A Systolic Parallel Simulation System for Dynamic Traffic Assignment: SPSS-DTA

Kwangho Park* · Wonkyu Kim**

Abstract

This paper presents a first year report of an ongoing multi-year project to develop a systolic parallel simulation system for dynamic traffic assignment. The fundamental approach to the simulation is systolic parallel processing based on autonomous agent modeling. Agents continuously act on their own initiatives and access to database to get the status of the simulation world. Various agents are defined in order to populate the simulation world. In particular, existing models and algorithms were incorporated in designing the behavior of relevant agents such as car-following model, headway distribution, Frank-Wolf algorithm and so on. Simulation is based on predetermined routes between centroids that are computed off-line by a conventional optimal path-finding algorithm. Iterating the cycles of optimization-then-simulation, the proposed system will provide a realistic and valuable traffic assignment. Gangnum-Gu district in Seoul is selected for the target area for the modeling. It is expected that realtime traffic assignment services can be provided on the Internet within 3 years.

Keywords : Dynamic Traffic Assignment, Simulation, Parallel Processing, Systolic Parallel Processing, Agent-Based Modeling

* Associate Professor Dept. of Business Administration Hanyang University

** Assistant Professor Dept. of Air Transportation Hankuk Aviation University

1. Introduction

The simulation for the dynamic traffic assignment is to assign traveling cars the next link consecutively until the cars arrive at the destination. Cars come into the area randomly at a source link, drive around links, and finally sink at the destination link. The dynamic traffic assignment system must guarantee an optimal path for each car, minimizing operation costs. However, the assignment must be dynamic because link conditions change in real time.

The purpose of the ongoing project called SPSS-DTA(Systolic Parallel Simulation System for Dynamic Traffic Assignment) is to develop a model and software for a systolic parallel simulation system for dynamic traffic assignment using autonomous agents modeling. The scope of first year study is to establish the basic framework for the simulation system. Currently, the second year research is launched and focuses on the implementation of the first-year design.

The dynamic traffic assignment is widely recognized as an important aspect of modeling in transport studies. The key aspect of dynamic traffic assignment over and above the static counterpart is recognition that the variation in travel conditions that arises during peak periods will engender variations in traveling car behavior and choice. Together, these will have appreciable influence on traffic and travel conditions, and hence on the development of appropriate traffic management and control measures.

Static traffic assignment is mature and largely well understood modeling process[Sheffi, 1985; Bell and Iida, 1997]. However, the same is not true of the dynamic counterpart. Several technical issues arise in the move from static to dynamic modeling that make the latter substantially more complicated than the former. This additional complication brings with it a computational demand that extends current computing capabilities for all but the simplest road systems. Addressing these demands in itself raises further technical issues and challenges that stem from computational considerations but which cannot ultimately be separated from the transport modeling aspects. Beyond this, the effective management of transport networks when traveling cars are free to choose their routes is known to be difficult, even when dynamic effects are not considered explicitly. Although some notable progress has been made on this recently[Yang, 1997; Chiou, 1998], this remains an area of considerable research interest. The extension to include explicit consideration of dynamic effects can reasonably be expected to lead to problems of substantial difficulty and with great computational demand.

SPSS-DTA seeks to address the issues that arise in the dynamic management of congested road networks, taking into account traveling cars' responses. The general approach is one of micro-simulation using model development in the form of autonomous agents. Simulation is a numerical technique for conducting experiments on a digital computer, which may include stochastic characteristics, be microscopic or macroscopic in nature, and involve mathematical models that describe the behavior of a transportation system over extended periods of real time[May, 1990]. More references on simulation can be found[Drew,

1968; Evans et al., 1967; Martin, 1968; Mize and Cox, 1968; Schmidt Taylor, 1970; Shannon, 1975]. Simulation has been a controversial subject because of mixed outcomes of the past approaches. Strengths and weaknesses of simulation are well outlined in [May, 1990].

With the recent remarkable development of commercial multi-processor platforms, the simulation in this project is targeting individual car simulation that has been considered as an unrealistic due to the limitation of computing power at hand. The simulation system is under implementation on an NT workstation with multiple processors. The development tool chosen is a software version of PIM(Parallel Inference Machine) from Flavors, in Manchester, NH, USA.

