6.033 Spring 2017 Design Project

See also: DP FAQ *(last update: 3/24)*, DP Errata *(last update: 4/7)*

Due Dates and Deliverables

There are four deliverables for this design project.

1. A preliminary report of approximately 2000 words, due on March 24, 2017 at 5:00pm (see also: preliminary report writing guidelines, including a description of the cover memo).
2. A brief oral presentation given to your recitation instructor, to be scheduled with your recitation instructor, for some time between April 19, 2017 and April 25, 2017. The oral presentation will assess your progress and provide some feedback prior to your final report submission.
3. A design report of approximately 5000 words, due on May 8, 2017 at **11:59pm**.
4. A peer review of approximately 500 words, due May 12, 2017, at 5:00pm.

Each deliverable will have specific guidelines, which will be linked above.

The preliminary report, final report, and peer review should be submitted via the submission site on the 6.033 website. As with real-life system designs, the 6.033 design project is under-specified, and it is your job to complete the specification in a sensible way given the stated requirements of the project. As with designs in practice, the specifications often need some adjustment as the design is fleshed out. Moreover, requirements will likely be added or modified as time goes on. We recommend that you start early so that you can evolve your design over time. A good design is likely to take more than just a few days to develop. A good design will avoid unnecessary complexity and be as modular as possible, to enable it to evolve to changing requirements.

Large systems are never built by a single person. To that end, you will be working in teams of three for this project. Part of the project is learning how to work productively on a long-term team project. All three people on a team **must** be in the same tutorial.

Note that although this is a team project, some of the deliverables may have individual components. See the individual assignment links for more information.

Late submission grading policy: If you submit any deliverable late, we will penalize you one letter grade per 48 hours, starting from the time of the deadline. For example, if you submit the report anywhere from 1 minute to 48 hours late, and your report would have otherwise received a grade of A, you will receive a B; if you submitted 49 hours late, you will receive a C.

As stated on the syllabus, you must complete each design project component to pass 6.033.

1 Introduction

The MBTA provides public transportation for the Greater Boston area via a network of buses, subways, commuter rails, and boats. For this project, we're going to consider only the buses.

To be a successful public transit system, the MBTA must meet various requirements for:

- **Availability.** Buses should operate during a reasonable part of the day—known as the *span of service*—and at a reasonable frequency. The bus network must also cover the entire Greater Boston area.
- **Accessibility.** Buses should be accessible to customers with disabilities; they must be ADA-compliant.[1]
- **Reliability.** Buses should depart on time, and maintain the timetable for scheduled stops.
- **Comfort.** Buses should not be too crowded.

The MBTA has already designed their bus network to meet the first two requirements: they have data on the population of the metro area and have chosen the span of service and routes to cover the area, and the buses are equipped with the necessary ADA-compliant devices.

However, assessing whether the bus network is meeting requirements for reliability and comfort requires real-time tracking, as does determining how the system will respond to failures.

Because the MBTA has recently been able to upgrade some of their infrastructure, they are soliciting design proposals for a new real-time tracking system. Your primary job in this project will be to design such a system, while also enabling some additional goals.

- Your system will allow the MBTA to collect data from buses so that it can evaluate its bus network on a variety of metrics (specified in Section **??**).
- Your system will allow the MBTA to react in a timely manner to unexpected events, such as route failures or unusually high demand.
- You will consider the implications of your system on the experience of the MBTA's passengers, and recommend a mechanism by which the MBTA can collect passenger feedback.

In designing this system, you will find that there are many constraints. The sensors on the buses, the communication network among the buses, and the capabilities of the server all place constraints on the amount of data that can be processed in a timely manner. Additionally, the MBTA is constrained by cost: they cannot hire new operators, or purchase new buses. The system will need to work well using the resources it currently has.

2 Existing Infrastructure

2.1 The MBTA Bus Fleet

The MBTA operates a bus fleet of 1036 buses on 177 routes over the Greater Boston area. On average, 991 are active at any given time; the other 45 are reserved for failure recovery.

