Design and Implementation of an Interface for the Integration of DynaMIT with the Traffic Management Center

by

Manish Mehta

B.E. (Hons.) University of Roorkee, India (1997)

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2001

Author . . . .
Department of Civil and Environmental Engineering
May 11,2001

Certified by . . . .
Moshe E. Ben-Akiva
Edmund K.Turner Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by . . . .
Haris N. Koutsopoulos
Operations Research Analyst, Volpe National Transportation System Center
Thesis Supervisor

Accepted by . . . .
Oral Buyukozturk
Chairman, Department Committee on Graduate Students

Design and Implementation of an Interface for the Integration of DynaMIT with the Traffic Management Center

Manish Mehta

Submitted to the Department of Civil and Environmental Engineering
on May 11, 2001
in partial fulfillment of the requirements for the degree of
Master of Science in Transportation

Abstract

DynaMIT (Dynamic Network Assignment for the Management of Information to Travelers) is a simulation based real-time system with traffic prediction and guidance generation capabilities. It is intended to operate within Traffic Management Centers (TMC) to provide decision support and pro-active route guidance. The objective of this research is to develop an interface for the Integration of DynaMIT with Traffic Management Centers. A successful integration of DynaMIT within the TMC requires a continuous and real-time exchange of data from multitude of legacy information sources. The proposed interface intends to synchronize the operation of the different information sources and the various components within DynaMIT through a common time server. The interface also converts the sensor and incident data (needed by DynaMIT for state estimation) into a common readable format and similarly converts the guidance provided by DynaMIT into an acceptable format. The system is designed over an extensible Common Object Request Broker Architecture (CORBA), which can help in the implementation of the system across multiple platforms and remote hosts.

The same interface is also used to integrate DynaMIT with MITSIMLab, a laboratory for the evaluation of Dynamic Traffic Management Systems. The Traffic Management Simulator (TMS), within MITSIMLab, emulates the TMC operations and hence can be a good test for the evaluation of the interface as well as DynaMIT, before applying it on real world. Suitable adapters have also been developed on the MITSIMLab side to transfer and receive required data. The interface is applied and tested on two case studies - the Central Artery/Tunnel Network in Boston and a test network in Irvine California. In the Irvine case study the interfaces between MITSIMLab and DynaMIT will replicate exactly the interfaces at the traffic control centers in the test site.

Thesis Supervisor: **Moshe E. Ben-Akiva**
Professor of Civil and Environmental Engineering
Massachusetts Institute of Technology

Thesis Supervisor: **Haris N. Koutsopoulos**
Operations Research Analyst
Volpe National Transportation System Center

Acknowledgments

I would like to express my deep gratitude for the constant guidance and support from my advisors, Professor Moshe Ben-Akiva and Dr. Haris Koutsopoulos, during the course of my graduate study. Their insights, suggestions and criticism contributed in large measure to the success of this research. Dr. Koutsopoulos has been a special mentor for me during my stay at MIT and has spent hours going over the fine details and discussing the bigger picture of the implemented research. His patience, encouragement and help, during the ups and downs, have been invaluable and would be difficult to express in words.

My thanks also goes to the Oak Ridge National Laboratories and the Federal Highway Administration for their financial support. My special thanks also to the MIT Center for Transportation Studies for providing me an opportunity to pursue the graduate program, which itself has been a special experience for me.

My gratitude also goes to Dr. Didier Burton and Bruno for acquainting me with the systems in the MIT ITS Lab at the beginning of my research.

My stay at MIT and at the ITS Lab has given me many moments and experiences that I will cherish for all times to come. My friends and colleagues from the ITS program have confirmed my belief that there is always something to learn from everybody. Among these friends are Rama, Tomer, Yosef, Leanne, Angus, Marge, Kunal, Zhili, Srini, Deepak, Dan and many others to whom I will always be thankful. We have had many a useful debates and discussions on academic and not-so-academic matters, which have all contributed to my education.

Finally, my special thanks to my parents for their love, understanding and encouragement. They have set high standards for me and inspired me to pursue my goals with high integrity and determination.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increasing congestion and traffic problems in the major cities around the world, has led to tremendous amount of research and development in the field of Intelligent Transportation Systems (ITS). ITS technologies are being developed to improve efficiency, productivity, and safety of existing transportation facilities, increase mobility, and alleviate the impact of transportation on the environment. A lot of effort has been concentrated on the development of various tools for the optimization of transportation systems. These systems, jointly referred to as Dynamic Traffic Management Systems (DTMSs), are based on a whole ensemble of diverse and complex software systems. In the future, DTMSs, with coordinated Advanced Traffic Management System (ATMS) and Advanced Traveler Information System (ATIS)operations, are supposed to be operated in the Traffic Management Centers (TMCs) to provide advanced traffic management with dynamic route guidance and traffic control. Dynamic Traffic Assignment (DTA) has been one of the most recent developments in this field, receiving extensive attention from transportation researchers worldwide. The real-time DTA system is envisioned as an ATMS and ATIS support system that will reside within a TMC. DTA systems use advanced travel behavior and traffic models to analyze multi-source data to estimate and predict traffic network states. Dynamic modeling and control of a multi-destination traffic network in real-time make these systems fairly complex. DTA systems, in order to operate successfully within TMCs, require to be integrated with many important technologies and software systems. A successful integration of a DTA system within the TMC requires a continuous and real-time exchange of data from multitude of legacy information sources and diverse software systems, posing some difficult problems for system integration, data communication, interfacing, and synchronization. Design of such an interface for the integration of a DTA system (DynaMIT) with the TMC is the focus of this work.

1.1 Dynamic Traffic Assignment Systems

Dynamic Network Assignment (DTA) systems are the latest generation of real-time support systems, designed to reside in Traffic Management Centers for the support of ATIS and ATMS operations. DTA systems aim at providing route guidance and traffic control based on predicted rather than historically measured traffic conditions. These systems, thus, are dynamic and envisioned to receive real-time and continuous data and disseminate proactive strategies for optimal traffic assignment. As a TMC support system, DTA will interact with other systems within a TMC, including surveillance, incident detection and management, and variable message sign (VMS) systems. DTA systems currently are receiving extensive interest from the transportation research communities and are under active development, validation and evaluation. DynaMIT (Dynamic Network Assignment for the Management of Information to travellers) is one such DTA system being developed at the Massachusetts Institute of Technology (Ben-Akiva et al. (1996a)).

1.1.1 DynaMIT

DynaMIT is a simulation based real-time Dynamic Traffic Assignment system with traffic prediction and guidance capabilities. It is designed to operate in real-time, and accept real-time surveillance data (including real-time sensor and incident data). Based on the surveillance data, DynaMIT estimates and predicts time-dependent Origin-Destination (OD) flows. The system, using the predicted OD flows predicts future traffic conditions, and interfaces with the traffic control system to generate route guidance consistent with the predicted traffic conditions. Bottom et al. (1998) discuss the importance and generation of consistent anticipatory route guidance. Figure 1-1 gives the overall structure and implementation framework of DynaMIT. DynaMIT is organized around two major simulation tools: *Demand Simulator* and the *Supply Simulator*.

The **demand simulator** itself has two broad functions: (i) it estimates and predicts the time-dependent Origin-Destination (OD) flows, and (ii) calculates driver's decisions in terms of departure time, mode-choice and route-choice. DynaMIT starts with a network representation and a historical estimate of the OD flows. But the actual OD-flows in the network may be very different actual travel demand in the network due to random fluctuations and shifts. DynaMIT can estimate OD flows by using a Kalman Filter approach (Kalidas et al. (1997) and Kalidas (1996)), or using a more generic GLS approach (derived from Cascetta et al. (1993)). The driver's decisions are implemented through various *behavioral models*, which calculate the probabilities of selecting the various choices the drivers have from their respective choice sets. Utilities for all the available alternatives are calculated based on the various attributes of available choices and their calibrated coefficients. The individual probabilities are calculated using MNL-type models (Ben-Akiva and Lerman (1985)).

The supply simulator is designed as a **Mesoscopic Traffic Simulator**. It is a time-based simulation model, designed to operate in real-time. Its accuracy (like other time based simulation models) is controlled by the time steps. The simulation of the traffic proceeds through two steps, (i) *Update Phase* during which the values of the various quantities of interest (e.g. speed, density, etc.) are updated, and (ii)*Advance Phase*, which is used to advance the traffic packets to their new positions. The simulation network is represented through nodes, links segments and lanes. The supply simulator gets from the demand simulator a list of vehicles. Speed calculations are made using the macroscopic speed-density relationships. The evolution and dissipation of queues is modeled through a deterministic queueing model, and is lane based.

DynaMIT operates in a rolling-horizon as described and below (and shown in

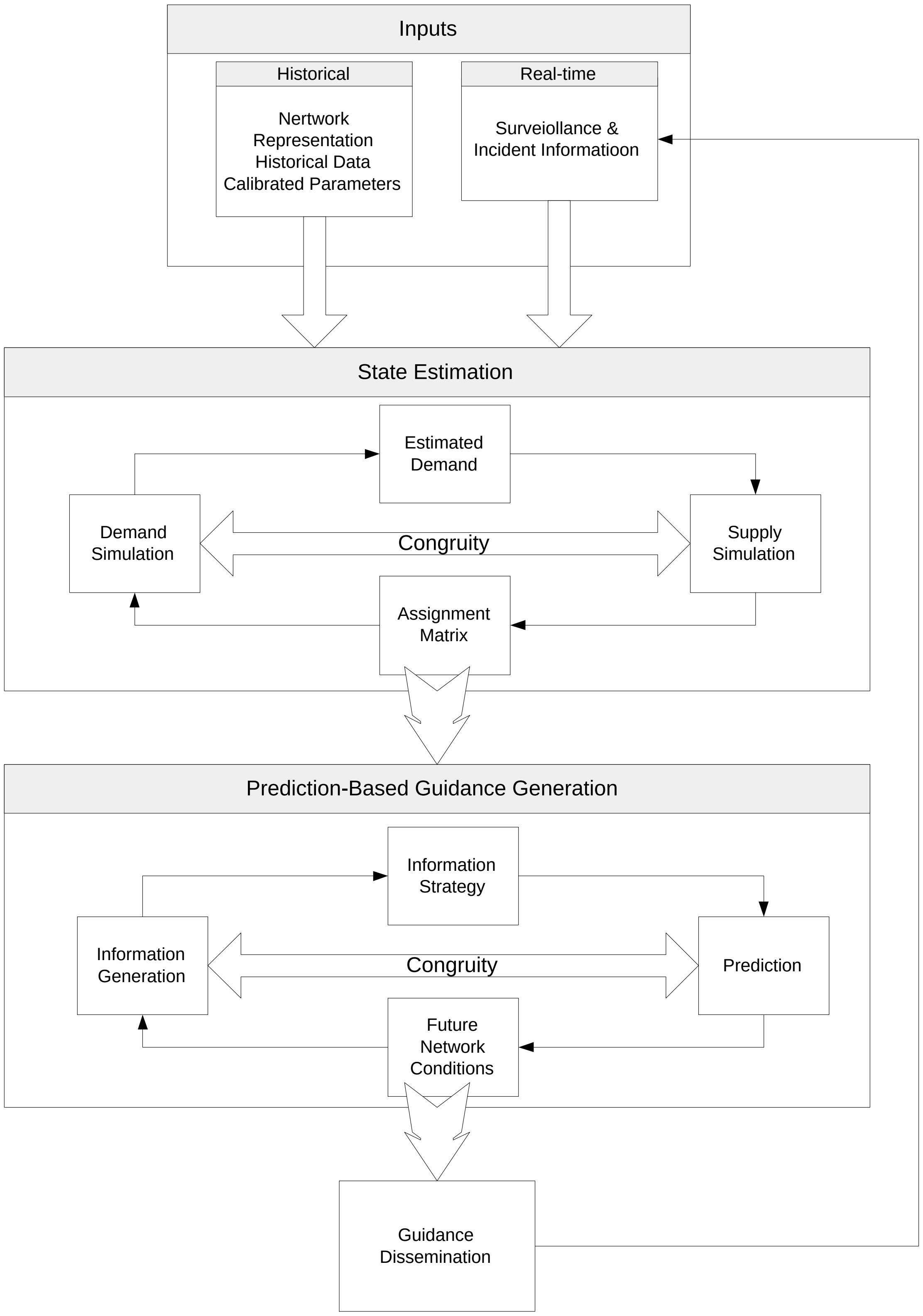


Figure 1-1: Overall System Structure of DynaMIT

Figure 1-2).

The figure shows two roll-overs for the rolling horizon implementation. When the current time is 8:00, DynaMIT does a state estimation past. In this example, the state estimation starts 15 minutes earlier and uses sensor counts during this time period. After the state estimation, DynaMIT would predict thirty minutes in future (in this example). The estimation and prediction computations, for example, may take upto seven minutes. In that case the guidance is disseminated at 8:07. The next set of fifteen minutes aggregated sensor counts would be available at 8:15. DynaMIT would thus roll-over at this time. It again estimates the state at 8:15 and predicts half an hour to the future and disseminates information at 8:22 (taking into account the computational delay). Estimations and predictions would proceed in a similar manner in subsequent intervals.

DynaMIT has two main functions: state estimation and prediction-based guidance generation.

State estimation provides the estimates of the current states in the form of network state (giving link or segment based flows, queues, speeds and densities) and OD flows. the state estimation is carried out through successive iterations between two simulation tools: *Demand Simulator* and the *Supply Simulator*. The OD estimation utilizes an assignment matrix, that maps OD flow to link counts.

