

A Dynamic Decentralized Traffic Light Management System: A TCP Inspired Approach

Omar Hiari

School of Electrical Engineering and
Information Technology
German Jordanian University
Amman, Jordan
Email: omar.hiari@gju.edu.jo

Ibraheem Nofal

School of Electrical Engineering and
Information Technology
German Jordanian University
Amman, Jordan
Email: i.nofal@gju.edu.jo

Abstract—As the number of vehicles on the road continues to increase, methods to deal with changes in traffic demand and density keep evolving. The intent is to ensure better management via traffic lights timing, phases, and cycles to reduce the delays that occur in increased demand scenarios. Typically, solutions have been divided into centralized and decentralized, each with their own advantages and disadvantages. Most existing solutions, however incur a high cost to adapt the infrastructure to. In this paper, we present a decentralized method for traffic management via controlling traffic lights that current systems can be adapted to. The proposed solution can be adapted to existing traffic solutions without huge infrastructure costs. Inspired by TCP congestion control, for its operation, the system relies on an RSSI-based vehicle direction detection system. The advantage of such a system is that it can be adapted to existing traffic networks without introducing a high infrastructure cost. The effectiveness of this solution is evaluated through simulations carried out using the Simulation of Urban Mobility (SUMO) tool and is compared to currently deployed solutions. A small-scale traffic intersection physical model is also developed that demonstrates the operation of the vehicle direction detection system.

Index Terms—Smart Cities; Smart Traffic Lights; TCP; Traffic Management.

I. INTRODUCTION

Throughout the past decades, the number of vehicles on roads has constantly continued to climb, causing over-saturation at many signalized intersections in urban areas worldwide [1], [2]. This over-saturation or congestion occurs when there are more vehicles on a part of the road network than is optimally desirable, resulting in lower travel speeds and an increased frequency of delays [3]. As a consequence of congestion, time which could have been used more productively is lost. Moreover, this results in increased vehicle emissions and higher operating costs. For example, in the greater Dublin area alone, the cost of time lost due to congestion in 2012 was estimated to be €358 million and is forecast to increase to €2.08 billion per year by 2033 [3]. Moreover, in the UK as a whole, the direct and indirect costs of congestion were estimated to be more than £37.7 billion in 2017, with London amounting to £9.5 billion alone [4]. Therefore, if the intention is to minimize losses, then its imperative that the issue of congestion is addressed effectively.

Traffic lights have typically been used as the technique to control traffic. In their first-generation, electric traffic lights were automatically controlled through pre-set signal timing plans [5]. Those systems were extremely simple and unable to react to changes in traffic demand, mainly due to their timing being based on historic statistical data. Following that, second-generation traffic control systems used detectors to sample input traffic flow at an intersection. The sampled inputs were used to select timing plans based on sensed conditions [6]. Second generation systems of actuated control evolved into what is referred to as Adaptive Traffic Control Systems (ATCS), which marked the birth of third generation systems [7]. ATCS are capable of making signal timing decisions in short intervals based on real-time data provided by sensors as well as traffic flow predictions. While ATCS undoubtedly perform better than first and second-generation systems, different implementations of ATCS still have room for improvement especially in deployment cost. Finally, fourth-generation systems are expected to be able to react to real-time traffic changes by being self-organized, decentralized, networked and distributed in terms of control.

Some currently deployed solutions resort to methods adopting what is referred to as gap-based actuation. Gap-based actuation works by having a pre-set maximum and minimum green values for a given intersection. Upon turning green, the traffic light stays green for the minimum value, and in the case of subsequent detection of traffic, the green interval is extended by a preset value up to a maximum green value, or till the gap between successive vehicles is too great that an actuation doesn't occur in the given window [7]. Examples of two gap-based actuated traffic control methods are SCOOT (Split Cycle Offset Optimization Technique), and SCAT (Sydney Coordinated Adaptive Traffic) [5]. In the case of SCOOT, compared to traditional fixed-interval traffic light switching, delay reductions of up to 16.9% were observed [6]. Nevertheless, in certain conditions, SCOOT performed in an inferior or indistinguishable manner to traditional systems.