[1] https://www.ada.gov/

In this project, a bus "route" refers to a round-trip route, starting and ending at the same location. Though we typically imagine a bus traveling the first half of its route, turning around, and then servicing those same stops in the reverse direction, this is not the case. For one, the second half of a bus route may be slightly different to accommodate one-way streets. For two, when traveling in the opposite direction, a bus serves stops on the opposite side of the street. Routes do have a specified "midpoint": the point at which the bus, effectively, turns around to travel back to the start.

The buses are designed to travel routes that have been pre-determined by the MBTA. These routes range in length from roughly 4 miles to 38 miles. Each bus has a unique, 48-bit ID assigned to it by the MBTA; these IDs do not change over the lifetime of the bus.

The MBTA also has 2000 bus operators, to drive the buses.[2] The bus operators are a part of your system (Section **??**), but we are not asking you to handle their work schedules; you can assume that each active bus has an operator available to drive it. Moreover, every operator is qualified to drive any route, though the MBTA prefers to keep the same operators on the same routes as long as possible.

At the start of each day, an operator must drive their bus from the centralized MBTA warehouse out to the start of their assigned route. If a shift change happens during the day, a new operator will travel to the route's origin and take over the route (that is, you can assume that unless there is a failure, a particular bus travels the same route all day). At the end of the day's service, MBTA operators drive the buses back to the centralized warehouse.

The MBTA aims to have each bus stop serviced at least once every twenty minutes during the span of service.[3] You can assume that the MBTA has scheduled its routes to meet this goal under normal operation (e.g., they've taken care of the edge cases that might occur at the start and end of a day). Meeting this frequency goal in the case of failures is part of the challenge of this project.

2.2 Devices on Buses

Each bus is equipped with a number of sensors that collect data, as well a small computer known as the *bus control* that reads that data and transmits it to the MBTA servers via the system-wide radio (described in Section **??**). Part of your job is to specify how the bus control works: what data it collects and stores, how often it collects that data, and how it communicates that data to the MBTA servers. The bus control has some amount of persistent storage available. You may specify how much you'd like it to use; the more storage it needs, the more you will have to justify that decision.

2.2.1 Positioning Sensors

The **GPS sensor** on each bus uses the Global Positioning System (GPS) to calculate the bus's position (latitude and longitude). The GPS sensor updates its data every one second. Each piece of

[2] In reality, the MBTA has closer to 1700 bus operators, not 2000. We've given you more operators to work with because we have made some additional simplifying assumptions about the frequency of bus service.

[3] In the real world, each route has a different target frequency (and a different span of service). We've made a simplifying assumption here for the purposes of the design project.

position data is 64 bits. This data is not automatically pushed to the bus control but may be read by the bus control at any point.

For the purposes of this project, we will assume that GPS data is reasonably accurate: it always reports a bus's location to within five meters of its actual location.[4]

2.2.2 Passenger-Count Sensors

The bus is equipped with a **beam sensor**, which use two infrared beams to determine when a passenger passes through the door of the bus. Each time a passenger passes through the door, the beam sensor sends that event to the bus control. The event also indicates whether the passenger was entering or exiting.

Data from the beam sensors is inherently noisy. Though they can distinguish between passengers entering or exiting, they often generate incorrect data when passengers are wearing large backpacks, using wheelchairs, or entering side-by-side. On average, beam sensors are 85-90% accurate. The MBTA buses are only equipped with beam sensors on the front doors of the bus, not the back doors.

Buses are also equipped with **security cameras**. By default, the video feed from these cameras is not sent to the bus control (and thus not sent to the MBTA's warehouse). However, this data, combined with computer-vision algorithms, results in passenger-count data that is 95-98% accurate. The security cameras collect 5 frames of video per second; each frame is 240 Kbytes. The MBTA buses are not equipped with devices to analyze the video feed; those algorithms must be run on the MBTA's in-house servers (Section **??**).

