

Distributed Large-scale Network Modeling with Paramics Implementation

Henry X. Liu, Wenteng Ma

*Civil and Environmental Engineering
Utah State University*

R. Jayakrishnan, Will Recker

*Institute of Transportation Studies
University of California Irvine*

ATMS Testbed Technical Report TTR3-07

This work was performed as part of the ATMS Testbed Research and Development Program of the University of California, Irvine.* The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views of polices of the State of California. This report does not constitute a standard, specification, or regulation.

June 2005

* In cooperation with the California Partners for Advanced Transit and Highways

Distributed Large-scale Network Modeling with Paramics Implementation

Henry X. Liu, Wenteng Ma, R. Jayakrishnan and Will Recker

Abstract — Distributed simulation modeling is in urgent need for two reasons: one is that the computational power demanded by the large-scale microscopic traffic simulation keeps increasing; the other is that independently and incrementally developed sub-networks of a large region need to be modeled simultaneously due to their interactions. In this paper, we propose a distributed modeling framework which including two categories of modeling strategies, namely, light global control / independent subnets vs. heavy global control / coordinated subnets. We have implemented the distributed scheme of light global control / independent subnets and the implemented details, such as communication techniques and vehicle transferring across the boundary of two subnets are discussed. Unlike the previous studies using the dedicated high performance machines, our efforts are to utilize the low-cost networked PCs that are commonly available. By using the Application Programming Interface (API) functions supported by off-the-shelf Paramics software, we are able to distribute the computational load of microscopic simulation to multiple single-processor PCs without access the proprietary source codes of the simulation program. Performance testing and analysis of the implemented prototype demonstrate that the proposed framework is very promising.

I. INTRODUCTION

SIMULATION modeling is an increasingly popular and effective tool for analyzing transportation problems. A traffic simulator for dynamic traffic management plays two distinct roles: as an off-line evaluation / design tool and as an on-line control / guidance tool. Both roles could be computationally intensive and demand fast simulator to fulfill their tasks.

Three primary types of simulation applications immediately show the need for faster simulations: (1) *Off-line Planning*: Many types of localized traffic jams, which can only be produced on the micro-simulation level, affect people's modal and route choice. Integrated planning using

Manuscript received March 1, 2005. This work was supported by the ATMS Testbed at UC-Irvine and the California Department of Transportation.

Henry X. Liu* and Wenteng Ma are with the Department of Civil and Environment Engineering, Utah State University, Logan UT, 84322 USA (*phone: 435-797-8289; fax: 435-797-1185; email: xliu@cc.usu.edu).

R. Jayakrishnan and Will Recker are with the Institute of Transportation Studies at the University of California, Irvine, USA.

microscopic modeling requires efficient simulations of large networks of the kind needed for planning exercises. (2) *Online Control/Guidance and Emergency modeling*: The online simulation applications such as prediction for traffic control require the model to run much faster than real time. This is even more important if modeling is to be used for emergency management requiring area-wide evacuations. (3) *Monte Carlo Analysis*: The apparent global stochasticity of traffic is captured in most modern simulation approaches. In order to generate credible results, multiple simulation runs are a must.

Although Moore's law (the speed of a microprocessor would double approximately every 18 months) has been proven remarkably accurate to date [1], the complexity of the systems we are interested in simulating also keep increasing. Given the increasing network size requirements for analysis, it would remain impossible in the near future for a single processor to provide satisfactory simulation performance. This implies that harnessing additional processors in parallel and decomposing the problem domain into sub-domains is an option of promise, and perhaps the only solution.

Although dedicated high performance computing systems can significantly reduce the computational time, most transportation agencies cannot afford them. The most optimistic and affordable computational environment in the near future for most transportation agencies is a network of personal computers (PCs) connected by local area network (LAN). By distributing the computational load demanded by large-scale simulation to the inexpensive networked PCs, our goal is to relieve the computational burden and speedup the simulation.

