Massachusetts Real Time Transit Control (MassRTTC)

A Group of 3 MIT Students in 6.033

1 Introduction

The MBTA bus network serves almost 400,000 people daily and is hence key to the economic functioning of Greater Boston. To provide high quality service to its customers, the MBTA needs to provide frequent, reliable, and comfortable service. However, although bus routes are scheduled to meet even peak demands, unpredictable congestion and fluctuating demands can lead to deteriorated quality of service.

To meet these needs, MassRTTC is an integrated real-time control system designed to meet the following goals:

- **Frequent service.** If routes are not serviced once every 20 minutes per stop, riders can perceive the service as unreliable, discouraging them from using it. Thus, frequent service both meets the needs of citizens and sustains ridership.
- **Low headway variability.** Inelastic bus demand means buses operating with scheduled headways can meet demand. However, when a bus is late, especially during peak hours, it picks up more passengers, increasing dwell times, further pushing it behind schedule. The following bus thus has fewer waiting passengers, pushing it ahead of schedule. Eventually, the buses form a pair, doubling waiting times and leaving the front bus overcrowded. This effect is known as bunching and is documented as a source of unreliability by Strathman et al. (2003) and Xuan et al. (2016)[3][4]. Preventing this is key to providing reliable service.
- **Passenger comfort.** The passenger to seat ratio on buses should be under 1.4, with less than one passenger standing for every passenger seated.
- **Resilience.** The bus network should keep running smoothly even in case of unexpected failures such as road closures or subway breakdowns.

MassRTTC includes automatic vehicle location and passenger counting systems. This is used by a central server to control and, if necessary, reroute buses. The server instructs operators to obey minimum and maximum dwell times to prevent bunching, informs them to new routing, and deploys reserve vehicles to meet unexpected demands. While automation will improve the efficiency of management, for certain decisions, the system defers authority to humans, who better understand the complexity of cities.

2 System Design

2.1 Components

2.1.1 Buses

The MBTA has 1036 total buses. Buses are equipped with automatic vehicle location, passenger counting and fare collection systems. A bus interface communicates with the driver. A payment interface records payments and CharlieCard information, including transfers.

2.1.2 Network

The buses, as well as the MBTA warehouse, each have a radio transmitter and receiver. Buses and the server can transmit and receive data simultaneously. During the day, buses can transmit data to the server or other buses using one of 10 frequencies, while the server can send data using its own dedicated frequency.

At the warehouse buses can wirelessly communicate with increased throughput. Server machines communicate with each other via wired network.

2.1.3 Server

The server has a master-worker architecture. The master is the machine that directly communicates with buses via the transmitter. It stores data, communicates with buses, and assigns tasks (described in section 2.3) to sub-masters. It also has an interface for system administrators to manually change system statuses, including deploying protocols for situations the server cannot detect, and querying the database.

A sub-master breaks tasks down further and assigns them to workers. It analyzes the results from the workers and sends the analysis to the master. The analysis can include protocols to deploy, or data to store or send to buses.

Sub-masters and workers are chosen randomly, and a sub-master for one task can be a worker for another. This ensures that tasks are split evenly, so that in case of a machine failure, less data of each type are lost and the system can recover more easily.

2.2 Bus Data Collection

Data are collected to allow the server to make informed decisions to maintain quality service.

2.2.1 Vehicle location

Buses collect GPS data each second. A bus decides that it has passed a stop when it passes in, and then out, of a 30m radius from a stop. The bus uses this information to display the next 3 stops on the bus interface. This slight overestimate prevents the display from changing before the bus has actually reached the stop.

2.2.2 Passenger counting

Buses rely on both beam sensor and security camera data to count passengers. The bus keeps a local count using its beam sensor, an overestimate because it cannot detect passengers alighting via the back door. If the passenger-seat ratio estimate exceeds 1.4, the bus sends 5 frames from its security camera to the servers for a more accurate count. Keeping an overestimate and only updating when the count is relevant reduces network and server load.

