10-1-2019

Fast Track: Allowing Bikes To Participate In A Smart-Transportation System

Stephen Fickas
University of Oregon

Marc Schlossberg
University of Oregon

NITC
NATIONAL INSTITUTE for TRANSPORTATION and COMMUNITIES

Final Report 1160
October 2019

Fast Track: Allowing Bikes to Participate in a Smart Transportation System

Stephen Fickas, Ph.D.
Marc Schlossberg, Ph.D.

UNIVERSITY OF OREGON

FAST TRACK: ALLOWING BIKES TO PARTICIPATE IN A SMART TRANSPORTATION SYSTEM

Final Report

NITC-RR-1160

by

Stephen Fickas
Marc Schlossberg
University of Oregon

for

National Institute for Transportation and Communities (NITC)
P.O. Box 751
Portland, OR 97207

October 2019

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NITC-RR-1160		

4. Title and Subtitle	5. Report Date
Fast Track: Allowing Bikes to Participate in a Smart Transportation System	October 2019
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Stephen Fickas and Marc Schlossberg	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Computer and Information Science Department Deschutes Hall University of Oregon	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Institute for Transportation and Communities (NITC) P.O. Box 751 Portland, OR 97207	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

17. Key Words	18. Distribution Statement
Smart Transportation, Active Transportation, V2X, IoT	No restrictions. Copies available from NITC: http://nitc.trec.pdx.edu

19. Security Classification (of report)	20. Security Classification (of page)	21. No. of Pages	22. Price
Unclassified	Unclassified	20	

ACKNOWLEDGMENTS

This project was supported with financial support from the National Institute of Transportation and Communities under grant number 1160. We are grateful to the City of Eugene Transportation Office, and Andrew Kading in particular. We also acknowledge the support provided to the project by software engineers involved with the McCain Transparity system.

DISCLAIMER

RECOMMENDED CITATION

Fickas, Stephen, Schlossberg, Marc. *Fast Track: Allowing bikes to participate in a smart transportation system*. NITC-ED-1160. Portland, OR: Transportation Research and Education Center (TREC), 2019.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

EXECUTIVE SUMMARY

This project focuses on a mode of transportation that is currently left out of V2X (vehicle-to-everything) conversations: bicycling. The project demonstrates how university researchers, city traffic engineers, and signal-controller manufacturers can come together to give bicyclists the same technology appearing on modern vehicles: Green Light Optimized Speed Advisory (GLOSA). GLOSA allows motorists to set their speed along corridors to maximize their chances of catching a “green wave” (i.e., not being forced to stop as they travel through the corridor). This project demonstrates how GLOSA can be used by bicyclists in the same way it is used by motorists on a busy car and bike corridor feeding the UO campus.

The project attempts to answer two questions: (1) Can we convey GLOSA information to a phone carried by a bicyclist? (2) Can we present that information in an effective, reasonable and safe manner using the interfaces available on the phone? For the former, we discuss our approach to combining (a) information provided by the city’s traffic office in terms of signal timing-tables with that of (b) a real-time traffic feed from the McCain Transparity server. We discuss a system architecture that we developed to use the real-time feed to keep an up-to-date view of signal state (current phase and its start time). However, if the feed was lost, the system continued to operate with its last known state information, potentially obtained by signal reset by the main office.

Our system architecture, and the app that employs it, uses the phone GPS and accelerometer sensors to keep an accurate view of the rider’s location, direction and speed.

For our second question, we describe an app we developed and a set of trials we ran to test it. Our overall results are positive. Testing revealed that our GLOSA-based app was effective, reasonable and safe on our test site.

As of July 2019, the system remains in beta testing with the goal to make it available to the public during the coming 12 months. Please contact Stephen Fickas (fickas@cs.uoregon.edu) for more information.

INTRODUCTION

In the rush to develop and install smart transportation (V2X) systems so that they can support connected and autonomous vehicles, people on bikes seem to be generally left out. At best, bicycles are viewed as obstacles to be “seen” and avoided by sensored-up cars. This is an extremely limited view of the future, especially at a time when cities are significantly investing in bicycle infrastructure and bike share systems, and where the private sector has discovered a possible transportation market in bike and scooter share systems.

This project is part of our larger interest in making bicycles and bicyclists full-fledged partners in all that is happening in the V2X world. In a prior NITC project, we focused on integrating cyclists purposefully into the connected systems environment by developing a direct means for people on bikes to interact virtually with actuated signal-controlled intersections in an eventual effort to improve safety and efficiency. Through the development of a low-cost traffic signal box add-on and a phone app, we were able to add virtual requests and useful information to cyclists' experiences at the location tested in Eugene, OR. See NITC report (Fickas, 2018) and Figure 1.

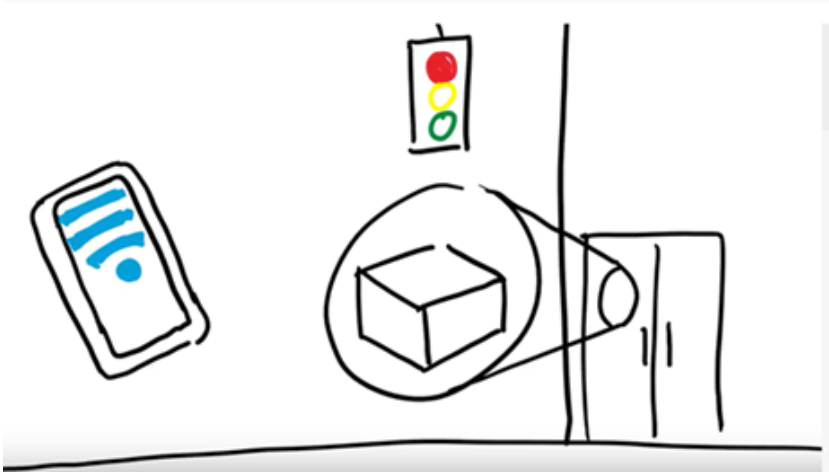

Figure 1: Illustration of Bike Connect System. As part of a prior-funded NITC project, we developed V2X technology by constructing a Bike Connect device that can communicate with a Bike Connect box (circled) to trigger a green light from a distance.

