 were generated, as well as the assumptions made:

- The voice and operator reports are rare and unpredictable events, and therefore cannot be considered in the day-to-day priority scheme. They take top priority, and in general, the table is ordered from highest to lowest priority.
- Payment/transfer logs and beam sensor data need only be sent once per stop, and we estimate the average time between stops to be 180 seconds.
- The data size for payment/transfer logs is determined by the number of passengers that get on at the stop. We have estimated the average number of passengers per stop to be 10, though the exact estimate is unimportant — the aggregate bandwidth is low for any estimate of the correct order of magnitude. Each transaction has a 32-bit UTC timestamp, 32-bit cost of ride, possibly a 64-bit Charlie card number, and a transfer bit, for 129 total bits per transaction.
- The transmission frequency of the image data was chosen to approximately saturate the aggregate bandwidth to provide “pressure” to the system so that it performs efficiently. Each packet contains 5 images. Finally, by averaging multiple image sets together that are spaced apart in time, we can slightly increase our passenger count accuracy — for example, passengers may move around and provide the cameras a better view.
- All data sizes include a 96 bit header containing the send and receive addresses, plus a 32 bit UTC timestamp of when the data was collected for 128 bits of overhead on each transmission. The UTC timestamp will later be compared to the server’s receive timestamp to determine data age, which will help with our evaluation metrics as detailed in Section 5.1.

It is important to note that the desired transmission frequencies for each type of data, as well as the parameters of the priority increases over time can be tuned for the system to perform optimally. If, for example, the radio network becomes clogged with images, forcing the low-bandwidth data to be severely delayed, the transmit priority could be tuned so that the system always attempts to leave one or two channels open for low-bandwidth communication. These parameters can be adjusted nightly when the

buses are connected to the central warehouse WiFi, which will ensure that all buses receive the update simultaneously, and the system remains fair across all buses.

We believe that this priority scheme allows Gigabus to maintain excellent performance and efficiency by maximizing network throughput, while still ensuring robust operation by prioritizing critical pieces of information. This enables Gigabus to reliably report passenger counts and bus locations in real-time, allowing the system or administrators to re-route buses so that the goal of a low passenger-to-seat ratio can be maintained.

3.2 Data Storage and Image Analysis

All data received from the buses is sent to MBTA headquarters and handled by the reliable server, one of the servers at headquarters, which never fails. The reliable server stores all low-bandwidth received data for the current day in a simple relational database, classified by data type and tagged with metadata including the bus, time and date, and bus stop or part of bus route. This database must support querying by each of those metadata classifications.

When the reliable server receives any high-bandwidth security camera images, it attempts to analyzes them or sends them to one of the other servers for image analysis in order to compute the number of people on the bus, depending on computing resources available (the bus dispatch algorithm, described in Section 3.4, takes priority). If no servers are available to perform analysis, camera images are queued up for computation, with the stalest images simply deleted if the reliable server runs out of storage. When the results of image analysis are completed by the reliable server or reported to it by other servers, it updates its database to replace the passenger counts returned by the beam sensors to the more accurate results from image analysis. We assume that the image processing algorithm runs roughly in real time for a single 30 fps image stream, so a half-dozen servers can process the aggregate data stream (5 photos per minute per bus from 1036 buses) in real time, even given network overhead to move image data around and server failures. This means that updated results will be returned promptly enough to be useful. Note that, as mentioned in Section 3.1, Gigabus can operate entirely on beam sensor data, so even if images are deleted without being processed, this only results in marginally less accurate passenger counts. Because image data becomes outdated rapidly and takes up a lot of storage space, it is discarded after being run through the image processing algorithm.

At night, when no buses are running and no data is received, the reliable server will write that day's data onto a replicated distributed database across the other nine servers. This is the only time data is written onto the other servers for long-term storage, so if the unreliable servers suffer any temporary failures, it will have minimal impact — the reliable server can simply wait and retry later, since there is lots of time. Night time can also be used to perform consistency checks across replicated data.

The reliable server also permanently stores the MBTA's pre-existing datasets, including census data, bus metadata, route and schedule data, and data about alternate stops. Since these data sets are static and relatively small, they do not need to be replicated. The reliable server also stores predictions and announced arrival times, described in the following sections, and similarly stores these predictions into

this replicated filesystem on day rollover. All of this data fits comfortably into the 10 TB storage of each server; we delay a more detailed analysis of the storage requirements to Section 5.2.

