

A RECONFIGURABLE COMPUTING ENVIRONMENT FOR URBAN TRAFFIC SYSTEMS

Mohamed Khalil and Evtim Peytchev
School of Computing and Informatics
Nottingham Trent University,
Burton Street, Nottingham, NG1 4BU, UK
E-mail: mohamed.khalil@ntu.ac.uk, evtim.peytchev@ntu.ac.uk

KEYWORDS

Telematics, distributed shared memory, non-locking, partial replication, reconfigurable

ABSTRACT

This paper presents a reconfigurable computing environment for building hierarchical traffic telematics distributed systems based on non-locking distributed shared memory algorithm. The algorithm aims mainly at minimising the total amount of time for data retrieval in network of workstations, considering the point of view of distributed traffic modules. The framework presented in this paper adopts a non-locking model to achieve the required performance. The presented framework develops further the successful features of DIME (developed and designed at SOCI, NTU) and at the same time avoids its shortcomings. The experimental results show that the new framework outperforms the old design of the system.

INTRODUCTION

The potential for enhancement of the performance of current urban traffic control (UTC) systems, through supervisory control layer making use of in-vehicle route guidance, has created the need for a flexible computing environment in which various new applications can be integrated with existing traffic control systems without adversely affecting their performance. Building parallel and distributed applications such as UTC system on NOW requires a middleware of software that can efficiently manage exchanging messages and data between different applications running on different machines. Traditionally there are two paradigms in building such middleware in distributed systems – the message passing (MP) paradigm and the distributed shared memory (DSM) paradigm. The former was predominantly used for building distributed systems, in which the programmers have to be conscious of where the data is and how the processes communicate with each other, making it hard to construct distributed systems using this paradigm.

Further research in finding more convenient alternatives led to the introduction of the distributed

shared memory paradigm, which provides an illusion of shared memory based on non-physical shared memory in a network of workstations where shared data reside in different address spaces. Unlike the MP paradigm, DSM algorithm facilitates accessing the shared memory and exchanging data via normal read and write operations, making life easier for programmers of parallel and distributed applications. Also, exchanging complex data between processes in different locations is supported by DSM Algorithms (Protic et al. 1996).

Research efforts in DSM paradigm have resulted in presenting number of different algorithms applying the concept of DSM abstraction (Argile et al. 1996), (Amza et al. 1996), (Li 1988). One important conclusion of the research in DSM algorithm is that, building distributed systems on network of workstations with DSM algorithm is a viable alternative to the traditional message-passing paradigm.

The Distributed Memory Environment (DIME) (Argile et al. 1996) is one of those systems that use a DSM algorithm. It provides an interface between software modules that execute on networked workstations. DIME system was designed specifically to support vast range of transport telematics applications and to offer a convenient interface to the applications programmer. As it was built as a user-level software DSM system, DIME provides an easy to use communication interface that simply and reliably delivers data and messages to all nodes in the system. This interface was built on top of Transmission Control Protocol/Internet Protocol. The implementation of the communication interface supports a variety of platforms such as DOS, Windows, UNIX and EPOC (operating system for palm-top computers PSION) (Peytchev and Bargiela 1998).

The new framework of DIME presented in this paper; called DIME-II; uses an original approach for measuring the performance of DSM algorithms, unlike many different approaches that have been introduced and taken into account in past and recent research to measure the performance of DSM systems. For example, Munin (Carter et al. 1995) adopts multiple relaxed consistency protocols in order to achieve good

performance through reducing the number of messages exchanged in the network. On the other hand, TreadMarks (Amza et al. 1996) adopts the same means, but to speed up the distributed system as a whole.

In this paper, we use an original approach for quantifying the performance of the produced DSM framework. We assume that the most important factor of measuring the performance of the system is the reduction of data retrieval time from the viewpoint of user applications. The motive behind this assumption is that the user application can have more time for performing its native tasks, which time is very often wasted in network communication. This paper presents a new software DSM framework based on a reconfigurable non-locking and partially-replicated model. It is presented as an improvement and extension to the current implementation of DIME system. The framework reflects features specific to urban traffic distributed system. This paper refers to current implementation of DIME as DIME-I, and the new improved framework as DIME-II, and when talking about common features will be using DIME.