Given the substantial computational demand of dynamic traffic assignment, use of advanced computing approaches is fully justified. Parallel computing technology provides a particularly interesting possibility in this respect. Many different approach to this are possible, varying from low-level and fine detail parallelization of arithmetic, through parallelization of important processes such as tree building, to parallel; treatment of different traveling cars. A small literature exists on application on parallel computing which may be some value in assessing this. This includes general works such as Bersekas and Tsitsiklis[1989], broadly based range of transport applications and work specific to the application of parallel computing to traffic assignment, including Van Grol[1992], Wisten and Smith[1996], and of particular interest in the present context, Greenwood and Taylor[1993].

2. Framework for SPSS-DTA

2.1 The Demand Model

The traffic is represented using a microscopic simulation approach. Individual vehicles have properties such as origin and destination, parameters for vehicle following, lane changing and gap acceptance; 4 different classes of vehicles are considered according to levels of driver information. Vehicles are generated at their origin according to an appropriate random process.

The mean level of demand appears to be determined a priori and hence the demand process is exogenous, though stochastic. This could be extended to encompass departure time choice; although an extension of that kind would call for more extensive modeling because it would entail specification of time-specific cost functions, it is a natural extension of equilibrium modeling into the dynamic context.

The use of discrete simulation of the route choice process in dynamic traffic assignment is well established in the UK Transport Research Laboratory (TRL) CONTRAM model[Leonard, et al., 1989; Taylor, 1990; Greenwood and Taylor, 1993]. In general, CONTRAM uses packets to represent several vehicles traveling together, but the value of 1 vehicle per packet would correspond exactly to the SPSS-DTA approach. Some experience established in the use of the CONTRAM model will be of interest to the present project.

2.2 The Supply Model

The supply model adopted includes a conventional link-node structure to represent the road net-

work, with centroid connectors to load and unload traffic at the end of journeys. Links have facility for signal control which can be used for traffic management and control. A volume-delay function is proposed for use in calculation of travel times of vehicles on links for use in planning of route choice. At present, vehicles are modeled as traveling through the network according to the micro-simulation model of vehicle following. Ideally, both route planning travel through the network would be addressed in the same way to ensure consistency between routes and travel times.

The TRL CONTRAM method addresses this by use of flow-delay and dynamic queuing relations to calculate travel times of packets through the network. The travel times calculated for one iteration are then used as the basis of route choice in the next, thus addressing the issue of mutual consistency.

In calculating flow propagation and travel times, certain simple models are known to satisfy all important considerations. These are the linear travel time model[Astarita, 1996] which implement the volume-delay concept as a function of the amount of traffic on the link at the time of entry. Only the case of linear sensitivity of travel time to volume of traffic on the link will ensure FIFO behavior under all circumstances. For the case of queuing to represent congestion at the exit to a link, the deterministic queuing model provides a satisfactory representation and has been used in this context[Smith and Ghali, 1990].

2.3 The Route Choice Model

At present, the SPSS-DTA model applies a single routing procedure. In view of the interest in levels of driver information, this will require some extension in the course of the project. This can be achieved without incurring the run time computational overhead of dynamic shortest path calculation by identifying in advance a suitable set of routes for each origin-destination pair. Traffic can be assigned to routes during execution of the simulation according to their pre-calculated cost at that time. Either Burrell's stochastic assignment approach which entails use of random additions to link or route costs, or Dial's logic approach to indicate probabilities could be adopted to achieve diversity in this.

Some helpful literature exists on the calculation of shortest routes in time-varying networks using travel times that reflect those encountered en route rather than those that prevail at the start of the journey. These include Kauffman and Smith[1993] who assume a regularity condition on rates of change of travel times which is equivalent to the FIFO condition that is satisfied implicitly by appropriate traffic models.

2.4 Dynamic Equilibrium

Achieving dynamic equilibrium in a large network will be an exacting task. Provided that an appropriate objective function can be identified, the approach to equilibrium can be monitored. Various algorithms can be applied to support this.

Smith's(1979) variational in-quality provides an explicit measure of departure from equilibrium which can be extended readily to dynamic assignment. If this is used as the optimization procedure, then the costs that are used should ideally reflect those arising

after the assignment of the traffic considered so that some feedback is achieved from that assignment to those costs.

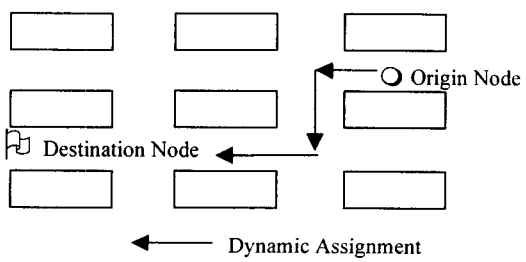
2.5 Traffic Control System

The SPSS-DTA framework includes facility for traffic management by signal control. This is seen as providing an ultimate test-bed for traffic-responsive signal control policies. This aspect of the work is particularly ambitious and the results of this aspect of the investigation will be especially interesting.