The MBTA database enables MBTA administrators to respond to customer complaints about bus arrival times, since the administrators can look up the data by the day, bus route, and stop. Such a database would also enable the MBTA administrators to quickly look up historical usage patterns in high-demand scenarios in anticipation of a similar situation, such as the recurring scenario of a Red Sox game, and allow them to inform the system about the possibility of such a situation through mechanisms outlined in Sections 3.3 and 3.4.

3.3 Traversal Time and Passenger Count Predictions

In the first step of data processing, the system generates two predictions for each route at each time: (1) the time required for a bus to travel a section of a route, and (2) the number of passengers that will want to travel that section of the route. Historical values of either type (or approximations thereof) are easily computed from data from the buses: the time to travel from stop A to stop B is the difference between timestamps when the bus is at A and B, according to the GPS; the number of passengers is given by the beam sensors or camera feed while the bus is in this section. For any route section, the system's default predictions of these values are simply averages of historical values, bucketed by 10-minute-long intervals throughout the day and day of week (6 intervals per hour, 18 hours per day, 7 days each week, for 756 total buckets), and weighted with an exponential decay factor:

$$New\ Avg.\ Traversal\ Time = 0.9 \times Old\ Avg.\ Traversal\ Time + 0.1 \times Measured\ Traversal\ Time \quad (1)$$

$$New\ Avg.\ Passenger\ Count = 0.9 \times Old\ Avg.\ Passenger\ Count + 0.1 \times Measured\ Passenger\ Count \quad (2)$$

The rolling average is bucketed by time of day and week to account for natural variations in travel time caused by peak traffic hours (people going to and coming from work) and weekday/weekend differences in transportation. When a measurement straddles multiple intervals, it is considered in all of those intervals; when multiple measurements occur in one interval, they are averaged. Measurements are given small weight to dampen the effects of outliers, while past measurements are weighted exponentially less as time continues so that predictions will adapt to reflect permanent and seasonal shifts in traffic pattern changes. The weighting can be adjusted appropriately to strike a balance between quickly adapting to seasonal changes and maintaining good historical relevance. The current weighting gives a total weight of about 0.5 to data which is more than 6 weeks old, which we believe represents a good tradeoff between quick adaptation to seasonal changes and valuable historical data input. If the system measures an outlier that is still sufficiently extreme to distort future predictions, administrators can mark that time in the database to be excluded from rolling averages.

Administrators can also manually input predictions or copy historical usage patterns to use as a prediction, as for issues with other transit systems that have appeared before or recurring events such as the Red Sox game mentioned above. To prevent against human error, the system will compare inputted predictions against its default predictions and warn administrators if they differ drastically.

In exceptional cases (e.g. MBTA trains breaking down and buses being used to cover for routes), the system may need to create predictions for bus routes that are not normally traversed and do not have a recent rolling average. In this case the system can copy a prediction from nearby roads with similar characteristics, or failing that, a simple global average. It is expected that such cases will be rare enough that administrators can scrutinize these exceptional predictions and override them as needed.

3.4 Bus Dispatch Algorithm

Equipped with these predictions, the system will be able to automatically dispatch buses to attempt to meet the MBTA's service targets, and predict stop times for buses. For each route, Gigabus tentatively plans to hit all service targets under predicted passenger counts and traffic patterns, then calculates how many buses would be needed and adjusts its plans if it does not have enough buses.

More precisely, for each route, from the current moment forward until the end of the day, Gigabus will predict the number of expected passengers along each route component during each 10-minute block through the method described in Section 3.3. Then, using a simple model where passengers wishing to traverse each component are assumed to arrive uniformly throughout each 10-minute-long block, Gigabus computes the first time at which the ratio of predicted passengers-to-seat ratio exceeds the current service target, which we call the "target time". Finally, the system computes the soonest bus dispatch time required to traverse all components of the route before each component's target time.