In this paper, we present a new approach to adaptive traffic lights, which is inspired by the Transmission Control Protocol (TCP). The proposed system is capable of detecting and managing vehicle flow from one intersection to the next i.e.

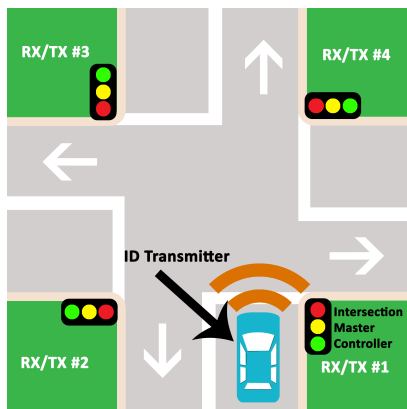


Fig. 1: The Intersection Model

treating vehicles like packets. Apart from light and phase control, the system also leverages the Received Signal Strength Indicator (RSSI) available in recent wireless solutions to determine vehicle travel direction. This gives a real-time insight into the conditions of the road network, which enables faster response to fluctuations in traffic demand. The system is also capable of accurately detecting and predicting traffic demand before it arrives at a given intersection. To test the proposed system, simulations were carried out to test the effectiveness of the proposed algorithms. The results were also compared against a benchmark gap-based actuated traffic control system. Moreover, to verify the ability of proposed approach to predict vehicular direction of travel, a small-scale model of a 4-way intersection was built and tested.

II. SYSTEM MODEL

The system model adopted treats each traffic light intersection as a node in the network. On the other hand, vehicles are observed as packets in a computer network system. As depicted in Fig. 1, the nodes would be equipped with wireless hardware for backbone, or node-to-node, communication and hardware for node-to-vehicle communication. Moreover, the vehicle-to-node simplex communication hardware exists at each traffic light direction. As a result, only one node-to-node communication hardware is needed and can be designated from any of the existing node-to vehicle hardware. This means that the vehicle itself needs to be equipped with a wireless module, which could be as simple as a passive RFID tag. Moreover, each intersection node handles congestion and flow control independently and exchanges information only with its direct neighbors. This could be through using an existing wireless infrastructure (Ex. cellphone towers). To detect direction of travel per vehicle ID, nodes leverage RSSI and a triangulation algorithm. Each node can individually manage detection of vehicle arrivals and control flow of vehicle departures based on vehicle departure directions. At the vehicle level, there is no computation required. The wireless node at the vehicle is needed only for the intersection node to acquire the vehicle ID and determine the direction of travel.

A. Route Prediction

In order to determine the vehicle direction for route prediction, RSSI provides a decibel (dB) value indicating how far the transmitting device is from the receiver. The higher the value, the closer the transmitter is to the receiver and vice versa. In order to predict the route, an algorithm continuously scans, in each light direction, for IDs and RSSI values of passing vehicles. Each light at a node stores the received vehicle ID, as well as the RSSI peak value. Upon finishing one scanning cycle per light, the RSSI value for a given vehicle is compared with the previously recorded value, keeping the greater of the two, as well as the time at which the value was collected. The scanning process per vehicle continues indefinitely until the RSSI value returned by all transceivers for a specific vehicle ID drops below a certain threshold value. This means that the vehicle is considered to have left the intersection on one of the leaving edges and its direction of travel is determined.

B. Adaptive Traffic Light Timing Algorithm

In TCP, hosts maintain a receive buffer and a transmit window for existing connections. In order to increase efficiency of bandwidth utilization, TCP allows for a certain amount of packets, defined by a window size *twnd* to be "in-flight". The packets are tracked by unique IDs and additional packets are not transmitted until previously transmitted ones are acknowledged by the receiver. This is necessary for the functioning of the flow control and congestion control services provided by TCP. As a result, TCP though controlling the *twnd* size controls the rate of transmission of packets. For flow control, the goal is to not overwhelm the receiver whereas for congestion control the goal is to not overwhelm the network. Flow control is maintained by signals sent back from the receiver indicating how much buffer space is left. Whereas congestion is implied by packet-loss through timeouts or duplicate acknowledgments.