2.6 Computational Approaches

The computational approach that has been adopted is one of active agents. These appear to offer facility to represent diverse elements of the system. In the first place, individual vehicles can be represented, and this is a fundamental element of the approach. In due course, the various routes between each origin-destination pair could be represented as active agents, competing for traffic according to their cheapness at each time.

Computational approaches to parallel dynamic assignment using the TRL CONTRAM model as reported by Greenwood and Talyor(1993) suggest that assigning all traffic and then updating all travel costs afterwards provides a relatively straightforward means to parallelize the assignment process and an efficient use of computational resources.



(Figure 1) Dynamic Traffic Assignment

3. Dynamic Traffic Assignment Simulation

3.1 Problem Definition

For a predetermined city area, the system is to assign traveling vehicles the next link consecutively until the vehicles arrive at the destination as shown in Figure 1. Vehicles come into the area randomly at an origin node called centroid based pm given hourly demand, drive around links, and finally sink at the destination node, also called centroid. The system must guarantee an optimal path for each vehicle, minimizing total operating costs. However, the assignment must be dynamic because link conditions change dynamically.

In order to well specify the problem, we made the following assumptions:

- Vehicles come into the area randomly at an origin node with a destination node and the predetermined trip generation rate.
- Each link has multiple lanes.
- Vehicles can change lanes if the current lane is not suitable for the next turn or necessary.
- When a vehicle change lanes, it must find a proper gap between two vehicles running at the

destination lane. This maneuvering is assumed to arise according to stochastic rules.

- To simulate running vehicles, speed and delay must be considered.
- Each link has a given capacity based on PCU (Passenger Car Unit). Delay or travel time of the vehicle can be calculated using link capacities.
- The system must dynamically assign vehicles to the next link when they reach an intersection.
- Each intersection has a traffic signal.
- Traffic signals must be controlled by fixed plans, or a TOD(Time of Day) plans, or a real-time adaptive signal control system.
- At some point of time or position, a vehicle must be assigned to the next link so that it can get into the right lane to facilitate an appropriate turn at the signal. The decision must be made considering the operating cost that changes dynamically. The operating cost includes travel time, toll and so on. Also, the vehicle must move toward the destination. Wandering around must be avoided.

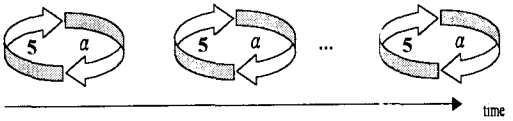
There are several types of drivers: 1) class 1: drivers with dynamic route plan, knowing the real time simulation results and modifies its path accordingly, 2) class 2: drivers with instantaneous real-time traffic information, coming out of the centroid with a minimum distance path that can be changed locally based on traffic information, 3) class 3: drivers with static traffic information, coming out of the centroid with a path that cannot be changed, 4) class 4: drivers without real-time traffic information, knowing minimum distance path by experience though.

3.2 Optimization and Simulation Cycle

The first phase of the project will not implement dynamic path optimization. Instead, optimal routes from each centroid to each centroid will be calculated off-line using some possibly an optimization method such Frank-Wolf algorithm[1956]. Based on the optimal routes, the system simulates the actual traffic situation for the next period within a tolerable time limit. Thus, the system repeats a cycle of optimization and simulation as shown in Figure 2.

The static optimal path information will be used to initialize the predetermined route from a centroid to another when a car is generated. Because the simulation cannot realistically start with zero cars on the streets, we will need a snapshot tool that can save and restore all of the internal states of the system. Note that the feedback information from the real road system is expressed in average flow rates at intersections and links. The issue is how this statistical information can be used to correct snapshot data that has diverged from the actual traffic during the process of the simulation. They are in different forms: the simulation information is in terms actual cars, the real world data is average flow rates. Certainly the flow rates will be used to throttle the global flow rate valves.

Applying the algorithm statically would not be



α : α minutes computation for an optimal path finding

5: 5 minutes simulation for the next 30 minutes

(Figure 2) Optimization and Simulation Cycle

meaningful because the links conditions change dynamically. However, such chaotic traffic environment makes optimization impractical. Although continuous computing power development makes optimization work easier, the value of optimization in real time is still low. As simulation of the traffic gets close to as it is, the result would be more practical. Nonetheless, for each cycle of the system, the algorithm would be practically applied because the dynamic shortest path calculation must be provided when drivers need that information.

