QUALITY CONTROL - LESSONS LEARNED FROM THE DEPLOYMENT AND EVALUATION OF GTFS-REALTIME FEEDS

Sean J. Barbeau, Ph.D. (Corresponding Author)
Center for Urban Transportation Research
University of South Florida
Tampa, FL 33620
813-974-7208
barbeau@cutr.usf.edu

November 15, 2017

6,678 words + 4 figures (1000) = 7,678 words

Abstract

Real-time transit information has many benefits to transit riders and agencies, including shorter perceived and actual wait times, a lower learning curve for new riders, an increased feeling of safety, and increased ridership. In the last few years, a real-time complement to the General Transit Feed Specification (GTFS) format, GTFS-realtime, has emerged. GTFS-realtime has the potential to standardize real-time data feeds and lead to widespread adoption for transit agencies and multimodal apps. However, GTFS-realtime suffers from a lack of clear documentation and openly available validation tools, which significantly increases the time and effort necessary to create and maintain GTFS-realtime feeds. More importantly, bad data has been shown to have a negative effect on ridership, the rider's opinion of the agency, and the rider's satisfaction with multimodal apps. This paper discusses the lessons learned in the deployment of a GTFS-realtime feed with an open-source mobile transit app as part of a regional transit information system for the Tampa Bay area in Florida. These experiences led to improvements to the GTFS-realtime specification itself, as well the creation of an open-source GTFS-realtime validation tool. An evaluation of 78 transit agency GTFS-realtime feeds using the validation tool showed integrity errors in 54 feeds and warnings in 58 feeds, indicating wide-spread problems with quality control. This paper concludes with recommendations going forward that will help reduce the time needed to develop, test, deploy, and maintain GTFS-realtime feeds, which will in turn lead to better quality real-time information for transit riders.

INTRODUCTION

Real-time transit information has been shown to have many benefits to transit riders, including shorter perceived wait time [1], shorter actual wait time [1], a lowered learning curve for new riders [2], and increased feeling of safety (e.g., at night) [3, 4]. Transit agencies who have deployed real-time information have also benefitted from increased ridership [5, 6], as well as a better perception of the agency and it's transit service, even if it's service hasn't actually changed [7].

Availability of transit schedule, stop, and route information to transit riders via mobile apps has historically been driven by agencies sharing this data in the GTFS format [8], which has become the dominant format for open schedule data in the transit industry and shared by over 1,500 agencies worldwide [9]. In the last few years, a real-time counterpart to GTFS, GTFS-realtime (GTFS-rt), has begun to emerge, with agencies sharing their real-time data in this format. Previously, real-time transit information had only been shared in proprietary formats specific to each vendor or agency. GTFS-rt offers the opportunity for application developers to create a mobile app that can function across a large number of cities and agencies, and for practitioners and researchers to be able to easily study and compare actual system performance across different transit systems using the same tools, without the overhead of manually transforming data into a consistent format. Having real-time transit data available in a common format is a key pillar for real-time multimodal information systems.

However, of equal importance to data availability is data quality. In fact, accuracy of real-time information is a key concern of transit riders. A survey of riders of a mobile transit app showed that 84% rely solely on real-time information instead of using the schedule [4]. Errors in predictions create a negative perception of the mobile app providing the information as well as the transit agency. For example, 74% of surveyed Puget Sound transit riders considered a difference between actual and estimated arrival times greater than 4 minutes as an “error”. In addition, 9% of surveyed riders said that they took the bus less often due to errors they experienced [4]. Prediction errors can also lead to reduced system performance if operations is making decisions based on this data.

The GTFS-rt format is relatively new, and, as with any emerging data format, this can result in various challenges. First, while the GTFS format for schedule data has several open-source GTFS feed validators [10], no such open validation tool has existed for GTFS-rt. And, due to the implementation details of the GTFS-rt specification [8], the GTFS-rt specification itself has not provided strong guidance for what data fields are required or optional for use cases of the data. Furthermore, the scale of datasets combined with the frequent refresh of real-time data makes manual inspection time-prohibitive. For example, in November 2017 Massachusetts Bay Transportation Authority (MBTA) in the Boston, Massachusetts area [11] has a GTFS dataset that contained 71,260 trips and 1,809,833 stop time records. MBTA's GTFS-realtime feed contains data for 489 vehicles with independent arrival or departure predictions for most stops on active trips that is refreshed around every 5 seconds. Lack of good documentation and validation tools results in confusion and disagreements between transit agencies, Automatic Vehicle Location (AVL) vendors, and application developers as to what data should actually appear in a GTFS-rt feed, which increases the time, effort, and cost to deploy a new GTFS-rt feed [12].

This paper discusses the lessons learned during the deployment of a GTFS-rt feed with an open-source mobile transit app as part of a regional passenger information system for the Tampa Bay area in Florida. To the author's knowledge, this is the first open documentation of such a GTFS-rt deployment. These experiences led to improvements to the GTFS-rt specification itself, as well as to the development of an open-source GTFS-rt validation tool. This enhanced guidance and openly available validation tool will reduce the time needed to develop, test, and deploy GTFS-rt feeds, which will reduce the cost of providing high quality real-time transit information to riders. As is demonstrated in this paper, many industry GTFS-rt feeds suffer from data problems that can be easily captured with such a tool.

The following sections give a brief introduction to the GTFS and GTFS-rt format and OneBusAway open-source mobile app. Subsequent sections discuss the lessons learned from the GTFS-rt and OneBusAway deployment in Tampa Bay, Florida, as well as the development and testing of the GTFS-rt validation tool.

GTFS – The Foundation of Real-time Data

GTFS forms the foundation for a GTFS-rt feed – a GTFS-rt feed cannot provide practical real-time prediction information without having a companion GTFS feed that defines the schedule. GTFS data is implemented as a set of comma-delimited text files added to a single zip file.

A subset of the full GTFS specification is required for a GTFS-rt feed – the following are key for understanding real-time information:

- stops.txt – All bus stops included in a feed, with each record including a stop_id (identifier internal to agency), stop_code (rider-facing stop identifier), stop location, location_type (a single stop or station with multiple stops), etc. For some agencies, stop_id and stop_code may be the same.
- routes.txt – All routes defined for an agency, including a route_id and short and long name
- calendar.txt and calendar_dates.txt – Includes service days and times, each identified via a service_id, that the agency provides service
- trip.txt – All trips defined for an agency, including to which route_id each trip belongs. A route may have multiple trip patterns, depending on the day and/or time. The day/time that each trip is operational is defined by a service_id that relates to calendar.txt and/or calendar_dates.txt
- stop_times.txt – The core schedule file that defines, for each trip_id, the ordered list of bus stops that will be visited, along with a scheduled arrival and departure time, and whether or not each stop is a timepoint (optional).