RELATED WORKS

TreadMarks (Amza et al. 1996) is a software DSM system where messages and data traffic is reduced by relaxing consistency semantics of the shared memory. Also, it is a user-level implementation of DSM relies on UNIX standard libraries in order to accomplish remote process communication, and memory management, therefore no need to make modifications on the operating system kernel. In (Lu et al. 1995) experimental results have shown that the separation of synchronization and data transfer and the request-response nature of data communication are responsible for lower performance comparing with PVM message-passing model. In DIME-II, there is no need for synchronization mechanism for a user application to have an exclusive access to the shared memory, since each user application is associated with an intermediate memory where all its operations are performed.

BDSM (Auld 2001) is a broadcast-based, fully replicated software distributed shared memory system. Similar to our framework, each user process has an associated DSM subsystem that manages the shared memory, however, each user process has a complete copy of the shared memory where it processes all reads and writes locally. Also, unlike our system, all writes to memory modify the local copy and arrange to broadcast the updated values to all the other processes. Another major difference with the presented framework is that BDSM allows one user process to be executed on a workstation.

AN OVERVIEW OF THE TRAFFIC CONTROL TRAFFIC INFORMATION FRAMEWORK IN NOTTINGHAM CITY

The ongoing collaboration between Nottingham Trent University (NTU) and the Nottingham Traffic Control Center (NTCC) has resulted in the prototype development of a distributed computers control environment. This environment offers an efficient access to real-time traffic data collected by the SCOOT Urban Traffic Control (UTC) computers and allows various telematics applications to communicate with each other while maintaining a unified logical view of the data. The communication harness is based on a standard TCP/IP protocol and client/server architecture, which makes it easily adaptable to various UTC systems, yet independent of the physical computing hardware and network type (Peytchev and Coggan, 1999).

The configuration of the traffic control system in Nottingham city contains an installation of the SCOOT system at NTCC, an installation of bus-tracking proprietary GPS system run by ACIS-UK, a distributed shared memory system developed by NTU (called DIME system) connected to both SCOOT and ACIS control centre in Cambridge and ATTAIN system - a supervisory layer of traffic control information service for serving user route finding requests (developed by NTU). On request the system is capable of accommodating variety of other software modules i.e. a predictive macro simulator PADSIM (developed by NTU), micro simulation program HUTSIM (developed at the Helsinki University of Technology) etc. The work presented in this paper is part of the ongoing collaboration between NTU and NTCC in the UTC research in Nottingham city, UK.

DIME-I'S CONFIGURATION

DIME system (as first introduced in (Argile et al. 1996)) is a flexible computing environment that provides a communication harness for the execution of software modules of urban traffic control systems, and allows all these modules to be effortlessly integrated. As in TreadMarks (Amza et al. 1996), the system provides user-level runtime library routines to perform read/write operation on the shared memory. But, unlike IVY (Li 1988), these routines require no modifications on the underlying operating system as they can be used and run on different kinds of environments and platforms.

DIME system is designed to work with two functionally different read operations and two functionally different write operations, supporting two different types of complex data structures (Peytchev 1999). The attributes of a shared element of any data type is user-defined, thus, this system can also be categorised as object-based DSM system. Moreover,

DIME is designed to work in a heterogeneous computing environment, however, because it does not facilitate data transactions between different platforms all applications that use DIME system should perform data conversion upon exchanging data between different platforms. Furthermore, new modules can be added to the system at runtime with no disruption to the running modules.

DIME-I system adopts a sequential consistency model and a centralized architecture of the memory manager. In a centralized algorithm all accesses to a shared memory are directed to a central server that controls the whole shared memory. Thereby, DIME-I does not require any hardware routines to detect accesses to pages of shared memory; therefore it can be ported to both UNIX and DOS (or any other similar operating platform) based systems with no modifications to the kernel of the hosting operating system. In DIME-I, each user application has an additional component linked to it, which provides the communication interface via DIME-I API with the shared memory system (Figure 1). The DIME-I library transfers requests for reading/writing data from/to the shared memory over the network to the memory manager task, where they are being processed and replies are sent back. There are two components of DIME: a) The shared memory manager (i.e. DIME-II-server), which owns the shared area, and b) the communication DIME libraries (i.e. DIME-II-client), which are linked to user applications and the DIME-server in order to interface to the network (Peytchev 1999).

