FastTrack: An Optimized Transit Tracking System for the MBTA

A Group of 3 MIT Students in 6.033

Contents

1 Introduction

The Massachusetts Bay Transit Authority (MBTA) is seeking a design for a new real-time tracking system for its network of buses across the greater Boston area. This tracking system will enable the MBTA to collect data on several metrics by which it measures the success of its network, including availability, accessibility, reliability, and passenger comfort. The MBTA's current bus system successfully meets its first two goals, but has historically failed to meet its goals for reliability and comfort. The MBTA has recently upgraded its infrastructure and is looking to improve on the latter two metrics through a real-time tracking system. The system proposed in this document utilizes the MBTA's existing buses, network, and central servers to evaluate and improve on the MBTA's success in meeting its specific goals for these metrics.

FastTrack collects and processes bus and passenger data at scale, using this information to adapt to changing demand and traffic conditions. The design prioritizes system reliability, user experience, and modularity in a variety of common and uncommon scenarios. FastTrack aims to achieve the MBTA's specified goals for bus frequency, punctuality, and comfort to provide the best travel experience to the MBTA's passengers. It uses the existing modules of the MBTA—buses, bus routes, radio network, and servers—in order to collect data relevant to the MBTA's goals, detect and respond to high demand or route issues, and obtain useful feedback from MBTA passengers. This paper details the design of the FastTrack system by describing each of these functions and their implementations, and specifying how these decisions impact the overall bus tracking system and the MBTA as a whole.

2 System Overview

The FastTrack bus system has three main concrete goals: collecting data from buses about the MBTA's goals, addressing overcrowded bus routes and other route failures, and getting feedback from passengers about their bus experience. In order to accomplish these goals, FastTrack uses the MBTA's existing infrastructure: its buses, equipped with security cameras, door and payment sensors, and onboard computers; its ten-frequency radio network which communicates between the buses and the main warehouse, and its warehouse servers connected via a wired connection. These goals and modules are described in Figures 1 and 2 respectively.

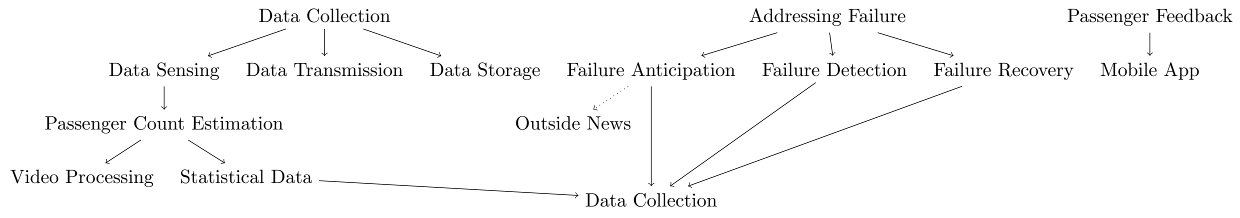

Figure 1: Functional Overview

2.1 Data Collection

The MBTA has several specific targets to meet regarding stop frequency, reliability, and comfort, and also wishes to track data about how many total passengers, transit-dependent passengers, and route transfers its system has per day. In order to evaluate these metrics, the system must be able to collect, store, and analyze various types of data from the buses. FastTrack collects five types of