A stop_times.txt file will look like the following:

trip_id	arrival_time	departure_time	stop_id	stop_sequence
2777	5:52:00	5:52:00	4301	1
2777	5:52:34	5:52:34	3471	2

2777	5:53:46	5:53:46	4456	3
2777	5:54:27	5:54:27	592	4
2777	5:55:11	5:55:11	593	5
2777	5:55:20	5:55:20	4457	6
2777	5:55:40	5:55:40	595	7
2777	5:56:34	5:56:34	596	8
2777	5:57:09	5:57:09	6898	9
2777	5:57:42	5:57:42	6899	10
2777	5:58:17	5:58:17	597	11
2777	5:58:56	5:58:56	599	12
2777	5:59:20	5:59:20	600	13
2777	5:59:50	5:59:50	601	14
2777	6:00:15	6:00:15	602	15

GTFS-realtime – An Open Format for Real-time Transit Data Exchange

The GTFS-rt specification can be broken down into three types of elements:

- **Trip Updates** – Real-time predictions for when vehicles arrive and depart. Predictions (stop_time_updates) are represented as an update to the time that the vehicle was scheduled to arrive or depart (defined in GTFS stop_times.txt), either as a relative "delay" or "time". stop_time_updates are identified using a trip ID from GTFS trips.txt.
- **Vehicle Positions** – Real-time vehicle location, trip assignment (defined using the trip ID from GTFS trips.txt), and occupancy information
- **Service Alerts** – Descriptions of events that affect transit service, along with the transit stops/routes that the event impacts. For example, "Route 5 is on detour due to flooding".

A GTFS-rt Trip Update for trip_id 2777 that predicts a bus running 60 seconds late for stop_id 4456 (stop_sequence 3), running on time for stop_id 592 (stop_sequence 4), and 60 seconds early for stop_id 593 (stop_sequence 5), would look like the following:

```
trip_update {
  trip {
    trip_id: "2777"
  }
  stop_time_update {
    stop_sequence: 3
    arrival {
      delay: 60  // Schedule deviation of 60 seconds (running late)
    }
    stop_id: "4456"
  }
  stop_time_update {
    stop_sequence: 4
    arrival {
      delay: 0  // Schedule deviation of 0 seconds (on time)
    }
    stop_id: "592"
```

```
        }
        stop_time_update {
          stop_sequence: 5
          arrival {
            delay: -60  // Schedule deviation of -60 seconds (running early)

          }
          stop_id: "593"
        }
      }
```

The architecture of a real-time transit information system can be divided up into two components [13]:

1. The Producer - The system generating the GTFS-rt feed (typically the automatic vehicle location (AVL) system)
2. The Consumer – The system reading the GTFS-rt feed (typically a server and mobile app displaying the information to a transit rider)

While GTFS datasets are typically updated 3-4 times per year (e.g., when new schedules are published), a GTFS-rt Trip Updates and Vehicle Positions feed can be updated as often as every few seconds and are typically driven by an automatic vehicle location (AVL) system.

GTFS-rt datasets are formatted in the Protocol Buffer format [14], which is a very efficient binary representation of the information in the feed. As a result, the actual GTFS-rt messages produced and consumed by applications require special software to convert them to human-readable plain text.

Both the frequency of real-time updates as well as the required conversion from binary to a plain text make it very challenging to manually identify and troubleshoot problems in the feed. Historically, these types of feeds have been tested by passing the information into a third party application, such as a mobile transit app.

The following section discusses OneBusAway, the open-source mobile transit app used by the research team to test the new GTFS-rt feed deployed in the Tampa Bay region.

Deployment of a Mobile Transit App with the Real-time Feed

OneBusAway (OBA) is a mobile application for Android, iPhone, Amazon Alexa, Google Glass, and Pebble Smartwatches that provides real-time transit information for multiple regions in which the users can see arrival times or updates (e.g., early arrivals, delays) for each bus stop [15]. OneBusAway Tampa (http://tampa.onebusaway.org) was officially launched in August 2013 as part of a research project partnership between the Center for Urban Transportation Research at the University of South Florida and Hillsborough Area Regional Transit (HART). A regional effort to add the neighboring transit agency on the west side of Tampa Bay, Pinellas Suncoast Transit Authority (PSTA), started in 2015 as part of a project to provide a single mobile transit app for the greater Tampa Bay area. This project also piloted technology designed to streamline the process of transit riders reporting multimodal issues back to public agencies, including transit agencies, departments of transportation, and city/county government [16].

Unlike other transit apps, OneBusAway is entirely open-source software, which means that anyone can download, deploy, and modify the software for their own use. The research team used the OneBusAway server software [17] and mobile apps as the testing tool for the new feed deployed at PSTA, provided by PSTA's AVL vendor Clever Devices.

GTFS-REALTIME FEED DEPLOYMENT – LESSONS LEARNED

The PSTA GTFS-rt feeds used with OneBusAway were created by PSTA's AVL vendor, Clever Devices. The following sections discuss the various issues encountered during the deployment of OneBusAway with the new GTFS-rt feed.

Erroneous GTFS-rt arrival times were attributed to three sources, which are each discussed in subsequent sections:

1. **Producer Issues** - Bugs within the GTFS-rt generation software and/or AVL system
2. **Consumer Issues** - Bugs or insufficient support of GTFS-rt data within the OneBusAway software
3. **Different interpretations of the GTFS-rt specification** – Some areas of the GTFS-rt documentation have not been well-defined, and therefore consumers and producers may expect different output for these gray areas in the specification

GTFS-rt Producer Issues

PSTA has been providing GTFS data to third party app developers for many years using the export feature of their HASTUS scheduling software. However, one key requirement for maintaining GTFS and GTFS-rt data is that the IDs within the GTFS data (trip_id, stop_id, etc.) must match the IDs in the GTFS-rt data. To properly support matching IDs, PSTA transitioned from exporting their GTFS from HASTUS to exporting it from Clever Devices system, the same vendor being used for the AVL system. As a result, PSTA was creating a brand new version of their GTFS data in addition to the new GTFS-rt feed.

The research team used the GTFS Feed Validator [12] to quickly identify and generate a report about issues in the new GTFS data, which included the following:

- **Incorrect route_long_names in routes.txt** – In PSTA's previous GTFS data, the route_long_name contained the descriptive name of the route like "Gateway Mall / Tyrone Square Mall", while route_short_name was "1". The new route_long_name contained the text "Route 1", which is an incorrect description of the route.
- **agency_url and timezone fields missing in agency.txt** – The agency.txt agency_url and timezone fields, which are both required by the GTFS specification to provide proper contact points and timezone information, were missing.
- **Stops have duplicate stop_codes in stops.txt** – The stop_code value should be the user-facing identifier displayed on a bus stop sign or shelter. However, for several stops the same stop_code was assigned to more than one stop. This resulted in duplicate stops being shown in the app for search results, one of which was missing a schedule (i.e., it showed "no arrivals or departures").

- **Duplicate times within trips in stops_times.txt** – arrival_time and departure_time must increase for each stop along the trip. Several trips showed the bus arriving at several stops in a row at the same exact time, which is incorrect.
- **"Too fast travel" warning for stop_times.txt** - This problem was a secondary issue resulting from the duplicate times within trips (above). Because the amount of time between sequential stops was very low (i.e., 0), the validator flagged the trips as traveling too fast for reality.
- **Bad shape data** - The shape data provided in GTFS shapes.txt to describe the actual travel path of the bus had some errors where a point would significantly deviate from the path of the vehicle. Because OneBusAway interpolates the vehicle position on the map based on the progress along the trip when no real-time information is available, this resulted in a strange display of information to the user where the vehicle is significantly off-route. This error was not flagged by the GTFS validation tool, but was found when manually testing the application.