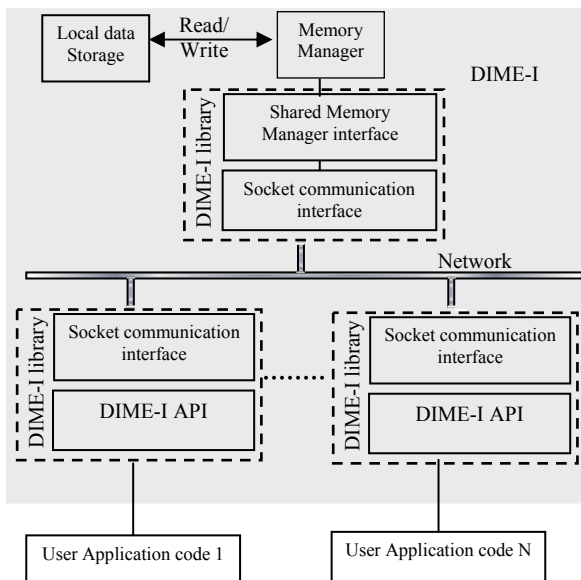


Figure1: DIME-I Configuration

In supporting the traffic simulation, monitoring and control applications, DIME system maintains two types of data structures. One type is dynamic data (data buffers), which are used for data stream collection by the real-time traffic control system. This type of data is

characterised by its high volume- in excess of 120 Mbytes per day. The other type is static data (data areas), and is updated in a much longer period of time. The purpose of this data structure, in general, is to make output results from the traffic modules available for reading by the other functional modules in the system. In the shared data areas every new write operation will destroy the old copy of data considering it obsolete (Peytchev 1999).

THE LIMITATIONS OF DIME-I

DIME-I system adopts Sequential consistency (SC) model (Lamport 1979), which is the most commonly assumed model by the programmers due to its intuitively expected throughput (Weiwu et al. 1998), (Hill 1998). However, SC definition is very strict and usually has a problem of poor performance. As shown in (Lipton and Sandberg 1988) experiments proved that any attempt, in a sequentially consistent distributed shared memory system, to change the protocol in order to improve reading operations performance makes writing operations performance worse, and vice versa. Equally important, many performance enhancement techniques, such as prefetching, multiple-issue, and write buffer, are not allowed in a sequentially consistent machine (Tanenbaum and Steen 2002).

In brief, with the definition of the sequential consistency model, a DSM system will not provide a high-quality performance, and moreover, any attempt to improve the performance using improvement techniques may not be successful or may not even be possible to implement. Therefore, subsequent definitions have been introduced to provide a consistent memory in DSM systems with relaxed or weak constraints in accordance with the nature of the required computing environment, while maintaining the usability and the programmability of the consistency model.

The main memory is controlled by only one central manager that is responsible for servicing read/write requests from/into the shared memory from all the applications in the system. This central mechanism used by the memory manager is liable to bottleneck problem, because all requests are directed to only one server. Moreover, such mechanism is unreliable, particularly if the server crashes, in which case the distributed modules of the system will lose communication with each other, leading to the failure of the entire system.

NON-LOCKING DSM MODEL

To overcome the previously detailed limitations of the current implementation of DIME-I system and in order to improve the performance of the system, DIME-I architecture has to be modified to support implementing non-locking approach, and the data

replication algorithm. This framework aims at improving the performance of DIME-I software DSM system mainly by minimizing the time of data retrieval from the viewpoint of user application. With DSM algorithms, distributed applications often waste valuable time while retrieving data from the shared memory, this time is spent by the middleware system during exchanging data and messages between different parts of the system to fetch the requested data. However, with the algorithm presented in this paper, an application retrieves the required data from a memory associated with that particular application. This intermediate memory contains copies of the data required by that application (not a whole replica of the main memory) and accessed only by that application.

The burden of making the memory consistent all around the system is entirely left to the middleware system, and applications can retrieve the data from their associated memories with no competition with each other. Thus, user applications will always find the required data without significant delay, bearing in mind that the data is retrieved directly from the intermediate memory. Providing an application with the requested data in a relatively short period of time is considered the one single most important factor of measuring the performance of the system. The motive behind this assumption is that user applications can have more time for performing their native tasks, which time is very often wasted in network communication.

Having its architecture in figure 1, DIME-I can be viewed as a system of three components, which are: DIME-I-server (the shared memory manager), DIME-I-client (DIME-I APIs), and user applications, wherein all shared data are resident in one machine controlled by one central memory manager. In DIME-I, DIME-I-client acts as an inactive process that is activated by a user application upon performing read/write operations on the central shared memory. Typically, a user application performs reads/writes on the shared memory via calling DIME-I-client routines that contact DIME-I-server to execute the prescribed operation, and then returning to its previous inactive state along with the result of the operation.

