Research of Real-time Control Algorithm for Traffic Lights Based on CPU Process Scheduling

Jinrong Zhou[1,*] Xiaofang Zhou[1] Zhibin Chen[1] Xiao Chen[1]
[1]Department of Physics and Electronic Information
Zhangzhou Normal University
Zhangzhou, China,363000
[*]Email: jinrongzhou@163.com

Abstract—**To solve the problem that current transport systems cannot provide real-time response to traffic changes, this paper proposes a real-time control algorithm for traffic lights based on CPU process scheduling to reduce the waiting time of vehicles. This paper proposes a system model of implementing the real-time algorithm. First, this paper designs the appropriate hardware system based on different algorithm requirements for traffic flow information. Then extract useful information through data processing. Finally, based on different requirements for fairness and access efficiency, this paper designs five different control algorithms. The algorithm this paper proposed can support countdown-time traffic lights; it will effectively improve the efficiency of traffic.**

Keywords-intelligent transport; traffic lights; real-time control algorithms; process scheduling model.

I. INTRODUCTION

The current intelligent traffic-lights control system can be categorized into two main classes: one is an independent control system designed for single traffic intersection, such as infrared sensor traffic flow; the other is global optimal scheduling of the whole region traffic signal system using expert system. The former method is weak in stability, slow in response speed. It is not widely used in practice. Now, the traffic lights real-time intelligent control system is used widely by image processing and DSP [1]. In theory, it may achieve higher efficiency. But the method makes it difficult to implement countdown function. The second kind of systems is mainly using neural network and other intelligent control methods, such as genetic-clustering algorithm [2]. Although in theory this approach can achieve good results. But the method cannot provide a clear framework for network knowledge representation, which make it not applicable in practice. To achieve real-time control of traffic lights system must be analyzed and modeling of traffic flow, such as the use of intelligent control of traffic lights enhanced learning algorithm. This paper presents an algorithm based on independent single intersection control systems, the algorithm uses countdown-style timer and has a strong real-time control capability and relatively low hardware requirement. What's more, in this algorithm choices can be made according to traffic efficiency, control fairness, accuracy and hardware complexity.

For the common intersection traffic light control system, the prevailing rule used in China is that the straight line should be separated from the left line and the right line is a full-time passage. This traffic rule is better than the traditional rule that the right line, the left line, and the straight line share one common traffic light. This rule, as we are using now, reaches our international standard but it still has two problems: first, when there are sidewalks in intersections, as the right line is a full-time passage; problems may appear when the right-moving vehicles and pedestrians meet. To solve this problem, we can close the right-line when the pedestrians are crossing the road. Second, in some junctions, there may be few left-moving vehicles due to some geographical factors. As it may cause unnecessary waiting time, so it may save a lot of time that skipping this direction and making the vehicles other directions pass through first. It means the access order can be optimized according to real situation.

In order to reduce the total waiting time of vehicles, this paper proposes a real-time traffic light control algorithm based on CPU process scheduling, and also discusses its algorithm design model.

II. TRAFFIC LIGHT CONTROL MODEL BASED ON CPU PROCESS SCHEDULING

In a computer, the operating system achieves the allocation of CPU resources through the process of scheduling. The basic definition is as follows: Process is a running procedure on a data collection. It is a unit used for resource allocation and scheduling of the system. The thread is a relatively independent process flow or control flow .It is an entity the processor allocated. In the transport system, access procedure of different directions can be recognized as separated processes, the passage of different directions can be recognized as different threads. The threads are included in the processes. As shown in Figure 1, the schematic diagram shows the different traffic intersection numbers: 1,3,5,7 are the left-moving lines. 2,4,6,8 are straight lines and 9 to 12 are right-moving lines. The system set up has the following properties:

- The process here is a special process, and it will never finish running. Therefore, preemptive scheduling algorithm must be used.

This word has been supported by the Natural Science Foundation of Fujian Province and the Research Foundation of Zhangzhou Normal University ,china. (NOS.2008J0036,SK09009)

978-1-61284-632-3/11/$26.00 ©2011 IEEE

- The system comprises four processes: two straight-line processes and two left–line processes. Because right-moving lines are closed when the straight-moving lines are open, the left-moving processes have two left-moving threads and four straight–moving threads, and the straight-moving process has two straight-moving threads.
- Every process has totally three states: operation, ready access and waiting access. The relationship of the three states is as shown in Figure 2. Among them, an operation is the state that the process (set the direction of movement) is open. Ready access means the process has been qualified and it is waiting for the end of the countdown time (in fact, it runs more like a quasi-state).Waiting access means the process is waiting, and it does not have the traffic conditions.