The AVL vendor fixed these issues identified in the GTFS data and generated new GTFS data that did not have these problems. Some of these issues (incorrect route_long_name, missing agency_url and timezone fields, duplicate stop_code) were not software bugs, but were due to the way that PSTA staff had coded data within the data management tool. In these cases, the PSTA staff edited the data to correct the problem.

Troubleshooting the GTFS-rt feed was significantly more challenging. The quality assurance process amounted to checking OneBusAway logs to determine if any errors were being identified, as well as physically visiting bus stops, checking arrival times show in the app, and comparing them against when the bus actually arrived at the stop. However, the OneBusAway server software was built to be an application, and not a validation tool. As a result, it often did not directly catch problems in the real-time feed or generate any errors. Instead, issues were identified when an abnormal arrival or departure time was manually identified within the OneBusAway mobile apps. Transit agency staff reported problems back to the research team, which then would attempt to identify the problem in logs and try to reproduce and/or manually catch the problem again in real-time. This was an extremely time-consuming process and involved significant communication between PSTA, the AVL vendor, and the research team.

The following issues were identified with the GTFS-rt feed [18]:

- **stop_time_updates not sorted by stop_sequence** – To enable efficient processing by consumers, the GTFS-rt specification requires that producers order predictions within a trip by stop_sequence. In other words, the predictions for stops within a real-time update should be in the same order as the stops occur within the trip, defined in GTFS stop_times.txt. The initial version of the PSTA TripUpdates feed did not include the optional stop_sequence field. The AVL vendor changed their software implementation to always sort stop_time_updates by stop_sequence, and eventually added the stop_sequence field to the GTFS-rt feed so it was easier to confirm that each trip did indeed have updates sorted by stop_sequence.
- **Wrong stop_ids were included in trip_updates** – Occasionally stop_time_update estimates appeared in a trip with a stop_id that didn't belong to that trip. This was caused by several problems, including more than one stop having the same stop_code in GTFS

stops.txt and the handling of routes that contain a loop where a stop is visited more than once in the same trip (discussed in detail in a later section). The AVL vendor coordinated with PSTA to resolve this issue.

- **Stop_codes instead of stop_ids were included in alerts** – In the GTFS-rt Alerts feed, alerts were published that related to particular stops. However, the stop_code, not the stop_id, appeared as the identifier in the alert. As a result, the alert couldn't be matched to the proper stop. The AVL vendor fixed this problem and published stop_ids to the alerts feed.
- **Invalid vehicle position data** – Occasionally a vehicle would have the latitude and longitude values of (0.0, 0.0) as a result of temporarily unavailable GPS data on-board the vehicle. The AVL vendor changed their feed to avoid publishing updates for vehicles with bad or unavailable GPS data.
- **Invalid vehicle route assignment data** – In the first version of the Vehicle Positions feed, vehicles that were not currently assigned to trips would appear in the feed with a route_id of "U" for "unassigned". Route_id should only be used for valid customer-facing routes that would appear in the GTFS routes.txt data, so these vehicles should not be included in the feed or should not have any route_id associated with then. The AVL vendor fixed the feed to remove this "unassigned" route information.
- **Unrealistic vehicle speeds** – In the initial version of the feed, very large vehicle speed values were observed (e.g., 129 miles per hour). This was because the speed values were being set in miles per hour, instead of the required units of meters per second. The vendor resolved this issue by converting to the correct units before outputting the data to the feed. However, even after this was fixed, abnormally high speed values were still observed. Apparently some vehicles were not calibrated to report speed accurately, so the AVL vendor worked on updating these vehicles to fix the reported speed.
- **Duplicate back-to-back stops in trip updates** – Some stops appeared more than once in sequence, each having a different predicted time of arrival in a stop_time_update. The AVL vendor fixed the problem to remove the duplicate stops and only have a single arrival time for each stop.

GTFS-rt Consumer Issues

The research team discovered a few problems with the OneBusAway open-source software that negatively impacted the predictions shown to riders. While OneBusAway already included basic support for GTFS-rt feeds, the research team encountered several scenarios in PSTA's data that OneBusAway did not properly handle. These issues mostly stemmed from the fact that the PSTA GTFS-rt feed provides many predictions (stop_time_updates) per trip – one for each stop.

All previous GTFS-rt feeds used in the various OneBusAway regions, including HART's GTFS-rt feed, had only provided one prediction per vehicle. This single arrival estimate indicated whether a bus was running ahead, behind, or on schedule for a particular stop, and this same delay value was then applied to all stops for the rest of the trip (i.e., all stops "downstream" of the prediction). In contrast, PSTA's GTFS-rt feed provides an individual predicted time for *each* stop on the trip. Presumably, the additional arrival estimates for each stop in the trip have been calculated using an advanced prediction algorithm that takes other information (e.g., the route configuration, historical arrival information) into account when producing estimates. Therefore, it is in the best interested of transit riders to correctly consume each of these individual predictions, as it should result in more accurate estimates being shown to the transit rider. The research team developed improvements to OneBusAway to correctly handle multiple predictions per trip, including the specific issues discussed in the following subsections.

Per stop predictions resulted in large delays for stops that the bus has passed

When testing OneBusAway with the PSTA GTFS-rt data, the research team saw large delays (e.g., 20 minutes) when viewing estimated arrival times in the mobile apps. OneBusAway was erroneously propagating predictions upstream of the stop for which the prediction was intended. This manifested in the app as a trip remaining in the upcoming arrivals list after the bus passes the stop, with a delay value that continues to grow until the bus has completed that trip.

The research team created a software patch to resolve this issue and stop propagating delays upstream of the intended stop [19].

Departure predictions were not used by OneBusAway

The research team encountered a problem where the initial prediction for the first stop in a trip wasn't showing up in the OneBusAway app. Upon further investigation, the research team found that OneBusAway was only designed to consume per stop *arrival* times from GTFS-rt feeds. The research team developed a software patch to resolve this issue and consume departure times [19].

Interpolation of missing arrival times in trips

The research team encountered an issue with OneBusAway's interpretation of missing arrival predictions.

For example, if the following stop_ids exist:

- 1
- 2
- 3
- 4

…and the following deviations from stop_time_updates are in GTFS-rt data:

- A
- --- (no data)
- B
- C

…when searching for the deviation for stop_id 2, OneBusAway attempted to interpolate the deviation value based on the A and C deviations. The interpolation software for OneBusAway was originally created prior to the development of GTFS-rt, and as a result this behavior did not follow the GTFS-rt specification. According to the GTFS-rt specification, the deviation A provided for stop_id 1 should be propagated to stop_id 2, without any modifications. These portions of OneBusAway were created prior to the existence of the GTFS-rt specification, and they needed to be updated to be compliant with the GTFS-rt format. The research team developed a software patch to resolve this problem and correctly follow the GTFS-rt propagation rules [19].