In addition to the above issues, decomposed and distributed simulation scheme is a must when independently and incrementally developing network data sets and debugging the simulation cases in any operational computer environment. For instance, it is nearly impossible to develop a network data set for all of Los Angeles basin or the San Francisco bay area without significant decomposition of simulation efforts across analysts and modeling personnel. Therefore after the independent sub-networks are developed, there is a need to run these models simultaneously due to the network interactions.

In this paper, we propose a distributed modeling framework for the large-scale microscopic traffic

simulation. In our proposed distributed simulation environment, the target large network will be divided into sub-networks, and each sub-network will be simulated on a separate desktop PC. Our distributed computing environment consists of a network of PCs operating under Windows XP and connected by a 100 Mbps LAN within a client-server framework. This type of computing environment is low-cost and commonly available to the transportation agencies, therefore the proposed framework can be adopted readily. We have selected Paramics (PARAllel MICROscopic Simulation) as the simulation platform and developed a software toolkit using Application Programming Interfaces (API) to demonstrate the distributed modeling framework.

This paper is organized as follows. We first offer a brief overview of previous efforts on the distributed traffic simulation. We then present the general architecture of the proposed distributed modeling framework. Two categories of modeling strategies, namely, light global control / independent subnets vs. heavy global control / coordinated subnets are described. We have implemented the distributed scheme of light global control / independent subnets and the implemented details, such as communication techniques and vehicle transferring across the boundary of two subnets are discussed. Performance testing and analysis of the implemented prototype are followed. Finally, we offer concluding remarks and future research directions in the last section.

II. LITERATURE REVIEW

A. Previous Efforts on Distributed Traffic Simulation

Distributed (or parallel) simulation is an application of distributed computing which aims at decreasing the computational time by engaging different processors of a multiprocessor system or different computers of a network to share the workload of a simulation program when latter is executed [2]. There are a few traffic simulation applications have adopted distributed computing techniques, including Transportation Analysis and Simulation System (TRANSIMS) [3], Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks (AIMSUN) [4], and Parallel Microscopic Simulation (Paramics) [5], etc.

TRANSIMS is an integrated travel forecasting model designed to give transportation planners information on traffic impacts, congestions, and pollution. Distributed implementation in TRANSIMS is based on a domain decomposition principle, where the network is partitioned into domains of approximately equal size, with each CPU of the distributed computer responsible for one of these domains [6]. AIMSUN is a microscopic simulation program originally developed as a sequential, but later exported to distributed computers [4]. For distributed

simulation implementation, AIMSUN uses a sequence of instructions that executed within the context of a process to handle a group of entities that need to be updated at every time step. A process therefore can be executed in distributed fashion by the multiple processors/computers system.

B. Paramics Simulation Tools

Paramics is a suite of microscopic simulation tools used to model the movement and behavior of individual vehicles on urban and highway road networks [7]. It simulates the components of traffic flow and congestion, and presents its graphical animation output simultaneously for traffic management and road network design. One important feature of Paramics is that it allows the user to customize many features of underlying simulation model through API, so communication techniques such as message passing can be programmed to link Paramics models in different computers in the network.

Paramics was developed originally for a shared-memory Connection Machine CM-200 with 16,000 processors in 1992, using a data-distributed approach and being able to simulate approximate 200,000 vehicles on 20,000 miles of road lanes [5]. However, to make good use of CM-200, the data must be in a parallel array form so that operations can occur in parallel on the elements of the array. In 1995, based on the previous success, Paramics was further developed using message-passing interfaces (MPI) and was targeted on a 256-node CRAY T3D [5]. This solution, named as Paramics-MP, could model 120,000 vehicles at three times the real-time rates on 32 nodes of the T3D. However, since Paramics Version 2.0, the commercial Paramics-MP is not available in the market. Another notable work on the Paramics parallelization is from the group at the National University of Singapore [2]. The idea was to divide the network into several regions and simulate under different instances of the program simultaneously, allowing transfer of vehicles at the boundaries of different regions. The method was implemented in Paramics API on a multi-processor UNIX System.