In this project we explore another component of the V2X space, the Green Light Optimized Speed Advisory (GLOSA) (Suzuki &Yoshitaka, 2018). This is more colloquially known as keeping a vehicle in the green wave: You are at a location and moving at a speed that will allow you to (theoretically) have a green light at each intersection you encounter along a corridor. If you are not in the green wave, then advice will be given on adjusting your speed. GLOSA systems are starting to appear in cities across the U.S. (e.g., Dallas; Denver; Gainesville, Fla.; Houston; Kansas City, Kan.; Las Vegas; Los Angeles; New York City; Orlando, Fla.; Phoenix; Portland, Ore.; San Francisco; and Washington, D.C.). Figure 2 shows a simple design of a GLOSA driver interface built into the car's speedometer. All of the GLOSA systems deployed to date have focused on motor vehicles. We need to bring the same convenience and safety benefits to the biking public.

Our interest is in providing an analog of Figure 2 for bike riders. We assume that a bike rider has a phone mounted on the handlebar and that our app (developed in this project) is active and visible. We expect the app to provide speed adjustments (increase speed, decrease speed, hold steady) to allow the rider to pass through the upcoming signal safely and without stopping. The following sections describe the test corridor we used, the challenges involved and the solutions we came up with.

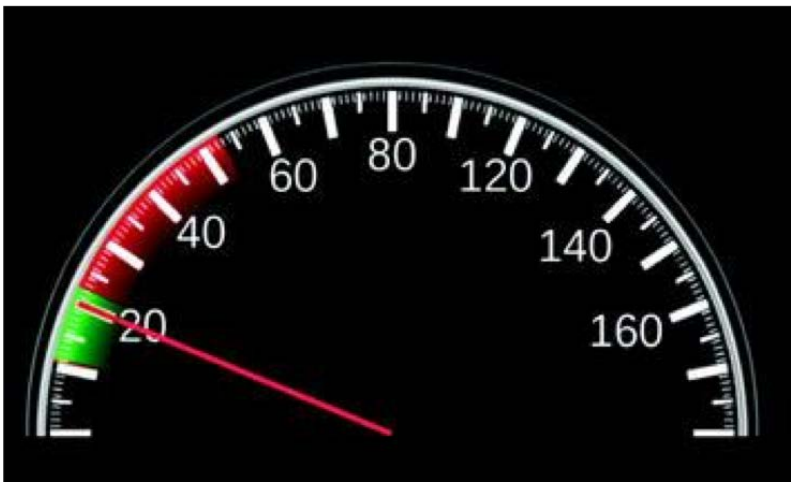

Figure 2: One of the GLOSA interfaces suggested for motor vehicles. The green band places the driver in the green wave.

TEST SITE: 13TH AVENUE CORRIDOR

For this project, we focused on a busy bike corridor that leads into the west campus entrance of the University of Oregon. The corridor lies along W 13th Avenue from Willamette Street (on the west end of the corridor) to Hilyard Street (on the east end of the corridor). The corridor is roughly .5 miles long and one-way east for both cars and bikes. It has six fixed-time semi-coordinated signals (discussed in more detail below). The clock reset from the main office for the signals is at 2 a.m. and8 p.m. each day. The speed limit along the corridor is 25 mph. Figure 3 shows a satellite view of the corridor with red dots denoting signals. Figure 4 shows a street-level view looking east on the corridor and approaching Willamette (1) with the bike lane on the right. The stop line in Figure 4 is the start of our trials. S

Signalized intersections:
1: Willamette Street
2: Oak Street
3: High Street
4: Pearl Street
5: Patterson Avenue
6: Hilyard Avenue

Non-Signalized intersections:
A: Mill Street
B: Ferry Street

Figure 3: The satellite view of the entire corridor. The start is signal 1 and the end is signal 6.

Figure 4: The view approaching the first signal on the corridor. The bike lane is on the right.

The corridor is semi-coordinated in that coordination exists between two pairs of signals but not for the entire corridor. In particular, the pair of signals at Willamette (1) and Oak (2) are coordinated. The pair of signals at High (3) and Pearl (4) are also coordinated for the speed limit. These gaps are **not** coordinated: between Oak (2) and High (3), between Pearl (4) and Patterson (5), and between Patterson (5) and Hilyard (6). Given that all signals are fixed-time, in theory they will all come into and out of alignment during the day. But given clock slippage, it is difficult to predict these times.

We view the semi-coordinated property of the corridor as a feature rather than a bug. It places more value on a GLOSA app, which needs to do real-time adjustments based on the shifting alignment in the gaps. In particular, if the entire corridor was coordinated, then a speed of 25 mph would typically allow a motorist to stay in the green wave for the entire corridor. But that is not the case with our test corridor. It is also not the case that we can reasonably expect a bike rider to maintain a 25 mph pace even if it maintained the green wave; a more reasonable biking speed is roughly half that. In summary, it becomes important to give a bike rider support in this rather challenging corridor.

OUR GLOSA GOALS

Our primary goal is to give bike riders along the 13th Avenue bike corridor a real-time display that shows GLOSA information. In particular, we want to let the bicyclist know, given their current location, direction and speed, whether they will reach the next signal

in their path with a green light (i.e., they are in the green wave for that signal). If they will not get a green given their current speed, we want to give them further information on *reasonable* speed adjustments they can take to bring them back into the green wave. Adjustment advice will specify whether to increase or decrease their speed and by how much.

There are three challenges to our goals. The back-end challenge is developing algorithms that will (a) take in the bike's current location and speed information using the device sensors and GPS, and (b) use built-in knowledge of the signal timing along the corridor to (c) produce speed adjustments that will place the rider in the green wave for the next signal. The front-end challenge is to take raw speed-adjustment data and display it to the bike rider in a usable fashion. Our expectation is that the bike rider will be using a phone with an internet connection mounted on the handlebars. Our third challenge is to architect a system that links technology components together to provide a scalable approach to the problem. We will discuss each of these challenges in the following sections.