Delay incorrectly prioritized over time for non-timepoints

The research team encountered a problem where OneBusAway was not showing real-time information for some stops in the middle of a trip with the following data:

```
stop_time_update {
   stop_sequence: 12
   arrival {
      time: 1436969397
      delay: 60
   }
   stop_id: "4995"
}
```

The PSTA GTFS schedule data did not provide scheduled arrival and departure times for this stop, as it was not a timepoint (the GTFS specification has since been updated to encourage provides to provide scheduled times for non-timepoints as well). And, OneBusAway was incorrectly prioritizing the “delay” value over the “time” value if both were provided in the feed. The end result was the app failing to show a real-time prediction for this stop, because there was no schedule value to apply the “delay” to, which was needed to calculate the final predicted arrival time. The research team modified OneBusAway to follow the GTFS-rt specification and use the provided “time”, if both “delay” and “time” values are in the GTFS-rt feed. This allowed OneBusAway to properly show the predicted arrival time to the user, even if the scheduled arrival time wasn’t specified at that stop.

Different interpretations of the GTFS-rt specification

Several scenarios were encountered where erroneous information was shown to transit riders, but the cause could not be attributed to a clear problem in the producer or the consumer software given the current wording of the GTFS-rt specification. Instead, these issues occurred because the producer and consumer interpreted certain portions of the GTFS-rt spec differently. These “gray areas” of the spec resulted in a discussion among the members of the GTFS-rt community, followed by a proposal by the research team to amend the specification and make the expected behavior of consumers and producers under these scenarios clear. The following subsections discuss each of these areas where the GTFS-rt specification was improved.

Scheduled times are shown if a GTFS-rt producer aggressively drops predictions

The research team encountered a problem when predictions were dropped from the GTFS-rt feed for a stop just before or after a bus visited that stop. In these cases, if a vehicle was running early the user would see real-time information in the app until the bus arrived, and then the arrival time would jump back to the scheduled arrival time (even though the data indicated that the vehicle already left).

The research team worked with the GTFS-rt community to clarify within the GTFS-rt specification the GTFS-rt feeds should not drop arrival predictions from a feed until after the scheduled arrival time for trips running early [20] and the AVL vendor updated their feed appropriately, and the research team developed a software patch to handle this issue in OBA until the AVL vendor was able to update their GTFS-rt feed.

Predictions for loop routes weren't matched if stop_sequence was missing

The research team encountered a problem where large, incorrect delays were being shown for loop trips in OneBusAway. The problem was eventually traced to the GTFS-rt feed providing ambiguous predictions for stops that appear twice in the trip – in other words, the GTFS-rt feed was missing the stop_sequence for loop trips.

For example, with the following GTFS schedule data:

- stop_id = 1756, stop_sequence=1
- …
- stop_id = 1756, stop_sequence=30

…if the GTFS-rt data includes an arrival prediction and only specifies that it should apply to stop_id = 1756, but not which "instance" or stop_sequence, OneBusAway does not have enough information to know which stop it should be matched to. In some cases, this resulted in arrival predictions for the later occurrence of the stop being applied to the earlier occurrence of the stop, which showed up in the app as large delays for each stops in the trip.

The research team worked with the GTFS-rt community to require that GTFS-rt feeds include the stop_sequence field if a stop is visited more than once in the same trip [21] and the AVL vendor updated their feed appropriately, and the research team also improved OneBusAway's handling of this situation.

Stops upstream of predictions have unknown delay

In the process of attempting to clarify behavior for producers as to when they are allowed to drop per-stop predictions, it became apparent that the AVL vendor was assuming that when using per-stop predictions, consumers could either propagate predictions upstream or hold over predictions from a previous feed update and show these to end users.

The research team proposed a clarification to the GTFS-rt spec that that in the absence of any predictions upstream of a stop-specific prediction, it should be assumed that these upstream stops have an unknown delay [22]. This proposal was accepted into the GTFS-rt specification following a vote by the community.

IMPROVING THE QUALITY OF GTFS-REALTIME FEEDS

The research team gained valuable insight into the challenges of launching a real-time transit information system during the efforts described in the previous section. Perhaps the most valuable lesson learned is that deploying a new GTFS-rt feed along with a mobile transit app can take a significant amount of effort due to the manual process required to detect problems from the transit rider's perspective, troubleshoot the cause of the issue (including searching log files or trying to reproduce the problem at the same time and day of the week), and resolve the problem in the producer and/or consumer software. This process increases the deployment costs to the AVL vendor (which are then passed along to the transit agency, both directly as a cost of the product as well as indirectly through transit agency staff time required to help troubleshoot problems) as well as the mobile transit app. Additionally, as discussed in the previous section, there may be certain areas of the GTFS-rt specification where the exact correct behavior of the producer and/or consumer isn't 100% clear – this ambiguity also causes additional troubleshooting time.

Based on these lessons learned, the research team pursued two efforts to reduce the effort required to launch and maintain high quality GTFS-rt feeds, which are discussed in the following sections.

Clarifying "Required" and "Optional" fields in the GTFS-realtime Specification

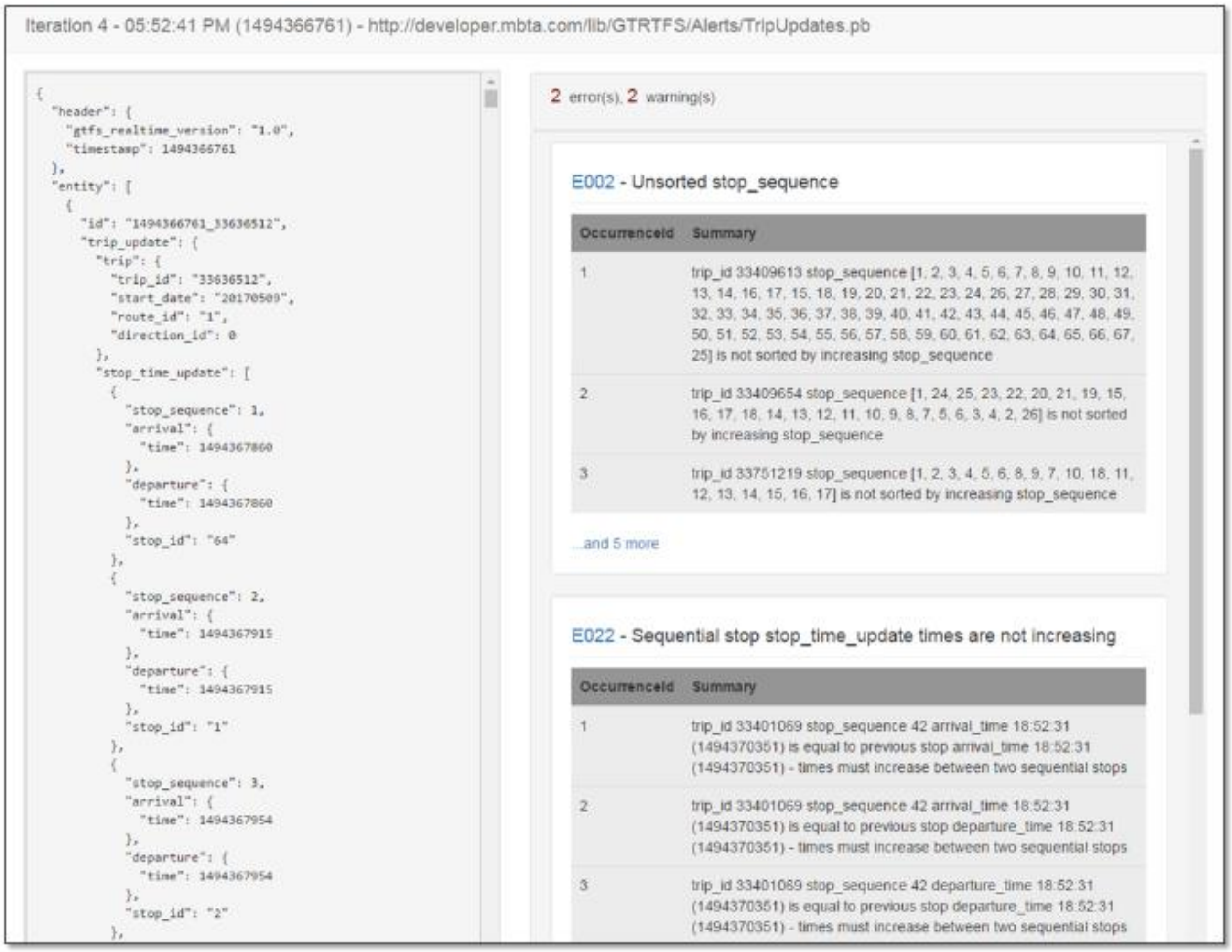
One common point of confusion with v1.0 of the GTFS-rt feed specification is that there has not been a clear indication of which data fields are required and which fields are optional. The root cause of this confusion is that while the specification has a field labeled "cardinality" that defines whether each field is "required", "repeated", and "optional", this is the Protocol Buffer cardinality, not the semantic cardinality, of each field [12, 23]. Protocol Buffer cardinality simply defines whether software parsing the binary message expects a field to exist – it has no direct mapping to GTFS or transit-specific logic. This becomes a problem because many software engineers choose not to label any Protocol Buffer fields as "required" because of forwards-compatibility issues with Protocol Buffer implementations [12, 24]. As a result, nearly all fields in the GTFS-rt specification are shown as "optional", even if that field is necessary for a transit app to show proper real-time information to a transit rider.

