TRAFFIC TELEMATICS SOFTWARE ENVIRONMENT

E. Peytchev, A. Bargiela.
Real Time Telemetry Systems - Simulation and Modelling Group, Department of Computing
The Nottingham Trent University, Burton Street, Nottingham, NG1 4BU, UK
e-mail: epe@doc.ntu.ac.uk, andre@doc.ntu.ac.uk

KEYWORDS

Control Systems, Urban Affairs, Transportation, Telecommunications, Distributed Processors.

ABSTRACT

This paper describes the design and implementation of a distributed computers software environment for urban traffic monitoring and control. It effortlessly accommodates diverse software modules and systems (such as real-time traffic control system SCOOT, macrosimulation prediction module PADSIM, microsimulation module HUTSIM, turning movement coefficients estimation module, mobile traffic information system etc.) without adversely affecting their performance and exemplifies the natural way for building hierarchical control systems in distributed computers environment. A control system built on the basis of such environment benefits from the use of data structures, specifically designed for real-time traffic monitoring and control (Argile et al. 1996). Easy to use communication interface provides simple and reliable delivery of the traffic data to all the nodes and modules in the system. The implementation of the communication software covers a variety of platforms such as DOS, Windows, UNIX and EPOC (operating system for palm-top computers PSION).

INTRODUCTION

The nature of traffic control systems is such that they need to operate in real-time (e.g. concurrently with the occurrence of the traffic). This requirement demands high performance computer systems. However the price of high-performance shared memory multiprocessor machines makes them, in general, uneconomical for traffic control applications so the hierarchical and distributed system architecture is the most natural solution for building a system for traffic management and control. It has been demonstrated, from applications in other fields, that hierarchically distributed systems have improved management and can collect and use large amounts of data (Kaysi 1992), (Moshe Ben-Akiva 1994), together with improved reliability and fault tolerance. The hierarchical distribution of information and control implies system modularity, which helps the implementation and the geographical growth of the system, and supports the introduction of new technologies and the evolution of existing ones. Therefore, the most promising approach to building a traffic control and management systems, satisfying the requirements listed above, is to use distributed computing resources.

The distributed computers shared memory environment DIME (which stands for a **DI**stributed **M**emory **E**nvironment), referred to in this paper as a communication environment on the basis of which the traffic control and management systems are built, was first introduced by (Argile et al. 1996). It has subsequently developed to provide a reliable and easy to use processing environment for the execution of software modules for urban traffic control systems.

URBAN TRAFFIC MONITORING AND CONTROL - PRINCIPAL STRUCTURE OF THE HIERARCHICAL CONTROL SYSTEM

Typical structure of a hierarchical traffic control system is shown on Figure 1. At the highest level of the hierarchy (District Node), the monitoring and the management of traffic flow is performed throughout a wide area, i.e. in the boundaries of a city, or in the case of a motorway control throughout the length of the controlled route. Directives containing new route or flow control instructions are obtained from prediction and control algorithms which use real-time traffic data, incident information and current control settings at the lower levels of the hierarchy.

The next level in the hierarchy is the regional node. This level coordinates traffic signals (i.e. this is the existing real-time traffic control system SCOOT level), provides freeway congestion advice and coordinates the control of local freeway segments. The primary tasks at a regional node include: using traffic flow counts to obtain a collection of traffic flow data from a defined set of local nodes; predicting and acting according to a set of regional origin and destination data; coordinating traffic signals for optimisation of the traffic flow control within and between the local nodes; detecting accidents within the geographical boundaries defined by the set of local nodes under control.

The lowest level of the hierarchy is the local node. Typical configuration of such a node consists of the traffic sensors (inductive loops) and a linked set of local processors controlling traffic signal changes (no optimisation at this level). It collects detailed traffic

flow measurements from a distributed set of sensors or from a local sensors group which measure key traffic variables in real-time (e.g. occupancy data for each link in the traffic network).