2.2.3 Data transfer

Buses send passenger count analysis, GPS and payment data to the server. Each bus stores its GPS reading every 5 seconds and sends them in batches to the server every minute. Data on the network are sent infrequently enough that the entire fleet can share the 10 frequencies without long delays.

Fare and transfer data are stored in the bus control and transferred to the server at night for security.

2.3 Server Data Processing

The server does heavier computation to maintain quality functioning of the bus network.

2.3.1 Passenger counting

Every time the server receives a request to count the passengers on a bus, it runs a computer vision algorithm on the video frames provided. It then discards the frames, stores the count, and sends the count to the requesting bus.

2.3.2 Detecting supply issues

Bunching is an important cause of crowding and unreliability (section 1). MassRTTC distinguishes between deploys protocols to alleviate different high demand scenarios. For each route, the server checks for pairs of buses that are less than 3 minutes apart and considers three cases:

- **The leader bus is crowded and the follower is not.** Separation is needed to balance the load and decrease wait times. The buses are flagged, Protocol 1 is initiated between the buses, and Protocol 2 is initiated on the follower (sections 2.4.1 and 2.4.2).
- **Neither bus is crowded.** Separation is required to meet frequency targets. The follower is flagged and Protocol 2 is initiated (section 2.4.2)
- **Both buses are crowded.** Neither strategy will help. This may indicate that the entire route is crowded. If a number of consecutive buses on the route are too crowded, Protocol 3 is initiated (section 2.4.3).

Buses are unflagged once their headways are restored.

Since the server counts passengers only on a subset of buses (section 2.2.2), it lacks accurate information about buses with low load. Hence, if a bus's passenger count is stale (older than 5 minutes), it assumes that the bus is under low load. If it detects a congestion situation, it only flags buses that have fresh data. Otherwise, it sends a request to that bus get video data for verification. In order to deal with changing traffic data staleness, this algorithm is run every minute.

2.3.3 Other Data Analysis

Since these data are not used for real-time control, the analyses are only run at night.

- **Reliability.** Based on location data, the server checks whether each bus reached its control points within 3 minutes of its schedule. The server also computes the mean and variance of headways, and uses this to estimate average waiting times. Ideal 20 minute service would result in 10 minute waiting times, and the server looks for routes that fail this target significantly.
- **Mobility patterns.** Gordon (2012) describes techniques to estimate passenger destinations and transfers even though passengers do not tap out or may use cash.[1] Combining this with census data can help planners better understand mobility patterns, including popular transfers, the prevalence of transit-dependent passengers, and the coverage of the network. This can inform long-term adjustments to service.

2.3.4 Storage

The master stores GPS and payment data from the buses. It also stores passenger counts and analysis results, totaling merely 350GB/year. With 10TB of memory, it can store the data for decades.

The master backs up data to a backup machine every night.

2.4 Protocols

2.4.1 Protocol 1: Bunching

The server tells each flagged bus whether it is the leader or follower, and the ID number of the other bus (for communication). Every minute, the follower sends the leader an estimate of the time difference between them. The leader displays NOT IN SERVICE instead of its destination to prevent people from entering. Its interface tells the driver about the situation. The driver then does not allow more passengers to board at the next stop, instead informing them how long before the next bus arrives.

Updating the time estimate every minute ensures that the driver always has a fresh estimate to tell the waiting passengers.

2.4.2 Protocol 2: Holding

When buses bunch, the server flags the follower bus, which informs the driver via the interface. The driver is then instructed to enforce maximum dwell times by simply waiting at stops. Sanchez-Martinez et al. (2016) states that this is the most effective and widely used method of preventing bunching. [2]

2.4.3 Protocol 3: Allocating reserve buses

In situations with unusually high demand, or where a subway breakdown or special event demands shuttling, the system notifies reserve crew to deploy reserve buses from the warehouse, if available. The system prioritizes first scheduled routes, and then special routes with highest demand.