The simulation rate must be 7 times faster than real time because the traffic prediction for the next 35 minutes must be made in every 5 minutes. It takes 30 minutes to transmit the results of the simulation to the physical traffic signals. Thus, updates arriving at 35 minutes intervals are acceptable. Therefore the simulation must run at 7 times faster than actual time. However, there is no need to run any faster that 25 times than real time. Thus, our system will restart in every 5+ minutes, assuming that is negligible.

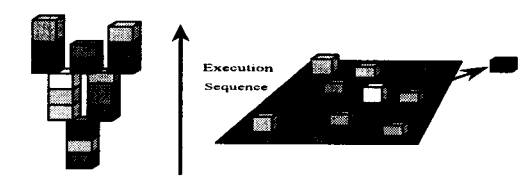
We have to show the behavior of the system in total. That means the system must simulate the behavior of individual vehicles, going straight, turning to the left or right, changing lanes and so on. The simulation must be too complex because we must consider the speed, the distance from the front vehicle, blocked vehicle, link capacity and so on. This movement can be modeled by some theories such as car-following theory.

One of the challenging part of the simulation is to allow vehicles to change lanes. Changing lanes is not simple. The intervening vehicle must find enough gap between intervened vehicles, maintain appropriate speed to bump into the space, and finally find the right time to get into the space. Some stochastic functions are available for the gap acceptance behaviors.

3.3 Agent Based Simulation

Agent is a self-directed software object with it's own value system. Agents continuously act on their own initiatives and have means to communicate with other agents. A multi-agent system is a group of identical and/or complimentary agents which act together to meet the goals of its designer. Here, agents are to simulate a real world as it is. Although agents are simple, they can have enough information to make immediate decisions with proper rules. An agent does not need to know irrelevant data, which makes simulation model simpler and faster. An agent should be a thing, instead of being a function. This means an agent must be treated as an independent object with its own rules. The rules follow the principles of real behavior of things that agents represent for.

On the other hand, conventional OR approaches determine optimal strategy based on a global view



Conventional	Agents
Touching any one module endangers the entire system	Independently add, remove, and change system modules

(Figure 3) Comparisons Between Conventional Model and Agents Model

(Table 1) Characteristics of Agents Model

Problem Characteristic	Match to Agents
Modular	Develop agents one at a time.
Decentralized	No need for single thread of innovation.
Changeable	Modify one agent without touching others.
Ill-Structured	Connectivity can develop as system operates.

of all relevant input data. Conventional approaches are open loop, while agents are closed loop. Conventional approaches are brittle, so trivial system perturbations can require a complete recalculation of optimal strategy. That makes the optimal strategy impractical in real time.

The characteristics of agents model are summarized in Table 1.

4. Agent Classes

4.1 Centroid

Centroids source cars using stochastic models driven by dynamic parameters. They also sink cars. They provide each car with its origin and destination centroids. The area being simulated is broken up into zones. Each zone has exactly one centroid that can both sink and source cars. Because there are dummy links that connect to the centroids, cars can begin or end trips anywhere in the zone. Dummy links have zero length.

When a centroid generates a car, it applies the headway distribution[May, 1990] that tells how often cars come out of the centroids. Normal distribution with dynamic distribution parameters for congested traffic, for sparse traffic, Poisson distribution, for in between, Pearson type 3 distribution (a type of negative exponential) are applied. The parameters for the distributions can vary with time and location.

4.2 Car

Major behaviors of cars are defined as follows:

- Connect with a centroid, leave the idle state and move into the first link.
- Travel within a link using the car-following algorithm.
- Select the next link in the route when it gets closer to the intersection.
- Reach the terminal centroid and go idle.

4.2.1 DoBid

DoBid agents connect with a centroid, leave the idle state and move into the first link. After reset, all cars will be in the idle state. When cars are looking for a new job, they would look in a round-robin manner at each of the centroids. If we stagger the centroid that they first look at after a reset, then the odds of a centroid waiting for service would be very low as long as there are enough available car agents.

