International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

SCHEDULING AND OPTIMIZATION OF TRAFFIC LIGHTS ON A T-JUNCTION USING A WIRELESS SENSOR NETWORK

Evarist Nabaasa[1] and Tonny E Bulega[2]

Department of Networks, Makerere University
Kampala, +256, Uganda
[1]enabaasa@must.ac.ug, [2]tbulega@cit.mak.ac.ug

ABSTRACT

This paper provides an overview of how to schedule and optimize traffic lights using Intelligent Transport Systems. In this work we present the scheduling and optimization algorithms that we developed and how we tested them by connecting different simulators to evaluate their efficiency. The developed algorithms have been tested on a T-Junction. The road network is simulated using MObility model generator for VEhicular networks (MOVE) and Simulation of Urban Mobility (SUMO) while the wireless sensor network is simulated by Network Simulator (NS).

Keywords: Scheduling, Optimization, Algorithms, Intelligent Transport Systems, Wireless Sensor Networks, Simulators.

1. INTRODUCTION

Road traffic control in cities around the world has continued to be a big challenge despite the attempts to expand road infrastructure by many city planners. What is very evident is that city planners can not keep expanding the road infrastructure as the vehicles increase in number.

In our research, we explore and develop algorithms for Scheduling and Optimization of traffic lights on a T-Junction via a wireless sensor network. Some researchers have done related work mostly considering resource rich networks. In our case we use a resource constrained wireless sensor network which is easy to deploy, maintain, expand among others. We achieved, air pollution, too much fuel consumption, in towns and cities can be reduced greatly among others.

Current road traffic simulation tools [2] [3] tend to specialize in simulation of road networks and the traffic that move on them. Wireless Sensor Networks (WSNs) simulators [4] [5] also tend to specialize in simulation of WSN scenarios, in this research we combine [1] [6] the two to implement the developed algorithms for scheduling and optimization of traffic lights on a T-Junction.

2. RELATED WORK

In [1] a Traffic Control Interface (TraCI) is used to interconnect Simulation of Urban Mobility (SUMO) with TinyOS SIMulator (TOSSIM) to have the two simulators exchange information, in this research there are many assumptions that were not clarified. In their work of Mobility Support for Wireless Sensor Networks Simulations for Road Intersection Safety Applications, it was assumed that the simulated vehicles running on a SUMO road network were automatically carrying the sensor nodes to form a Car2Car sensor node communication but this is not practically supported by SUMO.

The authors in [6] built an Adaptive Traffic Light Control with Wireless Sensor Networks. In their work, they

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

used Green Light District (GLD) simulator to simulate the road network with junctions, GLD is a microscopic road network simulator. They also used magnetometer sensors to form a sensor network for detecting vehicles at a junction. Their paper does not show how the magnetometer sensor network they used communicates with GLD simulator. The Green Light District simulator does not have an inbuilt support for simulating WSNs.

3. WIRELESS SENSOR NETWORK SIMULATORS

Wireless Sensor Networks have started gaining more ground on the stage of computer networks. Given the need of deploying many sensor nodes in a WSN, simulations become a more realistic choice in researching and understanding of their Application, Protocols, Design, Topologies, Security among others. These simulators help researchers to avoid the otherwise large and expensive real hardware network deployment. A considerable number of simulation tools have been developed. These tools tend to differ in many features, such as programming language, visual environment, supported protocols, freely available or with costs [1]. Below we give an overview of the common ones.

3.1 Ns2

Network Simulator (NS2) [5], is based on two languages: C++ for the simulator environment, and OTcl (Object oriented Tool Command Language) for executing user command scripts. Via OTcl scripts, the user is able to define particular network topologies, specific protocols, Applications he wishes to simulate and the form of output he wishes to obtain from the simulator. NS2 is a discrete event simulator, where the advance of time depends on the timing of events which are maintained by a scheduler. The Network Animator (NAM) application is used to graphically visualize the designed network and how the sensor nodes communicate.

3.2 Omnet++

OMNeT++ [7], is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. "Network" is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queuing networks, and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc., is provided by model frameworks, developed as independent projects. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools.

3.3 Tossim

TinyOS Simulator (TOSSIM) [8], is a discrete event simulator for TinyOS sensor networks. Instead of compiling a TinyOS application for a mote, users can compile it into the TOSSIM framework, which runs on a PC. This allows users to debug, test, and analyze algorithms in a controlled and repeatable environment. TOSSIM replaces low-level components of a TinyOS application by simulation-specific code to simulate radio chip, Analogue-to-Digital Converter (ADC) components and timer hardware [1]. The extension TOSSIM-SF is a serial forwarder extension which allows communication from outside to single nodes while simulation is in progress. Under TinyOS the compiler can be forced to produce code for TOSSIM by just adding the option sim or sim-sf respectively to the compiler call. TOSSIM allows simulation of very large networks (thousands of nodes), network links are simulated at bit level and it also allows asymmetric links.