BACK-END CHALLENGE: CALCULATING SPEED ADJUSTMENT

We obtained the timing table for all six signals along the corridor from the Eugene Traffic Operations Office. The timing table provides information on the length of each signal phase (only two phases per signal on our corridor). Theoretically, knowing the clock reset times and each signal's timing, we should be able to accurately predict greens all along the corridor at any time of day by dead reckoning; we can project out from reset time to current time using the timing tables. However, by doing on-site observations, we noticed significant drift in the theoretical start of green phases and the actual start of green phases. This ranged from 1.5 up to 8 seconds of drift. Traffic engineers noted that others had reported similar drift ranges on other fixed-time corridors in Eugene. The engineers could list likely reasons for this drift, including preemption by emergency vehicles and older, less sophisticated clocks on the controllers along the corridor. Whatever the reason, we would have to tackle the drift problem to be successful.

We made a key decision at this point. Instead of doing a deep-dive into predicting drift based on rigorous on-site monitoring of all the lights for extended hours and days (with no guarantee that a predictive model could be found), we decided to use another source of information. The traffic office agreed to give us a feed to their McCain Transparity server. McCain is the company that supplies the controllers for the City of Eugene. They, like many other controller manufacturers, track the real-time state of the controllers they deploy. The tracking information is based on events (phase changes) and not the clock on the controller. Phase-change events are time-stamped. This information is made available by a McCain Transparity server. The city pays McCain to gain access to the event information (i.e., they pay (rightly or wrongly) to get access to their own data). The city agreed to extend their license to give us access. With the

McCain feed, we can calibrate timing with real-time event feeds. It is analogous to accurately “resetting the clock” at frequent intervals (at the second level).

We followed the reset-the-clock analogy literally in our actual algorithm development. We noticed that there could be a delay of up to 5 seconds in transmitting event data to our phone app. And, in extreme cases, the server could go down for several hours. However, when events were produced, the time-stamp remained accurate, being done immediately upon receipt by the Transparity server and before the introduction of downstream transmission delays affecting delivery to our app. We wanted to be resilient to these delivery delays, including total loss of the server feed. Our approach is to write the speed-adjustment algorithm to rely on the last time of clock resetting, and then to augment clock resetting with receipt of the event data from McCain. In the ideal case, the simulated clock in our app would be reset second by second, based on actual phase change. In the worst case, the loss of McCain data for extended periods of time, the algorithm falls back to projecting forward from its last clock value. We expected to do no worse than what was possible before the inclusion of the McCain data and, on average, to do much better.

Of specific note, we are using a single-segment look-ahead in this exploratory study. Others have used algorithms to plot multi-segment adjustments that optimize speed by planning ahead. For instance, Seredynski et al. (2013) use a genetic algorithm to compute the “best” set of adjustments on all of the segments in front of a driver, where best is defined to be lowest fuel consumption, a function of speed and velocity (e.g., amount of necessary rapid acceleration). As discussed in the next section, our user interface defines best as any speed that will allow the bicyclist to make the next green within reasonable speed adjustments.

FRONT-END CHALLENGE: AN EFFECTIVE USER INTERFACE

Our goal is to give a bicyclist GLOSA-type information for a fixed-time bike corridor. We would like the interface to be effective (it provides speed adjustments that give the rider the best chance of making green lights), reasonable (it does not ask the bike rider for super-human performance), and safe (it does not force the rider to attend to the interface in a way that distracts from situational awareness). We chose to use two modes of information delivery based on our findings from a previous three-year NSF study of user interfaces for transportation information (Fickas et al., 2008). Our earlier study found a visual interface slightly easier to use, but also distracting in a way that audio is not. Some cyclists could use the visual interface effectively without being distracted while others worked most effectively with audio only.

We chose to use a cell phone as the interface device. For testing, we placed the phone in a holder on the handlebar and provided both visual and audio information to the bicyclist (see Figure 5). However, we also ran several extra tests with the phone in a backpack to test the audibility of the audio interface alone.

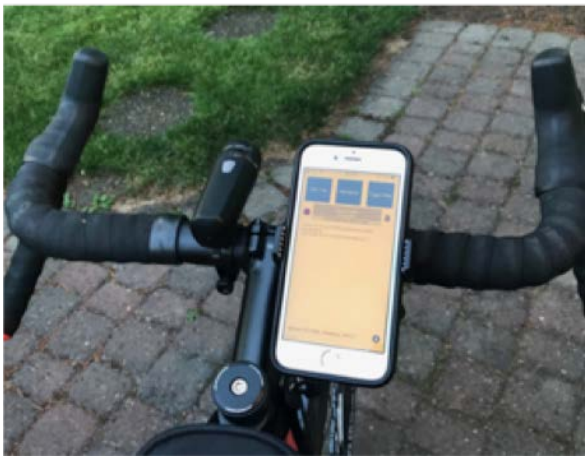

Figure 5: The project supplied the phone and holder for the test trials.

The visual interface we used in our trials is shown in Figure 6. It consists of two separate information displays. The first is a large area for a set of icons we developed for the project. The figure shows the check-mark icon. All possible icons are shown in Figure 7. When the user is stopped at a signal the X icon appears and remains until the light turns green (as predicted by the app).

For the trials we linked both small-adjustment arrows to changes of 2 mph or less. The large arrows we linked to changes greater than 2 mph. The one exception is when a rider is starting from a full stop. In this case, the app will display the small up-arrow icon for 5 seconds before calculating the actual adjustment. All of these values can be changed to a user's preference, but were held steady in our trials.