As illustrated in figure 2, the process of DIME-I-client is placed in a separate layer along with an intermediate memory holding copies of part of the whole shared memory in terms of data areas and buffers; this process is called DIME-II-client. This part of the shared memory contains the data required by the user application that uses the services of DIME-II system through its DIME-II-client. In the architecture, the shared memory consists of two types of data structures: data buffers that represent dynamic data, and data areas representing static data. DIME-II-client tasks are: establishing a persistent connection with DIME-II-server, saving updates in the intermediate memory as they are received from DIME-II-server,

sending updates to the server as they are received from its user application, sending data to the user application when requested, DIME-II-client is responsible for requests retransmission when no acknowledgement has been received as a response from the server in a predefined timeout.

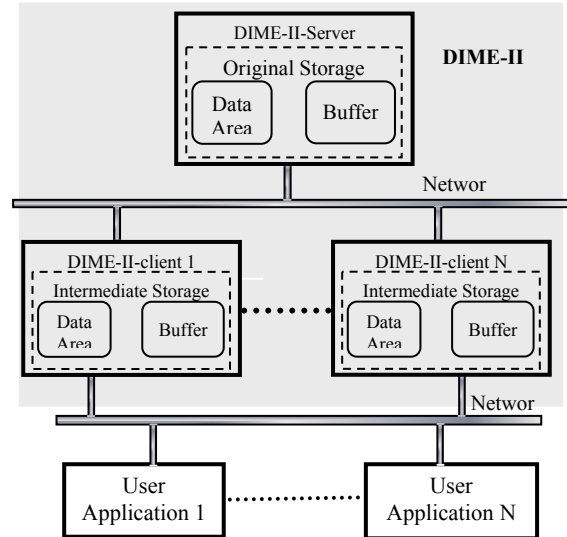


Figure 2: Non-Locking and Partially-replicated Model –DIME-II Structural Design

Thereby, a user application can perform any read or write operation on the intermediate memory rather than dealing directly with the main server, saving valuable time that can be used to perform its native tasks. In this framework, user applications; which are traffic control system modules; perform any operation on the local shared memory leaving the time delay burden of contacting the server to DIME-II-client for making the intermediate memory up to date, and reflecting updates on the original memory. On the other hand, DIME-II-server takes control over the original shared memory system, and has the role of monitoring any modification on the central memory in order to keep all intermediate memories throughout the system consistent with the original one. Unlike DIME-I, the shared memory is split over the network, and controlled by at least one memory manager. This support of the existence of several data replicas needs a special care to be taken into account to avoid data inconsistency in such architecture; a consistency model was introduced in (Khalil and Peytchev 2003) to avoid such inconsistency.

One important feature inherited from the DIME-I, is that each read/write command is considered as a single atomic operation, therefore no resources locking takes place and the system relies on the natural sequencing of the commands occurring in TCP/IP-networked environment. In regard with this feature, and because there is no need for any explicit synchronization for an application to have an exclusive access on the shared memory, the improved architecture of DIME (i.e. DIME-II) is categorized as a non-locking algorithm.

EVALUATING THE NEW FRAMEWORK

To examine the new framework, experiments were launched to quantify the performance of DIME-II system in comparison with the old system DIME-I. Both systems employ a non-locking algorithm as no synchronization mechanisms needed for having exclusive access to the shared memory, but unlike DIME-I, DIME-II system uses an approach that allows applications to have replicas of the shared memory in their proximity. This approach is expected to speed up data retrieval rates as an application can retrieve data from the local replica rather than fetching the data from the main server over the network. Therefore, in these experiments the data retrieval time from the viewpoint of the applications is considered as a major measurement factor for the performance.

For the experiments two kinds of codes was used to compare the performance of the two systems. The first code, writer, continuously performs write operations on the shared memory, whereas, the second code, reader, keeps reading from it. As it can be perceived, such codes make a significant overhead on the system, and make the network busy all the time, as the read/write operations require a persistent communication and continuous data exchanging between DIME-II-server and its DIME-II-clients. It can also provide a good idea about how the system performs with highly demanding applications. The codes were executed on three Windows2000-based machines, while the server was based on a UNIX-based machine. The experiments were conducted on the university LAN at different times of the day (i.e. included peak and off-peak time). The evaluation was based on different workloads; each workload executes only one writer and different number of readers. Workload 1 means one reader and one writer, and workload 2 means two readers and one writer, and so on.