It must be noted that the above parallel simulation efforts are quite different in nature with what is proposed here. The majorities of them are targeted on the dedicated high performance system with multi-processor and share memory running UNIX system. Cares must be taken for the programming of a parallel code to prevent simultaneous access of same data in the share memory. The realization of distributed simulation usually requires the access the proprietary source codes of simulation software. Instead, our efforts are to utilize the low-cost networked PCs that are commonly available. By using the API supported by off-the-shelf Paramics software, we are able to distribute the computational load of microscopic simulation to multiple single-processor PCs without access the

proprietary source codes of the simulation program. The methodology proposed in this paper is a distributed modeling framework especially suitable for path-level computations across sub-network simulations.

III. DISTRIBUTED MODELING FRAMEWORK

A. General Architecture

Before deciding the distributed architecture, CPU time consumed at each stage of the simulation process needs to be measured in order to identify which of them should be distributed. From the experience of parallelization efforts other traffic simulation software and also verified by the computation load study in our previous research [8], individual vehicle updating at each time step is most time consuming, and traffic detection and control is the second. Therefore, the basic distributed architecture is to decompose the large network into several sub-networks, and each sub-network will be simulated on a separate desktop PC.

Distributed computing systems predominantly use client-server model. Under this model, one processor called the client coordinates the other processors in the system, called servers, to function as a single computational unit. Coordination is performed through the message passing among the client and the servers. Therefore, the general distributed architecture will include at least one controller (client) and several sub-network simulators (servers). Although the controller may have various tasks related to coordinating the traffic simulation itself, the essential task from a computational architecture standpoint is the synchronization of the time in each sub-network, either at every simulation time-step or at specified intervals of times. To synchronize the simulation time, the controller will have to be able to start and stop the sub-network simulation at any time.

During a simulation run, the controller and simulators communicate over the distributed platform. The sub-network simulators act as slaves to the controller. During a time step of simulation or certain time interval, a simulator executes a non-blocking loop (asynchronous communication) while waiting for a new request from the controller. A request is simply a message associated with a specific task. When the request arrives into a sub-network simulator, it starts with an execution of the corresponding sequential code. When all simulators are “checked in”, the simulation master clock advances by one step and broadcasts the new times to every simulator in the system. Each simulator then proceeds until it reaches the master clock time. A pictorial description of the scheme to be used for distributed processing is shown in Figure 1. Depending on different simulation strategies, the controller-simulator architecture could lead to different styles of design,

including light global control / independent subnets and heavy global control / coordinated subnets, as described in the following.

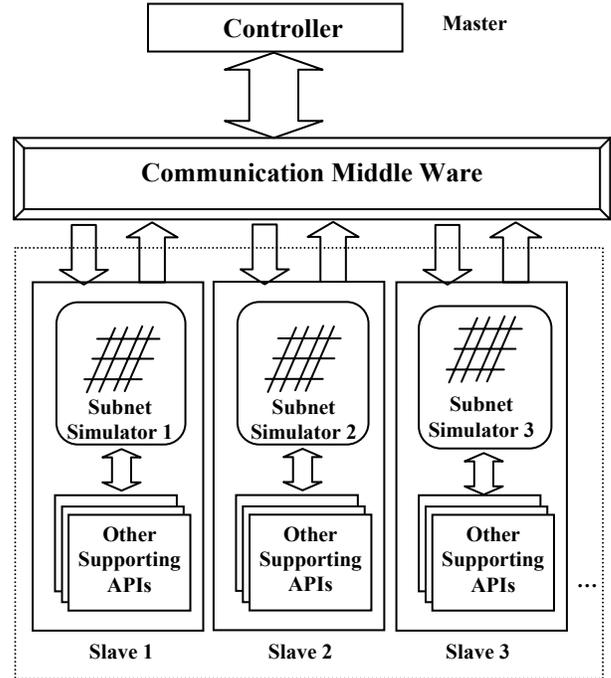


Fig. 1. General distributed modeling architecture

B. Light Global Control / Independent Subnets

The light global control / independent subnets design is the simple form of distributed simulation. In this case, each sub-network simulator has its own origin-destination demand matrix and its own route tables, but its simulation clock time is synchronized by the controller. In other words, if the time synchronization from the controller is removed, each sub-network simulator should be able to perform its simulation independently. The only task for the controller is to synchronize the time clock for each sub-network.