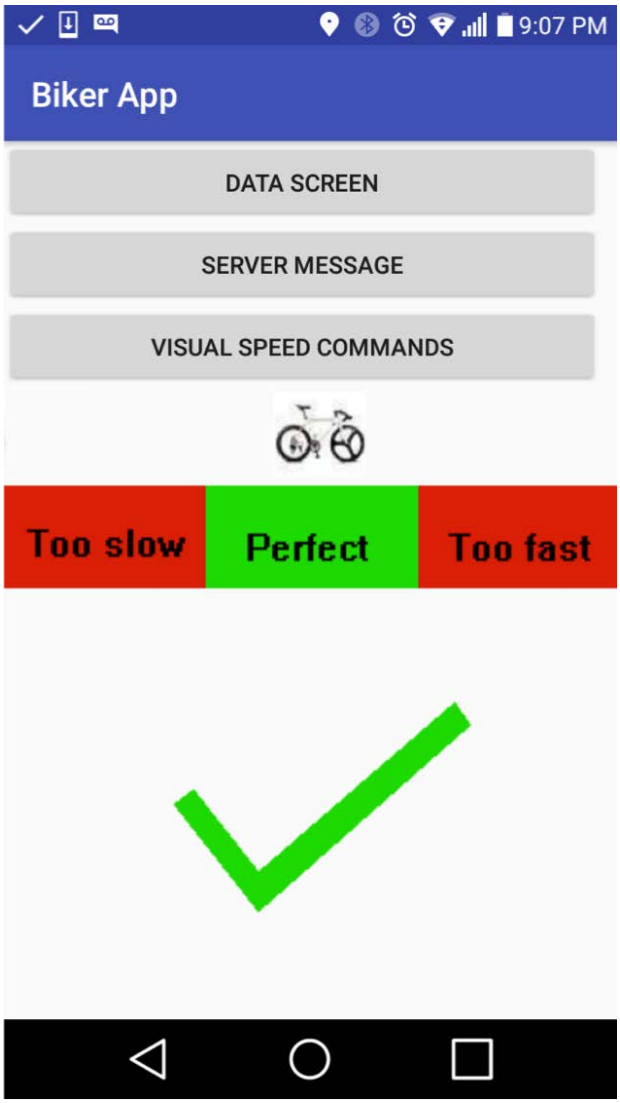

Figure 6: Delivers GLOSA information to a bike rider on a corridor. In the case shown, the rider is in great position to catch the next green.

The second visual display is of a bicycle that travels back and forth across the top of the three text boxes. It is meant to give a more fine-grained picture of where a bike rider is in terms of the green-wave interval. For instance, if the bike is straddling the Too slow and Perfect text boxes, the user can see that they are on the edge of falling out of the green-wave interval. When the user is stopped at a red light, the bicycle parks on the left-hand edge of the display.

The audio interface consists of six alternative messages: (1) “increase speed” (2) “reduce speed” (3) “in green-wave” (4) “impossible” (5) “entering corridor” and (6) “exiting corridor.” After initial tests, the repetitiveness of the adjustment messages was set at every 5 seconds. The green-wave message was delivered immediately when transitioning into the green wave and then every 10 seconds. The impossible message was delivered just once, as were the entering and exiting messages. We also note that these are all settable to each individual rider’s preference. In particular, at least one early tester would have liked to be reminded they were doing well more often; hearing

the green-wave message made riding “more fun.” However, we eventually set them to the values above for our final trials.

The tabs on the top of the screen are for use in debugging and were not used in the trials.

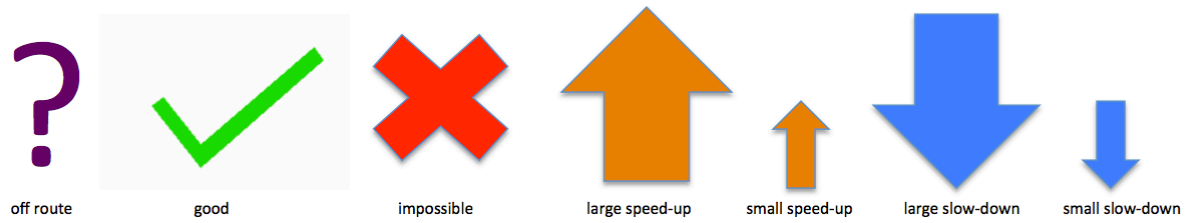

Figure 7: The possible icons available to the app to convey GLOSA adjustments to the rider.

THE SCALABLE ARCHITECTURE CHALLENGE

The architecture of our system is based on a publish/subscribe (pub/sub) model, and internet communication from our pub/sub server in the cloud to the phone. From a previous NITC project (Fickas, 2018), we found that a pub/sub architecture greatly simplifies the actual application code that needs to be written. And simpler code leads to fewer bugs and is easier to understand and maintain. The components of our architecture are as follows (please reference Figure 8 for a visual illustration):

- McCain maintains their own proprietary system for obtaining phase-change events from the signal controllers in Eugene. This is shown generally as the McCain Transparity server in Figure 8. The city and our project have access to this server through a certificate-protected API. The city bought a certificate that allows our project to access the API. The information available through the API is the current phase of any signal in the city, along with when that phase became active. This information is time-stamped. Knowing the timing table of each fixed-time signal allows us to predict when the current phase will end and the next phase will become active.
- We also maintain a V2X server in the cloud. We added a C# client to access the McCain API. The McCain API claims to support a push mechanism but we found it unreliable. Instead, we poll every second to get the latest information. Our C# client becomes a publisher to an MQTT broker that we describe next.
- Our MQTT broker also resides on our V2X server. It accepts published events from our C# client and redistributes that information to all subscribers.
- The client subscribers in our system are phones running our app. Once they subscribe to the broker (subscribe and unsubscribe actions are easy and fast), they receive events that they care about. To avoid receiving messages about signals not in their vacinity, the app can, in real time, tell the broker to filter out uninteresting data (e.g., only send events for the signals along the 13th Avenue corridor when the biker enters that corridor). Once the biker is out of the corridor, the app can tell the broker to cease any further messages until it approaches another corridor.

We found that the architecture we have chosen is a good fit with our project. We have a single publisher, the McCain Transparity server, and an unlimited number of app clients (bicyclists). These clients come and go in terms of their interest in subscribing during the day. The MQTT subscribe/unsubscribe protocol is quick and painless. The broker takes on the task of distributing information to clients who subscribe and allows those clients to filter what information they receive. We next discuss the actual algorithm we use in our phone app.