The RTT (round-trip time); defined as the amount of time between the sender sending a packet and receiving an acknowledgment from the receiver, is used as an indicator of network congestion, where typically more congested networks lead to longer RTT times. These methods are necessary in TCP because its difficult to predict network traffic, as is the case with vehicular traffic. Therefore, it is necessary to be able to quickly detect and adapt to changes in network traffic, which is what flow control and congestion control accomplish.

In the proposed adaptation for a vehicle traffic light network, the number of vehicles sent by a given intersection towards the receiving intersection is limited through a transmit window as well. To do so, it is necessary to have an identifier for each vehicle for which we propose using the vehicles VIN (Vehicle Identification Number), analogous to the packet ID in TCP. However, unlike in computer networks, vehicles direction of travel cannot be dictated and hence the flow to one particular receiver alone cannot be stopped. As a result, we assume that overflow occurs when all receiving intersections exceed a set maximum of vehicles; halting sending until the vehicle count in the receiver traffic light buffers dip below an allowed

threshold. This set threshold is analogous to the receiver window size and is determined in this case by the streets maximum vehicular capacity.

Additionally, in our implementation flow is controlled by completely halting transmission of vehicles by switching to the next traffic light phase. This also happens when the out flowing traffic rate is slower than the allowed minimum in a certain direction. Alternatively, this can be implemented using changes in RTT. Moreover, vehicles, unlike packets, cannot be re-transmitted when a loss occurs. As a result, when a timeout occurs, the vehicle is assumed to have left the network edge. For simplicity, a fixed timeout value was used, however it can be easily expanded to a calculated timeout value by sampling RTTs as in TCP.

In the proposed algorithm, at the start, the sending intersection initializes the RemainingPhases variable such that all 4 phases are available as switching options for a 4 light intersection. At the beginning, it is assumed that at least one traffic light starts green. As a result, the program has two loops, one for switching phases and updating states, and one for updating cycles. For simplicity, the phases utilized allow movement in all directions from each traffic light when its green (straight, right turn, left turn, u-turn). A cycle is defined as the switching on of each traffic light once; i.e, cycling through all phases once.

Inside the inner loop, the program checks for outbound traffic, obtaining the outbound vehicles' directions, then updating the receive buffers and transmit windows for every traffic light on the sending and receiving sides. If the system detects that all receivers' windows are full, then all traffic lights at the current intersection are switched to red and maintained red until at least one of the receivers' windows is no longer full. However, if the condition of AllTransmitWindowsFull() isn't satisfied, the system maintains the green phase as long as the following conditions are false: 1) The outbound traffic leaving the intersection slows down and dips below the minimum speed limit on all edges leaving the intersection for an x amount of seconds. 2) The current green phase isn't expecting any more vehicles. In the case that any of aforementioned conditions are true, the current green traffic light is set to yellow for 3 seconds, followed by red. The current phase is removed from the set of allowed moves in the current cycle, then the next phase is switched on. The next phase is selected by finding the least filled receiver buffer, and directing traffic towards that receiver. Afterwards, the current phase is switched off and the next phase is switched on, with each traffic light's phase being switched only once per cycle. At the same time, the sending intersection has independent counters for the vehicles being sent in each direction, incrementing the respective counter whenever a vehicle departs, and decrementing whenever a timeout occurs or an acknowledgment is received.