EXPERIMENTAL RESULTS

The individual execution of the two systems with the codes on different workloads yields the results shown in table 1. The results have shown that with DIME-II system, an application can retrieve more data from the shared memory comparing to the old implementation of DIME-I. With DIME-II, data retrieval rate ranges between 5.3 Kilo Byte/ second and 26.3 Kilo Byte/ second, whereas, with DIME-I, it has a range of 1.7-2.7 KB/second. The performance chart in figure 3 shows that the performance of the system deteriorates sharply as the number of running applications increase. This undesirable scalability is explained in the next section.

Table 1: The Performance in DIME-I and DIME-II Measured as Data Retrieval Rates (Kilobytes/Second)

Workload	DIME-I	DIME-II
Workload1	2.7	26.3
Workload2	2.5	23.4
Workload3	2.4	21.6
Workload4	2.3	16.3
Workload5	2.3	12.9
Workload6	2.0	12.6
Workload7	1.7	5.3

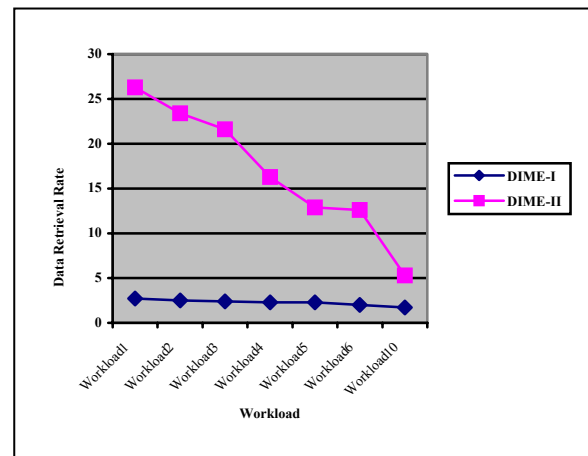


Figure 3: The Performances of DIME-II in Comparison with DIME-I

SUMMARIZING AND EVALUATING THE RESULTS USING CONFIDENCE INTERVAL METHOD

To evaluate the performance of the DIME-II system with DIME-I, a method of confidence interval is used (Weisong et al. 1998). The basic idea behind the use of this method is that a definite statement can not be made about the characteristics of all DSM systems, but a probabilistic statement about the range in which the characteristics of most systems would drop can be made. The variety of applications determines that it is not possible for one system to be better than others in all cases (Adve et al. 1996). In this evaluation, the steps of the paired confidence interval method are literally followed.

- The differences between the two systems are:
23.6 20.8 19.2 13.9 10.7 10.6 3.6
- Sample mean = 14.6
- Sample variance = 49.2
- Sample standard deviation = 7.0
- The 0.95-quantile of a t-variate with 6 degrees of freedom is 1.9.
- 90% Confidence interval for difference =
 $14.6 \pm t \sqrt{49.2/7} = 14.6 \pm 2.65 \times 1.94 = (9.459, 19.74)$

According to the paired confidence level method we can draw the conclusion that with 90% confidence level DIME-II is better than DIME-I considering that the performance is measured as data retrieval rates from the viewpoint of system modules. However, the results show that, unlike DIME-II, there is no great variation in the rates using DIME-I system as the number of applications increases. Therefore, DIME-I scales better than DIME-II.

THE TIME (IN MILLISECONDS) DIME SERVER SPENT IN LISTENING TO MESSAGES FROM THE NETWORK

In order to find out the reason of the high variation in data retrieval rates in DIME-II system, and based on the fact that the time of computation is far less than the time of communication, the time that DIME-II-server spent on the socket listening and waiting for messages from the network is roughly measured (table 2).

Table 2: The Time DIME-II-Server Spent Listening To Messages from the Network

Workload	Time	Time%
Workload1	238747	79.6%
Workload2	269753	89.9%
Workload3	282269	94.1%
Workload4	217572	72.5%
Workload5	243709	81.2%
Workload6	274755	91.6%
Workload7	234964	78.3%

This experiment shows that DIME-server approximately spends 72%-94% from the execution time listening to the network - depending on the state of the network, which means it has few time to perform the main task of executing commands on the shared memory. Put in other words, as the number of applications in the system increases the rates of data retrieval decrease due to the fact that DIME server always has the same few amount of time to share out between the increased number of applications. This kind of implicit waiting is one of the factors that degrade the performance of parallel-processing systems (Lai et al. 1997). Therefore, as the size of the system becomes large the likelihood of performance deterioration in DIME-II system becomes unavoidable. This limitation of the framework is overcome by adopting an optimization algorithm as shown next.