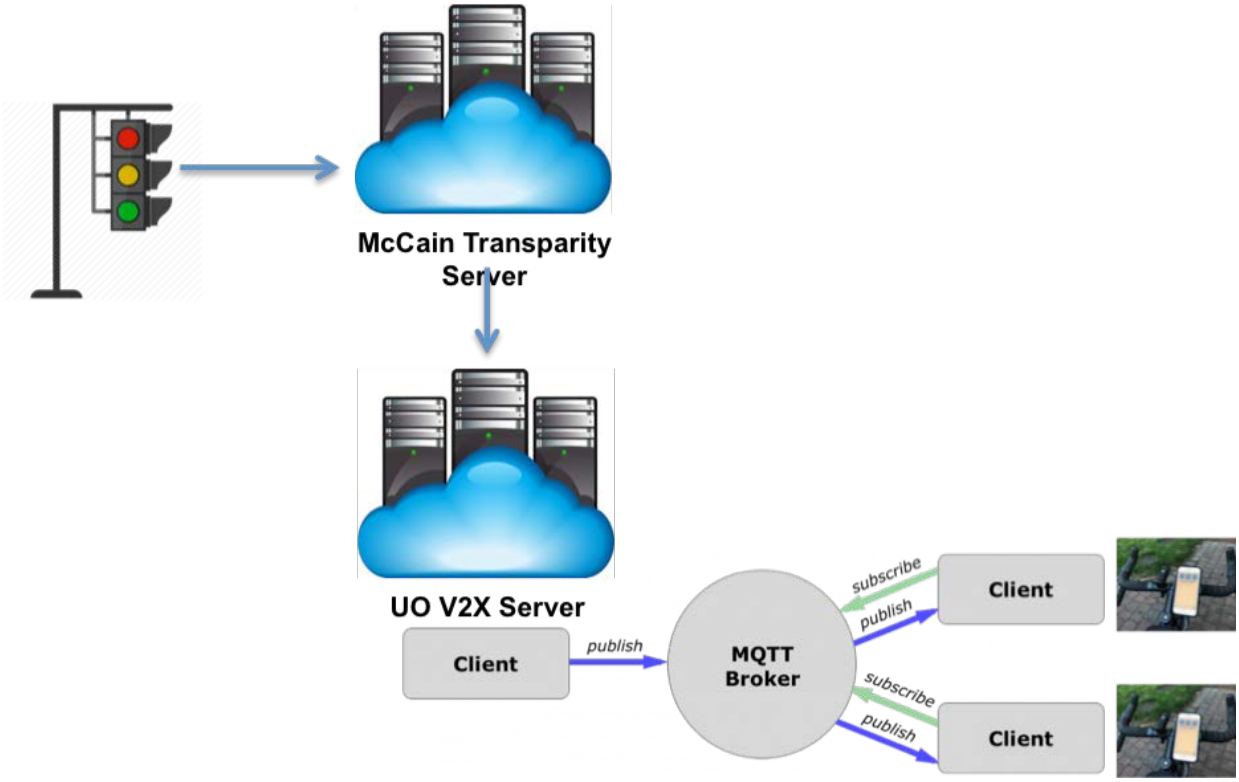

Figure 8: A view of the major components of the system architecture. The project developed the UO V2X Server and everything downstream from it.

THE APP ALGORITHM

The algorithm we use is necessarily asynchronous. Our app is responding to information it receives both from the outside (main office resets, MQTT broker) and from phone-based sensors and GPS. Receiving of information can be ordered and interleaved in arbitrary ways that we cannot predict ahead of time. We find that modeling interacting, communicating processes such as this is well suited to a formal modeling tool called LTSA (Labeled Transition System Analyzer). More can be found at https://www.doc.ic.ac.uk/ltsa/. We present our application architecture as a formal model in LTSA in Figure 9. For simplicity, we define the architecture for just one signal. By employing several of the powerful modeling tools that LTSA provides for abstraction, we have also modeled the full system (i.e., a system that allows an arbitrary number of signals to be modeled).

Referring to Figure 9, the top portion of the model represents the processes of the app and the bottom represents the environment the app operates in. Looking at the app

portion, we define a set of processes that can be viewed as event handlers. There are five in total. Each can be thought of as a process running on a separate thread. Each process is in a loop that waits for reception of a message/event. This is followed by the actions to perform. We compose the five processes into a larger component we call APP.

The EXTERNAL process has four possible sequences of actions. These sequences can be taken in any order. We cannot predict ahead of time the exact order we will see external messages and events. So, we leave the ordering as non-deterministic.

We pull the two components together in SIGNAL_SYSTEM. The code starting with "/{" sets up the communication paths between the separate processes.

There are several benefits in building an LTSA model. One, it provides the blueprint for the actual implementation. Our app program closely follows the model with five separate threads for the handlers. Two, we can formally verify that the model is correct for important properties. These include liveness, which guarantees that any action is possible in the future: no action is locked out permanently from occurring. *Deadlock* is a well-known example of failing the liveness property. Safety is another property we can check. In essence, are there a sequence of actions we never should take? If the LTSA analyzer finds a path to such an action sequence, the safety property is violated. We analyzed the model in Figure 9 and verified that neither liveness nor safety properties were violated.

```
/*
The clock is set to the last known start-time of the green phase. With this information
we can calculate forward (knowing the signal phase-timing) to future start times.

There are 2 ways for the clock to be reset: (1) it gets a clock reset signal from the main office,
or (2) we receive an mqtt message that gives us the start-time within seconds of it occurring.
*/
CLOCK_HANDLER = (
    receive_reset_msg -> get_reset_time -> move_forward -> rset_clock_signal -> CLOCK_HANDLER|
    receive_mqtt_msg -> move_forward -> mset_clock_signal -> CLOCK_HANDLER
    ).

LOC_DIR_HANDLER = (     //GPS gives both location and direction
    receive_gps_update -> set_user_location -> set_user_direction -> LOC_DIR_HANDLER
    ).

SPEED_HANDLER = (   //calulated by accelerometer
    receive_accelerometer_update -> set_user_speed -> SPEED_HANDLER
    ).

DISTANCE_HANDLER = (    //use location to compute distance to signal[ID]
    receive_location_update -> compute_distance_to_signal -> DISTANCE_HANDLER
    ).

ADJUSTMENT_HANDLER = (  //use distance to signal[ID] and speed and clock to compute adjustment
    receive_distance_update -> compute_adjustment_for_signal -> alert_user -> ADJUSTMENT_HANDLER
    ).

||APP = (CLOCK_HANDLER || LOC_DIR_HANDLER || SPEED_HANDLER || DISTANCE_HANDLER || ADJUSTMENT_HANDLER).

/*
The EXTERNAL process embodies all of the external events we might see. There
is no ordering on events. We want our system to work under any ordering.
*/
EXTERNAL = (
    clock_reset_event -> send_reset_msg -> EXTERNAL |
    mqtt_clock_publish -> send_mqtt_msg -> EXTERNAL |
    gps_timer_trigger -> read_gps_data -> send_gps_update -> EXTERNAL |
    accelerometer_timer_trigger -> read_accelerometer_data-> send_speed_update -> EXTERNAL
    ).

/*
We now integrate the app and external components into a single system. The majority of the code
below is defining the communication channels between external and app processes.
*/
||SIGNAL_SYSTEM = (EXTERNAL || APP)
            /{send_clock_reset_event/receive_reset, send_mqtt_message/receive_mqtt_msg,
            send_gps_update/receive_gps_event, send_speed_update/receive_accelerometer_update,
            set_user_location/receive_location_update, compute_distance_to_signal/receive_distance_update}.
```