Other popular Wireless Sensor Network Simulation tools are JiST/SWANS [9] and OPNET [10].

4. ROAD TRAFFIC SIMULATORS

Simulation of road traffic usually caters for simulating, road networks, mobility of individual participants, rules followed by the participants and sometimes, rules for traffic lights control at intersections. Road traffic simulators support either of the following traffic models: macroscopic, mesoscopic and microscopic.

Macroscopic simulation models analyze traffic flow in which the description of reality is shifted from individuals to “averaged” variables like flow and density.

Mesoscopic simulation models use methods of statistical mechanics to provide a probability function which expresses the probability to have a certain vehicle at a certain position.

Microscopic simulation, simulates each entity (car, train, person) of reality individually by considering all relevant properties. A number of microscopic simulation environments have been developed [12]. Below is an overview of some common mobility simulators.

4.1 Sumo

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

Simulation of Urban Mobility (SUMO) [3], is an open source, highly portable, microscopic road traffic simulation package designed to handle large road networks. It can perform GUI or Command line based simulations, import various types of maps, consider multiple lanes, traffic signs and right-of-way rules and configure individual routes for vehicles [1]. It does not have inbuilt sensor network modules though it allows communication with such networks via TraCI.

SUMO, uses poison distribution to generate vehicles which are injected into the road network. XML scripts are written to generate nodes, links, routes and the network files among others, finally basing on the network file, a configuration file is generated for the simulation scenario.

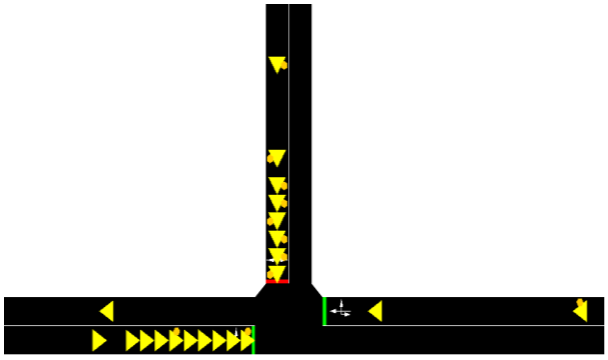


Fig. 1. Simulated T-Junction

With a simulation scenario like in Fig. 1, SUMO provides inbuilt Traffic Light Control (TLC) models on how the traffic lights handle the vehicles that are in or out bound of the T-Junction. With a WSN for detecting vehicles along the T-Junction lanes, the TLC models supported by SUMO would achieve more meaningful results.

4.2 Green Light District

GLD allows us to edit infrastructures using the mouse, to set different spawning frequencies creating different traffic patterns, to compare a large number of implemented controllers (new ones can be easily added), and to evaluate the controllers using different statistical measures (such as average waiting time of cars). The simulator itself is based on a cellular automaton model [12], and therefore a microscopic model which can be used for modeling various amounts of detail forexample the road user could be a car, bus, police car.

4.2.1 Infrastructures

An infrastructure consists of roads and nodes. A road connects two nodes, and can have several lanes in each direction (see Fig. 2). The length of each road is expressed in units. A node is either a junction where traffic lights are operational (although when it connects only two roads, no traffic lights are used), or an edge-node.

4.2.2 Agents

There are two types of agents that occupy an infrastructure; vehicles and traffic lights. All agents act autonomously, following some simple rules, and get updated every time-step. Vehicles enter the network at the edge-nodes.

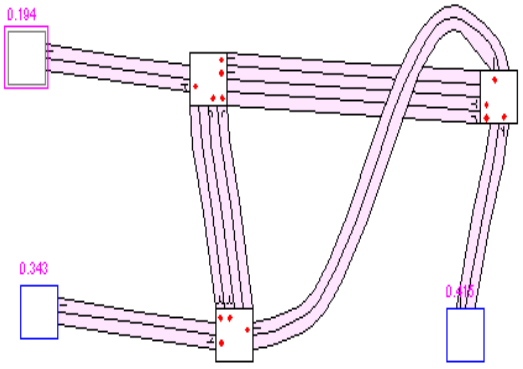


Fig. 2. Simple Road Network

Each edge-node has a certain probability of generating a vehicle at each time step. Each vehicle that is generated is assigned a destination, which is one of the other edge-nodes. The distribution of destinations for each edgenode can be adjusted. There are several types of vehicles, defined by their speed, length, and number of passengers. The state of each vehicle is updated every time step. It either moves with the distance given by its speed, or stops when there is another vehicle or a red traffic light ahead. At a junction, a car decides to which lane it should go next according to its driving policy. Once a car has entered a lane, it cannot switch lanes.