Communication between subnet simulators is required when an object (a vehicle) moves from one sub-network to another. A message is sent from the originating simulator to the receiving simulator describing the object, and the “ownership” of the objects is thus transferred. Once the transfer is confirmed, the vehicle object disappears at the destination zone of the originating sub-network, and the corresponding vehicle will be generated from the origin zone in the receiving sub-network.

This design features easy implementation since the vehicle routing has been taken care by the individual simulator. On the other hand, since each simulator knows only the local traffic condition in the sub-network, without knowing the big picture of the large network, this design may suffer from the unpractical or unrealistic routes taken

by some vehicles. Therefore the congestion pattern from the simulation may be distorted from reality.

C. Heavy Global Control / Coordinated Subnets

The design with a heavy global controller and coordinated subnets is at the other end of the spectrum, compared to the previous design. In this case, not only will the controller synchronize the time clock of simulators but also contain the global abstract network, global O-D matrix and global routing table. The global abstract network is a simplified network from the original large network, and used only for routing purposes. The controller will control the vehicle generation in the sub-networks and the individual vehicles' paths. The local traffic condition in the sub-networks will be reported back to the controller and used in the dynamic update of the global routing table. When a vehicle comes to the boundary of the originating sub-network, the controller will notify the receiving network to generate an identical vehicle, and continue to route the vehicle to the destination. Here, each sub-network simulator is only used to update the individual vehicle's location according to the car-following, lane-changing and gap-acceptance models, which are the most time-consuming parts in microscopic simulation.

The benefit from this design is that vehicle's origin-destination and its path are all controlled at the global level, as opposed to the local level in the previous design. In this aspect, the design is similar to the simulation over single processor in term of routing, with the distinction of updating vehicle's location over distributed processors. But the communication load between the controller and simulators is also significant higher than that of previous design, which may slow down the simulation.

D. Communication Techniques

There are several communication technologies that are popular used for distributed computation, including Distributed Component Object Model (DCOM), Common Object Request Broker Architecture (CORBA), and Windows Socket Programming (Winsock). DCOM [9] allows for peer-to-peer communications between computers, and it allows for greater flexibility in the Windows environment. The standard CORBA includes three levels, including ORB, public object services, and public applications [10]. CORBA is supported on almost every combination of hardware and operating system in existence. Windows Sockets enables programmers to create advanced Internet, intranet, and other network-capable applications to transmit application data across the wire, independent of the network protocol being used. It defines a standard service provider interface (SPI) between the application programming interface (API), with its exported functions and the TCP/IP protocol stacks [11]. Winsock is a lower level but still effective communication technique

comparing with DCOM and CORBA. Considering the complexity of the implementation of the DCOM and CORBA techniques, Winsock programming is employed in the proposed methodology.

E. Load Balancing

Since synchronous communication is used among simulators and controller, each simulator can only run as fast as the slowest one. So a proper and balanced decomposition of the network is critical to the overall performance. The computation load study in our previous research [8] shows the total computational requirement for a microscopic traffic simulation is dominated by the number of vehicles in the network at any time, the ideal division of network is to create N regions that each has exactly V/N vehicles, where V is total number of vehicles in the simulation and N is the target number of processors. The speed-up performance of the distributed processing is also dependent on the communication to computation overhead: if there are a large number of communication operations for each computational operation, the overall process will reduce in speed. In order to minimize the communication overhead, distributed simulations require methodological decomposition of the large network to find a subdivision where there are as few boundaries as possible and the computational load is spread evenly across the processors.