Figure 9: A model of the app algorithm as a set of asynchronous communicating processes.

PROJECT RESULTS

Bike Trials

Our interest is in whether bicyclists can be plugged into the same technology that car manufacturers are making available to motorists. Can bicyclists continue to make a place for themselves in the car-centric world of V2X? In particular, can we give bike riders GLOSA information based on a real-time traffic feed (currently developed to feed to motorists) on a traffic corridor timed for motorists?

To answer these questions, we carried out an exploratory, qualitative study. Our focus was narrowly on an interface that could use V2X to be effective, reasonable and safe under typical conditions. We describe the methodology we used for our focus question next.

Methodology

We recruited bike riders who do bike regularly but do not normally ride the 13th Avenue corridor. We asked them to make individual trips down the corridor with our app visible on a handlebar-mounted phone. We gave them desired days and time slots to make their trip. We supplied the phone and the handlebar holder. Riders were first shown the app and given information on following its possible adjustments. Riders were shown how the app identifies that it is fully operational and ready to go. We asked riders to follow the app's directions as closely as possible within safety constraints, even if they believed the app's advice is incorrect.

We told riders that the app is set to "give up" (show an X) if it calculates a necessary speed adjustment that seems excessively slow or fast. (They would also see the X when stopped waiting for a green.) Note that we did **not** tell them the actual range we were using, discussed below. Our hope was that this would lead to a more honest assessment of their personal comfort zone when biking.

The actual setting on the app was between 8 and 18 mph. We based our values on statistics gathered in Copenhagen for urban biking speed when on a green wave: (Copenhagen study). The Copenhagen study found 13 mph as the average urban biking speed and we expanded by plus or minus 5 to get our range. If the app computed a speed less than 8 or greater than 18, it would show an X to the user. While we have the means to personalize the range for each rider, we held the range steady during the trials.

Each rider was given a backpack with a clipboard and pencil. A blank form of the table, as shown in Figure 10, was clipped to the clipboard. We also provided a separate sheet for individual comments. Riders were given an explanation of each of the table codes as follows.

In general, please ride within your comfort zone. If the app asks you to perform below or above what you feel comfortable doing, ignore the app's adjustment and use the codes below to note the problem after your ride.

Never follow the app's advice if it will lead to an outcome you judge unsafe. Instead, ignore the advice and use the codes below to note the problem after your ride.

Otherwise, follow the app's advice as closely as possible.

After each ride through the corridor, stop to fill out a new row using the codes below. For SLOT, use 1 of these 3 values based on the time you started your ride:

PRE 0600 to 0745

RUSH 0746 to 0945

POST 0946 to 1300

For each of the 6 cross streets:

- *Use **+** if the app successfully guided you through the signal.*

- *Use **0** if the app showed an X correctly. The app judges a green to be unmakeable and you concur. You could not have made it while staying in your comfort zone.*
- *Use **–R** if the app incorrectly shows an X. You judge a green light to be makeable but the app says it is unmakeable. You could have made it while staying in your comfort zone.*
- *Use **–G** if the app incorrectly shows adjustments. You judge a green light unmakeable but the app says it is makeable with the adjustments it provides. You could not have made it while staying in your comfort zone.*
- *Use -**E** for any problems not covered by other codes. Please leave a comment to provide details.*

Finally, please leave comments on any aspect of your ride that needs mentioning.

The qualitative nature of our study is clear from the directions above. We are relying on the perception of each participant on their own ability to make a light or not based on what they can see in front of them and self-awareness of their limitations. We also rely on open-ended comments to supply further details for the codes they choose.

We divided times between the riders to correspond to pre-rush hour (weekday morning from 0600 to 0745), rush hour (0746 to 0945), and post-rush hour (0946 to 1300). We asked each bike rider to start their ride with a red on 13th and Willamette (signal 1) heading east toward campus.

SLOT	1 -----	----- 2	3 -----	----- 4	5	6
PRE	+	+	+	+	+	0
PRE	+	+	+	0	+	+
PRE	+	+	0	+	+	+
PRE	+	+	+	-G	+	+
PRE	+	+	+	+	+	+
PRE	+	+	0	+	+	+
PRE	+	+	0	+	+	0
PRE	+	+	+	-G	+	+
PRE	+	+	+	+	+	+
PRE	+	+	+	+	+	-R
RUSH	+	+	+	+	+	+
RUSH	+	+	+	+	-E	+
RUSH	+	+	+	+	+	+
RUSH	+	+	+	+	+	-E
RUSH	+	+	+	0	+	0
RUSH	+	+	+	+	+	+
RUSH	+	+	0	+	+	+
RUSH	+	+	+	+	-E	+
RUSH	+	+	+	0	+	+
RUSH	+	+	-R	+	+	-E
POST	+	+	+	+	+	+
POST	+	+	+	+	+	0
POST	+	+	+	+	+	+
POST	+	+	0	+	+	+
POST	+	+	+	+	+	0
POST	+	+	+	+	+	+
POST	+	+	+	+	+	+
POST	+	+	+	0	+	+
POST	+	+	+	+	+	0
POST	+	+	+	+	-E	+

Figure 10: Table results: Coding from 30 individual rides on the corridor.