The assignment matrix is obtained through the network state estimation modules which consist of the two subsystems - the *mesoscopic simulator* and the *Behavioral Module* that captures the driver behavior on the network, by predicting driver path choices and driver response to information and travel guidance. The mesoscopic traffic simulator (described above) simulates the traffic conditions in the network and generates the assignment matrix. The traffic simulator also measures link counts at the sensor locations which are then compared with the sensor counts obtained from the field. If the two match well we achieve congruency. Otherwise, the assignment matrix is again used for OD-estimation and the process is repeated till we achieve congruency.

Once the congruency is achieved (or the pre-specified maximum number of

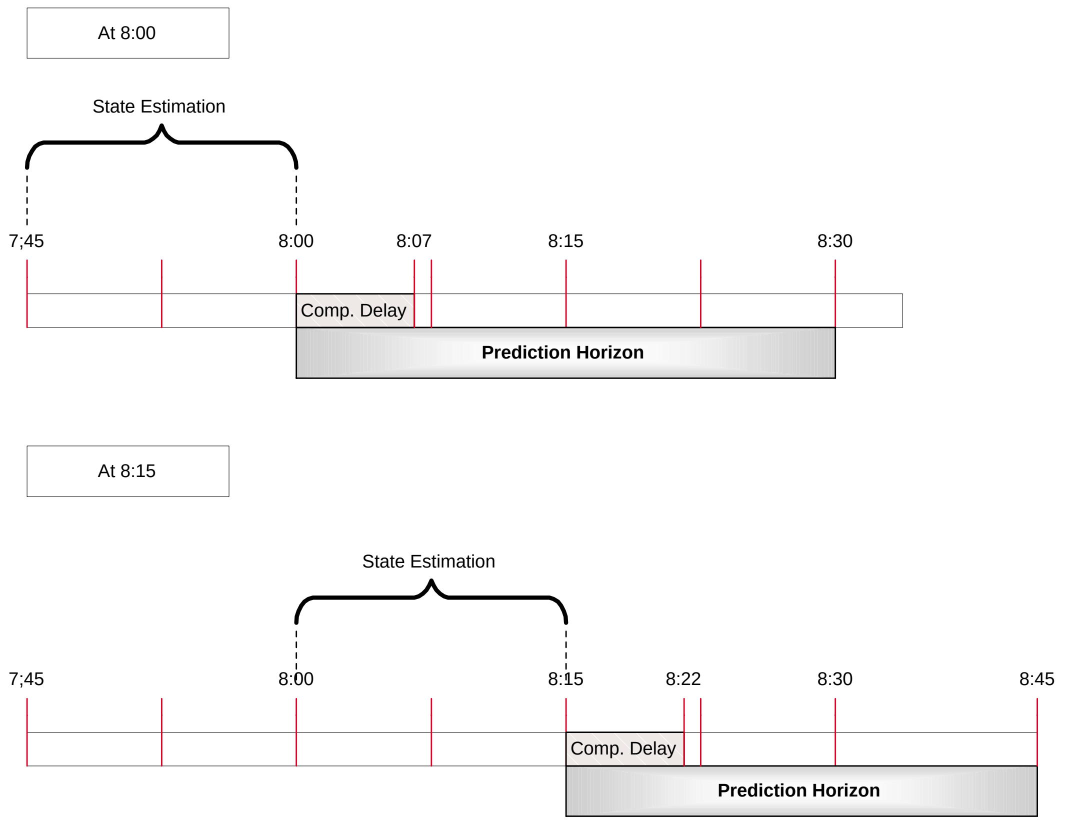


Figure 1-2: The Rolling Horizon Implementation of DynaMIT

iterations are made), the network conditions (which include the estimated OD-flows, link flows, speeds, queues and densities) are preserved as the estimated state.

Prediction-based Guidance Generation is solved as a fixed point problem. The guidance depends on the expected future traffic conditions, which depend on drivers choice, which in turn depends on the guidance provided. Ben-Akiva et al. (1996a) proposed an iterative approach to solve this problem. We begin with a base guidance case and simulate through the mesoscopic traffic simulator to generate a guidance (based on link travel times). We then do a route-choice based on these travel counts and again generate the travel times. The two travel times are compared till we reach consistency. The best guidance is dissipated to the users. The quality of prediction depends on the current state estimation, which thus needs to be updated at a regular basis. This formulates the rolling horizon approach of solution, as shown in Figure 1-2.

DynaMIT, when installed at the traffic control center, must interact in real-time with the various elements and systems in the TMC. The integration of DynaMIT with TMC depends on the system as well as the communication architecture. At the modeling level, DynaMIT needs to have sufficient flexibility to take continuous real-time inputs and disseminate continuous or discrete set of guidance as required by its user. The TMC should also have an open architecture to support integration with the DTA system. We would next look at the state of art practices at the traffic management centers and then formulate a set of requirements for interfacing DynaMIT with the TMCs.

1.2 Irvine Traffic Management Center Testbed-An Overview

DynaMIT is envisioned to operate within a Traffic Management Center as a support for ATMS/ATIS operations. A complete integration of DynaMIT within TMC depends on the system and communication architecture of the TMC. A TMC itself consists of multiple ITS systems, which include the systems that provide the basic functionality (e.g. controlling signals, collecting data, etc.) and some others that are interfaced to enhance the operational capabilities of the TMC (e.g. the adaptive control systems, route-guidance systems, etc.).

Different TMCs have widely varying architectures and number and types of individual subsystems. The interfacing requirements and the data acquisition and processing capabilities of individual TMCs also vary a lot. This makes the development of a generic interface that would support all the different TMC types, a very difficult problem. All interfaces require some kind of customization at the TMC level. Customization at the TMC-level, would standardize all the data and information that comes out the TMC and this would help in increasing the flexibility of interfacing with multiple types of different support systems. Even though an adaptor specific to the TMC is required for complete application, a good design should be able to produce a fairly generic interface that can be adopted to the individual

TMCs with minimal modifications.

In order to develop a complete set of interfaces for the individual subsystems, we need to look more specifically at the TMC in hand. In this section we look at the functionality and operations of a specific TMC. Many features and characteristics are common to most TMCs. However as with every TMC, certain features and characteristics are unique.

DynaMIT is intended to be field tested at a testbed developed at University of California, Irvine, which receives data directly from the City of Irvine Traffic Management Center, the Caltrans District 12 TMC and the City of Anaheim Traffic Management Center. We will use this test-bed as the focus of our description and describe the different subsystems we need to interface with in order to integrate the DynaMIT operations in real-time. Many of the characteristics discussed here are generic to most of the TMCs and hence this discussion can be extended to represent general characteristics of subsystems in a Traffic Management Centers.

We next describe the different subsystems which need to be interfaced and are of specific interest to this study:

1.2.1 Surveillance Systems

The TMC testbed has developed an extensive surveillance support system to support field testing. Traffic information is collected using a variety of technologies including loop detectors (sensors), video surveillance and closed-circuit television (CCTV), infrared sensors, vehicle probes, and mobile videos. A series of portable Video Image Processing (VIP) systems and supporting wireless communication infrastructure have been deployed as a part of new technology initiative to capture real-time surveillance. These systems work on Spread Spectrum Radio (SSR) technology and have been deployed under the federal Mobile Video Surveillance / Communication (MVSC) field operation test project. The video image processing surveillance function processes video images received from two fixed-field-of-view (FFOV) cameras to generate the pertinent data.

Each VIP sensor node can send in the following data: (i) date and time, (ii) detector ID, (iii) vehicle loop count, (iv) lane count, (v) speed, (vi) occupancy, (vii) density, and (viii) queueing data. The information is captured by the sensors, processed locally, and aggregated for transmission to the TMC. Regular scanning frequencies are 1/240s and broadcasting frequencies are about 30 seconds.

1.2.2 Control and Guidance Systems

TMCs can have a centralized, distributed, or a hierarchical control mechanism. In a centralized environment, a central facility collects traffic status data and makes traffic control decisions. In a distributed environment, control is performed locally, generally at the intersection level. A hierarchical control configuration is a hybrid between central and distributed control. The Irvine TMC control and guidance systems are designed to be centralized, with a variety of control mechanisms, including Variable Message Signs (VMS), ramp metering, real-time traffic adaptive signal control, highway advisory radio, lane usage control, etc.

In addition, Anaheim has been developing a Motorist Information System (MIS). There are four main component of this system (FHWA (1993)): information kiosks, highway advisory radio, highway advisory telephone, and a CATV feed. These will be strategically placed and play recorded messages to provide guidance. These messages will be updated at a periodic rate to maintain latest information.

1.2.3 Incident Management Systems

The TMC test-bed incorporates automatic incident detection by autonomously spotting conditions of non-recurring congestion. The incident is usually detected through a video input or specific information form the mobile sources. TMC requires some specific information processing capabilities like, integrated data management, real-time traffic simulation model execution, image processing for area-wide surveillance and incident detection, to provide incident management (FHWA (1993)).

Expert systems generally guide through the steps necessary to quickly and efficiently respond to an incident. The control options include sending advisory messages to VMS, route diversion alternatives via an interface to ATIS or radio broadcasts.

1.2.4 Communication Systems

The testbed system is built upon a wide-area communications network backbone linking the cities of Anaheim and Irvine Traffic Management Centers to the California Department of Transportation's District 12 TMC and with the ATMS research laboratories at the University of California, Irvine, Institute of Transportation Studies. The communication network is based on an ATM infrastructure, designed to be compatible with the existing Teleos ISDN PRI network established by the Caltrans Wide Area Network (WAN). The ATM infrastructure is linked with the Caltrans District 12 TMC and the City of Irvine ITRAC via an OC3 155 Mbps SONET fiber optics network, and with the City of Anaheim TMC via ATM T-1. The system also has a MPEG 1 video transmission system, allowing for selection and display of freeway video surveillance cameras within District 12.

The TMCs are equipped with UNIX workstations and numerous PC-compatible machines. The UCI laboratories have SGI IRIX 6.5, SunOS 5.6 and HP UX 11, besides PC-based computers. The fiber optic ATM links enable high speed exchange of traffic data and video images - both real-time and historical. Several communications interfaces or interties have been developed to perform these exchanges.

The testbed and TMC is distributed using CORBA (Common Object Request Broker Architecture) to provide to external agents the following services:

(a) real-time data in the form of LDS (raw loop detector data), VDS (processed loop controller data), RMS (processed ramp metering control data, and CMS (status and message on each Changeable Message Sign).
(b) CCTV switching
(c) ramp meter control

(d) Historical Data, and

(e) One switched video channel.

The CORBA services can also be used to develop own interfaces for communication with the rest of the system and for research and prototype deployment in the testbed.

1.3 Motivation and Research Objective

The real-time DTA systems are envisioned to be a part of the TMCs, and thus need to be integrated with the TMCs to realize their design objective. These systems are TMC-independent software systems intended to run in all configurations of TMCs. However the TMC configuration and communication architecture itself is far from standard and shows marked variations in different places. TMC must though comply with an adequate generic system architecture that allows an easy integration/interface of both internal systems and external systems. Efforts so far have been concentrated in developing stand-alone DTA systems. **?**)Fernandez:2000)has proposed an open standard architecture that could be possibly used to integrate Dynamic Traffic Management Systems (DTMSs)within TMCs. However, DTA systems work independently and they may not comply with this architecture to enable successful integration. An interface therefore needs to be designed that will act as an intermediate layer enabling the compliance of individual DTA systems with the overall system architecture within the Traffic Management Centers.

As a preclude to full-scale field application of the DTA systems, a strong need is felt to evaluate these systems off-line, using a ground-truth simulator[1]. Such a laboratory-evaluation system can be a very handy research tool and would also help in assessing the performance of the DTA systems before they are introduced on-line. The simulator can be calibrated with field data and hence replicate the ”real world”

[1]A traffic simulator like MIT’s MITSIM or Quadstone’s Paramics.

data that DTA systems are ultimately envisioned to work with. Such an off-line evaluation can serve the following important purposes before the DTA systems are finally integrated with the TMCs:

- Assess the quality of estimation and prediction capabilities of the DTA systems, using real operational data, and the performance of various models used in the system against real data.
- Assess the benefits and applicability of the outputs generated from these systems for the actual on-line TMC operations.
- Assess the efficiency and real-time performance of the system and make the required performance improvements.
- Understand and address any problems that might arise in the system and communication architecture of software.

In this research, a system interface is designed that allows integration of DynaMIT with the Traffic Management Centers. The architecture is *distributed*, using the Common Object Request Broker Architecture (CORBA). The proposed interface intends to synchronize the operation of the different information sources and the various components within DynaMIT through a common time server. The interface also converts the sensor and incident data (needed by DynaMIT for state estimation) into a common readable format and similarly converts the guidance provided by DynaMIT into an acceptable generic format.

The same interface is also used to integrate DynaMIT with MITSIMLab. Ben-Akiva et al. (1996b) proposed the applicability of MITSIMLab (Microscopic Traffic Simulation Laboratory) as a suitable simulation laboratory for the evaluation of Dynamic Traffic Management Systems. A suitable adapter is written on the MITSIMLab side to establish the communication between MITSIMLab and the rest of the system.

1.4 Literature review

As seen in section 1.3, we have two broadly defined objectives for this study - (i) interfacing of a Dynamic Traffic Assignment system (DynaMIT) with MITSIMLab, (ii) interfacing of DynaMIT with the Traffic Management Centers. In reviewing the previous literature and research, we would thus investigate the following areas of work in order to learn from the experience of the other researchers and their implemented frameworks:

- Dynamic Traffic Assignment and Dynamic Traffic Management systems,
- Tools and Methods used for investigating dynamic traffic management systems,
- Previous work in interfacing different dynamic traffic control and assignment systems.
- Proposed designs and architectures for the integration of DTMS with TMCs.