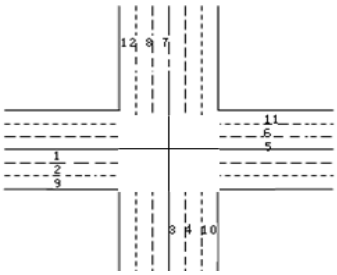

Figure 1. Traffic intersection number.

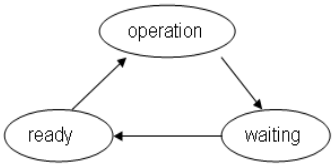

Figure 2. Process states.

The general flowchart is shown in Figure 3：

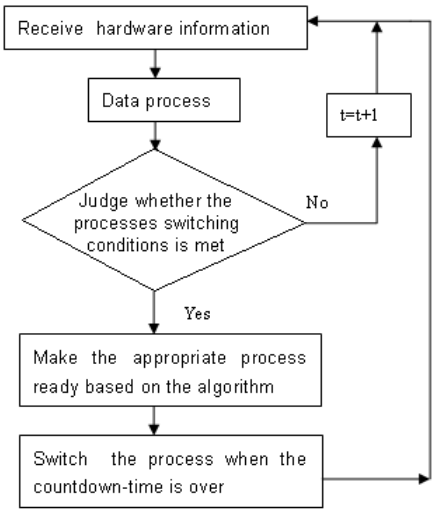

Figure 3. The flow chart of processes scheduling model.

III. HARDWARE SYSTEMS AND BASIC DATA PROCESSING METHODS

As the system the paper presents is a real-time control system, countdown-style lights and a master control with a strong computing ability must be applied. In addition, a monitor to detect traffic situation is also in need.

According to variable traffic information need of different algorithms, two different systems can be built up. In the first system, one monitor can be installed at the junctions and the other monitor nearby to measure the number of vehicles between the two monitors and provide the number of vehicles which went through the intersection during the countdown time.

We define the countdown time t_d ,the speed normal vehicles drive V_{normal} ,then the distance between the two monitor is $t_d \times V_{normal}$,the number of the vehicles X_i （"i" is the number of intersection）. If we add another monitor, and put it a little far from the end position of the longest queue of waiting vehicles in the peak period, we can find that the number of vehicles between this monitor and the first monitor is roughly equal to the number of waiting vehicles. We call the number the approximate numbers of the vehicles. and define it $N_{i\text{-}wait}$.

The second system provides information the first step needs and the digestion time of the waiting queue. This step requires high performance monitors. In this system, following functions can be implemented:

- Determine the operational status of a car (running or stop);
- Set up a broad monitoring distance with certain width;
- Divide the monitoring range into some small fixed areas (The length “s” is the distance a car normally drives within the next second).And it is able to add up the number of cars within the region at the same time.

At first, we define junction number as “i”. We already know the number of the monitor at the end of the queue k_{i1} , the total number of waiting vehicles is $\sum_{k=1}^{k_{i1}} M_{ik}$ .And M_{ik} is the number of vehicles in the region No. k. K_{i1} : is the monitor number at the end of the queue currently. Then calculate the number of the vehicles entering the waiting queue after the time t_d .We can define the distance between every two monitors is the distance a car pass by in one second. So after the time t_d , the number of vehicles entering the waiting queue is $\sum_{k=1}^{k_1+t_d} M_{ik}$, Then calculate the digestion time that all the waiting vehicles passing through the intersection will cost. The digestion time of waiting queue divides into two parts: The first part is the time all car starting motors should

cost. The second part is the time the end car of queue passing through the intersection will cost.

In the first stage, the number of the vehicles in the original queue is "Lnum" (Lnum= $\sum_{k=1}^{k_1+t_d} M_{ik}$). The distance between every two stopped cars is D_{stop}, when the car at the end of the queue starts, all the vehicles are moving slowly. The distance between every two cars is D_{mov} .And the ratio of the cars not through the intersection to the cars waiting originally is $\frac{D_{stop}+l}{D_{mov}+l}$ .At this time, the ratio of the cars already through the intersection to the cars waiting originally is :

$$K_{sm}=1-\frac{D_{stop}+l}{D_{mov}+l}=\frac{D_{mov}-D_{stop}}{D_{mov}+l} \quad (1)$$

The time it cost is: $T_{d1}=\frac{K_{sm}\times\sum_{k=1}^{k_1+t_d} M_{ik}}{N_0}+t_q$ (2)