4.2.3 Controllers

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

Every junction is controlled by a traffic light controller (TLC) that decides on the best configuration of red and green lights. A TLC will only consider safe configurations, that is, configurations in which moving cars do not intersect. A TLC can share information with other controllers to improve global performance. GLD has several built in TLCs, and allows for custom TLCs. GLD lucks standardized interface for communicating with other outside applications.

Other common mobility simulators are Sparamics [14], VISSIM [13], JSim [15], among others.

5. SCHEDULING AND OPTIMIZATION ALGORITHMS FOR THE T-JUNCTION

5.1 Scheduling Algorithms

Generally we categorized the designed and implemented scheduling algorithms into two i.e. Non Intelligent and Intelligent Algorithms. This technique was used to better understand the efficiency and robustness of the sensor nodes that formed the wireless sensor network vis-à-vis these algorithms.

5.1.1 Fixed-Phase Traffic Light Scheduling Algorithm

This is a non intelligent algorithm. Suppose there are ***n*** traffic lights on a junction and **i** represents the **i-th** selected traffic light. We select a traffic light by **i = (i + 1) mod *n***, constituting the traffic lights on a junction into a set **T, i** initialized with -1, and express the algorithm as below.

$T = \{T_0, T_1, \ldots\ldots\ldots\ldots, T_{n-1}\}$
j = i;
do
{
    j = (j + 1) mod *n*;

    trafficLightPhase(T_j);

    i = j;

}while(j != i);

A given traffic light that finishes its predetermined phase (Green time or Yellow time, or Red time) has to wait for all the others to first finish their phases before it resumes again.

On applying this traffic light scheduling algorithm it was observed that it always makes vehicles on a certain route wait even when there are no vehicles from the adjacent route crossing the junction. The algorithm gives Green time to empty routes on the junction (Non Intelligent). This algorithm is easy to implement but not efficient, it is very static, and often frustrates the road users.

The Fixed-Phase Traffic Light Scheduling Algorithm above offered a run-time complexity of $O(n^2)$, because the processing of each event required an entire traversal of the job pool.

5.1.2 Actuated Traffic Light Scheduling Algorithm

This is a slightly more intelligent algorithm compared to the above one. Suppose there are ***n*** traffic lights on a junction and **i** represents the **i-th** selected traffic light. We select a traffic light by **i = (i + 1) mod *n***, constituting the traffic lights on a junction into a set **T, i** initialized with -1 ***only if there is traffic waiting*** to cross the junction, and express the algorithm as below.

$T = \{T_0, T_1, \ldots\ldots\ldots\ldots, T_{n-1}\}$
j = i;
do
{
    j = (j + 1) mod n;
    if(trafficIsWaiting(T_j)
    {
        trafficLightPhase (T_j);
    }
    i= j;
}while(j != i);

This algorithm is more intelligent compared to the Fixed-Phase one above. Sensing nodes are used to determine the presence or absence of a vehicle(s) in front of a traffic light for a distance ***d meters*** from the junction. Once a vehicle(s) is detected, the lights change into an appropriate state in relation to their previous state, as they change they also consider the phases of the other cooperating traffic lights at the junction. The phase time is

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

also predetermined like in Fixed-Phase algorithm, though a phase of a given traffic light can be skipped if there are no vehicles waiting.

This algorithm allows a busier lane to take advantage of a lane that has no vehicles waiting since the latter's phase can be skipped. The disadvantage with this algorithm is, in a circulated environment i.e. when all roads are busy, it easily reduces to Fixed-Phase algorithm hence during rush hours the algorithm does not offer much compared to the Fixed-Phase one. This algorithm works well in less traveled lanes and junctions.

The above actuated traffic light scheduling algorithm offers a run-time complexity of *O*(ln n), because the processing of each event takes place only when there is a vehicle(s) waiting to cross the junction.

5.1.3 Ranked Actuated Traffic Light Scheduling Algorithm

In this section, we describe our ranked actuated traffic light scheduling algorithm. First, we describe the standard problem of ranked election and some relevant background. Then, we describe ranked actuated traffic light scheduling algorithm that is derived from ranked elections by establishing equivalence between the question of ranked actuated scheduling and ranked election.