2.2.3 Payment Interface

Also installed at the front of the bus is the **payment interface**. The MBTA accepts three forms of payment for buses: cash, tickets, and Charlie Cards.

When a passenger pays—regardless of the form of payment—the payment interface makes a record of the transaction and sends that data to the bus control. When a passenger uses cash, the transaction record only includes the timestamp (32-bit UTC timestamp) and the cost of the ride (32-bit floating-point number). When a passenger uses a ticket or a Charlie Card, the transaction record also includes the ID of the ticket or card (a 64-bit ID, read directly from the ticket/card), as well as information about whether the passenger is transferring from a different bus. The ticket/card IDs are unique, and generated when the passenger purchases the ticket or card.

For the purposes of this project, the primary difference between these forms of payment is that a passenger using a Charlie Card will use the same card over a long period of time (months, or perhaps years), because passengers can add monetary value to the cards. Tickets cannot be reloaded, and so are no longer used once their monetary value is zero.

[4] In reality, this is not true. For instance, GPS tends to be inaccurate in urban canyons. To deal with this, buses use additional sensors to perform *dead-reckoning*, an alternate means of calculating position.

2.2.4 Bus-Operator Interface

The **bus-operator interface** is used to display route information to the bus operator. In particular, it should always display the next three stops on the bus's route. This route data is available from the MBTA's centralized servers (Section **??**). You will need to specify how that data is communicated to the bus-operator interface, both at the start of the day and at any points when route data may need to be updated. The bus-operator interface can communicate directly with the bus control.

You can also assume that the bus operator has a radio transmitter and receiver that allow them to talk to someone at the MBTA, though this set-up comes with some constraints (Section **??**).

2.3 Devices and Data at the MBTA

At its centralized warehouse, the MBTA has a small number of machines to serve as servers; how you use each of these machines (which ones store what data, whether data is replicated across servers, whether servers need to communicate, etc.) is up to you. Each of these machines have 10 TB of storage. These machines can communicate with each other via a wired connection. One of them can communicate with the bus controls via the system-wide radio (Section **??**).

Of the pool of servers, one of them—the Reliable Server—never fails. The others may fail for up to two minutes at a time. A failed server will recover after two minutes; no data stored on it will be lost, and you are not responsible for describing this recovery process.

Additionally, the MBTA is expecting that your system is going to be used heavily; you will not be able to assume that all of your data can be stored on a single server.

2.3.1 Data

The MBTA already has a number of datasets that it must store:

- **Census data**, which the MBTA uses to plan its bus routes, to determine which areas have a high population density, and to determine which areas have a low average income. The MBTA prioritizes both of these types of areas: high-density areas since more people require more buses, and low-income areas since residents in low-income areas are typically more dependent on public transportation. This dataset is 600 MB.
- **Bus meta-data**, which stores information about each bus: its (unique) ID, its current operator and route assignment, and some additional information (the bus's make, model, etc.) This dataset is 10 MB.
- **Route and schedule data**, which stores information about each route. Each route has a unique 64-bit ID, as does each stop. The information about a route contains a list of the stops (including their IDs, lat/lon, and a text description of the stop) and the timetables for each day's trips on that route. Those timetables specify the times at which a bus should arrive at a particular stop. This dataset is 100 MB.
- **Alternate stop data**, which stores information (IDs, lat/lon, a text description) about locations that might be used as alternative stops. This dataset is not needed during normal operation, but may be useful as part of your failure-recovery plan. This dataset is 100 MB.

- **Operator data**, which stores information about each bus operator: their operator ID (a unique, 64-bit ID), whether they are currently on duty, the route they are currently driving, and a list of routes they have driven in the past month. This dataset is 10 MB.

You are responsible for specifying how the servers store these datasets. In addition, you are responsible for specifying how the servers store the data from the buses. For example, what data they store, how it's formatted, how long they store the data for, etc.