Obviously, during this time ,the coming vehicles will enter the queue from the back of the queue tail.These cars are in the region K_{i1} to $\overline{k_2}=INT(k_1+T_{d1})$,add up the number of middle cars $\sum_{k=k_{i1}}^{\overline{k_2}} M_{ik}$,take these cars into consideration .The time from the first car starts to the last cars we mentioned above starts is :

$$T_{d2}=T_{d1}+\frac{K_{sm}\times\sum_{k=k_{i1}}^{k_{i2}} M_{ik}}{S} \quad (3)$$

("S" is the distance between two monitors, it is the distance a car pass by in one second).The monitor number at the end of the queue is:

$$k_2=k_1+\frac{\sum_{k=k_{i1}}^{\overline{k_{i2}}} M_{ik}\times(l+D_{stop})}{S} \quad (4)$$

When the last car starts, no vehicles enter the queue, stop the iteration if $k_n=k_{n-1}$.We will know the place"$\overline{k_0}$" the last car starts at. The place"k_0" that the car on the starting stop at and the time "T_d" the first step costs is：

$$H=\sum_{K=1}^{\overline{k_{i0}}} M_{ik}\times(D_{stop}+l) \quad (5)$$

The second step, because speed of back car is faster than the speed of the front car, so there will be another part of the vehicle catching up with queue, ratio of them is:

$$K_{sn}=\frac{V_{normal}}{V_{slow}} \quad (6)$$

（Note: $V_{slow}=N_0\times(l+D_{mov})$）

Obviously, the vehicles in range will enter the queue, then the last vehicle's place is

$$K_{final}=\frac{K_{sn}\times H}{S} \quad (7)$$

Since the distance between every two monitors is the distance a car pass by in one second. So the digestion time $T_{digest}=K_{final}$, Figure 4 is as follows.

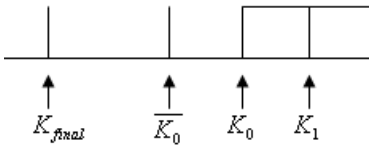

Figure 4. The picture of waiting queue

K_{final}: the place the last car starts in;

$\overline{K_0}$: at the end of K_0 , the position of car when we start time;

K_0: the queue length when the last car starts;.

K_1: the current queue length;

The algorithm flow chart of the two steps above is shown in Figure 5.

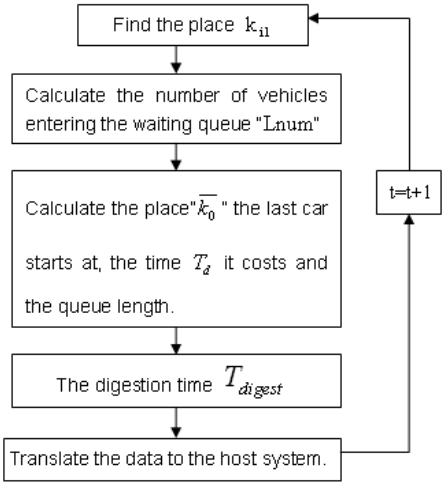

Figure 5. The algorithm flow chart of waiting queue

IV. TRAFFIC LIGHTS CONTROL MODEL BASED ON CPU PROCESS SCHEDULING

In Figure 5, the algorithm mainly consists of two parts. One step is the decision of switching processes, and the other step is the selection of the new processes.

A. Decision Algorithm of the switching processes

Because the time of traditional traffic lights is fixed in each

direction. So the condition of switching processes is the time of one direction is over. Obviously, it is not a good idea. To solve this problem, we switch the processes when the number of cars down to a certain number after the time t_d .But it is hard to know the number of cars passing through the intersection after t_d , and we can't sure the number of vehicles keeps the highest level during the adjacent time. Maybe there will be few cars passing through the intersection after t_d , but there will be many cars coming in a short time after that .It is not right to switch the traffic lights in this case. So we define "u" , "u" is in the range of 0 to 1.We compare "u" with the ratio of vehicles passing through the intersection after t_d to the max number. We take the system error in to consideration. The condition is met when the ratio above is smaller than "u". There is another problem need to considerate at the time traffic light turn green. The first car needs a period of time to start the engine. We define the time t_q .Because in this period, there is no vehicles passing through the intersection, so we add up the max number of vehicles except this time.