The research team created a proposal for v2.0 of the GTFS-rt specification [25] that clearly defines semantic cardinality for each field, or the conditions under which each field is required, conditionally required, or optional based on transit-specific logic and use cases. GTFS-rt v2.0 was approved by the community and published on August 29th, 2017 [26].

GTFS-realtime Validation Tool

As mentioned earlier, the process to manually identify and troubleshoot problems in GTFS-rt feeds can be extremely time consuming. Additionally, the process to examine feeds requires a significant amount of expertise with the GTFS and GTFS-rt specification, which limits the number of people that can evaluate a feed for potential problems.

To address these problems, the research team created an open-source GTFS-realtime Validator software tool [27] that can monitor GTFS-rt feeds (Trip Updates, Vehicle Positions, Service Alerts) and log any encountered problems.



Iteration 4 - 05:52:41 PM (1494366761) - http://developer.mbta.com/lib/GTRTFS/Alerts/TripUpdates.pb

```
{
  "header": {
    "gtfs_realtime_version": "1.0",
    "timestamp": 1494366761
  },
  "entity": [
    {
      "id": "1494366761_33636512",
      "trip_update": {
        "trip": {
          "trip_id": "33636512",
          "start_date": "20170509",
          "route_id": "1",
          "direction_id": 0
        },
        "stop_time_update": [
          {
            "stop_sequence": 1,
            "arrival": {
              "time": 1494367860
            },
            "departure": {
              "time": 1494367860
            },
            "stop_id": "64"
          },
          {
            "stop_sequence": 2,
            "arrival": {
              "time": 1494367915
            },
            "departure": {
              "time": 1494367915
            },
            "stop_id": "1"
          },
          {
            "stop_sequence": 3,
            "arrival": {
              "time": 1494367954
            },
            "departure": {
              "time": 1494367954
            },
            "stop_id": "2"
          },
```

2 error(s), 2 warning(s)

E002 - Unsorted stop_sequence

OccurrenceId	Summary
1	trip_id 33409613 stop_sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 15, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 25] is not sorted by increasing stop_sequence
2	trip_id 33409654 stop_sequence [1, 24, 25, 23, 22, 20, 21, 19, 15, 16, 17, 18, 14, 13, 12, 11, 10, 9, 8, 7, 5, 6, 3, 4, 2, 26] is not sorted by increasing stop_sequence
3	trip_id 33751219 stop_sequence [1, 2, 3, 4, 5, 6, 8, 9, 7, 10, 18, 11, 12, 13, 14, 15, 16, 17] is not sorted by increasing stop_sequence

...and 5 more

E022 - Sequential stop stop_time_update times are not increasing

OccurrenceId	Summary
1	trip_id 33401069 stop_sequence 42 arrival_time 18:52:31 (1494370351) is equal to previous stop arrival_time 18:52:31 (1494370351) - times must increase between two sequential stops
2	trip_id 33401069 stop_sequence 42 arrival_time 18:52:31 (1494370351) is equal to previous stop departure_time 18:52:31 (1494370351) - times must increase between two sequential stops
3	trip_id 33401069 stop_sequence 42 departure_time 18:52:31 (1494370351) is equal to previous stop departure_time 18:52:31 (1494370351) - times must increase between two sequential stops

Figure 1 – The GTFS-realtime Validator tool shows feed data (left) along with any errors/warnings (right)

The user simply enters URLs for their GTFS and GTFS-rt datasets, as well as how frequently the tool should fetch GTFS-rt updates (the default is 10 seconds). After starting the monitoring session, the user is shown a log view with the types of errors logged for each iteration (i.e., fetch) of the GTFS-rt feed. The user can click on the iteration ID to see all the occurrences of the errors and warnings for that iteration (Figure 1) – there can be multiple occurences of most errors and warnings in a single feed iteration. Because GTFS-rt feeds can be updated every few seconds, the tool enables an observer to capture critical data for troubleshooting problems in a log format that can be browsed and saved for further analysis.

The GTFS-realtime Validator has a modular rule architecture that allows new errors and warnings to be easily added to the tool as the GTFS-rt specification continues to evolve and new problems are discovered in feeds. As of November 15th, 2017 the research team has implemented rules to detect 45 types of errors and 9 types of warnings that appear in feeds, many of which were encountered in the team's experience described earlier in this paper. An error is logged when data in the feed is incorrect and would result in a transit rider seeing bad or missing real-time information as a result. A warning is logged when a feed contains data that would negatively affect some GTFS-rt consuming applications but either cannot be confirmed to be incorrect with 100% certainty based on data in the feed (e.g., a very large speed value for a vehicle) or the GTFS-rt specification does not clearly indicate that the data or behavior is incorrect (e.g., it is a best practice to refresh feed contents frequently, but the GTFS-rt specification doesn't require a minimum update frequency).