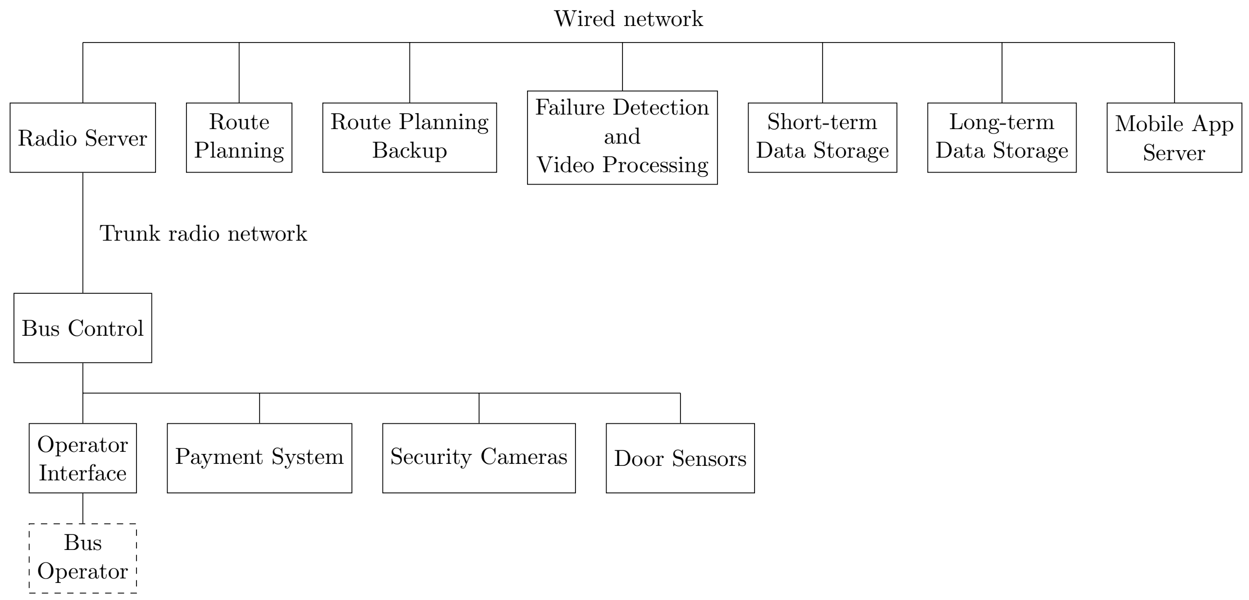

Figure 2: Module Overview

data from active buses: payment records, door sensor information, security camera photos, GPS data (and associated stop IDs), and timestamps. Payment counts are stored locally on the bus and combined with historical estimates to approximate how many passengers are on the bus at each stop. Estimated passenger counts and the other sensor data are transferred over the MBTA's trunked radio network to the radio server in the main warehouse, where the information is then distributed to the relevant servers for processing and storage. Bus data is sent to different servers depending on its purpose. Camera data is periodically sent to the failure-detection server, where it is analyzed to determine an exact count of the number of passengers on a bus. The failure-detection server then sends a message to the radio server to tell the bus its actual passenger count (to keep estimates from drifting too far), and also notifies the route planning server if a a route is overcrowded. GPS data and timestamps are stored in a database in the short-term data server as *stop records*, which record the bus ID, stop ID and time, and can be referenced later to calculate various metrics such as how often a stop was serviced and whether the stop was reached on time. Stop records are also sent to the mobile app server in order to keep track of the location of buses along each route.

2.2 Addressing Failures

In order to meet the MBTA's goals for service reliability and passenger comfort, the bus system must be able to detect when buses on a route are overcrowded and respond to both this situation and other types of route issues. To accomplish this, the failure-detection server keeps track of the number of passengers on each bus, and sends a message to the route planning server if too many buses on a route are over a certain threshold for comfort. When the route planning server receives this message, it uses the MBTA's static data sets to find nearby sources of buses (the warehouse and uncrowded routes), and reassigns buses from one of these to the overcrowded route. It then sends a message to the radio server, which passes on the assignment change to the affected buses. If a route needs to be temporarily changed due to construction or an emergency situation, the change is entered manually by an MBTA employee, and the route planning server notifies the radio server which notifies the affected buses. If a new route needs to be created, it is also entered manually, and buses are assigned to it in the same way as in an overcrowded route.

2.3 Passenger Feedback

The third major goal addressed by FastTrack is collecting and storing feedback from MBTA passengers. This system accomplishes this through the use of a mobile app, run on one of the warehouse servers. The app contains several features useful for MBTA passengers, including bus location tracking, schedules, and notifications of any unexpected route changes. The app also contains a simple rating form, which is the primary method of getting feedback from passengers about their level of satisfaction with their bus ride. The form allows passengers to rate their overall experience and add any other specific comments or complaints they have about their ride.

3 Design

3.1 Data Storage

Within the MBTA warehouse, there are seven servers, each with an internal storage of 10 TB, which have a wired connection with a throughput of 1 GB per second between them. Each server or pair of servers has a specific function and stores the relevant data locally when possible, in order to improve the modularity of the system and minimize data transfer between machines. All but one of the servers can fail for up to two minutes at a time, but the modularity of FastTrack's design and redundancy for important functions minimize the negative impact of any particular server failing. The functions of the servers are assigned as follows.