5.1.3.1 A. Ranked election

Definition 1 (Ranked election): There are *M* voters that vote for *C* candidates. Vote of each voter consists of a ranking (or permutation) of all *C* candidates. These votes can additionally carry quantitative values associated with their preferences. Let a_{mc} denote the value voter *m* gives to candidate *c*, for $1 \leq m \leq M$, $1 \leq c \leq C$. The goal of the election is relative order to all the *C* candidates as well as produce the ultimate winner in a manner that is *consistent* with the votes. The key for a good election lies in defining *consistency* of outcome of election with votes. The following are canonical postulates that are used in the literature on ranked election:

P1. Between any two candidates C and C', suppose that none of the *M* voters prefers C' to C and at least one voter prefers C to C'. Then C' should not be ranked higher than C in the output of election. This property corresponds to the economic notion of weak *Pareto optimality*.

P2. Suppose the voters are renumbered (or renamed) while keeping their votes the same. Then the outcome of election should remain the same. In other words, the election outcome is blind to the identity of the voters, that is election outcome is symmetric.

P3. Now, consider setup when votes are cardinal (i.e., quantitative). Suppose candidate C is preferred to C' by the election. Then, by adding same fixed constant to all a_{mc} and $a_{mc'}$ for $1 \leq m \leq M$, the relative order of candidates C and C' should not change. This makes sense because what matters is the difference in preference levels for two candidates, not the actual values.

It is well known that in the absence of cardinal (or quantitative) preferences there can not exist any election scheme that satisfies P1 and P2 simultaneously. We note that this result is an important corner stone in the field of theory of social choice. However, if voters have cardinal preferences, as in the traffic light scheduling case, then there is a unique ordering of candidates that satisfies P1-P2-P3 simultaneously. To support this, consider the following: let net score of a candidate C be $S_c = \sum_{m=1}^{M} a_{mc}$, then the following result can be obtained.

Theorem 1: Suppose the scores of all candidates be distinct. Rank candidates as follows: candidate C has higher ranking than C' if and only if $S_C > S_{C'}$. This ranking satisfies postulates P1-P2-P3. Further, this is the only such ranking.

5.1.3.2 B. Ranked Actuated Traffic Light Scheduling

Now, we shall establish a natural connection between selection of a ranked actuated traffic light scheduling for traffic lights and the problem of ranked election.

To define ranked actuated traffic light scheduling algorithm, we will use a shadow output queued traffic light controller. Specifically, a copy of every packet arriving to the Input Queued (IQ) traffic light controller is fed to a virtual shadow Out Put (OQ) traffic light controller. That is, (copy of) a packet arriving at input *i* for output *j* in the IQ traffic light controller immediately joins the queue at output *j* in the OQ traffic light controller. The departure from the queues at outputs in OQ traffic light controller happens according to appropriate ranked actuated traffic light scheduling policy, say $\mathcal{P}$, such as strict priority scheme, last-in-first-out or simply first-in-first-out applied at the output queues of OQ traffic light controller. Let d_{ij}^{k} be the departure time of k^{th} packet arriving at input i for output j from the OQ traffic light controller. Clearly, these departure

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

times are dependent on the policy $\mathcal{P}$ used in the traffic light controller. If OQ emulation was possible, the IQ controller would like to send out the packets at exactly the same times as $(d_{ij}^{k})_{i,j,k.}$ However, at speed up 1 it is not possible to emulate OQ perfectly. Thus, the whole challenge is to respect scheduling constraints of IQ while letting packets depart from the controller so that they are as faithful to departure times obtained from shadow OQ as possible.

The scheduling problem can be viewed as that of choosing one of the N! possible schedules from $\mathcal{M}$ every time. At each input, there are various packets waiting, possibly in different Virtual Output Queues (VOQs), to depart from it. Let us assume that, in a VOQ, packets are ordered in the increasing order of their departure times from the corresponding shadow OQ. Thus, for $k \neq k'$ if $d_{ij}^{k} > d_{ij}^{k'}$ then k' is ahead of k in the VOQ_{ij}. Thus, the most urgent packet as per the shadow OQ is the Head-Of-Line (HOL) packet in a queue. In what follows, queues will have preference over schedules given by the departure times of the HOL packets. Therefore, for the purpose of selecting ranked actuated traffic lights, it will be sufficient to consider such HOL packets. Assume that all N^2 queues are non-empty in a traffic light controller and hence have a HOL packet. Let $d_{ij}(n)$ be departure time w.r.t. the shadow OQ of the HOL packet in queue at input *i* for output *j* at time *n*.