Figure 10 shows the composite codes of a total of 30 rides by time slot. We have added information to the top row that shows which signals are coordinated on the route: While all signals are fixed-time, only two pairs are coordinated.

Our overall impression is very positive. We were able to tap into the car-centric V2X GLOSA world and provide accurate information to bicyclists as well. However, being an exploratory study, we were just as interested in the details. We discuss those below, organized by signal column.

Signal 1: The app was 100% reliable in starting the user off. It correctly predicted the change to green for the biker and suggested a gradual speed-up to make signal 2 (one block away). As a reminder, each trial started with the bike rider seeing a red on signal 1 (i.e., Willamette) going east.

Signal 2: Signal 1 and signal 2 are coordinated at a 25mph average speed, faster than our top speed range of 18 mph . However, the distance between the two signals is short, and all riders started with a new green on signal 1 (i.e., they had the full green-phase time to ride a block). Riders reported no problems getting a green with the app's adjustments.

Signal 3: There is no coordination between signals 2 and 3 and the distance between them is short (a single block). Given only a short distance to work with and constrained to stay within an 8-18 mph speed range, the app did advise that the signal was unmakeable on a number of occasions. In a majority of such cases, the bike rider judged the app correct in its assessment. However, in several cases the rider gave a code of **–R**: the rider judged that the light was makeable with reasonable acceleration. We note that the **–R** codes were often given, for this signal and others, by the same rider who left comments that the app was too pessimistic on the ability to speed up to make a light. This is an example of the need for personalization, discussed more broadly below.

Signal 4: Signals 3 and 4 are coordinated. As can be seen, our app judged in several cases that the light was unmakeable. Our conjecture is that the rider was on the tail end of the green on 3 and did not have the time and space (one block) to catch the green on 4 given our speed constraints.

Signal 5: Signals 4 and 5 are uncoordinated. However, they are three blocks apart. Riders generally reported few problems getting a green on 5 by following the app's adjustments. Our conjecture is that three blocks was enough of a cushion to line the user up for a green with adjustments that stayed within range. However, several trials were given an **–E**. We followed up with the riders to find out more. In all cases, the rider was stuck in a convoy and unable to follow the app's directions. **-E** was the closest category they saw to use. We discuss the convoy problem below.

Signal 6: Signal 6 is a bit like signal 3; there is no coordination with its upstream neighbor, signal 5. And the distance between is short (one block). We saw a similar pattern of unmakeable messages from the app. We also saw convoy issues for this signal (marked as **–E**).

User Comments

We asked each rider to write down any comments at the end of each trial. Overall, the riders found the interface effective, reasonable and non-distracting. However, they also noted room for improvement:

1. On bright days, the phone's auto-brightness adjustment was marginal (we preset auto-adjustment on). A suggestion was made that the app turn the screen to full brightness when actively in use.
2. The app did not take into account other riders "clogging" the bike lane. The app assumed an unobstructed path. In some cases, a rider could not comply with the app adjustment without dangerously swinging into the car lane to pass slower bike riders ahead. They ended up marking this event with **-E**.
3. One rider thought the app speed range was too conservative on the top end. Yet another rider felt the app asked for large increases that were on the edge of their comfort zone and would have preferred it be turned down.
4. One rider would like information on the wait for a green when stopped at a red, which is certainly within the app's power to give.
5. While the audio was recognizable with the volume at max (we preset this), some worried that they did not keep their audio at max in general. They suggested the app automatically increase the volume to max when in use.

6. The unanimous opinion was that the moving bicycle was too small and was distracting to attend to. Riders basically ignored it.

We also ran five extra trials with the phone in a backpack (supplied by us). Our question was whether the audio was loud enough to be audible by the rider *without* resorting to earbuds. Our interest in this was twofold: (1) our previous studies *with* an earbud (Fickas et al, 2008) showed that audio-only was the least distracting type of interface, and (2) from a practical uptake view, throwing your phone into a backpack without worrying about mounting it or wearing earbuds supports general adoption of the app. The comments we received from these trials were that the audio was still audible with modest ambient noise (e.g., car traffic). However, at busy rush-hour times, the riders felt they missed some messages because of ambient noise.

FUTURE WORK

The Need For Personalization

Our prior studies exploring user interfaces for individual transportation assistance had a big takeaway: our results showed that personalization was crucial for user acceptance (Fickas et al, 2008). Our GLOSA bike trials align with these previous findings. While the car-centric world of GLOSA currently takes a one-size-fits-all approach, our experience is that this is an erroneous position to take with bike riders. The comfort zone of the biking public varies. Some are happy with slow and steady. Some are ok with quick bursts of speed. Others are happy with sustained acceleration. We have constructed the app with this in mind: virtually everything is customizable. The question for our next round of interface design is how to allow the bicyclist to tell the app their preferences. One approach would be to allow riders to enter the codes in Figure 10 into the app in real time as events are happening. We could then adjust parameters accordingly. The challenge is to do this without adding distracting feedback components to the app. A second alternative is to have the app learn the rider's preferences by monitoring their actions as they attempt to follow the app's adjustments; failure to follow could trigger parameter changes. This frees the rider from distracting feedback actions by shifting the problem to building effective models of user behavior behind the scenes.

Situational Awareness

The app has access to the rider's location, speed and direction of travel now. What the app does not track is the landscape the rider is moving through. Is it hilly? If so, acceleration thresholds might be changed. Is parking such that car doors may open into the bike lane? If so, slower speeds might be much safer even if they mean missing greens. This type of rich environmental information is beginning to be made available by cities through APIs, allowing an app to use its basic GPS information to place the rider in a much richer context.