Your buses will likely be transmitting data back to the MBTA servers. You can assume that along with the servers, the MBTA has a single radio receiver—described in more detail below—that receives this data. That data can quickly be pushed to a single server of your choice, and disseminated from that server to other servers if necessary (you would describe that process). You do not need to consider a setup where each server has its own radio receiver, where different buses communicate with different servers, etc.

To communicate in the other direction—from the warehouse to buses—you can assume a single transmitter connected to the same server that the receiver is connected to. Data can be quickly pushed from that server to the transmitter, and transmitted to buses.

Additionally, you may use the servers to run the computer-vision software that analyzes the video feed from the bus cameras.

2.4 Networks

2.4.1 (Trunked) Radio Networks

Radio networks are wireless networks. In any wireless network, two users cannot transmit on the same frequency at the same time; when this happens, their transmissions "collide" and cannot be decoded.

Many wireless networks get around this problem by assigning different users different frequencies. When two users transmit on two different frequencies, their transmissions do not collide, and can both be decoded at the receiver in a process known as frequency division multiplexing.[5] The number of simultaneous transmissions is limited by the number of frequencies available to the system, among other things.

A downside of this model is that, since each frequency is assigned to a different user from the start, a user's frequency goes to waste when they aren't transmitting; other users don't have a means to "take over" that frequency for a period of time.

Trunked radio systems get around this limitation by not assigning frequencies up front. In a very simple trunked radio system, when a user wants to transmit, it requests a frequency from a centralized controller (there is a special frequency for communicating with the controller: the control frequency, or "control channel"). The controller finds a vacant frequency and gives it to the user to transmit on until its communication is complete.

The benefit of trunked radio systems is that frequencies are not as easily wasted. However, these systems can be quite complex.

[5] Technically, one also needs the frequencies to be "far enough apart" for this to work.

2.4.2 The MBTA Radio Network

Buses communicate with the MBTA warehouse via a trunked radio network, but for the purposes of this project, we will make some simplifying assumptions, as well as some upgrades, that allow you to ignore the complexities (and some drawbacks) of a trunked radio system while still reaping its benefits.

Bus-to-server communication: Each bus is equipped with a radio transmitter and receiver; the warehouse also has a transmitter and receiver (referenced in Section **??**). When a bus needs to transmit to the server, it will do so on whatever frequency is assigned to it by the trunk controller; you do *not* need to specify any details about this request beyond that it happens. You can also assume that the MBTA's server will be listening on all relevant frequencies; it will receive all data that is sent to it via the radio network.

Of course, there are limitations. The MBTA's trunked radio network only has 10 frequencies allocated to it, and there are 1036 buses in the system. As more buses attempt to communicate at the same time, the delay to receive a vacant frequency will increase. You can assume the network is reliable: transmissions between entities will not be dropped or corrupted.

Server-to-bus communication: If the MBTA server needs to transmit data to a bus, things go a bit differently. You can assume that the server has its own dedicated frequency—it doesn't have to worry about requesting anything from the controller—and that all bus receivers are tuned to that frequency (that frequency effectively provides broadcast from the server to the buses). This means that buses can transmit data to and receive data from the server at the same time. Similarly, the server can receive and transmit data simultaneously. To determine whether a piece of data from the server is meant for it or for another bus, the bus receiver checks the destination address in the packet header (see below).

On each frequency, the network is capable of transmitting 16 Mbit/sec.[6] The average latency between the bus and the server is 10 ms, but again, as more buses try to communicate at once, that delay will increase.

For a network to work, each entity needs an address. In this network, each address is 48 bits. The server's address is fixed as `0x00...0` (all zeroes). Buses are assigned an address at the beginning of each day and remain fixed for the day. By default, a bus's network address is the same as its ID. If you would like to design a different addressing scheme, you may.

In addition to sending data to a particular bus, the server can broadcast data to all buses simultaneously by sending data to the address `0xFF...F` (all ones).