When a centroid needs to release a car into the system, it writes the destination node value to StartCar[myCentroidNumber,DEST] and zero to StartCar [myCentroidNumber,CAR_ID]. Each of the idle cars will be looking at StartCar[myCentroidNumber, DEST]. When they see that it is non-zero, they will bid on the job by writing their ID into StartCar[myCentroidNumber,CAR_ID]. Because more than one car will bid on a new job in a single frame, odd/even bid

locking will have to be implemented to ensure that there is only one winner. When a car wins a bid, it fills in all of its parameters with initial values:

Car[myCarNum, DEST] is known from the bidding process
Car[myCarNum, POS] = 0 because it just started
Car[myCarNum, VEL] = 0 because it just started
Car[myCarNum, ACC] = +MAX_ACC because it just started
Car[myCarNum, LINK] = Route[source, dest, #link&] where #link& = 1

4.2.2 Travel

Travel agents travel within a link using the car-following algorithm[May, 1990] that applies a formula based acceleration. It is a function of the position, velocity and acceleration of the car in front of it. In sparse traffic, cars will travel at the speed limit. Nearly all developed car-following models could be related to most macroscopic traffic stream models [May, 1990]. Therefore, it can be easily understood if volume on a link increases, then the delay on the link will increase in a non-linear manner, usually proportional to the fourth power of the traffic volume. In short, a travel agent computes the next position of the car in each frame of execution.

Cars must obey traffic signals. Thus, based on the state of a signal, cars must behave properly as follows:

- Yellow: If within a specified distance from a yellow light, the car-following model by a stochastic factor is applied.
- Red: Stop in a stochastic period after the light turns red.
- Green: When light turns green, multiply the car-following model by a factor that starts out at 0 and ramps linearly up to 1 is applied. This applies only to the first car.

4.2.3 DoLaneChange

If the path that is known by a car requires a lane change, then the car will change to the proper lane when the gap is acceptable. Gap accepted by the driver can be computed by gap acceptance function. The car must know how many lanes the link has so that it does not change to a non-existing lane. If a car has not managed to change lanes before it gets to the intersection, it must change its route plan because it could not make the desired turn.

Whenever cars enter or leave a link, the list of cars in the link must be modified. When cars change lanes, the new lane's list must be propagated and the old lane's list must be shortened. This means that a car will always have ready access to its immediate predecessor's position so that it can be used by both the gap acceptance and car-following algorithms.

4.2.4 DoLinkChange

DoLinkChange agents select the next link in the route when it gets closer to an intersection. And when they reach the terminal centroid, then they go idle. When the car-following algorithm sees that Car[myCarNum, POSITION] > Link[#linkNum&, LENGTH], then it is time for a link change. Before a link change can occur, the signal at the node must be obeyed.

When Link[#linkNum&,TONODE] = Car[myCarNum, DEST], this link is the last link in the route. When this link terminates, the Car should go idle and then start bidding on new jobs that its centroid posts.

In order to simplify the problem, we made the following simplifications:

- A car can make a left hand turn without waiting for an opening in the traffic.
- A car immediately halts when it reaches the intersection regardless of the state of the traffic light at the intersection.
- Cars never slow down when they make turns; they simply halt at the light and then proceed from node as soon as the light turns green.

4.3 Signal

All road intersections have traffic signals; but some signals might constantly be green based on a static model. Signal agents must broadcast car behavior modifiers - to be used by the cars in front of the traffic signal.

Currently, the signals run at a fixed, predetermined rates but have different periods and number of phases per period. Third year project will include real-time signal optimizer.

4.4 Link

Link agents keep a doubly linked list of cars that leave and enter the link on a lane-by-lane basis. This is a doubly linked list of the rank ordering of cars within each lane for each link. Whenever cars enter or leave the link, the list must be modified. When cars change lanes, the new lane's list must be propagated and the old lane's list must be shortened. This means that a car will always have ready access to its immediate predecessor's position so that it can be used by both the gap acceptance and car-following algorithms.

This is a huge array that needs to be split up into two much smaller arrays. The value for MAXCARS is based on the longest lane in the net (2.3Km). Since the average lane length is .46km, Link[] should be split up into two classes of links; one for short links (with a much smaller value or MAXCARS) and the other for long links.

(Table 2) Number of Agents for Gangnam-Gu, Seoul

Quantity	Agent	Brief Description
44	Centroid	Sources and sinks cars into and out of the system using stochastic models with dynamic variable parameters.
113K*4 = 452K	Cars	Represent individual vehicles in one of four driver classes
68	Signals	Traffic lights at some of the intersections
583	Links	Represent the one direction of the individual traffic segments

4.5 Design for Gangnam-Gu

Simulation for Gangnam-Gu includes agents as shown in Table 2.