1.4.1 Dynamic Traffic Management Systems

Dynamic traffic management systems include a whole assembly of different integrated and stand-alone dynamic traffic control and guidance systems. Development of such dynamic route guidance systems and dynamic traffic control strategies has been receiving increasing attention over the last decade.

Dynamic Traffic Assignment (DTA) systems have been envisioned to be at the core of many of these different subsystems and processes (FHWA (2000)). The success of many of the DTMS and other ITS subsystems, like the Advanced Traveler Information System (ATIS), Advanced Traffic Management Systems (ATMS), Advanced Public Transportation Systems (APTS), Commercial Vehicle Operations (CVO), and Emergency Management Systems (EMS), is dependent on the availability of timely and accurate estimates of the prevailing and emerging traffic conditions.

DTA systems can provide this capability of traffic estimation and prediction and are anticipated to be ATMS and ATIS support systems. Dynamic traffic assignment has been a relatively recent development and has received extensive attention from the transportation research communities (DYNA (1992-1995);FHWA

(1995), Mahmassani et al. (1994), MIT (1996)). Gartner and Stamatiadis (1997) and Chen and Hsueh (1997) presented the framework for integrating dynamic traffic assignment with real time traffic adaptive control.

DYNA (DYNA (1992-1995)) was developed under the European DRIVE II program. It has been designed as a real-time traffic prediction system for an inter-urban motor-way network. The system collects network-wide data through on-line roadside measurement stations, and uses prediction-error feedback to update the traffic state. Figure 1-3 provides the operational framework of the system. The system regularly communicates with the operators at a number of levels, and continuously forecasts in a rolling horizon implementation. DYNA uses both traffic control and travel information strategies, and is based on variety of models which cab be sub-grouped into two categories. *Behavioural Traffic Model (B.T.M.)* concentrates on OD flow representation and sub-models that emulate human behavior and traffic assignment. The second category, *Statistical Traffic Model (S.T.M.)* focusses on the link density representation of the network traffic. DYNA has been extended by the European Union's Fourth Framework project DACCORD (Hague Group (1997)), to include the implementation and demonstration of a DTMS. DACCORD includes the EUROCOR (Middelham et al. (1994))traffic control project, and the GERDIEN project for systems architecture and traffic prediction.

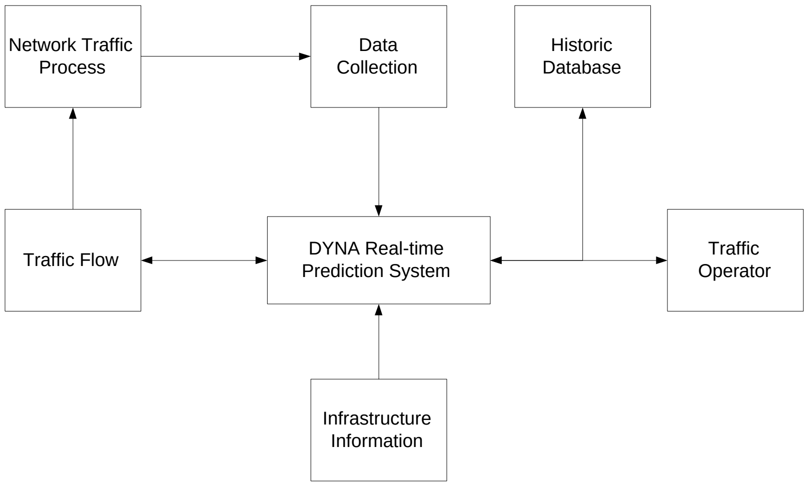


Figure 1-3: Operational Framework of DYNA

The United States Federal Highway Administration (FHWA) initiated a DTA project in 1994, to meet the rapidly emerging needs for a traffic estimation and prediction system. Two separate research project began in 1995 under this initiative under the management of Oak Ridge National Laboratory(ORNL).

DynaMIT (Dynamic Traffic Assignment for the Management of Information to Travellers) was initiated at Massachusetts of Institute of Technology (MIT). DynaMIT (MIT, 1996; Ben-Akiva et al., 1997) is a real-time system designed to reside in TMCs for the support of ATIS operations. The system architecture and operational features of DynaMIT have been discussed in section 1.1.1.

The second system **DYNASMART-X** (DYnamic Network Assignment Simulation Model for Advanced Road Telematics) has been developed by University of Texas at Austin (UTX). DYNASMART-X (Mahmassani et al. (1994))is a real-time traffic estimation and prediction system for support of ATMS and ATIS operations. It uses network algorithms and models for trip-maker behavior in response to information in an assignment-simulation framework. DYNASMART-X provides control actions, in the form of information to users about traffic conditions and routes to follow as well as signal control strategies. DYNASMART-X uses microsimulation for individual user decisions, and mesosimulation for traffic flow. Similar to DynaMIT it tries to achieve consistency between predicted network conditions, supplied information and user decisions. It is implemented in a rolling horizon framework and has features to recognizes multiple user classes. Origin-Destination estimation and prediction is an essential part for the simulation-assignment that is being implemented by DYNASMART-X. It adopts a distributed software implementation using CORBA.

Variety of route guidance systems have been field-tested in the past or are currently under the testing stage. ADVANCE is a real-time in-vehicle route-guidance system that has been tested in suburban area of Chicago.

RT-TRACS (Real-Time Traffic Adaptive Signal Control Strategy) is a program sponsored by FHWA to improve traffic control by performing signal optimization in real-time. RT-TRACS (Tarnoff and Gartner (1993))is intended to be a multi-level system that consists of a number of real-time control prototypes that each function optimally under different traffic and geometric conditions and are accordingly invoked and switched from one to another dynamically to produce best possible results. Five prototypes have been developed and are being evaluated for use in the RT-TRACS program. Three of these prototypes, RHODES from University of Arizona, OPAC from PB Farradyne, and RTACL from the University of Pittsburgh are at the advance state of development.

1.4.2 Evaluation of Dynamic Traffic Management Systems

Traffic management systems can be evaluated either through field tests or through computer-based simulations.

Field Testing is the most direct and definite way of testing traffic management systems.

A systematic study (McNally (1999)) on the performance evaluation of Advanced Traffic Control Systems was conducted from 1994 to 1998 in the city of Anaheim, California. Adaptive signal control technologies were evaluated, including SCOOT (2nd generation model) and a 1.5 generation control (1.5GC) approach, and a video traffic detection system (VTDS). The project, in addition to evaluating the performance of the traffic control technologies, examined institutional issues, as well.

A number of other field studies have been conducted under the California PATH (Partners for Advanced Transit and Highways) ATMIS (Advanced Traveler Management and Information Systems) research. These include the evaluation of Phase I of a PeMS (performance Evaluation Measurement System), called Transacct. The system was located at the University of California, Berkeley, campus, and was used for studies like, estimation of reliable real time speeds, congestion measures, etc. Another study was Dynamic Origin/Destination estimation using true section densities (Sun (1999)). Yet another study has been the Freeway service Patrol project (Petty et al. (1996)), which gets real-time information on the vehicle trip time and distance traveled, using patrol service as probe vehicles.

An example for evaluation of traffic control systems is the study of INFORM (Information for Motorists), a traffic management system designed for a 40-mile long highway corridor in Long Island, New York (Smith and Perez (1992)). In the evaluation, motorists response to and effectiveness of ramp metering and variable message sign strategies were evaluated.

A number of projects have also been field tested to evaluate the Advanced Traveler Information Systems. TravInfo (Miller (1998)) was a field operational test in the San Francisco Bay Area, sponsored by FHWA. The evaluation studied the institutional, technological, and traveler response elements of TravInfo. The technology element focussed on the operational effectiveness of TravInfo's Traveler Information Center and a study of Information Service Providers reaction to and use of TravInfo information. Traveler response investigated public access to and use of different types of information by TravInfo and changes in individual traveler behavior. The ADVANCE project (Bowcott (1993);Saricks et al. (1997)) has another example of field-evaluation of ATIS. A field test for evaluating ADVANCE dynamic route guidance system (Schofer et al. (1996)) was conducted in Chicago's suburban areas. Vehicles equipped with MNA (Mobile Navigation Assistant) acted as probes, sending real time travel information to a traffic management center, which was in turn relayed the information to equipped vehicles to aid in dynamic route planning.

Computer-simulation based evaluations provide a very good alternative to field tests because they are much less expensive and allow greater flexibility in testing lot of different strategies, in a controlled environment.

There have been extensive studies of the performance of traffic control and route guidance systems using simulation. For example, simulation has been used to evaluate the design of ramp metering strategies (Payne (1973)), urban traffic signal controls (Sibley (1985); Yauch et al. (1988)), route diversion (Stephanedes et al. (1989); Barcelo and Ferrer (1995)) and mainline metering (Haboian (1995)). Yang (1997) used a microsimulator to evaluate a dynamic traffic management system. Hasan (1999) used a microsimulator (MITSIM) to evaluate ramp control algorithms.

Studies integrated traffic management systems, using simulation, have been relatively rare. CORSIM (FHWA (1996)) is being used to study the prototypes of RT-TRACS. Reiss and Gartner (1991) also conducted some simulation based studies on an earlier version of INFORMS (IMIS).

Ben-Akiva et al. (1995) identified the different requirements for evaluating Dynamic Traffic Management Systems using simulation and provided an evaluation framework for DTMSs (shown in Figure 1-4). Based on the goals and objectives of the management system, a control strategy is formulated and fed into the simulation laboratory, which then operates on a set of scenarios. Based on the results of the pre-set performance measures, modifications are done in the control strategy till we observe the desired set of results. Because it is capable of realistically simulating the traffic flow in the network and its dynamic interrelationship with the control and route guidance system under consideration (which many of the other simulators lack), MITSIMLab (Yang (1997)) is an effective simulation laboratory for evaluating DTMS (Ben-Akiva et al. (1996b)).

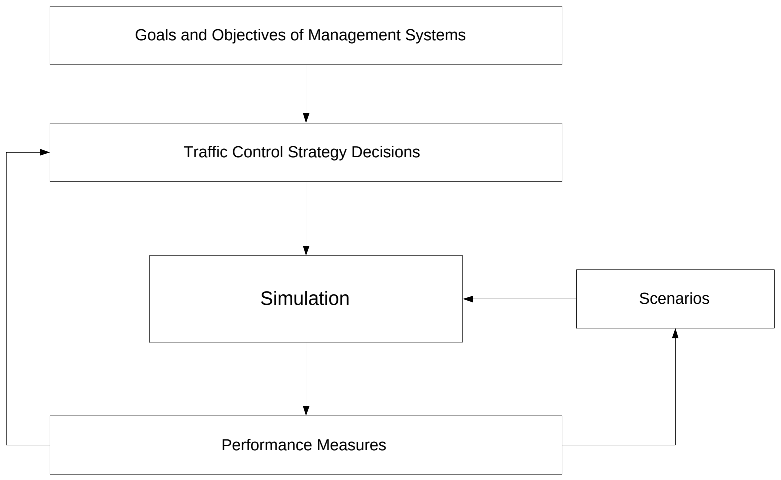

Goals and Objectives of Management Systems
Traffic Control Strategy Decisions
Simulation
Scenarios
Performance Measures

Figure 1-4: Evaluation Framework for DTMS

1.4.3 Interfacing of ATMS/ATIS subsystems

Literature on the interfaces of dynamic traffic assignment systems with other TMC-based subsystems has very limited. But there are many examples of other guidance and traffic control systems that have been interfaced in the past.

The ADVANCE project, during its testing in the Chicago suburban area, was interfaced with different internal and external subsystems which operated in sync.

ADVANCE consisted of five subsystems, namely (i) the Traffic Information Center (TIC) which contained the central facility, the operator interface, etc.; (ii) the Traffic Related Functions (TRF) which includes the traffic algorithms; (iii) the Communications Subsystem (COM) which provides message carrying capability between the TIC and the vehicles in the field; (iv)the Mobile Navigation Assistant (MNA) which contains in-vehicle route planning and display capabilities; and (v) the Help Center (HC) which provides for roadside vehicle assistance requests and driver queries.

The data flow between the MNA and TIC was through the COM interfaces on the TIC and MNA sides. the TRF internally contained multiple subsystems - each specific to an algorithmic role, which were interfaced with the TIC which in turn communicated with the MNA in each vehicle. Figure 1-5 shows overall interfacing framework adopted by ADVANCE.

Federal Highway Administrations's, Turner-Fairbank Highway Research Center [2] has taken many initiatives to interface various simulation and traffic control tools. One of the interfacing tool is TSIS (Traffic Software integrated System) which provides state-of-the-art environment for managing, controlling, and coordinating the application of various traffic engineering analysis and simulation tools. It provides a framework for exchanging data between different traffic engineering tools and standard interfaces for communicating with real-time traffic control systems and hardware devices.

The FHWA's Traffic Research Laboratory (TReL)[3] has also integrated a hardware-in-the-loop simulation with a video camera sensor collection tool and a real-time traffic adaptive signal control algorithm. It has also developed a communication interface between the signal control algorithm, the sensors, and the CORSIM simulation engine. Elaborate interfacing for CORSIM is being planned in order to allow it to operate as a simulation-based evaluation tool for traffic management systems.