All transmissions in this network have the same format:

```
|src addr | dst addr | data |
```

Where:

- `src addr` is the 48-bit source address (either the bus's address or the server's address).
- `dst addr` is the 48-bit destination address (which could be a bus address, the server's address, or the broadcast address). The bus's radio receiver uses this address to determine

[6] This is the primary networking upgrade that we have made for you. In reality, radio networks are *much* slower. We've given you a network with speeds closer to 802.11 wireless in part because we want to see what cool things you do with it, and also because it's likely that in the near future such network speeds would be available for these purposes.

whether communication from the server is meant for it or for another bus (the server uses this address to determine which bus is communicating with it).

- `data` contains the actual data of the communication, which you will specify.

2.4.3 Using the MBTA Radio Network for Voice Communication

The previous section describes using the radio network to transmit general data. It is possible to use the network to transmit voice communications from a bus operator to a system administrator at the MBTA. In that case:

- You do not need to describe the format of the data; you can assume it's transmitted using a standard audio format, which does not take up a large portion of network bandwidth.
- Beyond that, you can treat the conversation like any other data in this network: the bus operator's voice data will be transmitted over the bus's assigned frequency, and the MBTA system administrator's voice data will be transmitted over the server's frequency. Because of how voice communication works, you should assume that those frequencies are occupied for the duration of the conversation.

The bus operator's primary job is to drive the bus. You should **not** expect to be able to use the operator to, e.g., report the number of passengers who have boarded the bus.

2.4.4 Networks at the MBTA Warehouse

At the MBTA warehouse, there are a few other networks to consider:

- Whenever they are in the warehouse, buses can communicate with the server(s) via an in-warehouse wireless network. This is a faster network than the radio network, capable of 54 Mbit/sec.
- The MBTA servers can communicate with each other via a wired network. You can assume that it operates as a "normal" network, with a reliable-transport protocol already in place. The average latency between servers is negligible. The maximum throughput between servers is 1Gbit/second.

2.5 People

Bus Operators: The MBTA has 2000 bus operators, which you should consider to be a part of your system. You are **not** responsible for dealing with bus-operator work schedules; MBTA management handles that.

System Administrators: The MBTA also has 10 system administrators who are responsible for managing the warehouse operations and data. You may choose to use them to make decisions that you believe should not be automated.

Passengers: Though you are building this system for the MBTA, its performance impacts the MBTA's 446,700 daily bus passengers. You should consider their experience as you design your system.

3 Requirements

The existing MBTA infrastructure already imposes some requirements on your system: you cannot exceed the storage or processing power of any component, nor the maximum speed of the networks involved.

Moreover, your system must work at scale. Assume that, in the worst cases, all 1036 buses will be running at once, all routes will covered, and the routes will span the entire Greater Boston area.

3.1 Fleet Monitoring

In addition to those infrastructure-imposed requirements, the MBTA wants to monitor their system for availability, reliability, comfort, and quality. They will do this monitor using the data available on their servers, most of which your system provides.

- **Frequency of service:** For the duration of the span of service, are buses departing on each route at least once every twenty minutes on average?
- **Coverage:** How many citizens in the metro area live within .5 miles of a bus stop? The MBTA plans its route coverage to meet their target (see Table **??**) in the presence of no failures.
- **Reliability:** How many buses are arriving at their origin, midpoint, and destination timepoints within three minutes of their scheduled arrival time?
- **Comfort:** What is the maximum passenger-to-seat ratio for each ride?
- **Total Load:** How many total people are using the buses, and how many transit-dependent passengers are there? These are passengers who use the bus routes in low-income areas.
- **Value to network:** How many passengers who use a bus transfer from one bus to another? The number of transferring passengers gives the MBTA a sense of how valuable its network is.

If your system does **not** allow for the accurate calculation of any of these metrics, you must justify that decision.

3.2 Failures

Beyond basic fleet monitoring, the MBTA will also use your system to detect and recover from failures. They are concerned with three types of failures:

1. **High demand.** In a high-demand scenario, there is more demand for at least one bus route than the route provides (indicated by a passenger-to-seat ratio higher than the MBTA's target). This scenario can occur when there are more people using a route than usual or when the route is not meeting its frequency target, perhaps because a bus breaks down. When there is high-demand on a route, the MBTA must choose whether to add more buses to that route to compensate.