1. The radio server is connected to the radio transmitter/receiver pair and handles communication between active buses and the servers in the warehouse. It receives sensor data from buses and distributes it to the appropriate servers, and sends route change alerts to buses when prompted by another server to do so. The radio server is crucial for communication between the buses and the other servers, and needs maximal uptime since most computation at the warehouse requires data received from buses. Additionally, bus operators must always be able to contact the warehouse in the case of an emergency. To address this need, FastTrack uses the MBTA's reliable hardware for the radio server, since it is the one server that the others must depend on. Additionally, the radio server also stores the bus metadata and operator datasets since it needs to refer to these during the course of its operation.

2. The route planning server handles scenarios where bus routes must be added or modified, or certain routes are overcrowded and need more buses to reach the MBTA's comfort goals. This server stores the MBTA's static data sets – census data, bus meta-data, route and schedule data, alternate stop data, and operator data – all of which are useful for addressing route or capacity failures. This data is stored locally in order to improve the modularity of the system.

3. The backup route planning server is identical to the route planning server and is able to take over its functionality in case the first server fails. One of FastTrack's major priorities is being able to respond to route and overcrowding issues quickly, so the system includes some redundancy in the task of addressing failures. Recalculating routes and reassigning buses are both very time-sensitive tasks, so having a backup server avoids any potential delays caused by the first server failing and improves the average response time of the system.

4. The failure detection server processes payment, door sensor, and video data from buses in order to determine bus occupancy and alerts the route planning servers if it detects that a bus route is overcrowded.

5. The short-term data server stores a database of per-route information on frequency of service, bus punctuality, estimated passenger counts, and number of transfers over the past day. This server also calculates and stores the estimated likelihood that a passenger will remain on the bus at each stop.

6. The long-term data server stores a database of records of specific failures to meet comfort or reliability goals over the previous two weeks, as well as daily records for total passenger count, low-income passenger count, and transfer rate. It also stores daily averages over all stops of the short-term data listed above, as well as long-term averages over time for each stop.

7. The mobile app server runs the MBTA mobile app and stores passenger feedback and recent bus location data.

The server subsystem is designed to be highly modular, so each server is responsible for one aspect of the bus system. If one server fails, other functions of the system are not affected, since servers work on independent tasks and store all relevant data locally. The communication server is an exception to this modularity since only one server can be connected to the radio network and all information transferred to or from buses must rely on it. However, this server uses the reliable hardware and is guaranteed not to experience any failures, so it is acceptable for other servers to rely on it. The modularity also ensures that the rate of data transfer between servers in real time is significantly less than the 1 GB/s wired connection between them, so throughput is not a major concern.

3.2 Data Collection and Distribution

FastTrack collects and stores a variety of data from the buses. Each bus contains a bus control, a computer with 128 MB of storage, which centrally manages the bus's sensors, radio connection, and operator interface. Each bus control must measure and transmit data on location, timing, and the number of passengers to the central server. This data is used to detect failures and assess reliability and comfort standards, taking into account the potential inaccuracy of the collected data. Furthermore, the limited bandwidth on the trunk radio limits potential communication between the bus control and server, meaning that much of the communication takes place overnight.

3.2.1 Network protocol

The servers and buses communicate to each other via a trunked radio network. In order for these modules to interoperate smoothly, they use a common protocol to pass messages across this channel. The protocol both ensures that messages are understood by the correct recipient, and disambiguates between different types of messages.

The radio network has ten frequencies for communication, each of which has a capacity of 16 Mbit/s, in addition to the radio server's dedicated transmission frequency. The server transmits all of its messages over this frequency; all bus controls listen on this dedicated frequency but do not transmit to it. In order for a bus control to transmit a message, it requests a frequency from the trunk controller, and then transmits the message over the assigned frequency. Each message has the format described in Figure 3.