The DYNA project also required significant interfacing before it was applied to

[2]http://www.tfhrc.gov
[3]http://www.its.dot.gov

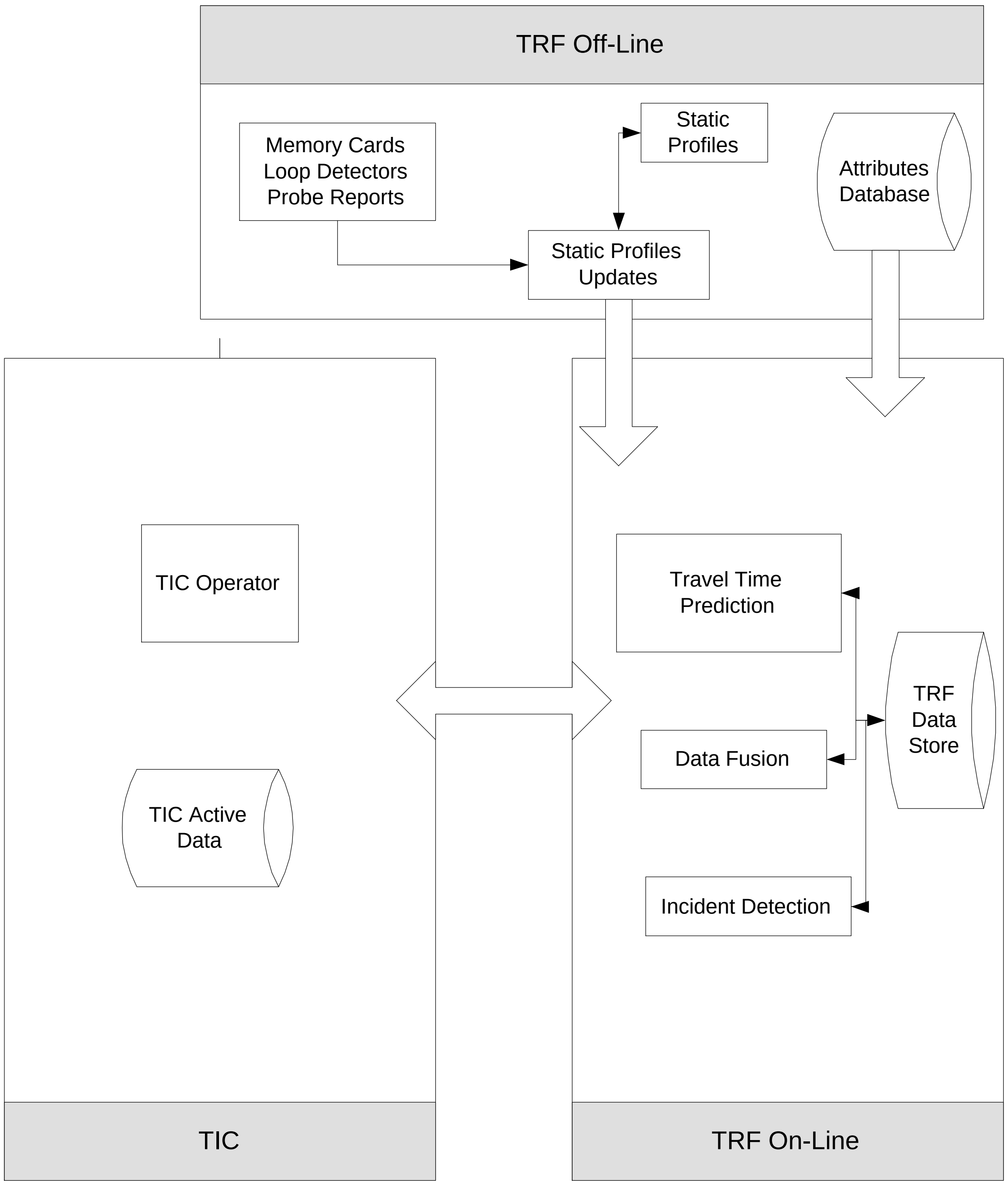


Figure 1-5: Interfacing Framework for ADVANCE

the Rotterdam network in Holland. The framework for surveillance interfacing is shown in Figure 1-6. The measurements are obtained from the surveillance station which are average one minute flows, one minute velocities and a congestion indicator that flags a time sequence for which link velocity falls below a critical value. The measurements then pass through several stages of processing, starting with detection at the road surface, local processing to aggregate over 1 minute interval, transmission to intermediate portals where the time stamp is added, and finally transmission to the traffic center. The data arriving at the TMC is then added to a file, which is used by the system subsequently.

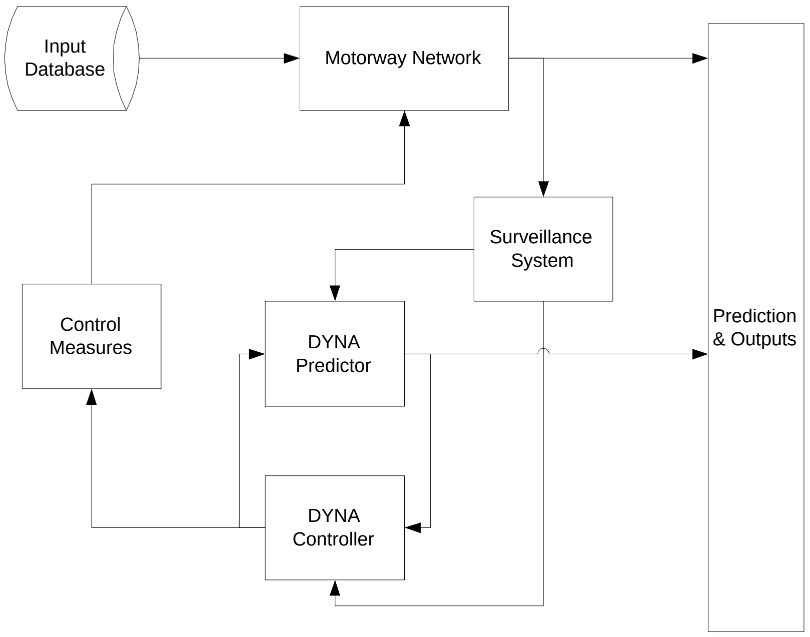


Figure 1-6: Interfacing Framework for DYNA

1.4.4 Architecture for Integration with TMC subsystems

The real-time traffic estimation and prediction capabilities can be very useful to many subsystems that operate within and outside the Traffic Management Centers. The Travel Management Team at Office of Operations Research and Development (FHWA) identified almost fifty independent ITS (Intelligent Transportation System) packages and subsystems that would support TMC operations and which can use the data provided by DTA systems.

The U.S. Department of Transportation (DOT) has developed a National ITS Architecture[4] to promote development of Intelligent Transportation Systems that are uniform and inter-operable. The architecture provides a framework in which the DTA systems are intended to operate and interface. Figure 1-7 provides an overview of the envisioned framework of operation of DTA systems and provides a wholistic view for integration and inetrfacing requirements of DTA systems.

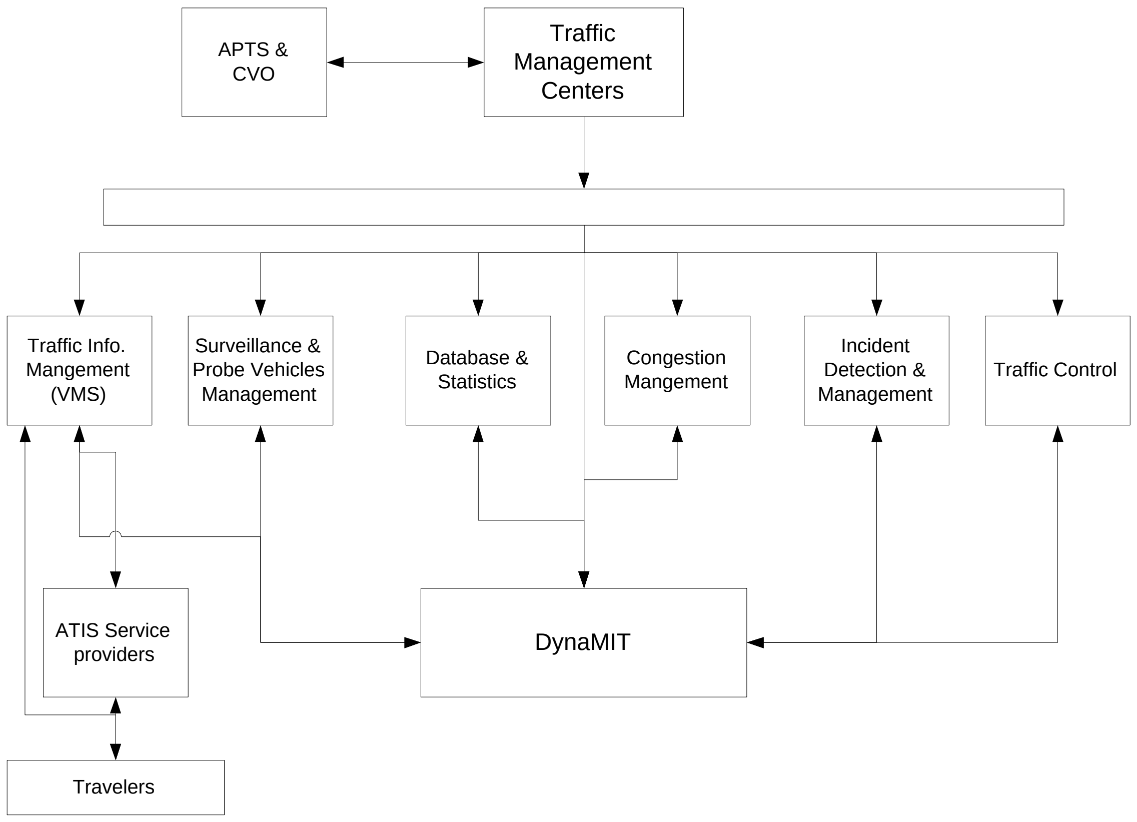

APTS & CVO
Traffic Management Centers
Traffic Info. Mangement (VMS)
Surveillance & Probe Vehicles Management
Database & Statistics
Congestion Mangement
Incident Detection & Management
Traffic Control
ATIS Service providers
DynaMIT
Travelers

Figure 1-7: DynaMIT's role with ATMS/ATIS subsystems in the Traffic Management Centers

There is currently no working example of an integrated TMC-DTA system. Ruiz (2000) provided an architecture for integration of Dynamic Traffic Management Systems. The architecture was based on a generic distribution mechanism using CORBA (Common Object Request Broker Architecture), and based on an abstract factory pattern that permits anonymous use of DTMSs within TMCs. The

[4] Available at http://www.itsonline.com/archls.html

architecture sought to implement a Publisher/Subscriber pattern to provide parallel programming on top of CORBA's synchronous communication paradigm.

1.5 Thesis Outline

Thesis is organized as follows: Chapter 2 specifies the off-line evaluation requirements for DynaMIT and the implementation requirements for the interface. It also provides the requirements for on-line integration of DynaMIT with the Irvine Traffic Management Center.

Chapter 3 discusses the overall system architecture for interfacing DynaMIT with MITSIMLab for off-line evaluation. It also discuss the design architecture for integration with the Traffic Management Center at Irvine.

The first part of Chapter 4 presents the implementation framework for combined MITSIMLab-DynaMIT operations. The second part discusses the systematic way in which the interface is implemented for DynaMIT-TMC operations. It will also discuss the interfaces's compatibility with the Interface Control Document (ICD) proposed by FHWA.

Chapter 5 includes two case studies that demonstrate the functional viability of the interface. The first case study is based on the Central Artery Network in Boston. The case study presents results that demonstrate the usefulness of predictive route guidance. It also demonstrates the flexibility of the interfaced MITSIMLab-DynaMIT tool in studying various scenarios useful for evaluating DynaMIT. The second case study focusses on the Irvine network. The case study illustrates the effective operational viability of the interface in supporting large networks. It also constitutes a big step forward in terms of the ultimate goal of testing DynaMIT with the real data from Irvine test network. The final chapter concludes the work presented in this study and gives directions for future work in this field.

Chapter 2

Integration and Evaluation Requirements

We divide this chapter into two parts. In the first part we describe the design requirements of the Interface so that DynaMIT can work in a closed-loop format with MITSIMLab - our ground truth simulator. As mentioned in the previous chapter, such a closed-loop system can be an excellent research tool and a good precursor to the actual on-line integration of DynaMIT with the TMCs. The second part describes the requirements for the integration of DynaMIT with the Traffic Management Centers. Special attention is paid to the Traffic Management Center at Orange County, California, where DynaMIT is intended to be first integrated for field-testing.

2.1 Off-line Evaluation Requirements

Computer Simulation systems provide an effective tool for testing alternate system designs before conducting expensive on-field operational tests. Various individual control elements of traffic management systems have been studied using simulation before. A similar simulation-based approach for evaluating Dynamic Traffic Management Systems has been proposed by Ben-Akiva et al. (1994a). In the following section we investigate the system and modeling requirements to pursue this line of approach.

2.1.1 Methodology

Ben-Akiva et al. (1996b) proposed an overall evaluation and design refinement framework for evaluating Dynamic Traffic Management Systems (see Figure 1-4). A simulation laboratory is used to compare the candidate designs against a base case. The inputs to the simulation laboratory are (i) the elements (e.g. surveillance and control devices) and the logic that determines how the system under investigation operates; and (ii) the scenarios against which the system will be tested. A set of performance measures are computed, and then the results are analyzed to suggest any improvements.