Each of the N^2 HOL packets has a preference list over the N! schedules or matchings. Consider schedule $\sigma \in \mathcal{M}$: it matches input *i* to output σ(i) for $1 \leq i \leq N$. Now, when a queue at input *i* for output *j* is served it should provide value that is higher if $d_{ij}(n)$ is smaller and vice versa. Specifically, we assign value $-d_{ij}(n)$ to serving queue at input *i* for output *j*. Therefore, net value of a schedule σ at time *n* is given as:

$$value(\sigma)(n) = -\sum_{i=1}^{N} d_{i\sigma(i)}(n)$$

The postulates P1-P2-P3 translate into the following postulates for the ranked actuated traffic light scheduling algorithm.

P1'. Between any two matchings σ1 and σ2, suppose that none of the N^2 HOL packet prefers σ2 to σ1 and at least one HOL packet prefers σ1 to σ2. Then, we should not choose σ2.

P2'. For given HOL packets, let σ be the outcome of the election as per the above preferences for schedules. Then, by renumbering queues while retaining the same HOL preferences, the outcome of election should be only renumbered σ. In other words, the election does not give unfair priority to any port and thus is symmetric in its inputs.

P3'. Suppose matching σ1 is preferred to σ2 by the election. By adding fixed constant to $-d_{ij}(n)$ for all *i, j*, the outcome of the election should remain unchanged then the following schedule $S^*(n)$ should be chosen:

$$S^*(n) \in \arg\max_{\sigma \in M} -\sum_{i=1}^{N} d_{i\sigma(i)}(n)$$

Thus, the algorithm is the maximum weight schedule (MWS) where weight is given by $-d_{ij}(n)$ at time n.

Algorithm. We call the above algorithm as the most ranked light first (MRLF). We will explicitly denote dependence of the algorithm on the policy, $\mathcal{P}$, used in the shadow OQ traffic light controller for scheduling at the output queues. That is, we call it algorithm MRLF($\mathcal{P}$). Now, using postulate P3', we use weights of nonempty VOQ_{ij} as $n - d_{ij}(n)$. Call it urgency of VOQ_{ij} at time n, denoted by $U_{ij}(n)$. Now we define urgency for empty queue. Let VOQ_{ij} be empty at time n. Then define $U_{ij}(n)$ as $\max\{0, -\min_{\{i'j':Q_{i'j'}(n)\neq 0\}} U_{i'j'}(n)\}$

Thus, the MRLF algorithm chooses schedule $S^*(n)$ at time n where

$$S^*(n) \in \arg\max_{\sigma \in M} \sum_{i=1}^{N} U_{i\sigma(i)}(n)$$

MRLF($\mathcal{P}$) Algorithm: Throughput

The previous section described how we arrived at MRLF algorithm as a fair (most ranked light first) algorithm based on preferences obtained from a shadow output queued

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

traffic light controller. As established in the previous section, Theorem 1 implies that MRLF is the only algorithm that satisfies desirable postulates P1'-P2'-P3'. In this section, we state and prove the throughput optimality property of the MRLF algorithm.

Theorem 2: Any N ×N traffic light controller operating under MRLF($\mathcal{P}$) algorithm with Bernoulli i.i.d. arrival process with arrival rate matrix being strictly admissible is rate stable when $\mathcal{P}$ is FIFO.

Proof of Theorem 2

We ascertain that if one considers FIFO policy to prove theorem 2 it would further show the effectiveness and robustness of the developed algorithm and one would need to first highlight some important notation as below.

Consider the packet that is HOL for VOQ_{ij} at time n: let $a_{ij}(n)$ be its time of arrival at VOQ_{ij} ; as before $d_{ij}(n)$ be the time of its departure from the shadow OQ traffic controller; $U_{ij}(n)$ be its urgency as defined above and $W_{ij}(n)$ be its waiting time i.e. $n - a_{ij}(n)$. Also, define $\Delta_{ij}(n) = W_{ij}(n) - U_{ij}(n)$. We note that if VOQ_{ij} is empty, then $W_{ij}(n) = 0$ and $U_{ij}(n)$ is as defined above.

From literature we know that any strictly admissible arrival rate matrix λ can be decomposed as follows: for some $\beta \in (0,1)$,

$$\lambda \le \sum_{k=1}^{N^2} \alpha_k \pi^k; \alpha_k \ge 0, \sum_k \alpha_k = 1-\beta, \pi^k \in \mathrm{M}.$$

We would use Lemma 1 below and others to prove the Theorem but this has been left for future work due to deadline constraint for this journal.