2.4.4 Protocol 4: Route changes

For planned route changes, such as for construction, new schedules are uploaded to the bus at the warehouse at night. Signs are posted at affected stop locations directing passengers to the new stops.

In case of emergency changes, server administrators use the master machines interface to input new routes and send them to affected buses. Passengers on board learn of changes via the bus intercom.

All routing changes are posted on the MBTA website and Twitter page.

3 Passenger Feedback

We collect customer feedback using the MBTA Twitter account run by a system administrator. Twitter is preferred over an application because it comes with a ready-to-use interface that is already widespread, whereas an application needs to be deployed. In addition, passengers from low-income households may not own a smartphone for using an application. On the other hand, Twitter is available on the Web and can be accessed from public computers.

4 Discussion

4.1 Use Cases

The following are some example use cases of MassRTTC:

- **Normal operation.** During normal operation, the system continues monitoring the data and checking for high demand, route unavailability, and unexpected routes.
- **High demand.** Our system detects high demand by counting passengers. However, apparent high demand can result from either failures of supply-side control or actually high demand, and MassRTTC distinguishes between the two by choosing when to hold buses and when to add reserve vehicles (section 2.3.2).

- **Road closures.** In the event that construction makes a route unviable, an alternate route is selected as described in section 2.4.4.
- **Complaints.** The server stores the buses' control points service for decades, allowing system administrators to investigate reliability complaints. The server stores the passenger counts for all buses that report loads close to or exceeding the target, allowing system administrators to investigate passenger comfort complaints.

4.2 Scalability

The server architecture allows easy recovery in the event of a machine failure. Data is sent over the network sparingly, with less than 20 percent used. Physical limits on the capacity of a city makes added load on the server less concerning. The amount of data stored is also relatively small. Thus, while our system was not designed to scale, we can easily scale to a city with 5x more buses, with the same server and network capabilities.

4.3 Security

Our system addresses some security concerns. Sensitive data such CharlieCard IDs and payment data is not sent over the network. Video frames, which can have identifying information, are not stored on the servers. However, video frames are still sent over the network and can be intercepted.

5 Conclusion

Bus reliability and passenger comfort are paramount in a customer-driven system like the MBTA bus network. MassRTTC is a integrated real-time control system for the MBTA bus network that is designed to provide frequent service, decrease headway variability, and maintain passenger comfort.

MassRTTC collects data on service frequency, route coverage, passenger load, and network value from MBTA buses, and uses the data to evaluate whether it is meeting MBTAs service targets for reliability, coverage, and comfort. It can handle cases of high demand, route unavailability, and other unexpected events. Our system can scale to 5 times bigger bus networks with the same network and server resources. Such a system is crucial for maintaining high quality bus service.

References

[1] J. B. Gordon, Intermodal Passenger Flows on Londons Public Transport Network: Automated Inference of Full Passenger Journeys Using Fare-Transaction and Vehicle-Location Data, M.S.T. and M.C.P. thesis, Department of Civil and Environmental Engineering and Department of Urban Studies and Planning, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2012.

[2] G. E. Sanchez-Martinez, H. N. Koutsopoulos, and N. H. M. Wilson, Real-time holding control for high-frequency transit with dynamics, Transportation Research Part B: Methodological, vol 83, pp. 1-19, January 2016.

[3] J. G. Strathman, T. J. Kimpel, and S. Callas, Headway Deviation Effects on Bus Passenger Loads: Analysis of Tri-Mets Archived AVL-APC Data, Final Research Report for Tri-Met, 2003.

[4] Y. Xuan, J. Argote, and C. F. Daganzo, A Dynamic Holding Strategy to Improve Bus Schedule Reliability and Commercial Speed, unpublished, 2016.

6.033 Computer System Engineering
Spring 2018