Yang (1997) demonstrated the use of MITSIMLab as a possible simulation laboratory for evaluating Dynamic Traffic Management Systems. A more detailed description of MITSIMLab and its subsystems is provided in section 2.1.2 and 2.2.

MITSIMLab has the following essential characteristics that are required for evaluating DynaMIT:

- flexibility to simulate a wide array of traffic management system designs, networks and control strategies.
- representation of the surveillance system, including real-time sensor counts and incident detection.
- modeling the response of drivers to real-time traffic information.
- incorporation of guidance (generated by DynaMIT) through a wide array of information dissemination strategies.
- open architecture, which facilitates interfacing and integration.

The effectiveness of the guidance generated by DynaMIT can be tested in MITSIMLab against the base case of no-guidance. The sensitivity of results (benefits) to a variety of design characteristics associated with DynaMIT can also be tested.

2.1.2 MITSIMLab - A Traffic Simulation Laboratory

MITSIMLab (Yang and Koutsopoulos (1996); Koutsopoulos et al. (1994) is a traffic simulation laboratory, consisting of a traffic flow simulator (MITSIM) and a traffic management simulator (TMS). To capture the real-time control and routing strategies and simulate the surveillance system, the traffic and network elements are represented in a fairly detailed level. Such detailed modeling is able to capture the stochastic nature of traffic flow and drivers response to route guidance - a feature that is very important for evaluating dynamic traffic management systems.

The Microscopic Traffic Simulator (MITSIM) models the traffic in the network. The road network with the traffic surveillance and control devices are represented in detail to emulate the "real world" elements. The Network consists of nodes, links, segments and lanes. The Surveillance system consists of various forms of detectors. Traffic signal devices, message signs, lane use signs, and special facilities (like Toll Booths, etc.) are also represented. Vehicle trips are generated based on pre-determined origin-destination (OD) tables. MITSIM uses lane-changing and car-following models to move vehicles. Behavioral parameters are assigned randomly based on pre-determined and calibrated user characteristics. Vehicles are moved at a pre-fixed step size according to various constraints.

The Traffic Management Simulator (TMS) has the control and route guidance system. It receives input from the MITSIM surveillance system in the form of real-time traffic measurements. Based on the surveillance data, TMS generates control and route-guidance and updates the traffic signals and signs in the network. TMS can model both pre-timed and reactive systems(adaptive systems where control and route-guidance is provided based on the prevailing traffic conditions). Yang (1997) provides detailed description on design and models of TMS.

There is a very strong coupling between the traffic flow and control strategies. Traffic control and routing strategies effect the traffic flow and control strategies themselves are triggered by the traffic flow as measured by the surveillance system. As a result, TMS and MITSIMLAB can simulate a wide range of traffic control and advisory services, including:

- *Intersection controls* such as traffic signals, yield and stop signs.
- *Mainline Controls* such as lane use signs, variable speed limit signs, variable message signs, etc.
- *Ramp Controls* such as ramp metering and speed limit signs.

These signals are controlled by traffic signal controllers, which can be of four types - static, pre-timed, traffic adaptive and metering controllers. Each controller is characterized by a set of data items (such as *signal type, controller type, number of egresses, IDs, etc.*), and as the simulation progresses, the controller can switch from one type to another.

2.2 Interfacing Requirements for MITSIMLab subsystems

The TMS within MITSIMLab, emulates the operations in the Traffic Management Center. We discuss here the interfacing requirements for each of the subsystems in TMS in order to integrate DynaMIT with MITSIMLab. This would be in many ways very similar to the interfacing requirements for integration of DynaMIT with TMCs. The integration of DynaMIT with the TMS would require interfacing between the following components/ subsystems:

- Surveillance System.
- Incident Management System.
- Time Server.
- Guidance and Control System.
- Software and Communication System.

2.2.1 Time Server

Both MITSIMLab and DynaMIT have their individual time clock components. The time clock utility in MITSIMLab is called by multiple objects throughout the software, and plays an important role in determining the step sizes, vehicle movements and control regulation. The simulation speed is governed by the computational time of the algorithms. It can practically proceed as fast as the computational times allow it to. This time may be faster than or slower than the real-time. The speed, however, can be slowed down manually, by changing a parameter that would decrease the advance frequency of MITSIMLab's internal clock. Similarly a different clock component is extensively used within DynaMIT by its supply and demand simulators.

The rolling horizon implementation of MITSIMLab-DynaMIT closed loop would require the TMS to make multiple and discrete calls to DynaMIT for providing prediction-based guidance. Similarly DynaMIT gets the sensor counts and incident data in discrete steps. The exchange of data thus occurs with an accompanying time stamp, which signifies the time interval the data belongs to. This calls for a clear mechanism for communication of a common time between the two systems. Also, MITSIMLab cannot be blocked while DynaMIT is generating guidance. The communication has to be asynchronous. Therefore, the time communication has to be uni-directional rather than bi-directional. Since MITSIMLab simulates the reality in our case, it should act as the time server for closed system.

An interface, thus, is required to be able to provide the timing data from MITSIMLab to DynaMIT. MITSIM and TMS operate with a common clock, and this time message requires to be communicated to DynaMIT on a regular basis. That way DynaMIT always knows the current time in MITSIMLab and can associate appropriate time stamps with the surveillance, incident and guidance data.

2.2.2 Surveillance System

MITSIMLab can simulate the following type of sensors, Yang (1997):

- *Traffic sensors*, collecting traffic counts, occupancy and speed data at fixed points in the network.

- *Vehicle sensors*, which extract information on individual vehicles.
- *Point to point data sensors*, which extract information such as vehicle travel time from one point in the network to another.
- *Area wide sensors* such as radar detectors and video cameras.

In addition, external agency reports - e.g. a vehicle passing an incident using a cellular phone to report it - can also be simulated using a probablistic model. A working probability is assigned to each sensor to simulate sensor breakdowns.

MITSIM provides TMS with the real time surveillance data. The data can be provided as per the MITSIM step size or at whatever step size is required by TMS for its control operations.

DynaMIT requires sensor counts from TMC (or from MITSIMLab's TMS) in order to carry out the OD estimation. DynaMIT simulates traffic on its own and generates the traffic counts from the simulation. A comparison of the TMC counts (true counts) and the simulated counts is used to decide whether the OD estimation has converged. If not then DynaMIT readjusts the OD flows and simulates again to obtain a new set of sensor counts. The iterative procedure is repeated on till the TMC and simulation counts converge. DynaMIT has a surveillance component of its own that helps it in recording the sensor counts. All sensors in DynaMIT are link-wide. DyanMIT sensors measure vehicle counts, flows, speed and density.

An interface is required that sends the sensor counts, as observed by MITSIMLab, to DynaMIT. The sensor counts need to be aggregated for the time period DynaMIT estimates the state of the network. Also filtering of sensor and surveillance data is required to report the counts only from the type of sensors used by DynaMIT.

2.2.3 Incident Management System

MITSIMLab reads the incident data from a scenario definition file. Each incident is represented based on the location (segment) where it occurs, and the visibility and number of lanes affected. Lane-specific information then includes the severity of the incident and its length, maximum speed, start time and the expected duration. The maximum speed specifies the upper bound with which vehicles can move through the incident. The TMS can control the clearance time of the incident. For example after the start of the incident, TMS perceives that the incident may take longer time to clear-up, the TMS can accordingly increase the expected end-time. This would be very similar to the role of a typical TMC during the management of an incident.

When an incident occurs in MITSIM, a message is sent to TMS describing the characteristics of the incident. The TMS then formulates a response plan after adding appropriate delays (See Yang (1997) for more information on the Incident Management in MITSIMLab).

DynaMIT models incidents based on the following information: *location (segment on which the incident occurs), start-time, expected end-time,* and *capacity reduction.* DynaMIT does not take into account the location or length of incident within the segment. It reduces the capacity of the entire segment by the specified factor.

Hence, the interface is required to convey to DynaMIT: (i) the location of the incident, (ii) start-time, (iii) expected end-time, and (iv) the capacity reduction factor.

2.2.4 Guidance and Control System

In MITSIMLab traffic regulation and route guidance is conveyed to individual drivers in the simulated network through a variety of devices and information means such as Variable Message Signs (VMS), in-vehicle units, etc. Vehicles view these devices and signs, and respond to the given control and route guidance according to the behavioral models. Each driver in the simulation can be either *informed* or *uninformed* depending on access to in-vehicle information source. The state of signals and signs is managed by the controllers in TMS.

MITSIM has two sets of travel time information: historical and real-time. Historical travel times are specified through a pre-specified data files. The real-time link travel times need to be updated periodically whenever information is received from TMS. Upon receiving the information the guided vehicles may update their route decisions based on the updated travel times.

DynaMIT has to provide TMS information about the predicted traffic conditions. DynaMIT's predicted traffic conditions take into account the drivers response to the provided guidance. As Kaysi et al. (1993) pointed out, predictive route guidance can minimize the inconsistency between provided information and drivers' experience and hence avoid problems such as over-reaction.

Prediction based route guidance is provided by taking into account drivers' current position, destination and projected travel times on the alternate paths. Expected travel times on the paths are calculated based on the projected time-variant travel times. Hence, the interface is required to provide guidance from DynaMIT in the form of a time-variant travel time list for each link. Periodic updates of guidance should be possible, in order to comply with the rolling-horizon implementation.

In addition, in order to investigate the sensitivity of the results to design characteristics of ATIS, the following parameters are additional inputs which the interface should be able to manage:

- different frequencies of guidance updates;
- different lengths of the prediction horizon;
- computational delay; and,
- various time resolutions of the guidance provided.

2.2.5 Software and Communication System

MITSIMLab is written in C++ using object oriented programming principles. The software system Yang (1997) consists of two simulation programs: MITSIM and TMS. A master controller (SMC) launches and synchronizes the execution of the MITSIMLab modules. Figure 2-1 provides the software architecture of MITSIMLab (Yang (1997)).

The communication between the modules is handled using interprocess communication and data files. The interprocess communication is implemented using the Parallel Virtual Machine (PVM). PVM (Geist et al. (1994)) provides the distributed architecture to MITSIMLab.

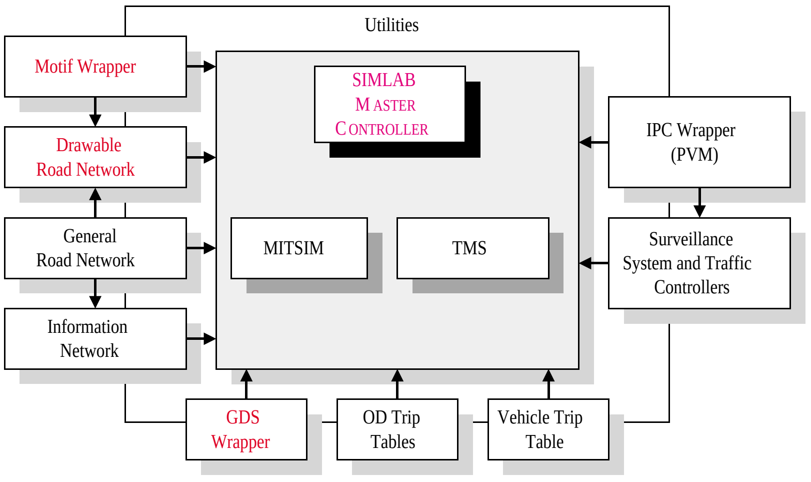


Figure 2-1: Software Architecture of MITSIMLab

DynaMIT is also written in C++ using the object-oriented paradigm. The system architecture is distributed and based on the Common Object Request Broker Architecture (CORBA).

The interface needs to provide the communication capability between CORBA-based DynaMIT and PVM-based MITSIMLab. Also, the basic design of the MITSIMLab-DynaMIT interface system should enable communication between the various subsystems located over multiple hosts. The different software systems may not be required to be running on the same host but may actually be located on different hosts in the network. This is required in order to emulate the actual subsystems in the TMCs which may be distributed across the network. Thus, the interface would be required to be able to locate and match different subsystem servers and clients throughout the network.

2.3 On-line Integration Requirements

Although the off-line evaluation serves as an excellent alternative for testing the capabilities and potential of traffic management systems, it is only a preliminary test in the evaluation process. The off-line tests can help in pointing out the potential deficiencies and shortcomings of the system ahead of the costly field tests. A first step towards full scale field deployment is the integration of the DTA system with the Traffic Management Center. The main objectives of the on-line evaluation would be to:

- assess the quality of the estimation and prediction capabilities using real operational data,
- assess the benefits and applicability of the outputs generated by the system,
- assess the real-time deployability, and
- pursue any refinements that might be necessary for further application.

As seen in section 1.2, TMCs have widely varying architectures and subsystem configurations. Design of a sufficiently generic interface which would require minimal improvisations to adopt to individual TMCs is a core requirement. Still, some level of customization for the individual TMC, where DynaMIT is to be installed, is unavoidable. In this section we discuss a set of requirements that make the designed interface fairly generic. We also discuss the integration requirements for complete interfacing with the TMC testbed at the University of California, Irvine. As in the previous section we also discuss the interfacing requirements for each of the subsystems.

2.3.1 Architectural Requirements

TMCs operate as a collection of multiple and sometimes highly dissimilar subsystems. Some of the functions are implemented on legacy sources that have highly varied system configurations are architectures. Furthermore, TMC data sources might themselves be actually located on multiple geographical locations and varied platforms. Figure 2-2 provides an example of the University of California, Irvine Testbed's information flow.