FRAMEWORK RE-CONFIGURABILITY

Khalil M. et al (Khalil and Peytchev 2005) has introduced a round-trip time-based algorithm that scales up the performance of the system by adjusting to the demands of the applications and the current state of the network. This algorithm aims at optimizing the performance of DSM systems by reconfiguring the system connectivity (i.e. the communication paths

connecting different parts of the system) to adjust to the current state of the network while preserving the main structure of the system. This reconfiguration can be achieved by initiating another level of service that can supply the running applications with the service rather than taking it directly from the main server.

In the presence of intermediate level of control, the communication paths of applications can be diverted from the main server and connected to nearby intermediate servers. The location of an intermediate server can be identified according to round-trip times between different parts of the system allowing the system to adjust to the current state of the system. Having several levels of control also allows more applications to be added to the system while maintaining good performance (i.e. good scalability). As shown in (Khalil and Peytchev 2005), the major requirement to implementing the algorithm is that the structural design of the DSM system has to be flexible, in order to reconfigure the communication paths without adversely deteriorating its performance. In other words, the DSM system must be flexible enough to allow the algorithm to redirect connection routes between the applications and the main server.

Having the architecture of DIME-II, it can be noticed that application communicates with the system via its DSM client (DIME-II client), and the DSM clients take the burden of sending/receiving updates to/from the main server. Therefore, it can be concluded that the structure of the DIME-II system is flexible in the sense that DSM clients can take the service from other servers residing in between DSM clients and the main server, as long as these intermediate servers have a tunnel for communication with the main server. Such flexibility is needed to implement the heuristic algorithm. On the basis of these findings, DIME-II system has been examined with and without the adaptive algorithm with 10 applications shown in figure 4.

The architectures appeared in the mentioned figure represent the different structures for DIME-II DSM system produced by the algorithm at run-time. It can be seen from the produced architectures that the algorithm has boosted the performance by up to 34% with 10 applications (figure 4b). Therefore, due to reconfigurability of the structural design of the DIME-II system, the heuristic algorithm can be implemented and improve the performance of the system, and then overcome the limitation of the original structure of DIME-II system.