Validation rules can be broken down into the following categories:

- Header – Checks if header fields (e.g., feed version) are populated correctly
- Timestamps – Checks integrity of feed timestamps (e.g., in POSIX format, age of feed, sequential arrival/departure times are in increasing order)
- Stop Time Updates – Checks the integrity of predictions provided for each trip (e.g., order by stop_sequence, missing field values, conflicts with GTFS stop_times.txt data)
- Stops – Checks that stop information provided in the feeds matches GTFS stops.txt (e.g., stop_id, location_type)
- Trip Descriptors – Checks integrity of trip properties (e.g., conflicts with GTFS data, missing data for certain use cases, trip start date formats)
- Vehicle – Checks integrity of vehicle properties (e.g., valid position/bearing formats, unrealistic speed values that may be unit conversion errors, proximity of real-time position to assigned GTFS trip)
- Cross Feed – If multiple feeds entity types exist (e.g., VehiclePositions and TripUpdates), checks if content in one set of entities matches the content in the other set of entities (e.g., that all trip_id and vehicle_id pairings are consistent)
- Frequency Type Zero Trips – Checks conditions specific to trips defined in frequencies.txt with exact_times = 0 (i.e., true frequency/headway-based service)
- Frequency Type One Trips – Checks conditions specific to trips defined in frequencies.txt with exact_times = 1 (i.e., scheduled service modeled using a specified headway interval)

A detailed description of all rules is documented on Github [28].

It is important to note that as of November 2017, the GTFS-realtime Validator tool does *not* detect errors in the arrival or departure predictions themselves (i.e., whether a vehicle actually arrived or departed when it was predicted). Current rules therefore focus on data integrity (i.e., if the data logically correct given the GTFS-realtime specification and GTFS schedule data). Prediction accuracy analysis, as discussed late, is a potential future area of work.

Evaluation of Industry GTFS-rt Feeds

To demonstrate the utility of the GTFS-realtime Validator, the research team developed another tool [30] to automate the validation of a large number of feeds.

This analysis tool:

1. Retrieves the URLs for GTFS-realtime feeds and corresponding GTFS data from the TransitFeeds.com GetFeeds API (a centralized directory for GTFS and GTFS-realtime feed URLs)
2. Downloads a snapshot of the GTFS-realtime and GTFS data from each agency's server into a subdirectory
3. Runs the GTFS-realtime Validator on each of the subdirectories
4. Produces summary statistics and graphs for all validated feeds

While TransitFeeds.com shows a total of 130 GTFS-rt feeds that have been registered with the system [29], the team has so far been able to automate the validation of 78 feeds (future work will focus on improving this number by supporting feeds that require API keys or use HTTP redirects).

Out of the 78 feeds evaluated, 54 of the feeds contained errors, and 58 of the feeds contained warnings (Figure 2).

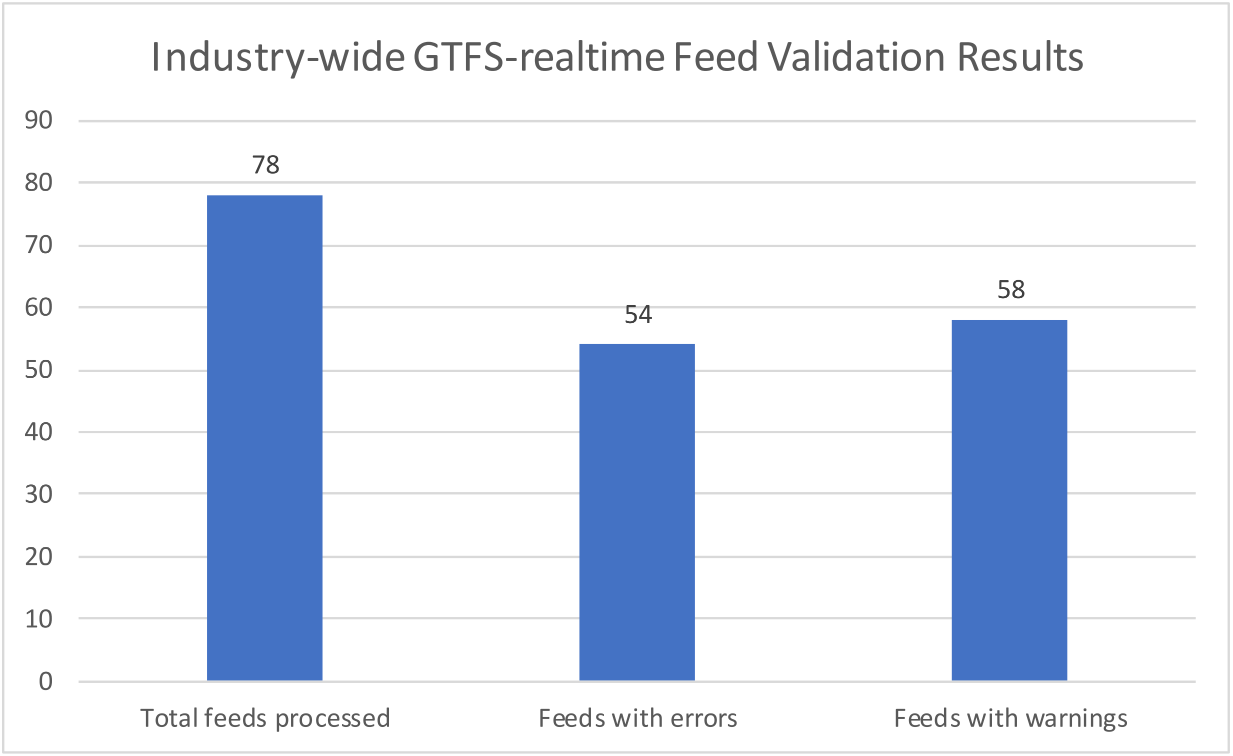


Figure 2 – 69% of tested GTFS-realtime feeds (n=78) contained significant errors, and 74% contained warnings