III. SETUP AND RESULTS

A. Small Scale Hardware Setup and Results

A real-life model, shown in Fig. 2, was built in order to test the route-prediction algorithm. For this, a Raspberry Pi with

while true do

RemainingPhases = [1,2,3,4] ;

while RemainingPhases.Length > 0 **do**

//Obtain sent vehicles directions;

//Update Lights' Receive Buffers and Transmit Windows;

if AllTransmitWindowsFull **then**

//Maintain Red on all lights while all receivers transmit windows are full;

SwitchToNextPhase();

//Check if a change in phase is needed;

else if ShouldSwitchPhases() **then**

RemainingPhases.Remove(CurrentPhase)

SwitchToNextPhase();

end

end

Algorithm 1: Adaptive Traffic Light Algorithm

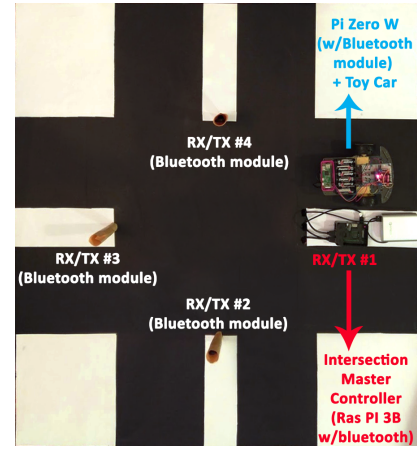


Fig. 2: The Intersection Small-Scale Hardware Model

4 Bluetooth modules connected was utilized. The Bluetooth modules were then placed on a 1.20-meter by 1.20-meter board meant to model a 4-way intersection, with the transceivers being mounted where a traffic light would be.

A Raspberry Pi controlling a toy car chassis that operates as a Bluetooth beacon transmitting a unique ID was also utilized. As a result, the traffic light module would scan for RSSI changes once every 0.35 seconds. The toy car was made to move around the model, while the traffic light transceivers would record the toy car's RSSI values. Depicted in Fig. 3, the RSSI values detected by each traffic light transceiver are shown for a single run. The plots roughly resemble the shape of a bell curve, with the transceiver closest to the direction of travel peaking in RSSI value later than other transceivers. Consequently, the algorithm was able to accurately predict the direction of travel for the vehicle in most runs. The last peak in the shown figure, at the time of one second, indicates that the vehicle went eastbound. Its worth noting that the prediction accuracy increased when the transceivers were placed further apart, performing excellently when they were

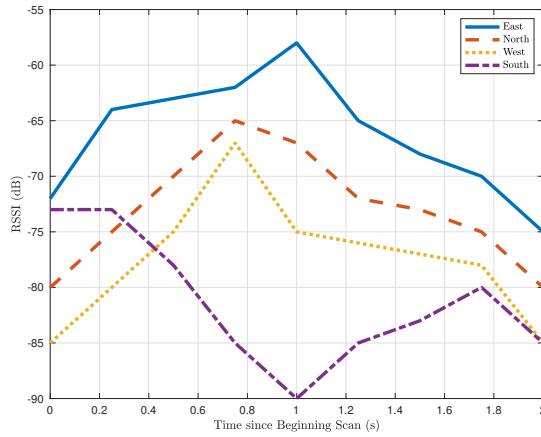


Fig. 3: Example RSSI Readings of Hardware Model

more than 50 centimeters apart. Rarely, the RSSI picked up by the transceiver in the cars direction of travel would peak at the same point in time as that picked up by another transceiver. In that case the two RSSI values are compared, and the direction of travel is determined to be the one where the RSSI value is greater; due to the vehicle being closer to the transceiver, the signal strength is greater.

B. Simulated Setup and Results

SUMO (Simulation of Urban Mobility) is an open source road traffic simulation software designed to handle large road networks, that's developed by the Institute of Transportation Systems at the German Aerospace Center [8]. It is suitable for modeling road networks as well as retrieving real-time data from the model in order to alter its parameters. SUMO provides a suite of tools to randomly generate, model and control traffic, among many other functions. For the proposed model, random vehicular traffic was generated using SUMO's different libraries with varying trip intervals. SUMO generates random traffic for a given road network using user-specified parameters, which is suitable when there's a lack of real-life based measurements for a given network. In our setup, the simulation was configured to model high traffic demand passing through central intersections. At each traffic light, the simulation also modeled our proposed solution in comparison to an existing gap-based solution.