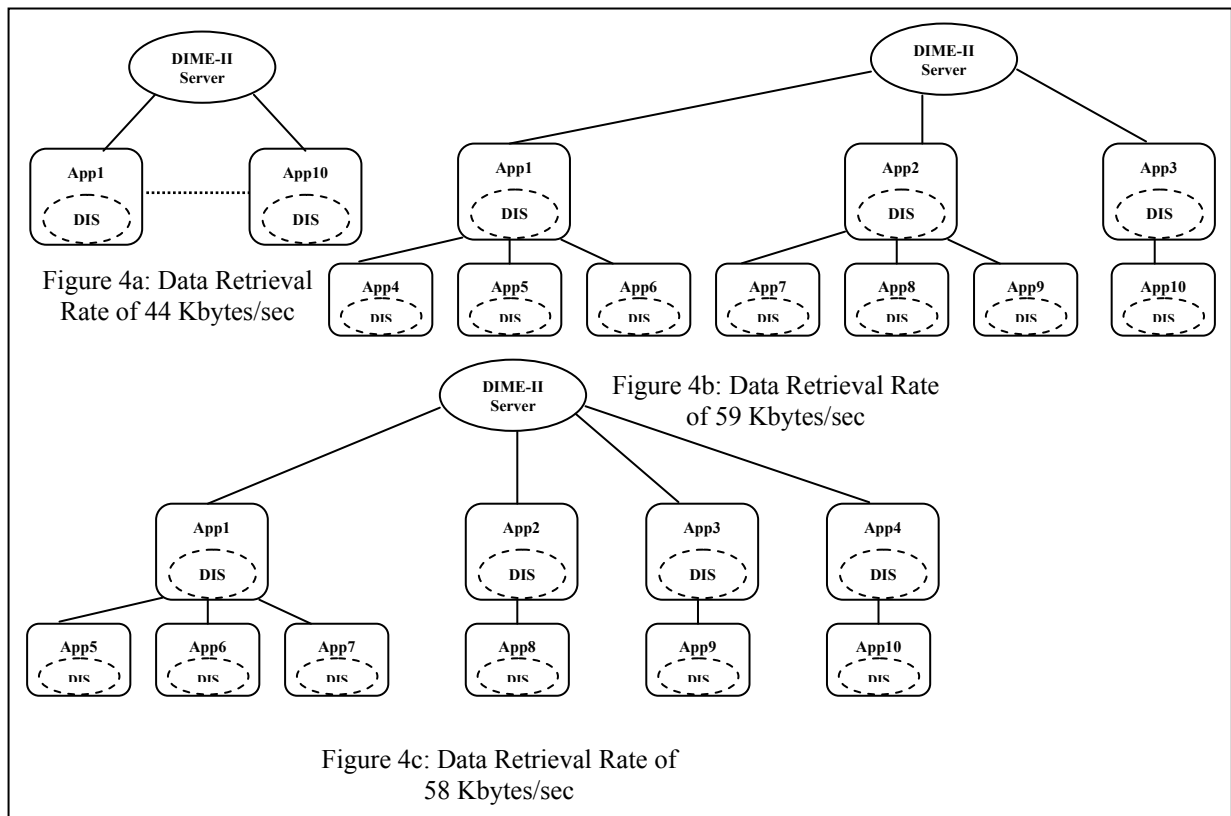


Figure 4: Different Structures of DIME-II Produced by the Heuristic Algorithm (10 applications)

CONCLUSIONS

This paper presents a description of a software DSM prototype for traffic telematics control systems. This prototype is sought to improve the performance of the whole system in many aspects: saving network resources by the exchanging the minimum number of data and messages, reducing data retrieval from user application viewpoint.

The framework of DIME-II with the non-locking approach and locality of references has show a minimization in the time of data retrieval from the viewpoint of the applications in the system. Therefore, in DIME-II, an application can have more time to execute its native tasks. On the other hand, DIME-I scales better than DIME-II in the sense that its performance in terms of data retrieval time does not change significantly. DIME-II has less scalability due to the fact that DIME server has to maintain replicas of the data required by the applications and with increased number of applications the server must maintain increased number of replicas that lead to the great variation in the data retrieval rates in the system. This limitation has been overcome by implementing a heuristic algorithm.

REFERENCES

- Adve S., A. L. Cox, S. Dwarkadas, R. Rajamony and W. Zwaenepoel, 1996 "A comparison of entry consistency and lazy release consistency implementations, *Proceedings of the 2nd High Performance Computer Architecture Conference*, 26-37, (Feb).
- Amza C., A. Cox, S. Dwarkadas, P. Keleher, H. Lu., R. Rajamony, W. Yu and W. Zwaenepoel, 1996 "TreadMarks: shared memory computing on networks of workstations", *IEEE Computer*, Vol. 29, No.2, 18-28, (Feb).
- Argile A., E. Peytchev, A. Bargiela and I. Kosonen, 1996 "DIME: A Shared Memory Environment for Distributed Simulation, Monitoring and Control of Urban Traffic, *Proceedings of European Simulation Symposium ESS'96*, Genoa, ISBN 1-56555-099-4, Vol.1, 152-156, (Oct).
- Auld P., 2001 "Broadcast Distributed Shared Memory", *PhD Thesis*, Department of Computer Science, The college of William and Mary, USA.
- Bargiela A., Peytchev E., "Intelligent Transportation Systems – towards integrated framework for traffic/transport telematics applications", *IEEE Semi-annual Vehicular Technology Conference* 6-11 October, 2001, New Jersey, USA.
- Carter J.B., J.K. Bennett and W. Zwaenepoel, 1995 "Techniques for Reducing Consistency-Related Information in Distributed Shared Memory Systems", *ACM Transactions on Computer Systems*, Vol. 13, No. 3, 205-243, (Aug).
- Hill M., 1998 "Multiprocessors Should Support Simple Memory Consistency Models", *IEEE Computer*

Journal, Vol.31, No.8, Publisher: IEEE Computer Society, USA, 28-34, (Aug).