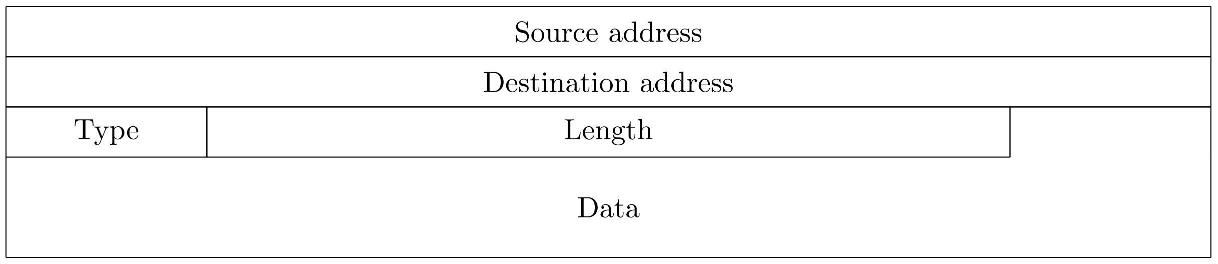

Figure 3: Radio message format

0. Heartbeat
1. Route update
2. Video data request
3. Video data
4. Video passenger count
5. Statistical information
6. Voice communication request
7. Voice communication data
8. Voice communication end
9. Error message

Figure 4: Radio message types

The addresses are unique 48-bit numbers, each of which identifies either a bus, the radio server, or "broadcast" (all of the buses). "Type" is an 8-bit number identifying the type of data being sent. The 11 possible types are enumerated in Figure 4; the format allows the protocol to be extended with up to 256 possible message types.

"Length" is a 32-bit number representing the length of the data in bytes. Since no single message should even approach 2^{32} bytes (that is, 4 GiB) in size, this is sufficient. The remainder of the message is the raw data, whose exact format depends on the type of data being sent. Voice communication data is treated specially in that the total length of the voice communication may be unknown when the transmission begins. Therefore, the "Length" field is ignored for voice communication, and the special data type "Voice communication end" is used to signal that the voice communication is ended.

3.2.2 Location data

At the beginning of each day, each bus receives information from the radio server about the day's route via the in-warehouse wireless network. In addition to the usual route information, this also includes an estimate of the rate at which passengers tend to remain on the bus at each stop, calculated by aggregating past data.

During normal bus operation, the bus control displays the name of the next three upcoming stops on the bus-operator interface. Meanwhile, it polls the GPS sensor to determine the bus's location. When the bus reaches a stop, the bus control records the ID of the stop and the current timestamp in memory, and updates the interface to include the next three stops.

3.2.3 Passenger data

The bus control attempts to maintain an estimate of the number of passengers riding the bus. At each stop, this count is scaled down by the estimated passenger retention rate for the stop, and is increased by one for each payment reported by the payment system. The bus control keeps track of the amount of time during which the estimated number of passengers is over a certain threshold for comfort, which we set at 1.2 times the number of seats. This threshold is intentionally below the MBTA's comfort requirements ratio, prioritizing customer experience over computation resources. After each stop, the bus control sends a "heartbeat" message to the server detailing the stop ID, current time, aggregate transfer data and current estimate for passenger count.

Video data is used to improve the accuracy of the passenger count tracking system. Keeping the passenger count accurate improves reliability, reducing the chances of false positives and negatives in the failure detection system. However, the video data must be processed on the server rather than on the bus control itself, owing to the cost of the video processing algorithm. We assume that five frames of video data suffice to calculate the passenger count with 95–98% accuracy.

To avoid overloading the radio network, the radio server maintains a priority queue of the buses, and notifies them to send video data. The server notifies five buses at a time, so that they can simultaneously transmit on different frequencies. Upon receiving this notification, the bus captures and transmits five frames of video data from the security cameras. The failure-detection server processes these and determines the number of passengers on the bus, sending this more accurate estimate back over the radio. The bus control updates its estimated passenger count to this value and stores a record of the accurate count. If the number of passengers in the video is greater than the comfort threshold, the failure-detection server marks the bus internally as overcrowded. Once all five buses' video data have been processed, the server notifies the next highest set of five buses in the priority queue. Waiting until the processing is complete before notifying the next set of buses ensures that neither the network nor the video processing server is overloaded, improving FastTrack's reliability. The server also maintains the rule that a bus not marked as overcrowded or potentially overcrowded may not be notified for video data collection more than once per 20 minutes. This decreases the potential for passengers' locations to be tracked using the collected video data, without compromising any accuracy for buses known to be crowded.