Two different trip interval values of 1 and 1.2 were utilized for the simulations where three random trip files were generated per interval. Trip intervals govern how many vehicles are spawned in the model per unit time. For instance, for a trip interval of 0.5 the vehicles spawned in 1 second would be equal to $1/0.5$. The simulation was also executed three times per trip interval per algorithm. Finally, a total of five traffic intersections were included in the model.

Tables I and II depict the results that were obtained for each trip interval. It is shown that our proposed algorithm performed better in all simulation attempts. Up to 13.7% enhancement in RTT was viewed over the gap-based solution for a trip

Run #	Gap-Based RTT (s)	Proposed Algorithm RTT (s)	Percent Enhancement
1	260	229	11.9%
2	262	226	13.7%
3	251	230	8.4%

TABLE I: Comparison of RTT values for a trip interval of 1

Run #	Gap-Based RTT (s)	Proposed Algorithm RTT (s)	Percent Enhancement
1	232	219	5.6%
2	229	218	4.8%
3	232	222	4.3%

TABLE II: Comparison of RTT values for a trip interval of 1.2

interval of 1. Meaning that the proposed algorithm performed better when the congestion rates were higher. We expect that the the algorithm can even perform better if the operating parameters are tuned to better estimate the capacity of a the street connecting to a particular intersection.

IV. CONCLUSION

In this paper, a model that provides a cost effective solution that can be built on using simple hardware existing traffic light networks is presented. Compared to existing solutions, it shown through simulations that the proposed model can provide up to 13.7% enhancement on top of currently deployed solutions. The results also have shown that the performance gets better as the traffic rate increases.

REFERENCES

- [1] Statista, "Number of vehicles in use worldwide 2015." [Online]. Available: <https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide/>
- [2] B. a. Park, C. J. Messer, and I. I. Thomas Urbanik, "Enhanced Genetic Algorithm for Signal-Timing Optimization of Oversaturated Intersections,," *Transportation Research Record*, Jan. 2000. [Online]. Available: <https://journals.sagepub.com/doi/10.3141/1727-05>
- [3] Gov.ie, "Gov.ie - Cost of Congestion - Main Report." [Online]. Available: <https://www.gov.ie/en/publication/679669-cost-of-congestion-main-report/>
- [4] INRIX, "Traffic Congestion Cost UK Motorists Over 37.7 Billion in 2017." [Online]. Available: <https://inrix.com/press-releases/scorecard-2017-uk/>
- [5] S. Goel, S. F. Bush, and C. Gershenson, "Self-Organization in Traffic Lights: Evolution of Signal Control with Advances in Sensors and Communications,," *arXiv:1708.07188 [cs]*, Jun. 2017, arXiv: 1708.07188. [Online]. Available: <http://arxiv.org/abs/1708.07188>
- [6] N. H. Gartner, C. Stamatiadis, and P. J. Tarnoff, "Development of Advanced Traffic Signal Controls Strategies for Intelligent Transportation Systems: Multilevel Design,," *Transportation Research Record*, no. 1494, 1995. [Online]. Available: <https://trid.trb.org/view/452642>
- [7] Y. Wang, X. Yang, H. Liang, and Y. Liu, "A Review of the Self-Adaptive Traffic Signal Control System Based on Future Traffic Environment,," 2018. [Online]. Available: <https://www.hindawi.com/journals/jat/2018/1096123/>
- [8] D. Krajewicz, G. Hertkorn, C. Ressel, and P. Wagner, "SUMO (Simulation of Urban Mobility) - an open-source traffic simulation,," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002)*, A. Al-Akaidi, Ed., Sharjah (United Arab Emirates), 2002, pp. 183–187. [Online]. Available: <https://elib.dlr.de/6661/>