Then, the conditions of switching the processes are as follows:

$$\sum_{\text{The current traffic direction}} X_i < u \times 2 \times N_0 \times (t_d - t_q + t) \quad , \quad t < t_q \qquad (8)$$

$$\sum_{\text{The current traffic direction}} X_i < u \times 2 \times N_0 \times t_d \quad , \quad t \geq t_q \qquad (9)$$

Among them, "t" is time duration from the moment traffic light turn green. When $t < t_q$,we use (8).When $t \geq t_q$,we use（9）.When all the conditions are met, we switch the processes.

B. the selection algorithm of processes

For the common intersection traffic light control system, the prevailing rule China uses is that the straight line is separated from the left line. The right line is a full-time passage. This rule, we use now, reaches our international standard but it still has some problems, in some junctions, and there will be few left-moving vehicles due to some geographical factors. It will cause some unnecessary waiting time. Then we can consider skipping this direction and making the vehicles other directions pass through first. It means the access order can change according to actual situation.

CPU process algorithm contains eight different algorithms. We modify them, draw the following four algorithms, and compared according to requirements of real-time control.

- Order rotation algorithm

  When the switching conditions are met, the next counterclockwise direction process will be ready.

- Long process first algorithm

  When the switching conditions are met, the direction process, which the most vehicles are waiting, will be ready. It is Max $\{ N_{\text{i-wait}} + N_{i+4-wait} \}$, i=1,2,3,4.

  This program has the highest efficiency. But there will be some directions that the cars are always waiting because there are few cars in these directions. So the fairness is poor.

- Algorithm of the highest response ratio

  These two algorithms above only need system one on the hardware. It is hard to take into account both fairness and traffic efficiency for the two algorithms.

  So we take the algorithm of the highest response ratio. We define the response ratio $(1+\frac{\text{waiting time}}{\text{serving time}})$, the waiting time $t_{\text{i-wait}}$, we can calculate it from the former traffic. The serving time is the digestion time of waiting queue. The response ratio $R_i = 1 + \frac{t_{i-wait}}{t_{i-digest}}$, choose Max R_i .This algorithm take into account both fairness and traffic efficiency, but since every intersection need digestion time .We have to use system two and the hardware will cost more than that of two algorithms above.

  The algorithm of the highest response ratio can take into account both fairness and efficiency, but this algorithm above use the system two, Although this algorithm is more precise on control of traffic information, but the algorithms will more. So we put forward algorithm of the highest response ratio, considering reducing the level of accuracy, and use the system one. The simplified algorithm is using $N_{\text{i-wait}} + N_{i+4-wait}$ instead of the serving time above that the algorithm of the highest response used. As their dimensions are different, the ratio adding 1 is not right. The ratio in the algorithm is $R_i = \frac{t_{i-wait}}{N_{i-wait} + N_{i+4-wait}}$, the condition of switching the processes "Max R_i ".

- Dynamic priority algorithm based on probability

  The Simplified algorithms above make $N_{\text{i-wait}} + N_{i+4-wait}$ replace the serving time. And it will result that it is impossible to reflect the future traffic flow. There are many possible situations in the future. Maybe the ratio of the current direction is not the highest, but it become the highest after t_d .It is to say, we can't sure the highest ratio direction can keep highest after t_d . So we take Dynamic priority algorithm based on probability into consideration. Based on the ratio of each current process, make the ratio equal to the weight, and normalize it. Its

weight decides whether choosing it.

Make the weight $r_i = R_i^w$ .When w=0, the processes selection style is whole process random process. When w=1, the property of every process is equal to the property of R_i .When w→∞, we choose the process of the highest ratio. In theory, the current processes of the highest ratio most likely keep still. So we make "w" bigger than 1, it is in the range of 3 to 5.

Then normalize them, make $\hat{r}_i = \frac{r_i}{\sum r_i}$ .In the end, choose one process randomly and make it ready. The property of every process chosen is $\hat{r}_i$.

V. FURTHER IMPROVEMENT

In the paper, the decision algorithm of switching process has a large room to improve. For example, critical coefficient is a fixed number. To improve the efficiency of traffic, "u" should be larger in rush hour and smaller when there are few cars. In practice, the dynamic value of "u" is one direction to improve the algorithm model.

REFERENCES

[1] A. M. ZHANG, W. J. KONG, "Research of Traffic Lamp Real-time Intelligent Control System Based on Image Processing and DSP," Journal of Zhengzhou University (Engineering Science), Vol.31, No.3, pp54–56, May 2010 (In Chinese).

[2] C. B. PENG, X. Y. LI and L. F. AN, "Control Algorithm for Traffic Lights with Countdown Timer," Computer Engineering, china, Vol. 35, No. 14, pp243–246, July 2009 (In Chinese).