2. **Route unavailability.** A portion of a route can become unavailable for unexpected reasons: fire or policy activity, construction, etc. When this happens, the MBTA must decide how to re-route the buses *and* how to alert passengers to the route change.
3. **Unexpected routes.** In some scenarios, the MBTA may have to add routes to its bus service. For instance, when a portion of the subway system isn't running, the MBTA will add routes that parallel the subway routes. You can assume that the MBTA knows what those routes should be; the problem here lies in making sure the MBTA can meet the demand that an influx of subway passengers will bring.

3.2.1 Failure Detection

Your system is responsible for detecting the first type of failure (a high-demand scenario). Your system is *not* responsible for detecting whether the second (route unavailability) or third (unexpected routes) types of failure occur. In these cases, an MBTA system administrator will be alerted (via some other means, for instance a police scanner).

3.2.2 Failure Recovery

In all three cases, your system must respond to the failure. Depending on its response, there may need to be communication between the MBTA's servers and one or more buses. You must describe how that communication happens even in the cases where your system was not responsible for detecting the failure.

In response to a failure, you may choose to:

- Re-route one or more buses. That is, calculate a new path from a bus's current location to the endpoint of its original route, and send the bus on this route. You can use the alternate stop dataset as a resource in this process. (This response is most useful in the case where a portion of a route has become unavailable; note that this is *not* the same as moving a bus from one route to another.)
- Put a currently-idle bus into operation, sending it from the MBTA warehouse out to a route. The idle bus can start its service at any stop on the route (i.e., not just the beginning of the route); once it has started, it will traverse the entire route as it would normally. Keep in mind that it will take a bus some time to travel from the MBTA's warehouse to the route.
- Pull a bus off of one route and add it to another. Buses can only be pulled off a route when they reach the route's origin, destination, or midpoint. As in the previous scenario, it will take a bus some time to travel from one route to another.
- Do nothing.

Whatever case(s) you choose, you must describe how that decision is made, and how it is executed. For instance, if you choose to pull a bus off of one route and move it to another, your design should describe that process. This may include specifying what data gets updated, which part of the system detects the failure, what messages are sent between the warehouse servers and the buses, etc. Recall that in the cases of updating a bus's routes, the bus-operator interface can alert them to route updates (Section **??**).

Standard	Target
Reliability	75% of the time, buses should meet their frequency-of-service demands (arriving at their origin, midpoint, or destination timepoints within three minutes of the scheduled arrival time)
Coverage	At least 75% of the Boston metro population should live within .5 miles of at least one bus stop. At least 85% of low-income households should be within .5 miles of at least one bus stop.
Comfort	96% of the time, the maximum passenger-to-seat ratio should be ≤ 1.4.

Table 1: MBTA Service Targets

In your consideration of how to handle these failures, it is helpful to know the MBTA's service targets: the level of service that it aims to provide to its passengers. Table **??** describes those targets.

3.2.3 Expectations for Your Failure Recovery Algorithm

The algorithmic problem of optimally allocating resources in a setting such as this can become arbitrarily complex. **We are not asking you to find an optimal algorithm for failure recovery.** We are asking you to make choices and justify your system's response to failure recovery, taking into account things such as the MBTA's service targets and user experience.

Remember that most good system design is iterative: we start with a simple solution and iterate from there. Starting out with a complex algorithm is a difficult design choice to justify from an iterative-design standpoint. (Of course, starting out with an algorithm that will not allow the MBTA to meet any of its service targets is also tough to justify.)

That said, we are not limiting your failure-recovery algorithm. If you want to build an algorithm that works by chaining eight neural networks together and then feeding the output into six different linear programs, go for it. Just be prepared to justify that design to your audience.