Any time a bus not already known to be overcrowded sends a heartbeat message indicating that its estimated passenger count is over the comfort threshold, the server marks that bus as potentially overcrowded, and increases that bus's priority in the queue so that it will be accurately counted sooner. To ensure all buses are counted accurately, this reprioritization is only permitted to occur once for a given bus before it must wait through the entire queue again.

To improve the estimates of passenger retention rate, the bus control also captures and stores video data one stop after it transmits video data to the server. This video data is stored in the bus control, along with the ID of the previous stop, to be uploaded and analyzed once the bus returns to the warehouse. To ensure that the amount of data stored in this way is reasonably bounded, the bus control does not perform this storage more than once per 20 minutes. At the end of the day, the bus control sends this video data to the radio server over the in-warehouse network. Since the radio, route planning, and route planning backup servers are otherwise idle at night, FastTrack parallelizes the task of running the video processing algorithms across these three servers as well as the failure-detection server. These servers analyze the video data and send the passenger count to the short-term data storage server, which determines the difference in passenger count from before and after the stop. This data is averaged over time to improve estimates and is sent to the bus controls every day along with the route information. Other statistics are also transmitted to the server at the end of day and aggregated: the time at which each stop was reached, the total number of passengers, the total number of transfers, and the total amount of time the bus estimated it was over the comfort threshold. These statistics are used to calculate the MBTA's metrics for fleet monitoring.

3.3 Route Planning

3.3.1 Initial route assignments

As part of its data storage, the MBTA servers maintain historical data on the usage of different routes. This data provides a baseline for the allocation of buses to routes that happens each day. In addition to this historical data, an employee for the MBTA will be in charge of adding descriptions of high traffic events to the bus system, such as a convention or Red Sox game. Thus, any planned events which create significant congestion can be approached as a normal day, though more buses will be necessary.

3.3.2 Failure detection

As the buses go about their routes, they keep a running estimate of the number of passengers onboard, which is updated based on information from the central server. If a majority of the buses on a route are marked as overcrowded, then the failure-detection server notifies the route planning server of a need to add buses to the route. In the cases of a route becoming unavailable or a new route being added, MBTA administrators will be contacted directly via a process external to FastTrack. In both cases, the administrator will add or modify routes to the network as necessary. In all cases, the route planning server is sent the IDs of the routes being modified, as well as the number of buses that will need to be mobilized.

We choose to base failure detection on passenger counts rather than the frequency with which buses service stops. This is because we aim to prioritize passenger comfort over the reliability of bus service. In addition, a high passenger count on a route tends to indicate that there are not enough buses on that route, where adding more buses to the route is likely to alleviate the problem. On the other hand, bus lateness without a high passenger count is often a symptom of heavy traffic along the route. This is unlikely to be helped, and could even be made worse, by the addition of more buses. Thus FastTrack detects failures based on high passenger counts, rather than frequency of service.

3.3.3 Failure recovery

Whenever there is unexpected demand, the failure-detection server is notified of the route and the number of extra buses it requires. Once the system detects a route with insufficient buses, the server adds a bus to the route either by diverting a bus from another route or by sending a bus straight from the warehouse. If the route is close to the warehouse itself, buses are dispatched directly from the warehouse. Otherwise, the server moves a bus from a nearby route. The server looks up nearby routes and buses operating on them. Each route, along with a list of the buses currently on that route, stores the minimum number of necessary buses for normal operation. Of the routes which have a number of buses above the minimum, the server selects the route with the greatest difference between its total passenger capacity and total estimated passenger count. The radio server sends the bus closest to the endpoint or midpoint of the selected route, as recorded by the bus heartbeat messages, its new itinerary and route assignment, which are communicated to the bus operator by the bus control once the bus reaches the endpoint or midpoint. The route planning server then updates its bus assignment data set to indicate that this change has occurred.

Another type of failure occurs when a route needs to be modified in some way because certain stops or streets are inaccessible due to construction or an emergency situation. In this case, an MBTA system administrator uses the census and alternate stop data stored on the servers to calculate a new route. Once this is calculated, the route is modified on the server, and both the buses on the relevant route and the mobile app server are notified of this change.