While representing the global view for the traffic control system, the generic traffic management node structure and the hierarchical structure of an advanced traffic management system take into account several important considerations for the development of the modern traffic control systems:

(a) The control on a regional level together with the control for the clusters of local nodes is usually encapsulated into a Demand Responsive Signal Settings Real-Time Control System (such as SCOOT) and is already implemented for many urban traffic networks all around the world.

(b) There are only a limited number of interventions available which a district node could use (district node according to Figure 1.). They usually are strategic interventions, concerning a subset of regional nodes (or all of them), rather than interventions concerning the control for a specific local node. These interventions are defined on the highest level only and their distinctive characteristics are as follows: they are real-time control system specific and reflect the way it controls all timing plans in the traffic network; the strategic interventions work on a different time scale in comparison with the control sequences used for operational control (e.g. traffic lights settings). Typically, an intervention, which has a lifecycle equal to the time needed for a vehicle to cross the traffic network (usually between 15 and 30 min.), is a strategic intervention for the system. In comparison the operational control interventions have lifecycles equal to the cycle of the traffic lights in the system (approximately 90 sec).

(c) The traffic signal control system uses a local area communication network or indeed shared memory computer configuration, specifically designed to meet the requirements of the time-critical data transfer between the signal settings control system modules. By contrast, the highest level of the control system configuration is built around a general LAN/WAN network.

(d) There is a wide variety of modules running in parallel on the district node level and a LAN-based distributed environment is a natural solution of the problem of providing the computational power required. A less obvious solution is to use the Internet Network (Wide Area Network (WAN)) as a network basis for the implementation of the traffic management system. As indicated by (Lum at al. 1983) and (Peytchev E. 1998), such a choice does not have a negative influence on the inter-modules data exchange. Data transfer speed of the interface across the network is enough to deliver, on time, the data required. In the same time, this environment allows new applications to be fully integrated within an existing system without adversely affecting its performance. This approach potentially allows the use of the computational power of the control system to its full extent, executing new tasks on the machines with minimal workload.

The analysis of the results, obtained by installing different generations of traffic light control systems on the field, showed that the best (responsive) control is obtained when the control strategy is free to evolve in line with the detected demand (demand responsive control systems). This conclusion shows the relative independence of the region level of the hierarchy shown on Figure 1. However, it is clear that even better results can be achieved, if reliable prediction on a large scale is present. Since all cycle to cycle modelling and queue formation anticipation is done on region node level, the prediction on a larger scale can be performed on district node level only and it would concern the traffic network in general terms (strategic traffic control) and usually the process is