- Khalil M., E Peytchev 2003 "Traffic Telematics Computing Framework Based on Non-Locking and Housekeeping Distributed Shared Memory Algorithm", *Sixth United Kingdom Simulation Society Conference (UKSIM2003)*, Emmanuel college, Cambridge, UK. 201-206, (Apr).
- Khalil M., E. Peytchev 2005 "Heuristic Algorithm for Performance Optimization in Distributed Shared Memory Systems", *the Eighth International Conference on Computer Modelling and Simulation (8th ICCMS)*, St. John's College, Oxford, UK, 110-115, (Apr).
- Lai A. C., C. K. Shieh and Y. T. Kok, 1997 "Load Balancing in Distributed Shared Memory Systems", *The 1997 IEEE International Performance, Computing, and Communications Conference*, 152-158, (Feb).
- Lampert L., 1979, "How to make a Multiprocessor Computer that correctly executes Multiprocessor Programs", *IEEE Transactions on Computer*, Vol. C-29, No. 9, 690-691, (Sep).
- Li K., IVY, 1988 "A Shared Memory Virtual Memory System for Parallel Computing", *Proceedings of the 1988 International Conference on Parallel Processing*, II-94 - II-101.
- Lipton R. and J. Sandberg, 1988 "PRAM: a Scalable Shared Memory", *Technical Report CS-TR*, Princeton University, 180-188, (Sep).
- Lu H., S. Dwarkadas, A.L. Cox and W. Zwaenepoel, 1995, "Message Passing versus Distributed Shared Memory on Networks of Workstations", *Proceedings of the 1995 ACM/IEEE Supercomputing Conference (IEEE Cat. No. 95CB35990)*. *ACM Part*, New York, NY, USA, Vol.1, 865-906.
- Peytchev E. and A. Bargiela, 1998 "Traffic Telematics Software Environment, Simulation Technology: Science and Art *10th European Simulation Symposium ESS'98*, San Diego, CA, USA, 378-82.
- Peytchev E., 1999 "Integrative Framework for Discrete Systems Simulation and Monitoring", *Ph.D. thesis, Department of Computing, Nottingham Trent University*, Nottingham, England, (Feb).
- Peytchev E. and, J. Coggan, 1999, "See before you go", *Traffic Technology International* - December 1999/Jan 2000 Issue. ISSN 1356-9252, 69-72 (Dec).
- Protic J., M. Tomasevic and V. Milutinovic, 1996 "Distributed Shared Memory: Concepts and Systems", IEEE, USA.
- Tanenbaum A. and M. Steen, 2002 "Distributed Systems: Principles and Paradigm", New Jersey, Prentice Hall.
- Weisong S., H. Weiwu, T. Zhimin, 1998 "Using confidence Interval to Summarize the Evaluating Results of DSM Systems", *Technical Report, Center of High Performance Computing, Institute of Computing Technology, Chinese Academy of Sciences*, (Sep).
- Weiwu H., S. Weisong, T. Zhimin, 1998 "A Framework of Memory Consistency Models", *Journal of Computer Science & Technology, Publisher: Science Press, China*, Vol.13, No.2, 110-124, (Mar).

AUTHOR BIOGRAPHIES



DR. MOHAMED KHALIL is a Research Fellow at the School of Computing and Informatics, Nottingham Trent University. Dr. Khalil was awarded his PhD July 2004 for a research titled "Integrative Monitoring and Control Framework Based on Software Distributed Shared Memory Non-Locking Model" under the supervision of Dr. Evtim Peytchev and Prof. Andrzej Bargiela. He is currently working in the Traffimatics project aiming at providing traffic information to the public as well as the traffic control for decision-making. His research interests are: Distributed shared memory algorithms: prototyping and optimization, Traffic Telematics Systems, and Web Services & Technologies.



DR. EVTIM PEYTCHEV is a Senior Lecturer at the School of Computing and Mathematics, the Nottingham Trent University and has been a member of Intelligent Simulation and Modelling group for 11 years. Most of the recent research work in the group, dealing with the traffic control telematics, has been carried out by Dr. Peytchev under the supervision and leadership of the head of the RTTS group Prof. Andrzej Bargiela. As a result of the research work Dr. E. Peytchev has successfully presented his Ph.D. work entitled "Integrative Framework for Discrete Systems Simulation and Monitoring". He worked as a researcher for the successful conclusion of an EPSRC project "Integrative framework for the predictive evaluation of traffic control strategies" (GR/K16593) and most of his publications reflect the work under this project. Dr. Peytchev's interests span: traffics simulation modelling, traffic Telematics, mathematical modelling of the uncertainties in traffic, distributed computing environments, shared memory design, Telematics technology application in the urban traffic control. He is involved in International collaboration with the Transportation Systems Laboratory at the Helsinki University of Technology (Dr. I. Kossonen) and he is principal investigator for Nottingham Trent University in the DTI funded 'Traffimatics' project.