3.4 Mobile App

The MBTA mobile app serves as an important method of communication between the MBTA and its regular passengers, and supplements the MBTA's existing website. This app provides its users with several useful features. First, the app contains information about each of the bus routes, including their stop locations and schedule information. Additionally, the app provides its users with an estimate of when the closest bus will service a given stop, calculated based on bus location data and any bus reassignments made by the route-planning server. This app also alerts passengers of any unexpected route changes, and provides a feedback form through which passengers can submit ride ratings and specific complaints. The form is completely anonymous, and collects information about the route that was taken, the time of day, an overall ride rating out of five stars, and a text box for users to include specific comments or complaints.

4 Evaluation

4.1 Use Cases

The MBTA bus system needs to operate effectively under three major use cases: normal use, high demand scenarios, and unexpected route failures. In any situation, if customers have complaints about a recent ride, this can be addressed using the convenient feedback form and corroborated by the recent failure records in the storage server.

4.1.1 Normal conditions

Under normal conditions, the bus system meets all of its goals, as there are not enough passengers to exceed bus capacity, and traffic is light enough that buses reach all of their assigned stops in a timely fashion. Bus operators service their standard, most familiar route, and no failure correction is necessary.

4.1.2 High demand

During periods of high demand, buses will be reallocated from the warehouse or less crowded routes to more crowded ones in order to maximize the average comfort of the passengers. Since only nearby buses are chosen to be rerouted, this system responds to failures quickly and keep bus operators near areas they know. A historic limiting factor has been a lack of idle buses, which FastTrack alleviates by searching for buses from nearby routes rather than immediately using an idle bus.

4.1.3 Unexpected route issues

In the case of unexpected route issues, MBTA administrators use the census and alternate stop data, along with their experience maintaining the bus system, in order to calculate optimal route changes. When a new route is designed, the system handles allocating buses to it in the same way as during a high-demand situation, meaning the new route becomes operational very quickly.

4.1.4 Historical Data

In the case that the MBTA needs to investigate a claim that a specific bus was extremely late, it can query the records of specific failures dating back two weeks on the long-term data storage server. Thus, it is possible to address customer complaints; and in general these should be unnecessary for the MBTA to know that something went wrong. FastTrack keeps track of failures, allowing the MBTA system to be easily audited.

4.2 MBTA Requirements

We now evaluate FastTrack's ability to meet the MBTA requirements for fleet monitoring, passenger feedback, and failure response.

FastTrack is directly monitors the metrics required by the MBTA, including frequency of service, bus punctuality, per-bus passenger count, and total passenger count both overall and in low-income areas. It can also calculate route coverage directly from the census and route information databases. Finally, FastTrack monitors the number of passengers who use bus transfers, provided that the payment system detects these transfers (that is, they are made using Charlie Cards). FastTrack calculates and stores these metrics aggregated both over time and over the number of stops, allowing the MBTA to analyze the data for both geographically and physically correlated trends.

The mobile app recommended as part of FastTrack allows for passenger feedback to be collected and stored in a controlled and anonymous fashion, without increasing the complexity of the main FastTrack system. This strikes a balance between the additional complexity of using an integrated application to communicate through the bus control, and the difficulty of integrating, anonymizing, and storing data from uncontrolled backchannels.

We also evaluate FastTrack against MBTA's service targets for reliability and comfort. We do not evaluate route coverage, since it is outside FastTrack's scope. Because FastTrack efficiently redistributes buses from nearby routes in the case of failure, we expect that the service targets will be met unless FastTrack is unable to to detect a failure. This can occur if the passenger counts are underestimated. However, because the passengers are counted to within 95–98% accuracy at least once every 20 minutes, this provides an upper bound on the amount of time it can take before a failure is detected. Another way the service targets might not be met is if there are not enough buses to alleviate all failures. In this case, FastTrack still redistributes available buses among the routes as best it can. Since the design of FastTrack prioritizes comfort and reliability, these service targets should be attained in most cases.

4.3 Data Management

FastTrack uses large amounts of data in various ways. We evaluate FastTrack's data management to determine the feasibility with which its design stores these datasets, both in transit and long-term.

The static data-sets stored on the route planning servers (and replicated somewhat on the radio server) comprise 820 MB in total; this is negligible compared to the 10 TB of storage available to each server.