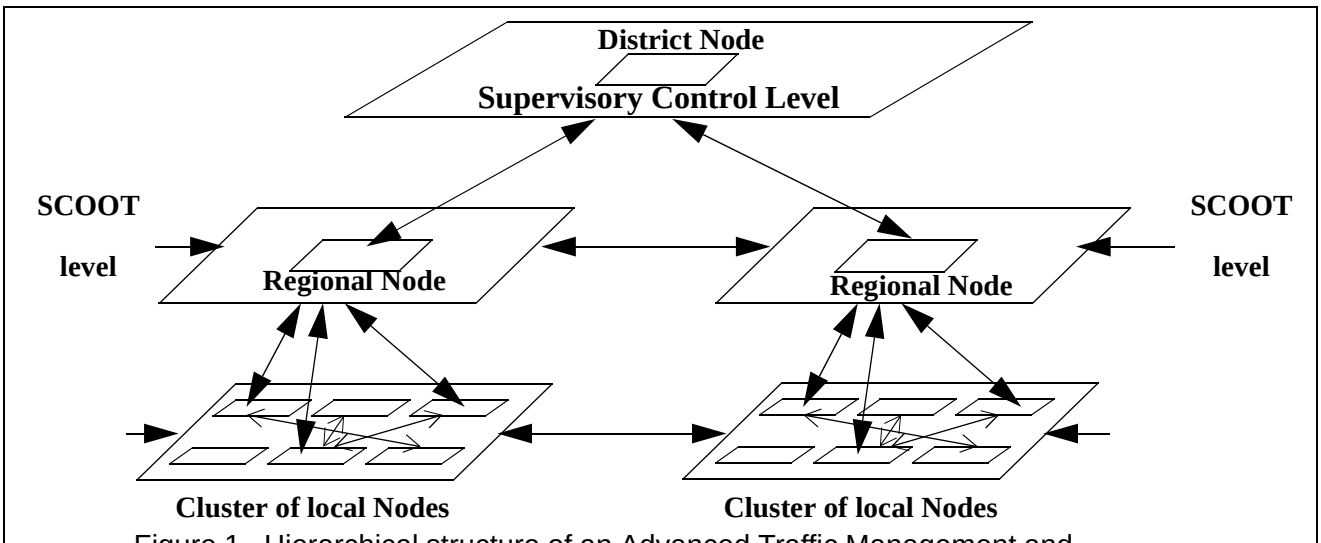

Figure 1. Hierarchical structure of an Advanced Traffic Management and Control System

referred to as supervisory control and the district node level is referred to as supervisory layer of control.

URBAN TRAFFIC MONITORING AND CONTROL - FUNCTIONAL STRUCTURE OF THE SUPERVISORY LAYER OF THE CONTROL SYSTEM

In terms of modularity, the supervisory layer of the traffic monitoring and control system consists of the following modules: Surveillance Module, Turning Movements Estimation and Prediction Module; Queue Prediction Module; Control Strategy Generation Module; Operational Control Generation Module and Communication Module (Peytchev E. 1998). The overall structure of the supervisory layer of control is shown on Figure 2.

The Surveillance Module (in an ideal system) communicates with all possible programs and devices capable of providing traffic flow information. Such programmes and devices may include traffic flow counts, in-vehicle devices, video-cameras etc.

The Turning Movement Coefficients Estimation and Prediction Module (in some systems it is replaced with OD-matrix estimation module) supplies essential information regarding the split of the traffic flow for every cross-road in the controlled traffic network. This information is later used by the Queue Prediction Module to simulate the traffic network in faster than the real-time mode and to determine the parameters of a future state of the system. These results are fed into the Control Strategy Generation Module, which generates supervisory commands (interventions), aimed at improving the overall state of the traffic. The commands are generated on the basis of the collected and predicted data in the system and are subsequently interpreted and applied on the road by the Operational Control Module (typically a system similar to SCOOT).

All interprocess communications and data exchanges are delegated to the Communication Module.

DESIGN AND IMPLEMENTATION OF THE COMMUNICATION MODULE FOR SUPERVISORY LAYER OF CONTROL

There are two alternatives for implementing the communication module: using message-passing approach and using distributed shared memory system.

The primary advantage of distributed computers shared memory systems over data-passing systems is the simpler abstraction provided to the application programmer, an abstraction the programmer already understands well. The access protocol used is consistent with the way sequential applications access data, allowing for a more natural transition from sequential to distributed computations. In principle, parallel and distributed computations written for a distributed shared memory system can be executed on a shared memory multiprocessor without the need for change. The shared memory system hides the remote communication mechanism from processes and allows complex structures to be passed by reference, substantially simplifying the programming of distributed applications. Moreover, data in distributed shared memory can persist beyond the lifetime of a process accessing the shared memory.

In contrast, the message passing models force programmers to be conscious of data movement between the processes at all times, since processes must explicitly use communication primitives and channels or ports. Also, since data in the data-passing model is passed between multiple address spaces, it is difficult to pass complex data structures. Data structures passed between processes must be packed before transmission and unpacked after reception by the application. For this reason, the code written for distributed shared memory is usually significantly shorter and easier to understand than equivalent programs that use data passing.