Lemma 1: Let $L(n) = \sum_{ij} W_{ij}^2(n)\lambda_{ij}$. Then, under MRLF(FIFO) algorithm with λ being strictly admissible, there exists ε > 0 such that

$$\mathbb{E}[L(n+1)] \le -\varepsilon \mathbb{E}\left[\sum_{ij} W_{ij}(n)\right] + 2\mathbb{E}\left[\sum_{ij} \Delta_{ij}(n)\right] + K$$

For some constant K.

6. SIMULATION SETUP AND RESULTS

6.1 Step 1

MOVE was used to create nodes, edges, the map, flow and turns, then the generated files were used to simulate road traffic on a T-Junction in SUMO and the senor nodes were simulated using NS.

Below is the logical flow and interconnection between the above three simulators that we based on to simulate vehicles, road network, and finally the wireless sensor network.

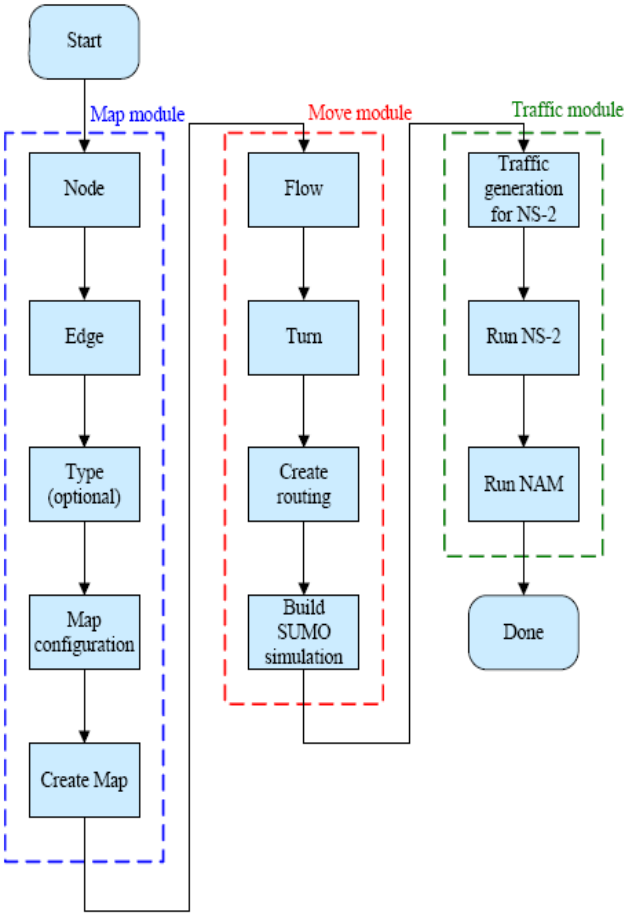


Fig 3: Interconnection of MOVE, SUMO and NS

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

Below are two screenshots showing the simulation process at particular times of the entire simulation duration.

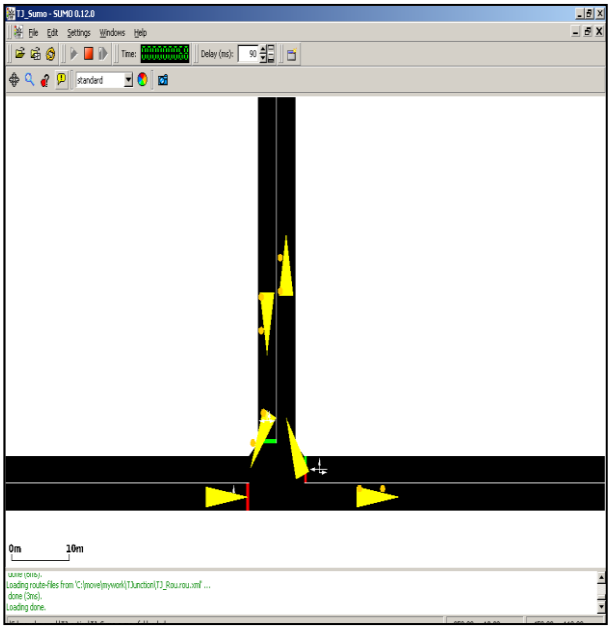


Fig 4: SUMO at sixty eight simulation time

From Fig 4 above, the traffic lights on the T-Junction are seen allocating green and red lights to the vehicles for different routes.