F. Synchronization Mechanism Using Conservative Time Window (CTW)

In the research, in order to reduce the synchronization overhead, we specify a time window that each sub-network simulator is independent within these windows and can be processed concurrently. The synchronization work will be done at the end of each conservative time window, which is greater or equal to the constant simulation time step. The problem is how to choose a proper time window. If the synchronization time window is too short, such as one simulation time-step, then the faster simulators will always waiting the slowest one and the synchronization overhead will counter the benefits from the distributed simulation. If the synchronization time is too long, the local causality constraint may be significantly violated, meaning that when the vehicle transferred from the slower simulators to the faster one, the vehicle may be transferred behind. The simulation would thus not be carried out correctly.

To circumvent this problem, we introduce the transfer vehicle forecasting technique. Figure 2 shows the vehicle-transfer status between sub-networks. Instead of detecting the information and transferring vehicle at the boundary of the sub-networks (Point B), a detector is put at certain distance to the boundary (Point A), collecting and transferring the vehicle with the forecasted simulation time clock, which depends on the distance of the detector to the

boundary, the speed of the vehicle, and etc. In this way, the local causality constraints are preserved, and then the vehicles across the boundaries between two sub-networks can be transferred at the right time, not ahead nor behind.

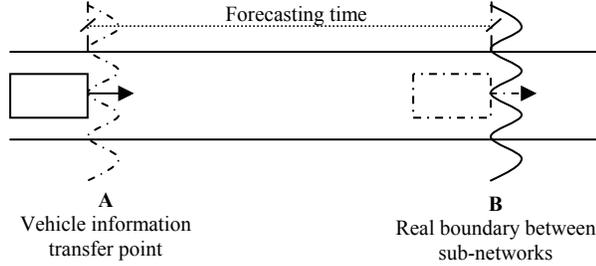


Fig. 2. Transfer vehicle forecasting

IV. PERFORMANCE TEST AND ANALYSIS

A. Prototype for Light Global Control / Independent Subnets

Figure 3 shows the prototype of the distributed modeling framework for the light global control / Independent Subnets. As we mentioned before, this framework is suitable for either decomposing the large-scale network into smaller sub-networks, or joining some independent but adjacent sub-networks and running them simultaneously with vehicles being transferred across the sub-networks.

The working process is described as follows. First, the controller will start the communication client and open a socket to listen and send messages, and all the sub-networks on the sever-computers will be loaded through remote control. Meanwhile, the sever-computers will automatic establish the connections with the client. After all the sub-networks are loaded and all the servers are successful connected with the controller client, Paramics with different sub-networks will start the simulations simultaneously. During each simulation run, the client and the servers will communicate through the Windows Socket platform. There are two types of message will be transferred, one is the synchronized time clock information, and the other is the vehicle transfer information. The server-computer usually has different simulation speed due to a variety of reasons. In order to synchronize all the server simulators, the faster simulators need to wait the slower ones after each synchronized time window that pre-defined by the user, such as 30 seconds. Once a vehicle arrive the boundary of the sub-networks, the vehicle information will be sent from the “upstream” simulator, collected and analyzed by the controller, and then transferred to the “downstream” simulator. Such communication processes will continue until all the servers finish the simulations. Finally, the controller can output the simulation results as a whole. If multiple runs needed, the

controller can also call the server simulators for the next run.