The data stored on the short-term data server in the form of stop records includes information on the time at which each bus serviced each stop over the last day, as well as the estimated number of passengers at that stop. It also includes the number of transfers and the percentage of passengers who remained on the bus at each stop. Since the route and schedule dataset also contains comparable amounts of information for each stop, we assume that this dataset is comparable to the route and schedule dataset (100 MB) in size. As before, this is negligible compared to the amount of storage available.

The long-term data server stores records of specific failures to meet comfort or reliability goals. It also stores daily records of statistics aggregated over all stops, as well as statistics for each stop aggregated over time. The records of failures and per-stop aggregate statistics are constant and negligible in size. Since the server stores a very small amount of additional data each day, it should be able to practically store this data for years without requiring additional space. The data stored by the mobile app server is similarly negligible in size.

The amount of data stored by each bus control, on the other hand, requires more consideration. We can expect that each route contains about 40 stops, and that each bus makes about 360 stops (about one stop every 3 minutes) per day. For each individual stop on a route, the bus must store a stop ID, the stop's location, the stop's description, and the times at which the bus is scheduled to service the stop. Assuming that each of these can be stored in a 64-bit number, this requires only 3.84 kB of data. In addition to this data, every 20 minutes the bus control may store five frames (240 kB) of video data to be analyzed later. Over the course of a day, this requires about 64.8 MB, which is within the bounds of the bus control's storage capacity, even if video data is queried more often.

We thus find that the amount of data stored by the servers is easily within the amount of storage available, leaving plenty of room either to scale to larger scenarios or to collect greater amounts of data. However, FastTrack's design does not require the storage of large amounts of data. In particular, FastTrack avoids storing video data for longer than is required to be processed for privacy reasons.

4.4 Scalability

In evaluating how FastTrack scales, the primary notions of scalability to consider are the number of buses necessary as well as the number of routes and the distance between them. In both notions of scaling, FastTrack keeps up with an increase in system size without requiring substantially more resources.

First, we will discuss the scenario where more buses are added to the same network to meet demand. As described in the previous section, the amount of data stored is well within the current capacity of FastTrack's servers, and adding more buses will not strain data storage. In terms of processing data, the major bottleneck is the treatment of visual data. However, the analysis of FastTrack's network load again shows that the radio network will not be running at full capacity, meaning that it can accommodate the traffic caused by new buses. Finally, the biggest concern when adding more buses is the time for video processing, both in real time during the day and overnight. Here, the system suffers the greatest loss in performance, and will likely mean that buses take longer between getting accurate passenger counts. However, this can also be partially alleviated by assigning more servers to the task of video processing, which would allow FastTrack to scale well when adding buses to meet overall higher demand.

Another important question of scalability is the ability to scale to different network topologies present in different cities, both when routes get sparser and further apart and when condensed into a small physical area. Here, the main concern will be the effectiveness of failure recovery, as an important part of the protocol is the transfer of buses towards busier routes. Failure recovery is composed of an algorithm that determines which bus will be moved, followed by that bus physically moving from one route to another. However, the algorithm given iterates over nearby routes and calculates the extra capacity on each route, a simple calculation. So, running the algorithm should not take much time regardless of network size, meaning the speed at which failure recovery occurs is controlled by the time it takes for a bus to move between routes. This means that having more routes closer together will improve the operation of FastTrack, as moving buses will be much easier, Additionally, it will also be easier to find the bus to move, as a denser topology will mean that routes have more neighbors and thus more chances to find a bus to move. For sparser networks, FastTrack scales considerably more poorly, as route changes will take longer to pull of and there will be less opportunity for buses to move to the routes where they are most needed.

4.5 Network and Data Processing

In order for FastTrack to function effectively, the buses and servers must communicate to each other with sufficient frequency both to keep the servers informed of the status of the buses, and to ensure that the buses' passenger count estimates are accurate. Therefore, we numerically estimate the frequency with which these communications can occur, and evaluate the ability of the servers to process this data in the amount of time required. We compute these estimates in the worst case, assuming that all 1036 of the buses are actively collecting as much data as FastTrack's design allows.