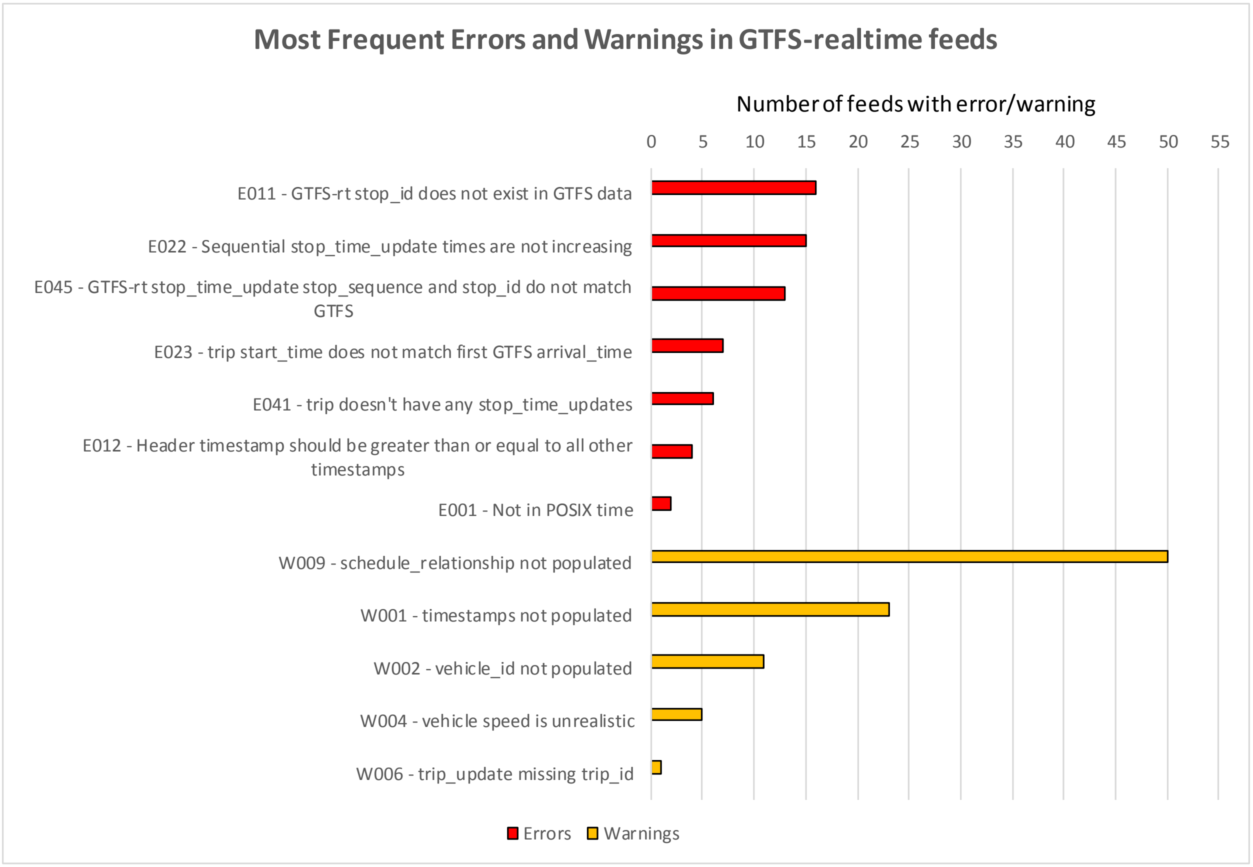


Figure 3 – GTFS-rt feeds containing stop_ids that don't appear in GTFS data was the top error

Figure 3 indicates the most common errors and warnings that appeared in feeds. "E011 – GTFS-rt stop_id does not exist in GTFS data" was the most common error, appearing in 16 feeds. E011 means that the GTFS schedule data has no record of a stop that the GTFS-rt data is showing a prediction for, indicating an incorrect stop_id either in the GTFS or GTFS-rt data. The 2nd most common error was "E022 – Sequential stop_time_update times are not increasing" appearing in 15 feeds (which indicates that predicted times are wrong - the vehicle would be traveling backwards in time). "E045 - GTFS-rt stop_time_update stop_sequence and stop_id do not match GTFS" appeared in 13 feeds – this means that the GTFS-rt data shows a conflicting order of arrival for stops for a trip when compared to the GTFS data.

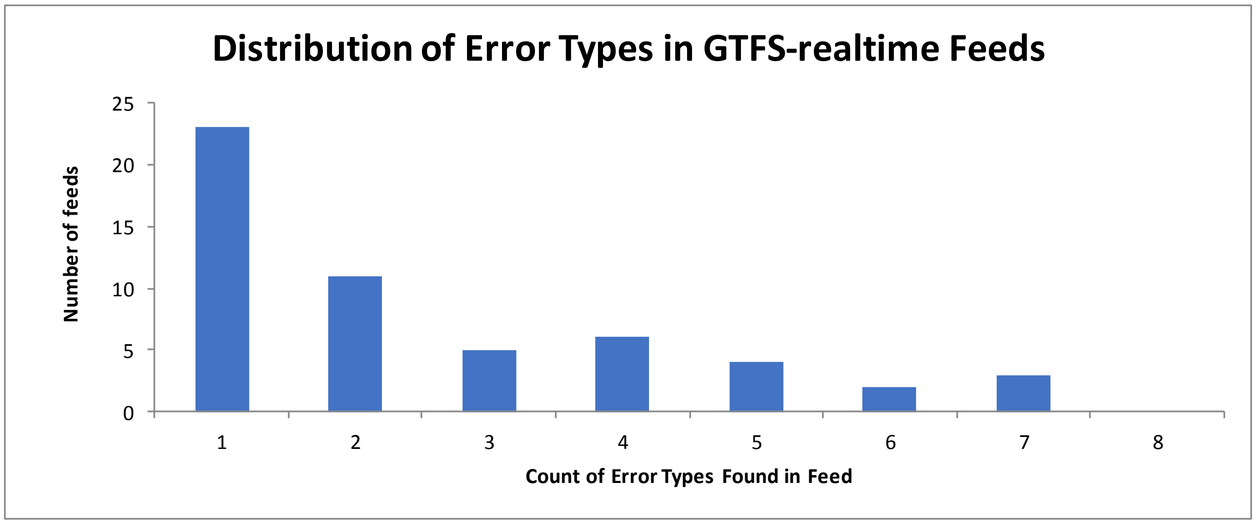


Figure 4 – Most feeds had 5 or fewer types of errors, but many occurrences of those errors

Figure 4 shows the distribution of the count of error types found in feeds. For example, the feed with the worst performance had 7 different types of errors found, while 23 feeds had only one error type found. Even though the majority of feeds had 2 or fewer types of errors, as mentioned earlier, some errors can occur multiple times in the same feed iteration, as well as in multiple iterations of the feed. For example, in Feed 51 that had eight different types of errors, there were 24 occurrences of "E022 - Sequential stop_time_update times are not increasing" in a *single* feed iteration. Each of these occurrences can have a significant impact on the transit rider experience, as discussed in the following section.

It should be noted that all of the above analysis is for a single iteration of each of the 78 evaluated feeds. It is highly likely that if the validator was executed over several hours of time additional errors and warnings would be found for each feed. Future work will focus on enhancing the analysis tool to automate data collection for a large number of feeds over an extended time period.

The research team is sharing this data, as well as the validation tool itself, with transit agencies and their AVL vendors to help them resolve issues in their real-time feed software, which should result in better real-time information for transit riders.

CONCLUSIONS

Based on the experience gained from deploying a GTFS-rt feed and multimodal transit app, as well as the development and testing of the GTFS-realtime Validator tool, the transit industry must focus on real-time data quality as well as data availability. The number of errors and warnings found in industry feeds reflect significant data issues that impact riders and, based on research, leads to reduced ridership and satisfaction with the transit agency and its service. Real-time data that contains integrity issues (e.g., trips with out-of-sequence predictions or conflicts with GTFS data) are very problematic for transit apps to parse; many transit apps, including Google Maps, the Transit App, and OneBusAway, will drop all predictions for that trip, resulting in users seeing the schedule information instead of real-time information. The good news, however, is that research shows good quality real-time data leads to increased ridership and satisfaction with the agency. Transit agencies can focus on improving data quality by getting involved with the GTFS-rt improvement process [31] and voting for proposals that clarify how producers and consumers should interact. Agency can also use the GTFS-realtime Validator tool when creating and maintaining GTFS and GTFS-rt feeds to ensure that no errors and warnings occur, and require that their AVL vendor (including during the Request for Proposals process) also use such a validation tool before feeds will be accepted.

As mentioned earlier, it should be noted that as of November 2017, the GTFS-realtime Validator tool does not detect errors in the predictions themselves (i.e., whether a vehicle actually arrived or departed when it was predicted), which is another significant source of problems encountered by riders. Future work should examine adding prediction accuracy analysis to the GTFS-realtime Validator.

Future work can also focus on enhancing the automated analysis tool to increase both the number and duration of feeds evaluated.