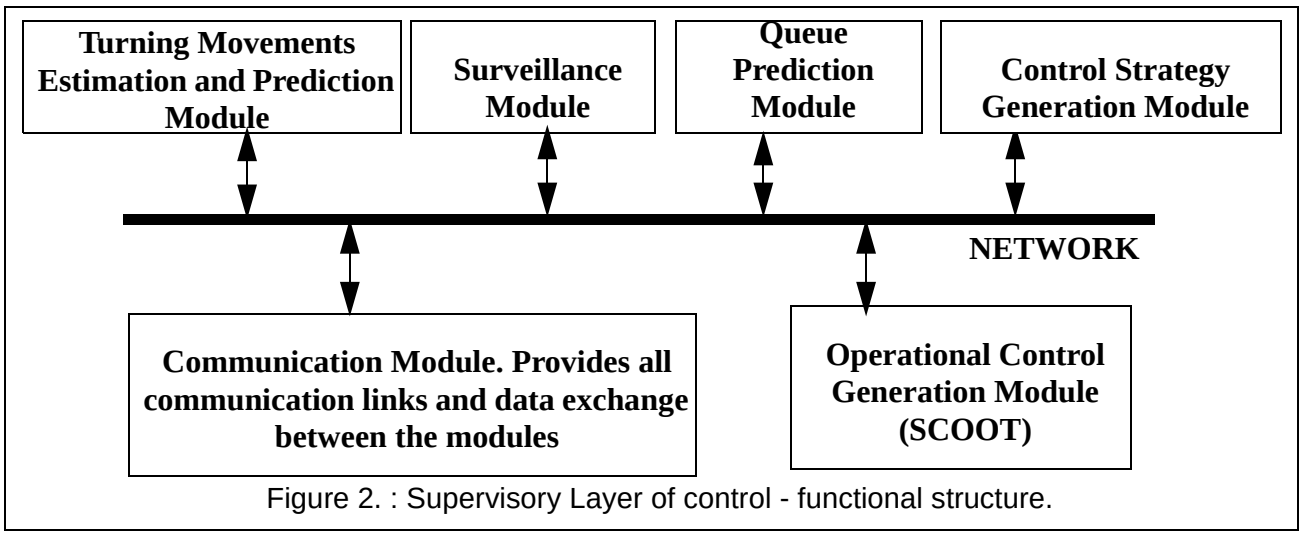

Figure 2. : Supervisory Layer of control - functional structure.

Integrating all the modules in the supervisory layer of control into one working system is a difficult task, therefore the simplicity of the communication is of paramount importance. The best choice in this case is to use distributed computers shared memory system and the DIME system introduced by (Argile 1996) is an ideal tool to perform this task.

DIME CONFIGURATION

DIME's configuration is represented on Figure 3.

Each user application code has an additional component linked to it, which provides the communication interface via DIME API with the shared memory system. The requests for reading/ writing data from/to the shared memory (creating or removing areas) are transferred by the DIME library 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 (SMM) task which owns the shared area and b) the communication DIME libraries which are linked to user applications and the memory manager in order to interface to the network.

The shared memory manager (SMM) component of DIME operates on a closed-loop basis, continually checking for the requests to access or to maintain the shared memory data structures. These requests are typically raised by the application programs executing appropriate API routines, but a provision has also been made for a keyboard entry of requests to the memory manager.

The DIME's API provides facilities for the creation and removal of shared memory structures and the read/write access to them. The synchronization of accesses to shared memory is implicit in the operation of the memory manager task thus it does not require the provision of separate library functions.

In supporting the traffic simulation, monitoring and control applications, DIME software implements two types of memory structures, an array of records and a circular buffer. These two structures are intended for use with static and dynamic data respectively.

(a) Shared Memory Structure "Array".

This type of structure represents the static type of data in the system and the most common description for its internal representation is an array of records. Typically, the array of records will store traffic network description data, that is shared between several concurrent applications and is read and updated infrequently (for example route descriptions and route travel times). Because of that, it is possible to afford a simple model for accessing the shared array, whereby always a complete array is read or written-to.

(b) Shared Memory Structure "Circular Buffer".