The radio network is capable of handling up to 16 Mbit/s on each of its frequencies, with 10 ms of latency. We assume that the time it takes for a bus to be assigned a frequency is negligible when one is available. Furthermore, most of the types of message data are negligibly small. For instance, a single heartbeat message from the bus to the server contains 136 bits of protocol overhead, a stop ID (64 bits), timestamp (64 bits), and passenger count estimate (at most 8 bits). Even with all 1036 buses attempting to send a heartbeat message at once, this would require only 1.8% of the available network capacity per second on a single frequency. Similarly, voice communication does not require a significant amount of bandwidth; furthermore it is required only in exceptional situations like emergencies. We can safely assume that the radio network can always accommodate these small messages using only a couple frequencies, so we will focus only on the messages which contain large amounts of data. These are the video data transmissions.

Each round of video collection consists of five buses each transferring five frames of video data to the radio server in parallel, which forwards the video data over the 1 Gbit/s wired network to the video processing server to be analyzed; the resulting passenger counts are then transmitted back to the buses. We also assume that the video processing algorithm used on the server requires 0.1 s per frame. The total time required to transmit the video data, process it, and return the passenger counts is about 3.17 s. We also note that only 6 MB of video data is ever held in memory either by the radio server or the video processing server during this process.

With 1036 buses, we can thus expect that each bus may receive an updated passenger count approximately every 11 minutes, while using only 5 out of the 10 available frequencies. For buses which are known to be overcrowded or potentially overcrowded, video data collection will occur at this rate. However, we intentionally reduce this frequency to once per 20 minutes for other buses; this is for privacy reasons to be discussed in the next section. In any case, it is clear that FastTrack's network is capable of handling the required amounts of data transmission.

We also need to consider the time taken to process the video data stored on the bus controls at the end of each day. Over the course of the day, each of the buses captures and stores about 270 frames, or 64.8 MB. This data is transferred over the 54 Mbit/s in-warehouse wireless network.

Assuming that four buses can simultaneously transfer data via this network to the four servers performing video processing, the data transfer takes about 42 minutes, while processing the video data itself takes approximately 2 hours in total. Thus this data processing takes place in well under the 6 hours of nighttime available.

4.6 Security

To evaluate the security of FastTrack, we first consider how much information someone listening in to the radio network could obtain. Much of the traffic on the network consists of location data for the buses, which is publicly available and thus has few consequences for security. Under normal operation, the sensitive data sent along the network consists of passenger counts and video data to be processed. The former piece of information is unlikely to be useful, as in general it will not be perfectly accurate and makes no reference to which individuals are on the bus. Finally, the video data is a potential point of abuse, but the choice to limit video data transfer to once every 20 minutes means that it is not possible to follow someone from bus to bus, as it will be essentially random when the data is sent. Because the video data is only sent sparingly, someone listening in to the radio network will not be able to get much information about when individual passengers get on or off a bus.

Another possible breach of security is an MBTA employee with access to the servers. However, this employee will not have much more data than someone listening to the radio network. All of the stored data is completely anonymized, with reference only to the number of people on or leaving a bus. With video data, it is easier to find when someone gets off, as they have access to the video data before and after a stop. While transfers are tracked, individuals who are transferring are not tracked, but rather an aggregate count, meaning that the employee has no information about specific passenger transfers. Overall, FastTrack is secure, only making a small sacrifice to meet MBTA performance targets by sending video data.

5 Conclusion

FastTrack is designed to meet the requirements and constraints inherent to the problem of providing real-time data collection, failure detection, and failure response for the MBTA bus network. By combining real-time data from the payment system and security cameras with historical data on passenger behavior, FastTrack calculates accurate passenger count estimates without exceeding the capabilities of the existing network infrastructure. It detects failures of high demand by analyzing these passenger counts and applying heuristics to determine when action is required, and responds to these failures by moving buses onto the overloaded route, either from the idle pool or from nearby routes as appropriate. In doing so, FastTrack improves reliability and user experience for the passengers while remaining as scalable and modular as possible. By fulfilling these design goals, FastTrack succeeds in fulfilling the MBTA's bus system goals under all conditions.

6 Acknowledgements

We would like to thank the entire 6.033 staff for their help in improving our knowledge of systems and helping us write this paper. Additionally, we would like to specifically thank Karen Sollins, our recitation instructor, Emily Zhang, our TA, and Michael Schandorf, our tutorial instructor for their instruction and help throughout the semester. Finally, we thank Katrina LaCurts for teaching the class and designing this project.