3.3 Passenger Feedback

In addition to all of the metrics discussed, the MBTA can also get feedback from passengers to evaluate its quality of service. The MBTA is considering multiple mechanisms for dealing with this:

1. Using an integrated application. Users would download an app to their phone, which would be capable of communicating with the bus control of the bus they were on.
2. Using an MBTA application, but not one that could communicate with the bus control. Such an application might give a structured feedback form for the passengers to fill out.
3. Using a "backchannel" that the MBTA does not control. For instance, getting customer feedback via Twitter or other social media. One of the system administrators would monitor the channel and respond to passengers when necessary.
4. Not using passenger feedback at all.

You do **not** need to design the MBTA's system for passenger feedback. However, as part of your proposal, you should recommend one of the four mechanisms, and justify your choice.

3.4 Security

While the MBTA has not imposed any security requirements on your system, they would like you to think about the following threats:

- An MBTA employee, with access to the data from your system as well as other data that the MBTA might collect (e.g., information about who purchased a CharlieCards).
- A person—not necessarily an MBTA employee—who is able to observe traffic on the radio network between the buses and the warehouse.

What ability to these two attackers have to track passengers? Could they track a passenger through Boston over multiple days? Could they infer where a person lives or works?

If the data you are collecting allows passengers to be tracked in some way, you must justify that decision.

4 Design Trade-offs, Use Cases, and Other Considerations

4.1 Design Trade-offs

While designing your system, you will be faced with a number of design trade-offs. We've summarized four of the biggest below.

- **Data accuracy:** In general, the more data you collect, the more accurately you can estimate various quantities. How will you trade-off the accuracy of your data with the performance of your system (for instance, the overhead of storing and transmitting data)?
- **Failure recovery:** What will you prioritize when recovering from failures? There may be cases when you cannot meet all of the MBTA's service requirements; how will you defend your choice of which ones to meet? How will your decisions impact the passenger experience?
- **Automation:** Should all parts of your system be automated? Are there points where MBTA operators should step in and make a decision? What are the repercussions of automating every part of a system that has social implications? Appendix **??** describes the specific points in your system where humans could potentially be helpful.
- **Security:** What are the security implications of your system by itself, and also if your system is integrated with other MBTA systems? For instance, the MBTA may also keep track of credit card numbers when customers purchase tickets or Charlie Cards. How would that data interact with the data that your system collects?

4.2 Use Cases

As part of your proposal, you should describe how your system would work under each of the following use cases. This is not an optional part of the design project; if your system cannot handle one of these cases, you must explain why.

- **Normal operation:** The pool of idle buses is large, all routes are running smoothly, and there are no unexpected failure conditions.
- **High demand:** There is a Red Sox game at Fenway park, and the D branch of the Green Line is not running due to a track problem. Because of the length of that branch and the passenger demand, it is unlikely that the pool of idle buses alone will be sufficient for handling this demand.[7]
- **Construction:** There is unexpected construction that affects multiple bus routes. This construction occurs in a low-income area where many passengers depend on the affected bus routes to get to work.
- **Historical Data:** The MBTA wants to investigate a complaint from a passenger. The passenger reported that their bus was over thirty minutes late one day last week.

Also consider how your system would scale beyond the MBTA. Suppose that a larger city wants to adopt your system. Will your system be able to adapt to such growth?

[7] The insufficiency of the idle pool of buses is a real problem for the MBTA.

5 Appendix - Humans vs. Robots

There are many points in your system in which you can choose to involve the MBTA system administrators. We outline a few below.

- **Data verification:** There are some cases where humans can perform more accurate data verification. For instance, given the video feed from the buses, a human can more accurately count the number of passengers on the bus than either the computer-vision methods or the beam sensors.
- **Bus re-routing:** A human may be able to choose a better alternate route for a bus. It is easy to automate shortest-path calculations, but humans have a better sense of traffic patterns and social implications of new bus routes (e.g., they can determine whether a proposed bus route will take a bus down a narrow, difficult-to-navigate residential street).
- **Adding buses:** Humans can anticipate whether any portion of the idle-bus pool should be reserved, for instance in preparation for a large sporting event happening later in the day.