For the time-varying data, such as the traffic flow measurements or the results produced by the real-time simulation and control software modules, the access to the supporting memory structures is requested by applications on a second-by-second basis, so the efficiency of access to the shared memory is of utmost priority. The circular buffer data structure provides an efficient access to an individual record and it readily keeps track of the time sequence of messages. The buffer maintains a global insertion pointer and an individual extraction pointer for each of the "readers", thus enabling applications to recover from sporadic communication delays that are inevitable in a distributed processing environment.

EVALUATION

Following the design of the traffic telematics

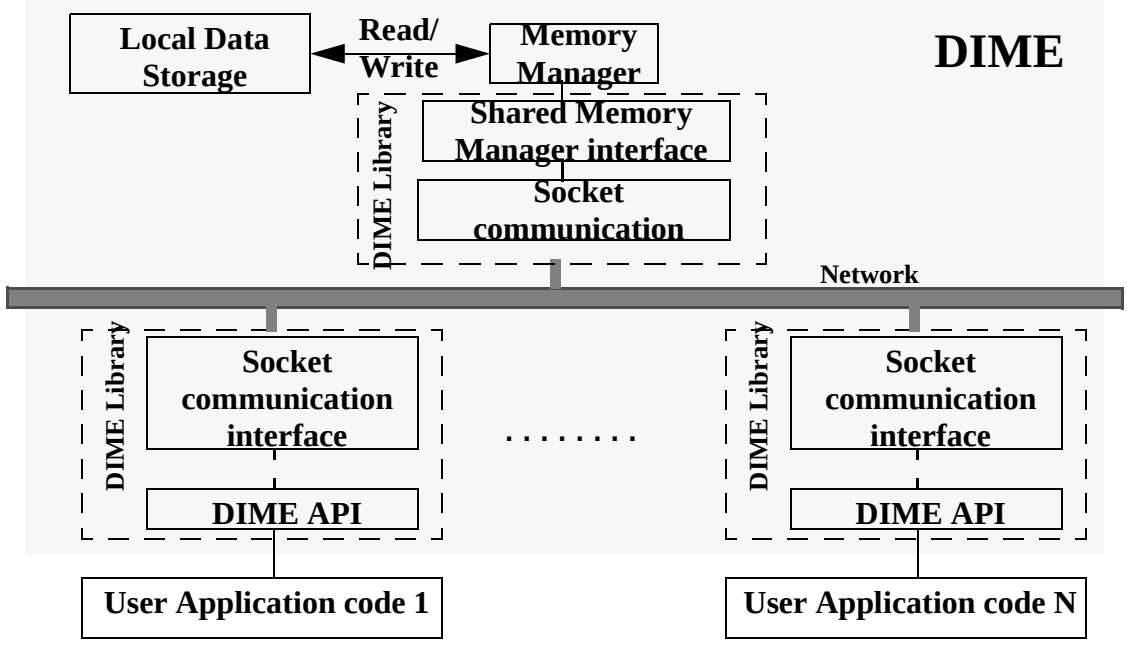

Figure 3. : DIME Configuration.

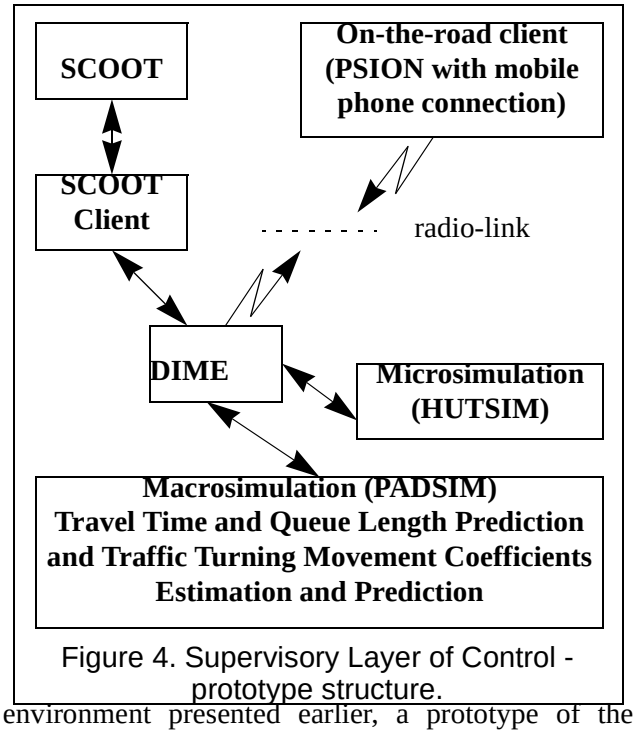

Figure 4. Supervisory Layer of Control - prototype structure.