In addition, Gigabus expects a list of advertised bus arrival frequencies for routes depending on the time of day and day of week (in the same 756 buckets) that will meet or exceed service targets during most periods of normal traffic and that do not require all buses of the MBTA fleet. These frequencies are intentionally underspecified in our system, as they may be chosen to meet design goals beyond simply hitting service targets — for example, it may be desirable to choose frequencies that are round numbers and stay consistent throughout the day, because passengers can easily remember them. If it is desirable for each stop to be serviced once every twenty minutes, for instance, this list of advertised frequencies will simply be twenty minutes for every route. Since they can be set once and need only be changed rarely, if at all, our system leaves it to the administrators to choose them.

For each route, Gigabus will tentatively plan to dispatch a bus to traverse that route to satisfy the passenger-to-seat ratio requirement or to match the advertised frequency, whichever is sooner, and it will repeat this calculation to create a tentative plan to dispatch buses to meet all service targets for the rest of the day. Note that this tentative plan is formed without regard to how many buses are actually available. While this algorithm is somewhat complex, it is a calculation using prediction data that fits comfortably in the reliable server's RAM, so it can be performed rapidly throughout the day, as well as instantaneously in response to traffic updates, bus failures, or overrides inputted by administrators. Because the calculation can be performed quickly in response to adjusting conditions, Gigabus can accommodate bus failures and other unexpected scenarios as quickly as buses can be re-routed from route end points or the MBTA warehouse.

If the tentative plan is feasible, Gigabus simply dispatches buses accordingly. Thus, during normal operation, Gigabus will simply dispatch buses at the advertised frequency if this would not lead to a passenger-to-seat ratio exceeding the service target of 1.4. In cases where the system predicts that the default bus schedule is not adequate for meeting service targets but the MBTA has sufficiently many idle buses, Gigabus will simply dispatch additional buses more frequently along that route.

If, however, Gigabus calculates that traffic or demand is so extreme that the tentative plan requires more buses than the MBTA's available fleet — which might occur due to, for instance, a Red Sox game combined with failure of the MBTA Green Line — the system will reduce its global service target, propose a new plan, and recalculate the number of buses needed until the reduced service target can be satisfied. The best achievable service target can be found quickly by binary searching on possible service targets. In high-demand or high-traffic scenarios, this strategy makes proportional sacrifices so that every route is treated fairly based on its demand or traffic. Thus, during a Red Sox game, Gigabus will holistically dispatch more buses along routes that take people to the game without abandoning other routes.

In cases where construction renders parts of routes or stops unserviceable, Gigabus relies on administrators to input this information, choose alternate routes and stops, and enter predictions for passenger counts across these stops. At that point it can run the bus dispatch algorithm as usual. This is expected to be a rare event that administrators can handle, and the system will assist administrators by providing historical data as an easily queryable database, as described in Section 3.2. If a stop is unserviceable, the administrator might, for instance, input a prediction that passengers who travel from that stop will travel from the previous or next stop instead.

Finally, equipped with bus dispatch times and traversal time predictions, Gigabus can predict the times at which each stop is visited, which it publishes to its passengers, integrating with feedback mechanisms outlined below.

4. Passenger Feedback

The MBTA is offered as a service to Boston area residents to help them get where they want to go, easily and comfortably. Therefore, the rider experience should be central to planning and decision making. We utilize on-board sensors and analyze troves of data to understand how well we are serving our constituents, but no one knows the rider experience better than the rider. Our system attempts to maximize feedback from riders whenever they are willing to share it. An online form will be advertised on the bus and at bus stops, linked by URLs and QR codes. This form would provide a convenient way for passengers to share their opinions and comments, and take the guess work out of trying to infer their experience. Passengers can tell administrators when a bus is late or broken, or if they couldn't find a seat because the bus was too full. This helps us cover edge cases where the on-board communication equipment breaks, rendering the driver unable to provide updates to headquarters.

As a secondary source of rider feedback, an MBTA mobile app would be a win-win for the riders and for the MBTA. Riders are used to checking the weather or the news with their phone, and it would be helpful

if they could check the status of their bus too. Providing a channel to alert riders to delays and service outages would help alleviate some of the associated problems, since this rider could choose to take an alternate route. If enough riders choose to take alternate routes, we could avoid the problem of accumulating passengers that is caused by a delay in service and results in overcrowded buses.