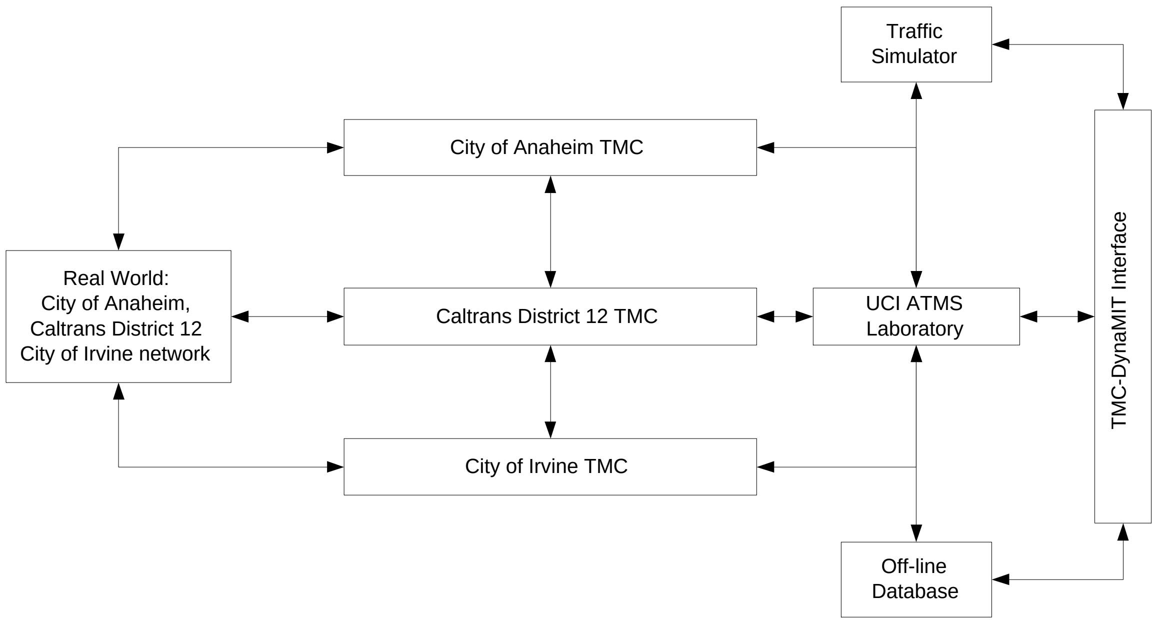


Figure 2-2: UC Irvine TMC Testbed Information Flow

The laboratory is actually receiving data from three individual TMCs (City of

Anaheim TMC, Caltarns District 12 TMC, and City of Irvine TMC). The three TMCs may have varied configurations which alter the frequency, accuracy and the type of data that is available for feeding into DynaMIT. Such varied configurations warrant the adoption of an architecture that can be easily *extensible* to different types of platforms, situated at multiple locations. This requires implementation of a suitable *middleware* technology that provides distributed implementation over multiple hosts and across multiple platforms.

The UCI Testbed communication has itself been implemented using CORBA (Comon Object Request Broker Architecture) over a large backbone of ATM network. The individual hosts are located over a TCP/IP network.

Another requirement of the interface architecture is the *parallel* implementation of multiple sources and users. For example, the interface should be able to allow simultaneous provision of data from a real-world source (like a TMC) and from another source like a Ground Truth Simulator (like MITSIMLab). The two sources of data may be implemented into two instances of DynaMIT running in parallel. Such a provision can be an excellent study tool and can be of great help to the TMC operators and researchers for comparison of multiple scenarios at the same time. The second source can be an off-line Database, rather than the simulator data. The interface should allow flexible *switching* to different server sources for this implementation to be possible. Figure 2-2 shows that such an implementation requirement has been planned for the evaluation of DynaMIT at UCI TMC testbed.

As previously pointed out, we cannot make an assumption of standardization or uniformity across all TMCs. Individual TMCs would invariably require at least some customization of the interface in order to comply with characteristics and features that are specific only to themselves. At the same time we should not consider an interface design that has to be changed completely for each TMC implementation. *Modularization* can facilitate the development of a flexible design. Hence, the interface will be implemented through multiple modules. This would require only one of these modules to be customized to the individual TMC at hand. The rest of the modules can be fairly generic and independent of the individual TMC implementation.

The interface is designed to be a real-time implementation tool as well as a research tool operating on-line and off-line data. Further research advances and practicalities of implementation would warrant inclusion of many more features and utilities into the interface than what had originally been planned. This would require the interface design to be sufficiently flexible for further additions. Also, the future implementation of theinterface would involve much bigger networks and enormous amount of data handling. Thus our interface should be *scalable* to bigger and more advanced implementations.

In addition to these requirements Ruiz (2000), proposed the following key features and requirements for an architecture that can support the operations of an integrated DTMS within the TMC:

- *Open:* An architecture that is easily expandable and can be adapted to very different applications.
- *Anonymous:* A system that has no hard-coded reference and an architecture in which the identification is done in real-time to give additional flexibility.
- *Distributed:* An architecture that can support various TMC operations that can located on any platform at any physical location.
- *Concurrent:* An architecture that allows different subsystems to run in parallel performing concurrent tasks.
- *Object-Oriented:* An architecture that is object-oriented to reduce long-term development costs and risks.
- *Secure:* An architecture that is secure against information leak and potential hacks.
- *Standard:* An architecture that follows the current software standards to insure widest possible applications and inter-operability.

2.3.2 Surveillance Interface

The surveillance system may consist of multiple sources. For example in the case of UCI testbed, the surveillance might be received from either one of the three individual

TMCs.

The interface should be able to *simultaneously process data* received from these varied sources. In addition, the interface should be able to *map* the field sensor locations and local identification numbers (IDs) to the corresponding IDs in DynaMIT.

Furthermore, the sensor data provided by the TMC may be at different levels of aggregation. For example, the City of Irvine TMC provides aggregate data at 5 minute intervals, while the Caltrans TMC at 30 seconds intervals. Thus, the interface should be able to provide the required *aggregation* capabilities, so that all sensor data are consistent for the DynaMIT specification and requirements.

Finally, the data received may be in a very different format than that acceptable by DynaMIT. The interface should be able to covert the data into DynaMIT-readable format and only transmit data that is useful for DynaMIT. It thus needs to be able to *filter* the information received from various sources.

2.3.3 Incident Detection Interface

Incident Detection usually can be a fairly non-uniform process and varies widely from one case to another. Incidents can either be reported by moving patrols or through drivers themselves. Such reports ultimately do end up with the TMC operators. In some cases, incidents may be directly observed by TMC operators through video cameras or other imaging devices. In some other cases they may be detected through incident detection algorithm using information from the surveillance system. The interface design should be able allow flexibility to incorporate all the above different possible scenarios of incident detection. It should also be able to map the incident location into the corresponding DynaMIT segment ID. It should also be able to broadcast the incident start time and the expected end time (as perceived by the traffic operators). The incident start time would usually be different from the reported time (because of lag in detection). DynaMIT should also be notified through the interface about the expected reduction in capacity.

2.3.4 Traffic Control and Route Guidance Interface

The Traffic Management Centers will use the predictions by DynaMIT to develop a control strategy and translate it to status for the physical signals and other devices (VMS, etc.).

The traffic control and route-guidance interface should be able to provide the guidance and traffic prediction generated by DynaMIT, with a format consistent with the one required by TMC. The predicted network state should be able to be visualized using graphical displays. The predicted state of the network should be provided through travel times and/or network flows, speeds, queues and density information.

The route guidance may be implemented through multiple remotely-located control subsystems. The guidance information, in such a case, is required to be broadcast to multiple clients over the network. Hence the interface should identify all the users of DynaMIT output, and the part of the data they require. The interface should then be able to broadcast the specific data to the respective clients.

Chapter 3

Interface: Architectural Design

As seen in chapter 2, the integration of DynaMIT within TMC requires an architecture that is able to support multiple legacy information sources, distributed over multiple hosts, operating in tandem and exchanging data with each other in real-time. This chapter describes the architecture that has been implemented to meet the requirements as outlined in the last chapter. The first part of the chapter describes the design of the generic architecture that has been implemented for the interface. The second part describes the implementation of this architecture for the MITSIMLab-DynaMIT integration. The third part describes how this design can be used for the Traffic Management Center at Orange County, California.

3.1 System Architecture

The architectural design that has been implemented for the interface strives to meet the requirements as mentioned in the previous chapter.

Figure 3-1, shows the overall interface architecture that has been adopted for the integration of DynaMIT with the TMC. The interface consists of three modules: the Traffic Management Center Adaptor (TMCA), the Dynaic Traffic Management Simulator (DTMS) and the DynaMIT Communicator. The DTMS and DynaMIT Communicator are generic modules, which would remain the same in all the interface implementations. The TMC adaptor would be specific to the TMC we would be integrating DynMIT with. The surveillance, guidance and the incident detection systems at the TMC communicate directly with the TMC adaptor. Through a clientt/server implementation (described in the sections later in the chapter) the TMCA communicates with the DTMS. The DynaMIT communicator has a similar client/server implementation on the DynaMIT side. The communicator exchanges the data with the DynaMIT modules. Xdta is the graphical interface for DynaMIT. It displays the flows, speeds and densities as estimated and predicted by DynaMIT. Xdta is instantiated by DynaMIT whenever it completes a prediction.

The following sections describe in detail the features and functionality of this architecture and how they meet the requirements we have previously discussed. We begin first by describing the software technologies that have been used for the interface and follow that by describing the distribution mechanism that has been adopted. We then describe in detail each of the separate elements of the interface (shown in Figure, 3-1), and follow that by a general description of the process.

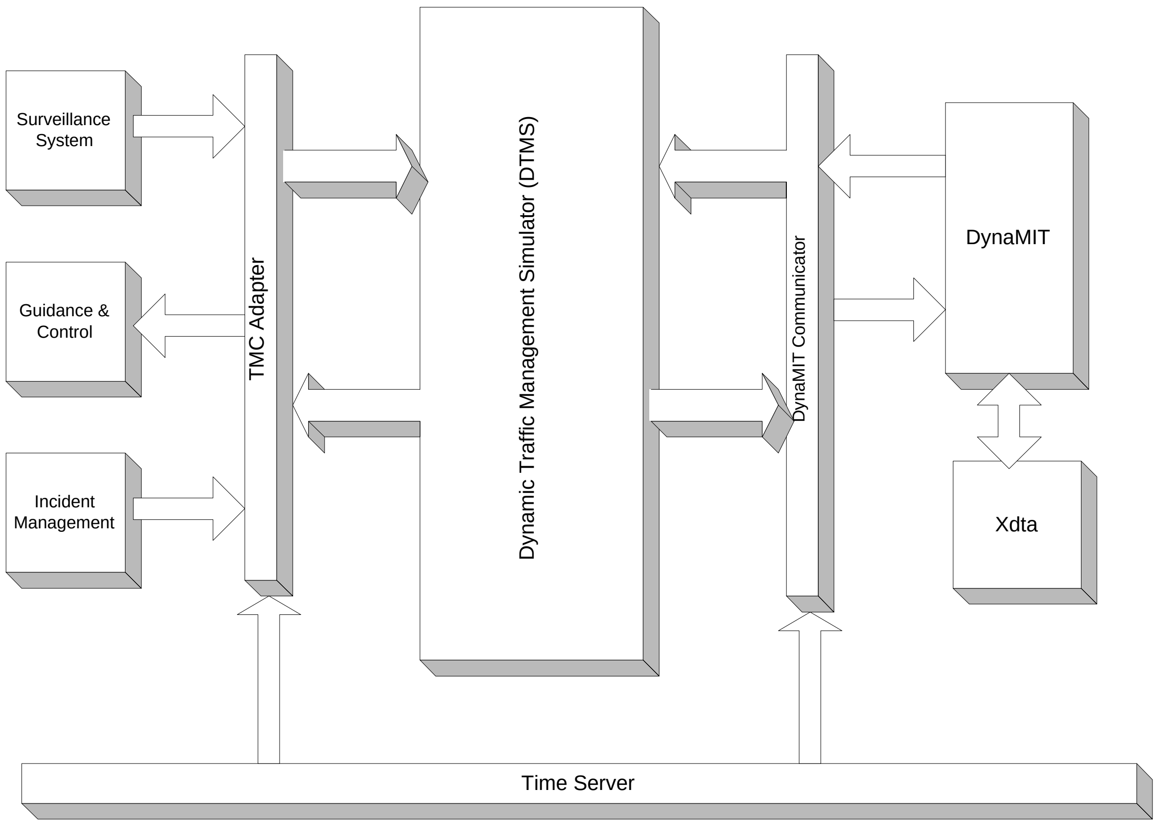


Figure 3-1: Overall Interface Architecture

3.1.1 Distributed Implementation

Orfali and Harkey (1998), studied various technologies available to implement distributed systems: sockets, CGI scripts, CORBA, DCOM, RMI, etc. Ruiz (2000) reviewed these technologies in order to select one that best fits the need of DTMS architecture. He proposed the use of the extended Common Object Request Broker Architecture (CORBA) as the object distribution implementation for use in Dynamic Traffic Management Systems. He mentions the following advantages of using CORBA as the distribution mechanism in DTMSs:

- *Platform Independence*: CORBA is hardware and software independent and provides an abstraction layer for development of systems in multi-platform environment. The different subsystems that would need to be integrated with the TMC can be expected to include different platforms across an heterogenous network.
- *Legacy System Inclusion*: CORBA can be implemented across different multi-vendor legacy systems, without incurring significant reengineering costs.
- *Security*: CORBA standards define a security mechanism that is always integrated in the implementations.