environment presented earlier, a prototype of the supervisory layer in the traffic control system has been created. The prototype integrates a client, delivering messages from the SCOOT real-time traffic control system, predictive simulation model PADSIM (includes turning movement coefficients estimation and prediction module), microsimulation model HUTSIM and one end-client on the road, connected to the system via mobile telephone link and running on a PSION palm-top computer. The structure of the system is shown on Figure 4.

In the presented configuration, the SCOOT traffic control system delivers two dynamic flow data (so called M14 and M19 messages) and the SCOOT client makes them available by writing the information into the DIME memory manager every second. These data are read by PADSIM, which estimates and predicts the traffic turning movement coefficients and makes a prediction for the state of the traffic network for up to 20 minutes ahead. Subsequently, PADSIM makes the obtained results available for the other modules in the system through DIME. It also calculates route travel times and fills in the appropriate structures in the DIME memory manager. Some of the data (traffic turning movement coefficients) are used by the HUTSIM for microsimulation and verification. The on-the-road client connects to the system and reads the route travel times data on request from the user and displays the obtained data on a small user-friendly graphical display.

The amount of data, currently available on-line for the Mansfield region, is delivered at a rate of 3000 bytes/second and this is well below the data exchange rate provided by the DIME system, which is around 35 kB/sec for a single client and in the range of 15-30 kB/sec for 2 - 5 clients (depending on the workload of the network).

CONCLUSIONS

This paper concentrates on the design and development of a new generation traffic telematics schemes. It recognises two principal levels of traffic control schemes: Supervisory Layer of Control (Predictive Control), in which actions are being taken according to results obtained by predictive simulation and Operational Control (Demand Responsive Control), in which decisions are being made according to the current measurements (demand). Various researchers (including the authors of this paper) have shown the potential of the distributed processors environment for building traffic control systems. This paper reports the implementation of a prototype of a Supervisory layer of control in such a traffic control system. The implementation, described in this paper, uses the distributed shared memory system DIME as a backbone for interprocess communication between the modules of the system. It has been shown, that the throughput of the communication subsystem is enough to deliver the necessary traffic data on-line. This choice enables also easy mobile phone access to the system and the fact has been illustrated by displaying the on-line traffic data on the screen of a hand-held computer PSION.

The DIME system is currently available for DOS, Windows95/NT, UNIX, EPOC operating systems.

REFRENCES:

Argile A., Peytchev E., Bargiela A., Kossonen I. “Dime: A Shared Memory Environment For Distributed Simulation, Monitoring And Control Of Urban Traffic”, 8th European Simulation Symposium, Genoa, Italy, 1996, ISBN 1-565555-099-4, Vol.1, pp. 152-156.

Kaysi I., “Framework and models for the provision or real-time driver information”, PhD thesis, Department of Civil Engineering, Massachusetts Institute of Technology, 1992.

-Lum M., Kinney L.L., Kumar K.S.P., “Feasibility of a Distributed Computer Traffic Control System”, IFAC Control in Transportation Systems, Baden-Baden 1983, pp. 157-163

Moshe Ben-Akiva, Haris N. Koutsopoulos, Anil Mukundan, “A Dynamic Traffic Model System For ATMS/ATIS Operations”, IVHS Journal, 1994, Vol. 2(1), pp. 1-19

Peytchev E., “Integrative Framework for Urban Traffic Simulation and Prediction and Confidence Limit Analysis of the Predicted Traffic Flows“, 1998, Ph.D. thesis (submitted), Department of Computing, The Nottingham Trent University, Nottingham, England.