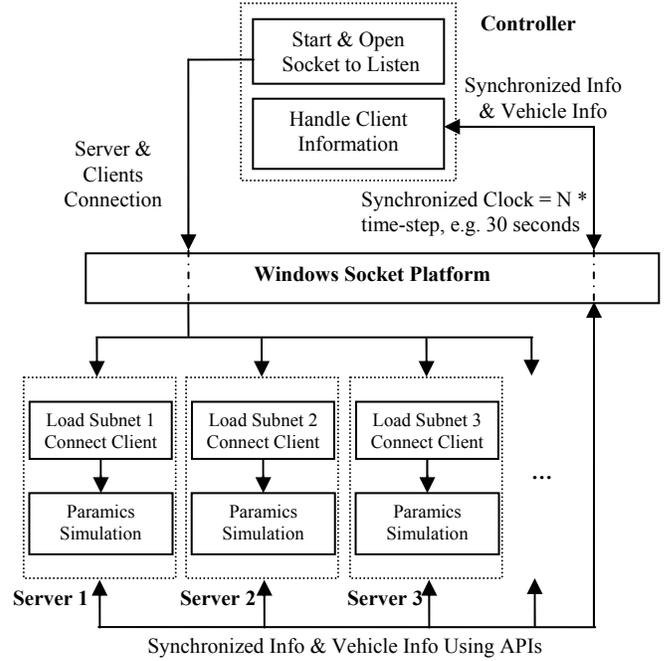


Fig. 3. Prototype of the distributed modeling framework

B. Example Problem and Test Environment

An example large-scale grid network, as shown in Figure 4, was built to test the performance of the proposed distributed modeling framework. The original network is coded with 22 arterials, 120 signalized intersections, 44 O-D demand zones, and the peak hour demand of the whole network is 33840 vehicles. Three scenarios were tested with this grid network. In Scenario 1, the grid network was evenly divided by 2 parts and simulated at 2 clients with the proposed distributed modeling framework. The sub-network boundary includes 10 connections or transfer zones. In Scenario 2, the grid network was evenly divided by 3 parts and distributed simulated on 3 server simulators, and in Scenario 3, four evenly divided networks were simulated on four server simulators, the connection zones are 10 for the left and right side parts , and 20 for the middle parts.

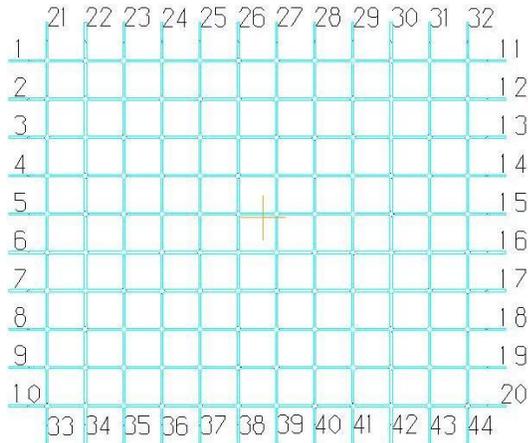


Fig. 4. Example large-scale grid

Testing was carried on four desktop PCs: Scenario 1 was carried at two desktops, each with one 2.8 GHz CPU processor and 1 GB RAM. Scenario 2 was carried with one additional workstation that has 3.2 GHz CPU processor and 2 GB RAM. Scenario 3 was carried with an additional desktop with 2.5 GHz CPU and 1 GB RAM. Each server computer was carried one part of the origin network in Paramics Version 4. All other traffic conditions and simulation configurations at each simulator are the same.

C. Benefits from the Distributed Modeling Framework

Figure 5 shows the benefits by dividing the large network with different number of subnets. It is obviously that we will get more benefits with more distributed servers. However, because the communication load will be significantly increased as the number of server increases, the speed-up benefits from the distributed computing schemes may be limited.