Figure five below shows the same T-Junction but when the applied algorithm is switching the traffic lights between yellow, red and green colors. Yellow color means about to stop, red color means stopping, while green means allowed to cross the junction.

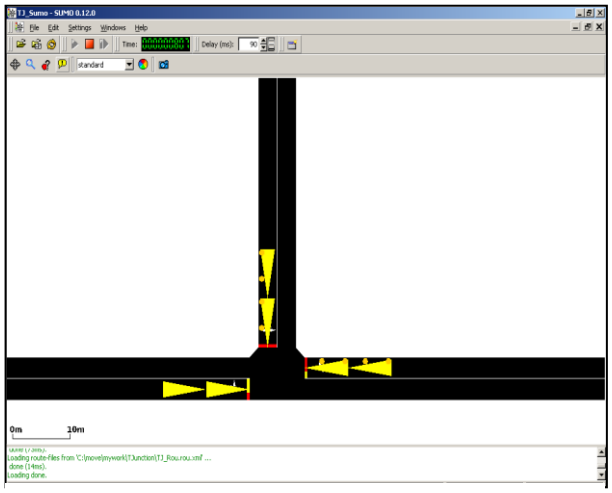


Fig 5: SUMO at eight hundred and seven simulation time

6.2 Step 2

After simulating the road traffic as shown in step 1 above, we generated trace and TCL files. We fed TCL file into NS which generated a NAM file that we used to visually see the behavior of wireless nodes in the network and how they controlled the traffic lights.

Network Animator (NAM) was used to visually see the wireless nodes in the network.

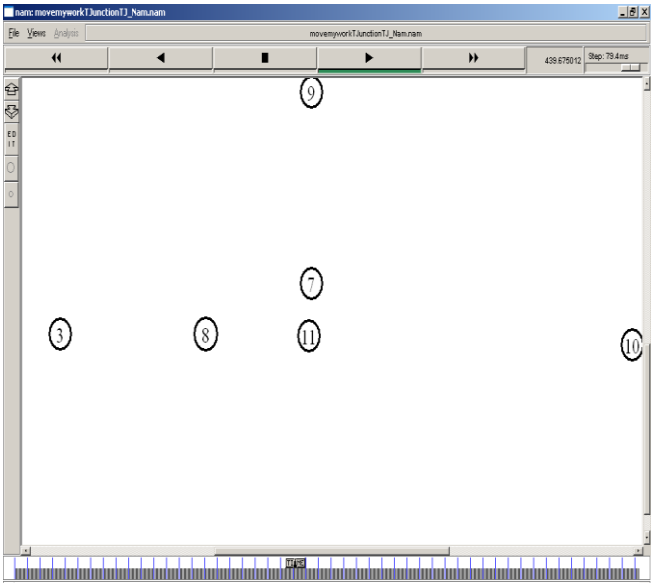


Fig 6: Wireless Nodes Moving on a T-Junction

In reference to figure 6 above, the wireless nodes are seen following the T-Junction Road Network that was produced by MOVE and implemented by SUMO. This screenshot is at Simulation Step: 79.4ms and shows node 11 exactly at the junction while the rest of the nodes are some distance away from the junction.

6.3 Gossiping –Epidemic Sensor nodes

Every node in the network gossips to its immediate neighbors about what it has generated or received, the neighbors do the same to their neighbors until all nodes in the network share data/information from every other node.

6.3.1 The Gossiping-Epidemic Algorithm Requirements

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

We considered a network of n nodes denoted by $V = \{1,..., n\}$. By letting $E \subset V \times V$ denote the set of bidirectional links along which node pairs communicate we generated a network graph denoted by G = (V,E). The network graph G changes over time in terms of V and E. Dynamics or uncertainty in the network were modeled by means of a stochastic probability matrix $P = [P_{ij}]$ where P_{ij} indicated probability that a node *i* can communicate to node j in a given time slot. In a given time slot a node communicates with at most one other neighbor.