5. System Design & Implementation

5.1 Simulation Initialization

Because the simulation cannot realistically start with zero cars on the streets, we will need a snapshot tool that can save and restore all of the internal states

of the system. Note that the feedback information from the real road system is expressed in average flow rates at intersections and links. The issue is how this statistical information can be used to correct the snapshot data that has diverged from the actual traffic during the process of the simulation.

5.2 Simulation Output Data

After a simulation, the following information must be generated:

- Average speed, and travel time for every link
  Traffic volume for every link
- Car trace capability for some limited number of cars (~10)
- Real time probe into an intersection, for a single intersection(statistics, queue length, volume vs. time plot, etc)
- Real time probe into a single link (same capabilities)

5.3 Data Structures

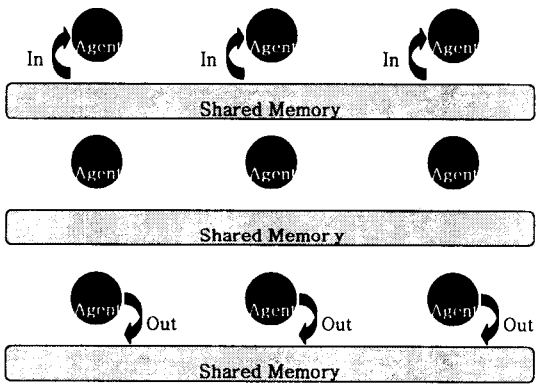
In order to speed up the simulation process, arrays are used to contain relevant data.

```
NUMNODES is the constant 2364.          // NUMNODES is the total number of nodes
XCOORD is the constant 1.               // X coordinate of the node
YCOORD is the constant 2.               // Y coordinate of the node
NTYPE is the constant 3.                // 0 => it's a centroid; 1 => it's a normal node
Node is an array[1..NUMNODES, 1..3] of paracell numbers initially 0.
// Link[432, FROMNODE] = source node of the 432nd link
// the corresponding traffic signal is always located at the end of the link
// traffic flows exclusively from the TONODE to the TONODE
NUMLINKS is the constant 583.           // 800 links
FROMNODE is the constant 1.             // source node of the link
TONODE is the constant 2.               // destination node of the link
LENGTH is the constant 3.               // length of the node in Km
LANE is the constant 4.                 // max number of lanes is 3
SIGNUM is the constant 5.               // if SIGNUM = 0, there is no traffic light
LEFTPHASE is the constant 6.            // the signal phase for turning left from this link
STRTPHASE is the constant 7.            // the signal phase for going straight from this link
LINK1 is the constant 8.                // if a car turns left from this link, it will end up this link
LINK2 is the constant 9.                // if a car goes straight, it will end up on this link
LINK3 is the constant 10.               // if a car goes right, it will end up on this link
Link is an array[1..NUMLINKS, 1..10] of paracell numbers initially 0.

MAXLANES is the constant 3.             // left, right, others
MAXCARS is the constant 200.            // at most 200 cars in a lane
CARNUM is the constant 1.               // the car number at this location in the list
```

```
NEXTINDX is the constant 2.                    // the index of the car-following this car
PREVINDX is the constant 3.                    // the index of the car in front of this car
LinkList is an array [1..NUMLINKS, 1..MAXLANES, 1..MAXCARS, 3] of machine integers.

// holds the head and tail pointers for the above doubly linked list
HEAD is the constant 1.                        // contains the HEAD of the above linked list
TAIL is the constant 2.                        // contains the TAIL of the above linked list
TAIL_P is the constant 3.                      // QQueue[ ] index where a car just entered
HeadTail is an array[1..NUMLINKS, , 1..NUMLANES, 1..3] of machine integers initially 0.

// if one of the two times is zero, then the light is always either on or off
// the signals definitions below refer to horizontal motion; for vertical motion, the definitions are
// reversed i.e. red is green and green is red
NUMSIGNALS is the constant NUMNODES.           // for this simple case, NUMNODES = NUMSIGNALS
GREEN_TIME is the constant 1.                  // the time the light is green
RED_TIME is the constant 2.                    // the time the light is red
SIG_STATE is the constant 3.                   // the current state of the signal
Signal is an array[1..NUMSIGNALS, 1..3] of paracell numbers initially 0.
// Demand[N, M] is the rate at which cars go from N to M.
NUMCENTROIDS is the constant 44.
Demand is an array[1..NUMCENTROIDS, 1..NUMCENTROIDS] of paracell numbers initially 0

// Route[I, J, N] = the link number of the Nth link in the path from I to J
// Route[ ] is read-only and calculated off-line by a separate program
MAXLINKS is the constant 10.  // no more than 10 links per path
Route is an array[1..NUMCENTROIDS, 1..NUMCENTROIDS, 1..MAXLINKS] of machine integers initially 0.

DEST is the constant 1.                        // the car's destination centroid
POS is the constant 2.                         // distance from the FROMNODE to the car's position
VEL is the constant 3.                         // current velocity of the car
ACC is the constant 4.                         // current acceleration of the car (pos or neg)
LINK is the constant 5.                        // the current link number for this car
LANE is the constant.                          // the current lane number for this car
STATE is the constant 7.                       // the current state of this car
// NEXT_LINK is the constant 7.                // ??? used for prefetch??
```