An extended CORBA implementation can include additional services like the Event Service, Naming Service, and the Trading Service. The CORBA event service provides a flexible model for asynchronous communication communication among objects. The CORBA Naming Service is central to most CORBA applications. It serves as a directory for CORBA objects, allowing objects on one host to locate objects on another host through user-defined names. The CORBA Trading Service can be seen as a generalization of the Naming Service; instead of merely providing a way for clients to search for available servers by name, it provides for servers to register their capabilities and clients to find them based on properties, specified via a simple constraint language.

Figure 3-2, shows an example of a simple CORBA application. Subsequent sections will present specific and more advanced implementations of this basic model.

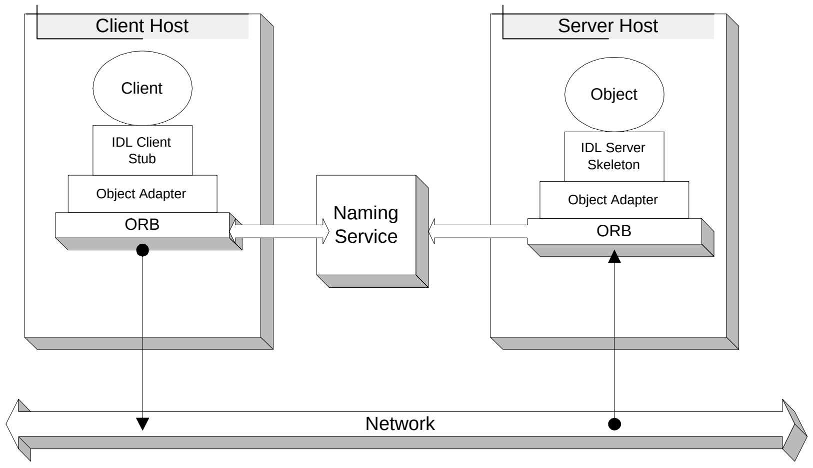


Figure 3-2: Overview of a CORBA Application

The client (e.g. the surveillance module in DynaMIT) seeks the services of a server (e.g. the surveillance information module residing within the TMC). When the server starts, it creates one or more objects and then publishes its object reference (the object handle) in the naming service. The client obtains the object reference for the server it wants to call from the naming service, and packages this information along with the method's name and its parameters, and sends it to the client stub as if it is making a local call. The client stubs are automatically generated, in the target language of the client (a feature that makes CORBA's language-independent and cross-platform implementation possible), when the server IDL was compiled. The client stub sends this packet in the form of a request.

The Object Request Broker (ORB) enables the communication between the clients and the servers through the physical network. ORB handles the language-independent requests and replies for the clients. For this, it needs the object references for the server it has to make the request call to. It gets this from the naming service. ORB sends the message to the server's Object Adaptor [1] via the Internet Inter-ORB protocol (IIOP) [2]. The server accesses ORB operations through the Object Adaptor. The server skeleton then turns the request data into the server's language-specific method call and then performs the call on the servers object implementation. The reverse process occurs while sending back the reply to the client.

3.1.2 Dynamic Traffic Management Simulator (DTMS)

As shown in Figure, 3-1, the DTMS layer instantiates the various servers to which the TMC Adaptor and DynaMIT communicator clients bind to while exchanging the data during the execution of the interface.

DTMS instantiates one server for each individual subsystem on the TMC side that communicate with the subsystems on the DynaMIT side. In addition, it has a Registry server (Figure 3-4) that keeps the names of all the servers belonging to a

[1]Object Adaptor is the entity that mediates between object implementations and the ORB.

[2]The standard mechanism by which communication takes place in between ORBs on TCP/IP based networks

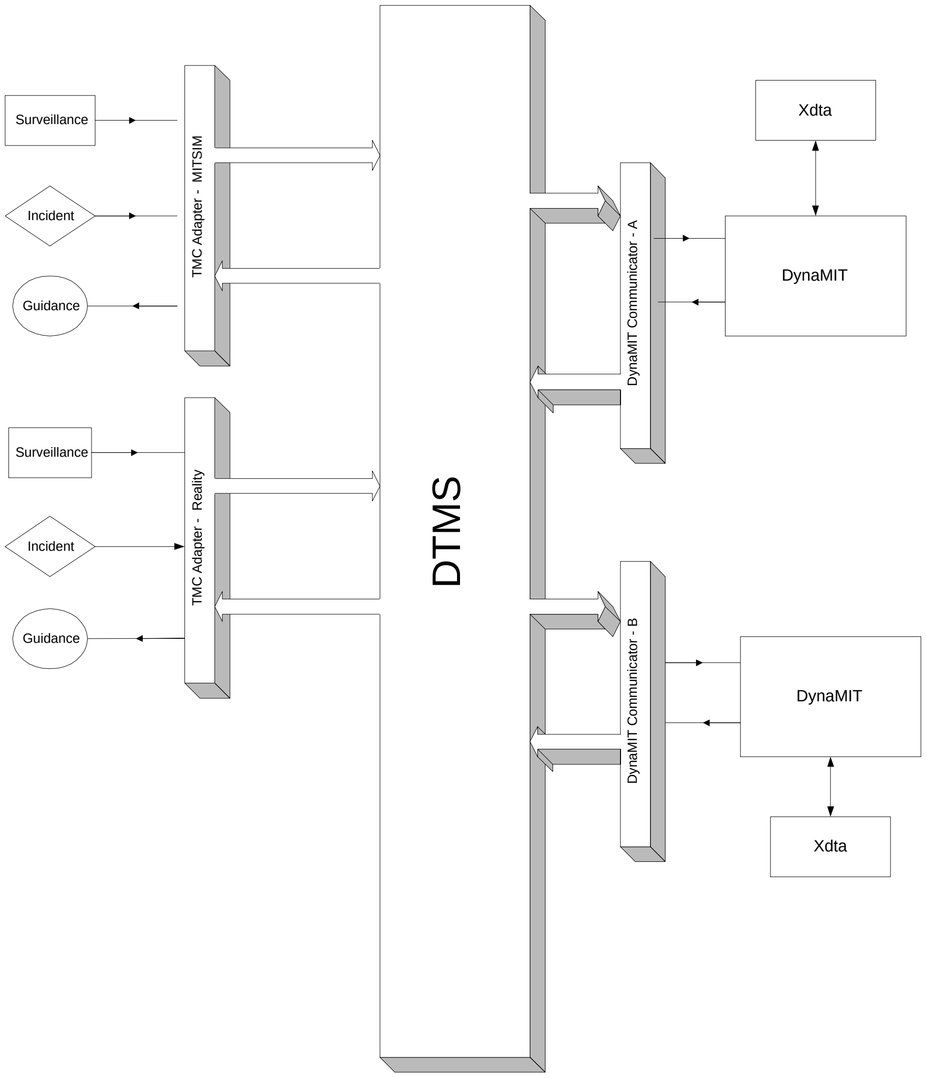


Figure 3-3: Simultaneous support of Multiple TMC-DynaMIT instances

specific system. DTMS can map simultaneously multiple TMC systems with their corresponding instances of DynaMIT (shown in Figure, 3-3). For example, we may require to run an instance of MITSIMLab and another instance of the TMC system with the field collected real-time data, together running along with DynaMIT. We can do this by instantiating two separate TMC adaptors (one for MITSIMLab and the other for the field TMC data) and link them with two separate instances of DynaMIT.

Each TMC-DynaMIT mapping pair has its own Registry server that maintains the names of the following servers:

- The Time Server,
- The Surveillance Information Server,
- The Incident Management Server, and
- The Guidance Server.

The Registry thus acts as a naming service that keeps the names of the object references and locations. A Registry server is the first one to be instantiated. Each Registry server has a unique name and its host location. All of the servers mentioned above are than added to their respective registries at instantiation. Each of these servers is instantiated by providing the server name, the registry name and their locations (Figure, 3-4). Appendix A shows an example script file that instantiates the above mentioned servers for the MITSIMLab-DynaMIT Interface.

Figure 3-5 shows what happens when a server is instantiated. All server instances and its objects reside in a process. The process address remains constant as long as the server is running. It will disappear or run as a new process, if the server is killed or restarted. CORBA provides a component known as the location domain, which tracks the current location of the server objects. The servers store their object adapter information in a centralized repository called the Implementation Repository (IMR).

IMR is itself managed by a locator daemon. When a server is instantiated, it registers its object adapter and object information with the locator. It also registers its host and port numbers (on which the Object Adaptor can be contacted), which

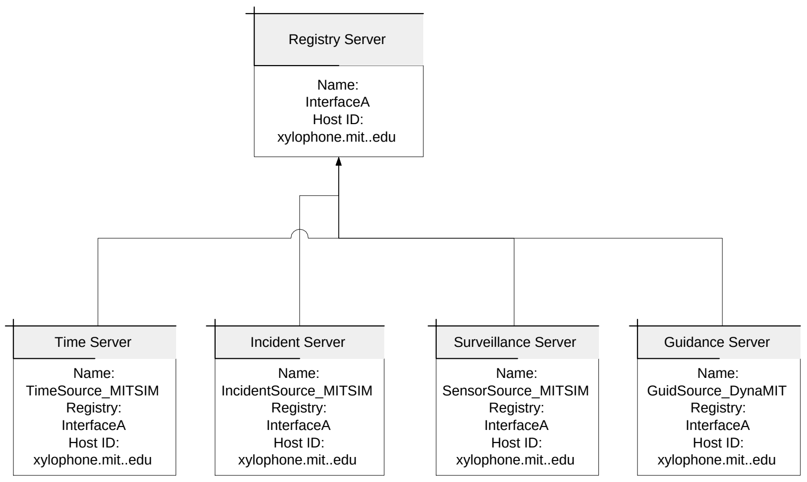


Figure 3-4: Multiple Server Instantiation for each TMC-DynaMIT pairing

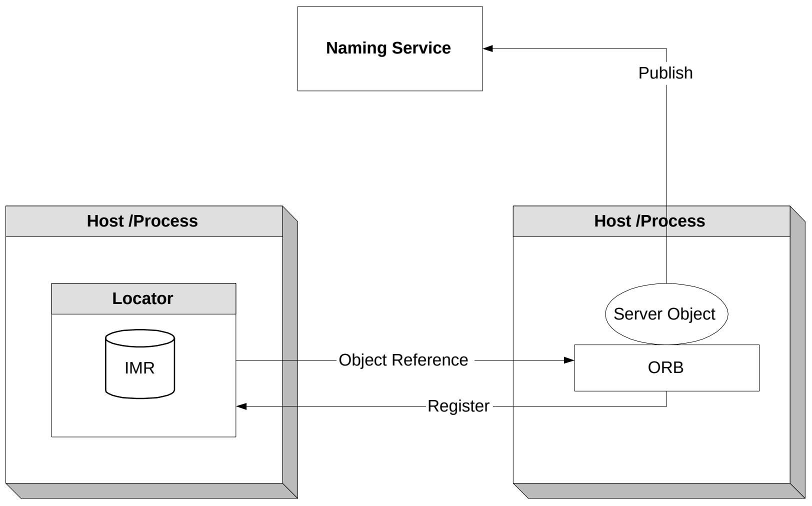


Figure 3-5: Registration of Server Information

may dynamically change. The locator in turn returns a modified reference that contains the address of the locator. The server object then publishes the modified object reference with the naming service like the DTMS Registry.

3.1.3 Traffic Management Center Adaptor (TMCA)

As seen in section 1.2, Traffic Management Centers have a widely varying system and communication architectures. The data provided by these systems also varies widely in format and resolution. The TMC Adaptors is a customization layer between the DTMS and the TMCs which helps in the translation and communication of data in a standard usable format. Broadly speaking TMC adaptors have two purposes:

- To establish a communication with the individual TMC subsystems, assimilate the data, and then communicate the needed information to the DTMS and subsequently to DynaMIT,
- To manipulate the data according to the required resolution and time interval and convert it into the acceptable format.

TMC adaptors, thus, are specific and customized to the individual TMC systems. Every TMC should have its own TMC Adaptor. Architecturally speaking, a TMC adaptor has two sides to it. One side is very similar to the architecture of the individual TMC subsystems and the other one resembles the architecture of the DTMS layer. It retrieves data through one layer and sends it through the other. Specific TMC adaptors are described more thoroughly in sections, 3.3 and 3.4.

A TMC Aadaptor acts as a client to the servers instantiated by the DTMS. It finds a registry it wants to bind with. It then binds with the individual Surveillance, Time, Incidence and Guidance servers. The TMC adaptor publishes information into the Surveillance and Incidence servers and retrieves information from the Guidance Server.

3.1.4 DynaMIT Communicator

The DynaMIT communicator is the interface between DTMS and DynaMIT. DynaMIT is implemented as a distributed set of CORBA-based servers. Each server represents a single-threaded process running on a single platform. Server processes are created automatically by the Orbix runtime system in response to the instantiation of the server assigned system objects. The number of servers and their platforms are configured at runtime, and the inter-operability between objects is provided through the CORBA interfaces tied to the system objects.

A different server, called the *Server manager*, keeps a dynamic-table of all the object-server-platform assignments in the system. The server manager has a CORBA interface that makes assignments, creates and destroys objects, creates and destroys servers, and looks up the location of a given object or a server. Every server in the server manager contains a single instance of an object called the *Object Manager*. The Object Manager is responsible for all the system objects contained within a specific server.

All objects in DynaMIT have a single instance [3]. All method calls to an object are handled by the same physical process and access the same object instantiation. Each server and each object are assigned a unique name, which is used to reference the system object whenever it is bound. Each system object contains the implementation of a single element of the overall system. System objects are themselves grouped into separate classes in the form of *utilities, components, modules* and *processes*, in the increasing order of hierarchy.