We want to utilize passengers' mobile devices to improve their experience by communicating important route and schedule information to them. We envision using multiple mediums for outbound communications, such as tweets, app notifications, and website updates. For instance, when a bus is rerouted, we would notify riders who have subscribed to this bus route's activity feed. To illustrate how this would work, consider the case of Ben, a hypothetical MBTA rider. Every weekday morning, Ben takes the 1 bus leaving 84 Massachusetts Avenue between 8 and 9 am to get to work. As such, Ben can subscribe to all the 1 bus schedules in this time range to be notified of any delays or schedule changes. This way, any abnormal schedule situations can be actively brought to Ben's attention as they occur, without the need for Ben to seek out the information. This subscription system would work for non-recurring subscriptions as well. If Ben is meeting his friends for dinner at 8 pm in Harvard Square, Ben can make a one-time subscription to all the 1 buses leaving 77 Massachusetts Avenue between 7 and 8 pm, so he can be aware of any delays that might arise and meet his friends on time.

One important point to keep in mind is a major design goal of the MBTA bus system: accessibility. Just as we want to help riders from every demographic get from point A to point B, we want to design all aspects of our system around this idea of inclusion. Thus, special considerations are required for passengers who do not own smartphones. For these passengers, we may not be able to connect with them through a smartphone app, but we would still like to send them important updates whenever possible. Thus, Gigabus features an SMS conduit to both our system alerts and our feedback forms (providing a number that riders can text to provide feedback). To protect the privacy of our riders, all incoming rider feedback with be stripped of personally identifiable information such as phone number or email address and stored anonymously. Privacy considerations are discussed further in Section 5.4.

5. Evaluation

5.1 Data Transfer Metrics

One of the core features of Gigabus is the data transfer prioritization scheme. We have designed the priority scheme for the goals of timely and reliable transmission of low-bandwidth data as well as maximum throughput of high-bandwidth data. We have also designed the priority scheme to be tunable to achieve maximum performance and to allow for expansion in the radio network or number of buses. Because Gigabus can operate entirely off of low-bandwidth data, the number of buses it supports can be expanded by more than an order of magnitude before being severely crippled by the radio network — this expansion will cause a decrease in passenger count accuracy, but the system will remain fully functional. Because we want to prioritize low-bandwidth data and use high-bandwidth data to fill the rest of the available bandwidth, Gigabus' data transfer performance is measured in two ways — data age for low-bandwidth data sources (the difference between the data receive timestamp and the data creation timestamp), and the total data throughput.

We want to keep both the average and 95th percentile data ages low. Our internal service targets for these metrics are an average data age of less than 1 second, and a 95th percentile data age of less than 5 seconds. This will ensure that bus tracking data is available for customers in a timely manner.

We also want to keep the average bandwidth usage high, since this corresponds to more accurate passenger counts and allows for more precise control over the distribution of buses to meet our passenger-to-seat ratio service target. This also increases accuracy of historical data, and will let Gigabus make better recommendations for bus distribution in the future. The maximum possible aggregate bandwidth is 160 Mbps. We are confident that Gigabus can achieve a high saturation level, so administrators can tune the high-bandwidth parameters aggressively in an attempt to reach 160 Mbps until data age metrics exceed internal targets.

In the case that the data age or bandwidth usage is unacceptable, Gigabus will alert administrators and suggest a method of tuning as follows:

- If the average data age is too high, the initial low-bandwidth priority can be set higher.
- If 95th percentile data age is too high, the exponential curve on low-bandwidth data can be adjusted to be steeper.
- If the bandwidth usage is too low, the high-bandwidth priority can be increased.

5.2 Data Storage Metrics

Based on our throughput estimations, we can compute that the low-bandwidth raw data received by the reliable server per day will roughly be 70 MB of payment data, 160 MB of GPS data, and 80 MB of beam sensor data, for a total of at most 310 MB per day. Image data is not factored into these calculations, because it is discarded after analysis. As for the rolling averages used for predictions, there are 108 buckets per day and fewer than 6000 stops in the worst case (fewer than 30 for each of 177 routes, where we distinguish stops visited by inbound and outbound buses, plus some additional buffer for possible alternate stops). There are two kinds of predictions, passenger counts and traversal times, and each is simply a floating-point value for which the accuracy of a 32-bit float easily suffices. This constitutes 6 MB of data per day.