ACKNOWLEDGMENTS

The project deploying OneBusAway to multiple agencies in Tampa Bay as part of an improved issue reporting infrastructure was funded by the National Center for Transit Research and the Florida Department of Transportation. The opinions, findings and conclusions expressed in this publication are those of the author(s) and not necessarily those of the Florida Department of Transportation or the U.S. Department of Transportation. The author would also like to thank the Florida Department of Transportation District 7 (including Project Manager Elba Lopez), Hillsborough Area Regional Transit and Pinellas Suncoast Transit Authority for their participation in the project, as well as Cambridge Systematics that supports OneBusAway for HART, as well as Clever Devices, PSTA's AVL vendor. Thanks also to Cagri Cetin for his invaluable work implementing many software components critical to the success of this project and troubleshooting and documenting data issues. The development of the GTFS-rt v2.0 proposal to define semantic cardinality and the development of the GTFS-rt Validator software has been funded by the National Institute for Transportation and Communities (NITC) and is based on earlier work funded by the Google Summer of Code program. The contents of this paper reflect the views of the authors, who are solely responsible for the facts and the accuracy of the material and information presented herein. Thanks to Mohan Gandhi Achchakkagari, and Nipuna Gunathilake for the software development work on the GTFS-rt Validator, and Surya Vamshi Kandukoori for software development on the tool as well as data collection and graphs for errors in production GTFS-rt feeds.

REFERENCES

[1] Kari Edison Watkins, Brian Ferris, Alan Borning, G. Scott Rutherford, and David Layton (2011), "Where Is My Bus? Impact of mobile real-time information on the perceived and actual wait time of transit riders," *Transportation Research Part A: Policy and Practice,* Vol. 45 pp. 839-848.

[2] C. Cluett, S. Bregman, and J. Richman (2003). "Customer Preferences for Transit ATIS," Federal Transit Administration.

[3] Brian Ferris, Kari Watkins, and Alan Borning, "OneBusAway: results from providing real-time arrival information for public transit," presented at the Proceedings of the 28th international conference on Human factors in computing systems, Atlanta, Georgia, USA, 2010.

[4] A. Gooze, K. Watkins, and A. Borning (2013), "Benefits of Real-Time Information and the Impacts of Data Accuracy on the Rider Experience," in *Transportation Research Board 92nd Annual Meeting*, Washington, D.C., January 13, 2013.

[5] Lei Tang and Piyushimita Thakuriah (2012), "Ridership effects of real-time bus information system: A case study in the City of Chicago," *Transportation Research Part C: Emerging Technologies*, Vol. 22 pp. 146-161.

[6] C. Brakewood, G. Macfarlane, and K. Watkins (2015), "The impact of real-time information on bus ridership in New York City," *Transportation Research Part C: Emerging Technologies*, Vol. 53 pp. 59-75.

[7] C. Brakewood, S. Barbeau, and K. Watkins (2014), "An experiment evaluating the impacts of real-time transit information on bus riders in Tampa, Florida," *Transportation Research Part A: Policy and Practice*, Vol. 69 pp. 409-422.

[8] Google, Inc. "General Transit Feed Specification Reference." Accessed July 31, 2017 from https://github.com/google/transit/blob/master/gtfs/spec/en/reference.md

[9] MapZen. "TransitLand - An Open Project - For Data Providers." Accessed July 31, 2017 from https://transit.land/an-open-project/

[10] Luqmaan Dawoodjee. "GTFS Validators." Accessed November 15, 2016 from https://github.com/luqmaan/awesome-transit#gtfs-validators

[11] Massashusetts Bay Transportation Authority. "Developer Portal." Accessed November 14, 2017 from http://realtime.mbta.com/Portal/Home/Documents

[12] GTFS-realtime Google Group. "Proposal: Make FeedHeader.timestamp a required field." Accessed January 2015 from https://groups.google.com/forum/#!msg/gtfs-realtime/wm3W7QIEZ9Y/DLyWKkknJyoJ

[13] S. Barbeau (2013), "Open Transit Data – A Developer's Perspective," in *APTA TransITech 2013*, Phoenix, Arizona, March 20th, 2013.

[14] Google, Inc. "Protocol Buffers." Accessed July 31, 2017 from https://developers.google.com/protocol-buffers/

[15] OneBusAway. "OneBusAway - The Open Source Platform for Real Time Transit Info." Accessed August 1, 2017 from http://onebusaway.org/

[16] Sean J. Barbeau (2018), "Closing the Loop - Improving Transit Through Crowdsourced Information," in *Transportation Research Board 97th Annual Meeting*, Washington, D.C., January 7-11, 2018.

[17] OneBusAway Organization. "OneBusAway Github Source Code Repository." Accessed July 31, 2017 from https://github.com/OneBusAway/onebusaway-application-modules

[18] S. Barbeau. "PSTA Data Issues." Accessed July 31, 2017 from https://github.com/CUTR-at-USF/psta-data/issues?q=is%3Aissue

[19] Sean J. Barbeau. "onebusaway-application-modules Pull Request #142 - Fix per-stop predictions (#127, #138, and #139)." Accessed November 14, 2017 from https://github.com/OneBusAway/onebusaway-application-modules/pull/142

[20] Sean J. Barbeau. "Pull Request #16 - Clarify behavior for dropping StopTimeUpdates for vehicles running ahead of schedule." Accessed November 14, 2017 from https://github.com/google/transit/pull/16

[21] Sean J. Barbeau. "Pull Request #20 - Conditionally require stop_sequence in StopTimeUpdate." Accessed November 14, 2017 from https://github.com/google/transit/pull/20

[22] Sean J. Barbeau. "Pull Request #18 - Clarify that stops upstream of predictions have unknown delay." Accessed November 14, 2017 from https://github.com/google/transit/pull/18

[23] Google, Inc. "GTFS Realtime Reference." Accessed August 1, 2017 from https://github.com/google/transit/blob/master/gtfs-realtime/spec/en/reference.md

[24] Google, Inc. "Protocol Buffers - Specifying Field Rules - Required is Forever." Accessed July 31, 2017 from https://developers.google.com/protocol-buffers/docs/proto#specifying-field-rules

[25] Sean J. Barbeau. "Pull Request #64 - Define semantic cardinality for GTFS-realtime fields." Accessed November 14, 2017 from https://github.com/google/transit/pull/64

[26] Sean J. Barbeau, "What's new in GTFS-realtime v2.0," Vol. 2017, ed. Medium, 2017.

[27] University of South Florida. "GTFS-realtime Validator." Accessed November 14, 2017 from https://github.com/CUTR-at-USF/gtfs-realtime-validator

[28] University of South Florida. "GTFS-realtime Validator - Implemented Rules." Accessed November 14, 2017 from https://github.com/CUTR-at-USF/gtfs-realtime-validator/blob/master/RULES.md

[29] TransitFeeds.com. "TransitFeeds.com." Accessed November 10, 2016 from http://transitfeeds.com/

[30] University of South Florida. "transit-feed-quality-calculator." Accessed November 15, 2017 from https://github.com/CUTR-at-USF/transit-feed-quality-calculator

[31] Google, Inc. "GTFS-realtime Specification Amendment Process." Accessed November 14, 2017 from https://github.com/google/transit/blob/master/gtfs-realtime/CHANGES.md