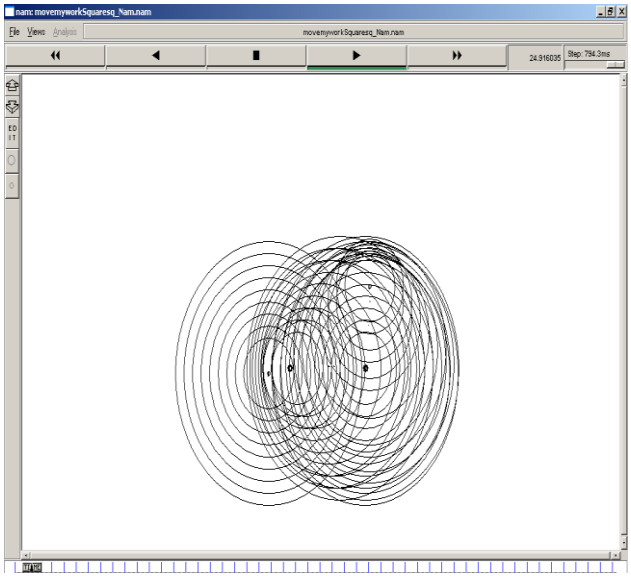


Fig 7: Four nodes moving and exchanging-gossiping messages

The performance of the gossiping algorithm was characterized in terms of the graph topology G and the dynamics matrix P. By letting d_i denote the degree of node *i* in G, i.e., $d_i = |N(i)|$, where $N(i) = \{j \in V : (i, j) \in E\}$. Without loss of generality, we assumed G to be connected (under P).

The 'Gossip' algorithms were operated at each of the n nodes of the network and below is the formal definition of these algorithms.

6.4 Gossiping Algorithm Definition

Under a Gossip algorithm we ensured that, the operation at any node $i \in V$, satisfied the following properties. (1) The algorithm only utilized information obtained from its neighbors $N(i) \triangleq \{j \in V : (i, j) \in E\}$, this helped us generate a set of gossiping neighbor nodes i and j. (2) The algorithm performed at most $O(d_i * \text{poly}(\log n))$ amount of computation per unit time. (3) We let $|F_i|$ be the amount of storage required at node i to generate its output. Then the algorithm maintained $O(\text{poly}(\log n) + |F_i|$ amount of storage at node i during its running. (4) The algorithm did not require synchronization between node i and its neighbors, $N(i)$, this is because node i just broadcasted to the nodes within range. (5) The eventual outcome of the algorithm was not affected by changes in $N(i)$ during the course of running of the algorithm.

The deployed Gossip algorithms computed a generic network function. We let each node have some information, x_i which denoted the information of node $i \in V$. The node $i \in V$ would always compute a function $f_i(x_1, \ldots, x_n)$ using a Gossip algorithm. Also, it would obtain a good estimate of $f_i(x_1, \ldots, x_n)$ as quickly as possible. Identifying the dependence of the computation time of the Gossip algorithm over the graph structure G and the functions of interest $f_1, \ldots, f_n$ were critical.

7. CONCLUSION

This paper has largely highlighted scheduling and optimization algorithms that we developed to control traffic lights on a T-Junction. It was evident that MRLF algorithm performed far much better compared to the rest. We have also given the overview of current road traffic simulators and WSNs simulators. The aim was to show that the current ITS simulators can be used to test the developed algorithms aimed at effectively controlling traffic in cities. This paper is by no means exhaustive of the algorithms for scheduling and optimization of traffic lights in cities. Our future work will expound on this and venture more on proving the theorems and lemmas, also we plan to extend such algorithms to bigger networks involving very many junctions in a big city.

International Journal of Scientific Knowledge
Computing and Information Technology
© 2012- 2013 IJSK & K.A.J. All rights reserved
www.ijsk.org

7. REFERENCES

[1] Lars Hoehmann, Anton Kummert, "Mobility Support for Wireless Sensor Networks Simulation for Road Intersection Safety Applications", in *Proceedings of Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium, 2009.*

[2] Green Light District Simulator, http://sourceforge.net/projects/stoplicht/

[3] Simulation of Urban Mobility, http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Main_Page

[4] http://www.tinyos.net/dist-2.0.0/tinyos-2.x/doc/html/tutorial/ lesson1.html

[5] http://www.isi.edu/nsnam/ns/

[6] Malik Tubaishat, Yi Shang, Hongchi Shi, "Adaptive Traffic Light Control with Wireless Sensor Networks", IEEE, 2007

[7] OMNet++, http://www.omnetpp.org/

[8] P. Levis, N. Lee, M. Welsh, D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*

[9] jist/swans, http://jist.ece.cornell.edu/.

[10] OPNET, http://www.opnet.com/

[11] E. Brockfeld, P. Wagner, "Testing and Benchmarking of Microscopic Traffic Flow Simulation Models", *In 10th World Conference on Transport Research (WCTR), Istanbul, Turkey, July 2004*

[12] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," J. Phys. I France, vol. 2, pp. 2221–2229

[13] http://www.ptvamerica.com/vissim.html

[14]http://www.sias.com/ng/sparamicshome/sparamicshome.htm

[15]http://portal.acm.org/citation.cfm?id=957289.957323