Car is an array [1..TOTAL_NUMBER_OF_CARS, 1..7] of pnumbers initially 0.

// StartCar[ ] is used by the centroids to release a car into the system.
// To start a new car, the centroid writes a destination centroid value into
// StartCar[ ]. A zero values means that there is no need for a new car
// DEST is the constant 1. // the destination centroid of the new car
// CAR_ID is the constant 2. // the ID of the car that wins the new job
StartCar is an array[1..NUMCENTROIDS, 1..2] of machine integers initially 0.



(Figure 4) Systolic Agent Execution

5.4 Systolic Parallel Processing

Each agent reads from memory, executes code, then writes results to memory as shown in Figure 4. Agents continuously repeat this 3-step execution until the system is terminated. We call such 3-step execution as frame. We intentionally devise a systolic parallel processing for the simulation. Thus, all of the agents are executed at a same speed. That is, they read data from the shared memory all at once, execute their own rules, and then, write back to the shared memory all at once. Only problem with such systolic processing occurs when multiple agents want to write to the same memory. In that case, we cannot guarantee which agent succeeds.

5.5 Implementation

The systolic parallel processing can be implemented by several methods. One alternative is to use multi-threading[Hughes and Hughes, 1997]. In our research, PIM(Parallel Inference Machine) NT version[Favors, 1999] is used. PIM is based on a state machine that tightly controls input and output variables in the systolic fashion. It has been reported that the performance of PIM is proportionally enhanced by hardware capacity.

By such systolic parallel processing, we can predict the simulation speed with given parameters, such as centroids, cars, links and so on. That's why it is well suited to real-time traffic prediction and control. In systolic architecture, no code placed in an agent can affect another agent's real-time behavior. Agents' guaranteed resources are imports from shared memory, exports to shared memory, local memory space and processing time.

6. Conclusions and Further Research

This paper presents a systolic and parallel simulation system for dynamic traffic assignment. Various previous findings are incorporated into the system.

Major decision variables are modeled as agents. Such small chunks of business rules together simulate the real traffic situation as exactly as it is. With the promising future of hardware development, the system will be proved to be a practical solution for the dynamic traffic assignment. Furthermore, realtime DTA services will be provided in the Internet through mobile telecommunication.

The SPSS-DTA modeling has addressed all of the key issues in modeling dynamic traffic assignment. Recently, a prototype system for a 4 by 4 district has been successfully implemented. Whilst the present state of development is one of a framework with some key elements in place, some important work has to be undertaken on identification of suitable approaches to all aspects.

As the project turns to the second phase, further refinement to the model has been attempted. For an example, major modifications on data structure are under processed in order to implement lane changing efficiently. With fine-tuning and rigorous testings, the system will be applied to Gangnam-Gu and provide real time assignment information to drivers.

In order to do that, however, appropriate hardware requirements must be projected because the cost factor is a major criterion to decide the practicality of DTA systems. The quality issues of a model of this kind and scale will be of great importance in successful development. For this reason, component models should be investigated and tested as they are added. Furthermore, the behavior of a system as a whole can be substantially different from that of its individual components. This suggests that the role and performance component models should be subject to review throughout the project, even if they appeared to provide satisfactory performance at their tie of addition.

Acknowledgement

This research has been supported by the Department of Science and Technology of Korea. Especially, we would like to thank Mr. Howell Mitchell of Favors Technology and Prof. Ben Heydecker of Univ. College London for their invaluable supports in modeling and outstanding consulting for this work.