Living With Convoys

If city planners are successful in laying out bike lanes along convenient corridors, they can expect heavy bike use. We observed that case along our test corridor of 13th Avenue leading into the UO campus. The problem from a GLOSA standpoint is that convoys form and GLOSA-based adjustments that are ignorant of these convoys are either ineffective or even dangerous when the rider is part of such a convoy (e.g., the only way to follow the app's speed-up adjustments is to swerve into the adjacent car lane). The high-tech solution is to look to a future when all bikes are connected into V2X and convoys can be spotted and taken into account by an app like ours. In this future world we expect all cars to also be linked to V2X. Our app can be made aware of a lack of cars in the adjacent lane, making it safe to use to pass a convoy. A much more practical, low-tech solution is to give bicyclists more space to move at their own pace. And to coordinate signals on the corridor to typical *bike* speeds. These are ideas that are being discussed now by city transportation planners in Eugene.

GLOSA For All

There is a segment of the population who do not carry cell phones: students in grades K-6. We have been thinking about this segment and how to support them in conjunction with our local chapter of Safe Routes To School (SRTS). A main goal of SRTS is to get more kids biking to school. Our vision is of a GLOSA interface built into the bike itself. We have prototyped several ideas and hope eventually to convince a bike manufacturer to build a new line of bikes that is V2X-ready.

Actuated Signals Remain A Challenge

We plan to test the app on a new corridor that adds a significant new wrinkle: an actuated signal. With a fixed-time signal, we found we were able to predict the start of a green with high accuracy given the real-time feed from the Transparity server. With an actuated signal, queues come into play leading to on-demand phases. In a prior NITC project (Fickas, 2018), we found that knowing the timing table of an actuated signal helps predict the start of a green phase, but without information on what demands have been placed on phases ahead of our phase of interest, prediction remains challenging. For instance, in our study intersection, potentially three phases were scheduled ahead of the bike phase. However, if there was no demand for one or more of these phases, they would be skipped. In the best case a green for bikes would come quickly (no demand so all skipped). In the worst case the bike rider would have to wait for three separate phases to cycle through before getting a green (all in demand so none skipped). We expect controllers of the future to provide information on internal queues, and this will help greatly; we can determine what phases will be skipped or not. However, the Transparity server, at this time, does not provide any information other than current active phase and when it became active. There is new work in the related field of traffic flow that shows some promise using machine learning and predictive models. For instance, Ran et al. (2019) effectively use a machine-learning model to predict traffic flow during special events. We conjecture that a form of these models can be successful with phase predictions as well. To follow up on this conjecture, we have stored a year's worth of Transparity data on our actuated-signal test site. We are now looking at training a deep-learning model using our historical data as training data.

CONCLUSION

This project is based on two questions: (1) Can we convey GLOSA information to a phone carried by a bicyclist? (2) Can we present that information in an effective, reasonable and safe manner using the interfaces available on the phone? For the former, we were able to combine (a) information provided by the City of Eugene's traffic office in terms of signal timing-tables with that of (b) a real-time traffic feed from the McCain Transparity server. We developed a system architecture that used the real-time feed to keep an up-to-date view of signal state (current phase and its start time). However, if the feed was lost, the system was designed to continue operating with its last known state information, potentially obtained by signal reset times maintained by the traffic office.

We used the phone GPS and accelerometer sensors to keep an accurate view of the rider's location and speed. In general, the system architecture we employed is within the reach of any city that has access to their signal-controller information through a real-time internet feed. Beyond the license fee paid to McCain, all other components of our system are free and open source.

Our second question was answered with a qualified yes. We took on an exploratory study based on our prior work in the area. We established that with two information modes, visual and audio, bike riders in our trial were given effective, reasonable and non-distracting GLOSA information. However, the rider comments reflected room for improvement, as well as a major stumbling block. In particular, the app has no situational awareness in terms of other cars and riders on the road. As long as bike lanes are narrow (or even non-existent), it may be difficult to move ahead of a convoy of fellow bikers.

Our results are also qualified by the lack of generality. We tested on a specific corridor in the city with its own peculiar mixture of fixed-time and coordinated signals. We supplied a project phone as opposed to allowing the riders to use their own phone. We did not attempt conditions that simulated the McCain server going down. We did not run trials in varying weather conditions, for instance moderate rain or wind. We did not run trials before dawn or after dusk. In summary, we have shown that is feasible for a city to consider delivering GLOSA information to their biking public, but there is room for new research that broaden our results.

REFERENCES

Copenhagen Study (2013) https://web.archive.org/web/20131212093813/http://subsite.kk.dk/sitecore/content/Subsites/CityOfCopenhagen/SubsiteFrontpage/LivingInCopenhagen/CityAndTraffic/CityOfCyclists/CycleStatistics.aspx

Fickas, Stephen. (2018) *V2X: Adding Bikes to the Mix*. NITC-ED-1027. Portland, OR: Transportation Research and Education Center (TREC), 2018.

Fickas, S., Sohlberg, M., Hung, P., (2008) Route-following assistance for travelers with cognitive impairments: A comparison of four prompt modes, *Int. J. Human–Computer Studies*, Volume 66, Issue 12, December 2008, Pages 876-888

Fickas, S., Lemoncello, R., Sohlberg, M. (2013) Requirements Engineering in a Mobile Setting, In *User Modeling and Adaptation for Daily Routines*, Martín, Haya, Carro (Eds.), Springer

Lemoncello, R., Sohlberg, M.M., & Fickas, S. (2010a). When directions fail: Investigation of getting lost behavior in adults with acquired brain injury. *Brain Injury*, 24, 550-559

Lemoncello, R., Sohlberg, M.M., & Fickas, S. (2010b). How best to orient travelers with acquired brain injury: A comparison of three directional prompts. *Brain Injury*, 24, 541-549.

Ran, X., Shan, Z., Fang, Y., & Lin, C. (2019). An LSTM-Based Method with Attention Mechanism for Travel Time Prediction. *Sensors*.

Ravi, N., Dandekar, N., Mysore, P., & Littman, M. L. (2005). Activity recognition from accelerometer data. In Proceedings of the *American Association for Artificial Intelligence* (Vol. 5, No. 2005, pp. 1541-1546)

Seredynski,M., Dorronsoro, B., Khadraoui, D. (2013) Comparison of Green Light Optimal Speed Advisory approaches, In proceedings of the *16th International IEEE Conference on Intelligent Transportation Systems* (ITSC) 2013