If we assume conservatively that database overhead will double the data size, we find that at most 750 MB of data is generated every day, which fits comfortably on the reliable server in addition to the static data sets, leaving ample space for queueing image data. If we replicate all the data by storing it on two other servers, we find that each 10 TB server allows the MBTA to store 36 years of data. In reality, data more than a few years old is likely to be useless and can simply be deleted or archived offsite.

5.3 Evaluating the Bus Dispatch Algorithm

Our system for scheduling and dispatching buses depends on moderately heavy administrator input: most importantly, they are responsible for predicting large spikes in demand and entering those spikes into the

system ahead of time or as soon as possible, although our archive attempts to assist with this by providing easy access to relevant data. Scaling this part of the system by hiring additional administrators is not hard.

Compared to other possible designs, our system does not respond as quickly to failures as possible, because it only changes the rate at which idle buses are dispatched from headquarters. In particular, it does not pull buses from the middle one route to service a different route, even if the buses are empty. We believe the additional complexity and unpredictability caused by pulling buses mid-route is not worth the potential gain in efficiency. Firstly, we provide arrival time predictions as service to passengers to make riding an MBTA bus more convenient, and in doing so, we enter into a sort of contract with our constituents. Although a bus may appear empty, a rider could be waiting at the next stop. Pulling this bus off its route would invalidate our published arrival estimates and break our agreement with our riders. Secondly, we believe passengers will be more understanding of a shortage of buses on visibly crowded routes, and more willing to consider alternate modes of transportation; whereas if we pull a bus from a likely less crowded route, passengers waiting for a bus on that route won't be able to easily see why their bus isn't coming, and will be more dissatisfied.

5.4 Privacy Concerns

Another important evaluation criteria for our data handling and storage design is user privacy. Gigabus will collect and store comprehensive sets of data captured from every bus' operations. Some of this data, such as CharlieCard taps or transfer information, may be personally identifiable. An example of a privacy concern might be that, in our datacenter we store logs of all recorded CharlieCard taps on any of the MBTA buses. While the CharlieCard data may itself not have any personally identifiable data on it, we may have payment information (which would likely include the payer's name) stored elsewhere on our servers. These two datasets in combination would reveal personally identifiable ridership patterns. In consideration of this, the data storage procedure of Gigabus strips all incoming data of personally identifiable fields. For instance, CharlieCard taps are only used to estimate the number of riders on a given bus — it is not important for us to know who this rider is or what other bus they may have transferred from. Thus, we can remove the serial number off an incoming CharlieCard tap at the server without sacrificing any of Gigabus' functionality. All of the data Gigabus processes can be fully anonymized, except the schedule subscription and mobile app login information, as we need to retain subscriptions' associated phone numbers to properly deliver updates.

On its servers, Gigabus encrypts data with standard public/private key encryption technology, adding a layer of protection against intrusion and exploitation by malicious actors. Our most sensitive data is of course our route alert subscription data, which we cannot completely anonymize for functionality purposes. In order to protect user privacy, this information must be encrypted. Other data on our servers can be encrypted if resources allow, but it is not imperative to encrypt it since it does not contain personally identifiable data.

6. Conclusion

The age of the MBTA bus system and its various components presents a challenge to maintaining its excellence. In this paper, we presented Gigabus, a system designed to improve MBTA service using existing infrastructure by prioritizing certain types of bus sensor data to maximize network utilization, combining historical data and administrator input to meet service targets, and implementing subscription and feedback mechanisms to communicate with passengers. Notably, we can leverage the technological resources of our riders to help circumvent some of our technical limitations.

Though we expect Gigabus to be a reliable, flexible, and responsive system, there is always room for improvement. One potential way to improve Gigabus is to reduce its dependency on administrator input. Administrator override should always be allowed, but more advanced algorithms or machine learning techniques could allow Gigabus to make more sensible default guesses for future travel times and passenger counts based on information it has received, easing the load on administrators. Another potential place for improvement is in the radio transmission scheme, where small data packets could be batched to decrease communication overhead.

Through Gigabus, the MBTA will be able to make better use of their existing hardware to provide a more reliable, accessible, and comfortable ride for all Bostonians.

Acknowledgements

We would like to thank the 6.033 staff for providing feedback on the paper.