References

[1] Astarita, V, The continuous time link based dynamic network loading problem: some theoretical considerations. Proceedings of the 24th PTRC European Transport Forum, Seminar D. PTRC, London, P404(1), 1996.

[2] Bell, M. G. H. and Iida, Y, Transportation network analysis. Wiley, Chichester., 1977.

[3] Bertsekas, D. P. and Tsitsiklis, J. N., Parallel and distributed computatoin: numerical methods. Prentice Hall International, London, 1989.

[4] Chandler, R. E., and Herman, R., and Montroll, E. W., Traffic Dynamics: Studies in Car-Following, Operations Research, Vol. 6, No.2, 1958, pp. 165-184.

[5] Chiou, S-W, Area traffic control optimization for equilibrium network flow. Proceedings of the 8th World Conference on Transportation Resaerch, Antwerp, July 12-17, 1998.

[6] Drew, D. R., Traffic Flow Theory and Control, McGraw-Hill, New York, 1968.

[7] Evans, G. W., Wallace, G. F., and Sutherland, G. L., Simulation Using Digital Computers, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[8] Forbes, T. W. et al., Measurement of Driver Reactions to Tunnel Conditions, Highway Research Board, Proceedings, Vol. 37, 1958, pp. 345-357.

[9] Forbes, T. W., Human Factor Considerations in Traffic Flow Theory, Highway Research Board, Record 15, HRB, Washington D. C., 1963, pp. 60-66.

[10] Forbes, T. W. and Simpson, M. E., Driver and Vehicle Response in Freeway Deceleration Waves, Transportation Science, Vol. 2, No. 1, 1968, pp. 77-104.

[11] Frank, M. and Wolfe, P., An Algorithm for Quadratic Programming, Naval Research Logistics Quarterly, Vol. 3, 1956.

[12] Gazis, D. C., Herman, R., and Potts, R. B., Car-Following Theory of Steady State Flow, Operations Research, Vol. 7, No. 4, 1959, pp. 499-505.

[13] Greenwood, D. G. and Taylor, N. B., Parallelising the CONTRAM traffic assignment model. Paper presented to the 1993 European Multisimulation Conference, Lyon, 7-9 June, 1993.

[14] Herman, R., Montroll, E.W., Potts, R., and Rothery, R. W., Traffic Dynamics: Analysis of Stability in Car-Following, Operations Research, Vol. 1, No. 7, 1959, pp. 86-106.

[15] Leonard, D. R. , Gower, P and Taylor, CONTRAM: structure of the model. TRL Report RR 178. Transport Research Laboratory, Crowthorne, 1989.

[16] Martin, F. F., Computer Modeling and Simulation, John Wiley & Sons, Inc., New York, 1968.

[17] May, A. D., Traffic Flow Fundamentals, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[18] Mize, J. H., Cox, J. G., Essentials of Simulation, Prentice-Hall, Englewood Cliffs, NJ, 1968.

[19] Pipes, L. A., An Operational Analysis of Traffic Dynamics, Journal of Applied Physics, Vol. 24, No. 3, 1953, pp. 274-287.

[20] Schmidt, J. W., and Taylor, R. E., Simulation and Analysis of Industrial Systems, Irwin, Homewood, IL, 1970.

[21] Shannon, R. E., Systems Simulation the Art and Science, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[22] Sheffi, Y, Urban transportation networks: equilibrium analysis with mathematical programming methods,.Englewood Cliffs: Prentice Hall., 1985Smith, M. J., The existence, uniqueness and stability of traffic equilibria. Transportation Research, 13B(4), 295-304, 1979.

[23] Smith, M. J. and Ghali, M, Dynamic traffic assignment and dynamic traffic control. In: Transportation and Traffic Theory (ed M. Khoshi). Elsevier, London, 273-90, 1990.

[24] Taylor, N. B., CONTRAM 5 and enhanced traffic assignment model. TRL Report RR 249. Transport Research Laboratory, Crowthorne, 1990.

[25] Van Grol, H. J. M., Traffic assignment problems solved by spedcial purpose hardware with emphasis on real time applications. Thesis for PhD, Technical University of Delft. ISBN 90-90054413, 1992

[26] Wisten, M and Smith, M. J., A distributed algorithm for the dynamic traffic equilibrium assignment problem. In: Transportation Networks: Recent Methodological Advances (ed MGH Bell). Pergamon, Oxford, 385-408, 1996.

[27] Yang, H, Sensitivity analysis for the elastic-demand network equilibrium problem with applications. Transportation Research, 31B, 55-70, 1997.