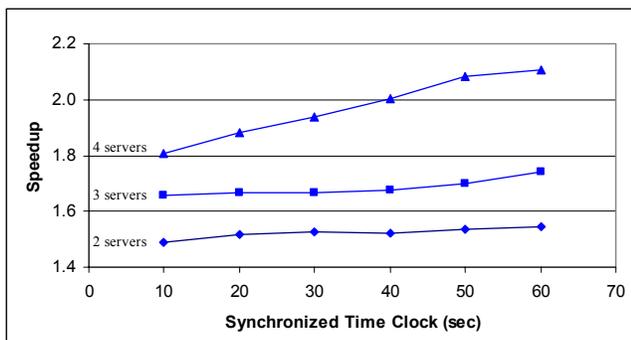


Fig. 5. Speed-up with different number of subnets

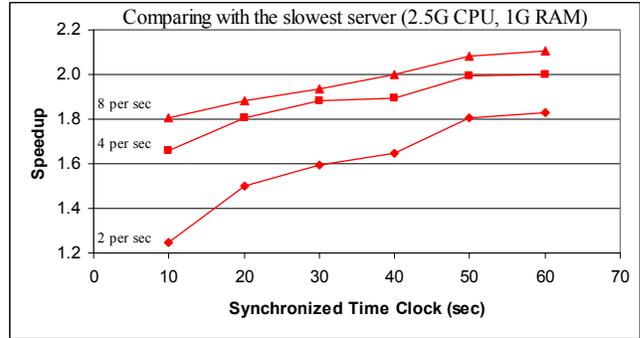


Fig. 6. Speed-up with different time-step—Four subnets

Figure 6 shows the speedup benefit of Scenario 3, the large-scale grid network was evenly divided by 4 parts and distributed simulated on four desktops. From the figure we can see that as the synchronized time window increase, we can achieve more benefits because of the communication overhead is decreased and the waiting times of the faster simulations are also decreased. Moreover, the benefit from increasing the time-step, i.e., increasing the computational load, is also significant. In this scenario, the benefit from the proposed distributed modeling framework was up to 1.8 with 2 per second time-step, and up to 2.1 with 8 per second time-step. Note that the pure computational load of 8 time steps per second is almost 4 times of that 2 time steps per second.

V. CONCLUDING REMARKS

A distributed simulation modeling methodology for large-scale network is proposed in this paper. In the proposed framework, the sub-networks from either the division of large-scale networks or being independently developed but adjacent to each other can be simulated over a cluster of networked PCs in a synchronized fashion. Windows socket programming is employed as the communication middle ware to transfer the synchronized time clock and vehicle information between the client controller and server simulators. The method was tested by a grid network with 44 O-D demand zones and 120 signalized intersections with very promising results.

The research demonstrated in this paper is a light global control and independent subnets design. To achieve a more accurate distributed simulation results, the heavy global controller and coordinated subnets design should be considered in the further research, including develop the algorithm for global routing, and network decomposition, etc.

REFERENCES

- [1] Moore, G. E. Cramping More Components onto Integrated Circuits. Electronics, Vol. 38, No. 8, April 19, 1965.
- [2] Lee, Der-Hong and Chandrasekar P. A Framework for Parallel Traffic Simulation Using Multiple Instancing of A Simulation

- Program, Intelligent Transportation Systems, Vol. 7, No. 3-4, pp. 279-294. 2002.
- [3] Bernauer, E. Breheret, L., Algers, S., Boero, M., Taranto, C. D., Dougherty, M., Fox, K. and Gabard, J. F., Review of Micro-Simulation Models Appendix D., Ref: SMARTTEST/D3, Institute of Transportation Studies, Leeds, U.K., University of Leeds, 1998.
 - [4] Barceló, J, Ferrer, J.L, García D., Florian, M. and E. Le Saux, The Distributedization of AIMSUN2 Microscopic Simulator for ITS Applications, Proc. 3rd. World Congress on Intelligent Transport Systems, Orlando, 1996.
 - [5] Cameron, G. and Duncan, G., PARAMICS-Distributed Microscopic Simulation of Road Traffic, The Journal of Supercomputing, Vol.10, pp.25-53, 1996.
 - [6] Nagel, K., and Rickert, M., Dynamic Traffic Assignment on Parallel Computers in TRANSIMS, Future generation computer systems, Vol. 17, Issue 5, pp 637-648, 2001
 - [7] Quadstone Limited, Paramics User Guide Version 4.0, Quadstone Limited, Edinburgh, UK, 2003.
 - [8] Liu, H., Ma, W. and Jayakrishnan, R. Distributed Modeling Framework for Large-scale Microscopic Traffic Simulation. Proc. of 84th Annual Meeting of the Transportation Research Board (CD-ROM), 2005.
 - [9] Microsoft Corporation. Distributed Component Object Model Protocol-DCOM/1.0, draft, November 1996. Available: <http://www.microsoft.com/Com/resources/comdocs.asp>
 - [10] Henning, Michi and Vinoski, Steve, Advanced CORBA Programming with C++, Addison-Wesley Professional, 1999.
 - [11] Microsoft Corporation. MSDN Library Online. Available:http://msdn.microsoft.com/library/en-us/winsock/winsock/windows_sockets_start_page_2.asp