The DynaMIT communicator binds with the objects in DynaMIT that deal with surveillance collection, incident management and guidance generation. The communicator has been designed to be launched as a separate server (along with others) through DynaMIT's Server Manager. Thus, the communicator is able to obtain the object references of the system objects it needs to establish communication from the DynaMIT side. The communicator can be launched on an entirely different

[3]All objects are instantiated in shared mode, unlike the un-shared or per-method mode of many CORBA applications.

host than the rest of the system objects. The host configuration of the Communicator can be established in the configuration file at run-time - just as it is being done for all other server objects through the server manager. The DynaMIT side implementation of the communicator can thus establish the data exchange with the different modules of DynaMIT.

The DTMS side exchange is a little more involved process. DTMS (as pointed out in section 3.1.2) can have multiple instances of TMC subsystems running simultaneously. The DynaMIT communicator needs to establish the server names of subsystems it wants to contact and the hosts they are running on. Also it needs the reference to the Registry where these server locations are stored. This information cannot be hard-coded and needs to be input for every instance of DynaMIT. The subsystem servers which DynaMIT needs to contact and their host names can be mentioned in a data file - which DynaMIT can read at execution and contact the appropriate servers and host locations. A sample host and server name file is included in Appendix A.

3.1.5 Software Environment

The interface has been implemented in C++ using the object-oriented paradigm. This provides the flexibility to enhance and modify the code for future applications. The entire system is implemented in distributed mode using the Common Object Request Broker Architecture (CORBA). Section, 3.1.1 provides a more detailed explanation of the distributed implementation.

FLEX++, a lexical analyzer generator (Levine et al. (1992)), and BISON++, a parser generator (based on a GNU version of BISON (Donnelly and Stallman (1992)), created by Alian Coetmeur), are used for developing the parsers for reading the data files that provide the host and factory information. FLEX++ and BISON++ generate the C++ code that reads the data files. Much of the data structures have been implemented using the Standard Template Library (STL) from C++.

The system has been compiled and run on SGI's IRIX 6.x and Sun Solaris 5.6 platforms. The compilers used are GNU C++ and the standard SGI and Solaris compilers. Orbix 2.0 by IONA Technologies is used for compiling the IDL files.

3.2 Process Description

The TMC-DynaMIT communication process gets instantiated by starting a registry server. Once the TMC to which DynaMIT will provide traffic prediction and assignment support is specified, a registry specific to that TMC is marked and started. The registry is started by providing it a unique name and a host location. The time server, surveillance information server, incident information server, and the guidance information server, are instantiated next. They are started on specific hosts and their location information is provided to the registry. More details on starting the DTMS servers was provided in section 3.1.2.

The TMC adaptor and the DynaMIT communicator then, start up their own clients - one each for time information, sensor information, incident information, and guidance information. These clients then locate and bind to their corresponding servers and send in their requests (as shown in Figure 3-6).

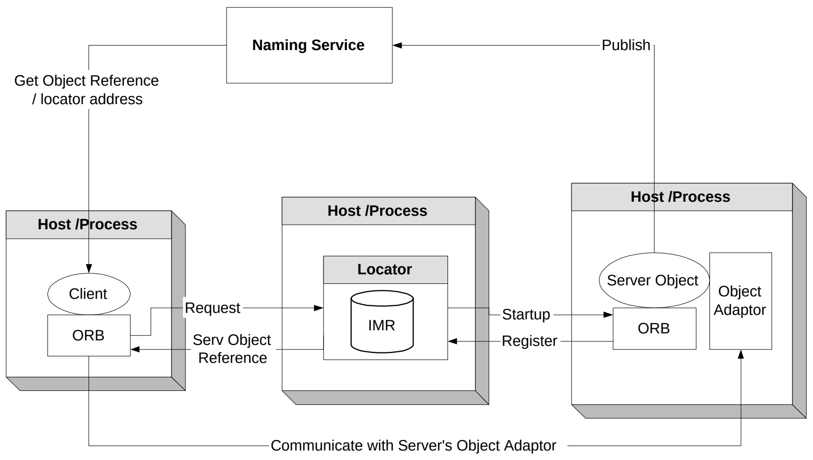


Figure 3-6: Client/Server Location and Communication Process

The clients retrieve the object reference of the server they want to bind to from the registry. The object reference consists of the object adaptor, object name and the locator address. The clients use the object reference obtained from the registry to call an operation on the server. The client ORB then, uses the object reference to send a request to the locator (described in section, 3.1.2). The locator [4] uses the Object Adaptor name to find the object reference from its database and then passes this back to the client.

The client starts using this object reference to establish communication with the server and uses this for all subsequent functional calls. All the clients on the TMC adaptor side and DynaMIT communicator side thus establish direct communications with their corresponding servers.

The communication between the respective subsystems in TMC and DynaMIT takes place through a push/pull mechanism. It is based on the Object Management Group's (OMG) Event Service model through the "publish/subscribe" paradigm. This model suggests inserting a third part between the client and the server in order to provide a high degree of decoupling between the two. This helps in supporting concurrent applications. Under the simple client/server model the server is blocked when it provides information to the clients. This would not be a good option in our case because, for instance the TMC subsystem operations cannot be blocked when DynaMIT is reading surveillance or incident data. The very nature of our subsystem requires a continuous operation with simultaneous publishing and retrieval. The provision of a middle buffering layer between the clients and servers helps in avoiding this kind of problem. Figure, 3-7, shows how the communication architecture is implemented under the publisher-subscriber paradigm using the push/pull model.

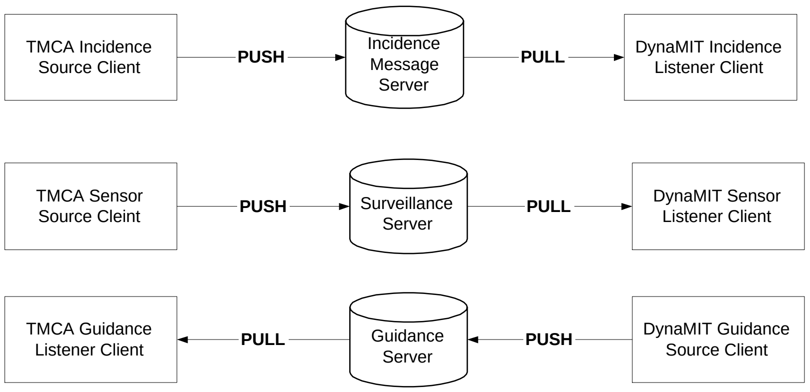

TMCA Incidence Source Client
PUSH
Incidence Message Server
PULL
DynaMIT Incidence Listener Client
TMCA Sensor Source Cleint
PUSH
Surveillance Server
PULL
DynaMIT Sensor Listener Client
TMCA Guidance Listener Client
PULL
Guidance Server
PUSH
DynaMIT Guidance Source Client

Figure 3-7: Push/Pull model for providing asynchronous communication

[4]The locator keeps communicating with the server to verify its presence

As discussed above, through the registry service all the clients know their servers and all the servers would know their clients. Whenever the client (for example, the Surveillance generating client on the TMC adaptor side) receives data (new sensor counts in this instance) it creates a message (by invoking the `createMessage()` call) and publishes (pushes) it to the corresponding server (the Surveillance server in this case) on DTMS. It enters the DTMS surveillance data base corresponding to that particular TMC.

The corresponding client on the DynaMIT communicator side (the surveillance reader in this case) invokes the `getMessage()` call and pulls the message whenever it requires it. This push/pull mechanism allows simultaneous and asynchronous operations on both the client and server (or publisher/subscriber) side without either waiting for completion of each others events.

This inclusion of additional buffering data layer in between would not be the most efficient in terms of memory management, but it is facilitates asynchronous and concurrent operations which is a necessary requirement. The database can, nevertheless, be periodically cleared after a sufficient elapse of time.

3.3 Architecture for MITSIMLab-DynaMIT Interface

As previously mentioned, TMS (Traffic Management Simulator) within MITSIMLab emulates the Traffic Management Center operations. The MITSIMLab architecture, as we will see in this section, also has certain features that are similar to the subsystem architectures found in many of the TMCs.

The DynaMIT-side (DTMS and DynaMIT communicator) implementation of the interface does not depend on the specific TMC system and its architecture, and thus, it is pretty much standard for all TMC instances. All the different types of TMC systems, though, require a TMC Adaptor interface that is very much specific to the TMC being used. Therefore, the MITSIMLab-DynaMIT interface needs its own specific design of the TMC Adaptor layer.

MITSIMLab is implemented as a distributed system using the Parallel Virtual Machine (PVM). The TMC Adaptor has to communicate between the PVM-based MITSIMLab and the CORBA-based DTMS and DynaMIT Communicator. The TMC adaptor thus, needs an architecture that supports both the PVM and CORBA implementation.

PVM enables a collection of heterogenous systems to be used cooperatively for concurrent and parallel computation. The computational tasks can be executed on a configured host pool which may include multiple machines or hardware multiprocessors. The unit of parallelism in PVM is a task and multiple tasks may be executed on the same processor. An application can be considered to be composed of multiple tasks, each responsible for a part of application's computational workload. The application can be parallelized along its functions (called *functional parallelism*) or in the other case the function to be executed is the same, but each task operates on a small part of the data. This is called *data parallelism*[5].

MITSIMLab uses functional parallelism for its multiple processes. The MITSIM and TMS components of MITSIMLab are distributed using PVM. This is to the overall framework in which TMC subsystems operate. TMC subsystems (for instance surveillance or control and guidance) are located on different remote hosts distributed over the network. The data collected may be relayed to the TMC facility which may be located elsewhere. The surveillance component of MITSIMLab (located in MITSIM) and the control and guidance component (located in TMS) can similarly be located on different hosts using PVM.

The TMC adaptor for MITSIMLab also needs to be integrated with the PVM architecture to allow it to communicate with the TMS. Figure 3-8 shows the communication architecture of the MITSIMLab-TMC Adaptor interface. MITSIM, TMS and TMCA (TMC adaptor) are controlled together through a master controller (called SMC - the Simlab Master Controller). All three can be located on separate hosts. MITSIM (located on HostA) exchanges the surveillance and control and

[5]This is also referred to as single-program multiple-data (SPMD) model of computing.

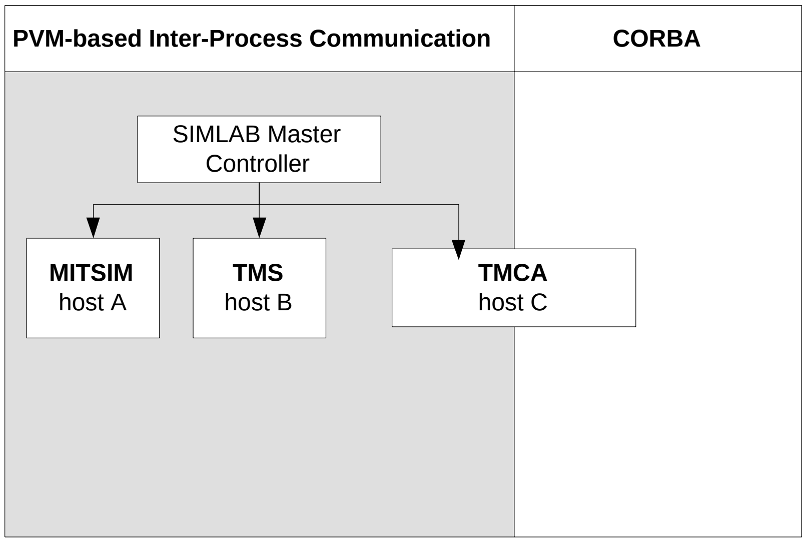


Figure 3-8: Communication between TMC Adaptor and MITSIMLab

guidance data regularly with TMS (located on HostB). The TMC adaptor has one part which uses the PVM communication and is instantiated as a part of the MITSIMLab master controller. The other part which talks to DTMS uses CORBA for communication. The TMC adaptor talks to TMS whenever it needs to get surveillance or report prediction-based guidance.

3.4 Architecture for TMC-DynaMIT Interface

This part discusses the interface architecture for integration of DynaMIT with a TMC. The discussion focuses on the UC Irvine Testbed, which in turn is interfaced with the City of Anaheim, City of Irvine, and Caltrans District 12 TMCs.

As in the previous case with MITSIMLab, the modularization of the interface allows us to maintain the DTMS and Dynamit Communicator part of the interface, without making any changes. All the changes, specific to the TMC testbed, need to be made only with the TMC Adaptor.

As specified in section, 2.3, the TMC-DynaMIT integration entails interfacing with the real-world data as well as with the TMC maintained database at the testbed.

The integration of DynaMIT with the TMC testbed proceeds in two stages. The first stage allows interfacing directly with the raw stream of data sent form the surveillance stations. It includes the freeway stream data and the arterial stream data. Figure 3-9 provides a schematic of the planned interface architecture. The first stage does not have any interfacing with the Database server (although the CORBA service will continuously feed the database client).

The interface architecture revolves around a CORBA-based Event Notification Service. The individual TMCs or surveillance subsystems collect the sensor data through various field-deployed sources and send them to a Memory Map File [6]. Each surveillance subsystem has its own event notification service.

The TMC adaptor instantiates clients for each of the surveillance subsystems. These clients register with the event notification service. Whenever the surveillance

[6]A disk file that contains a dump of all the data being received from the surveillance stations

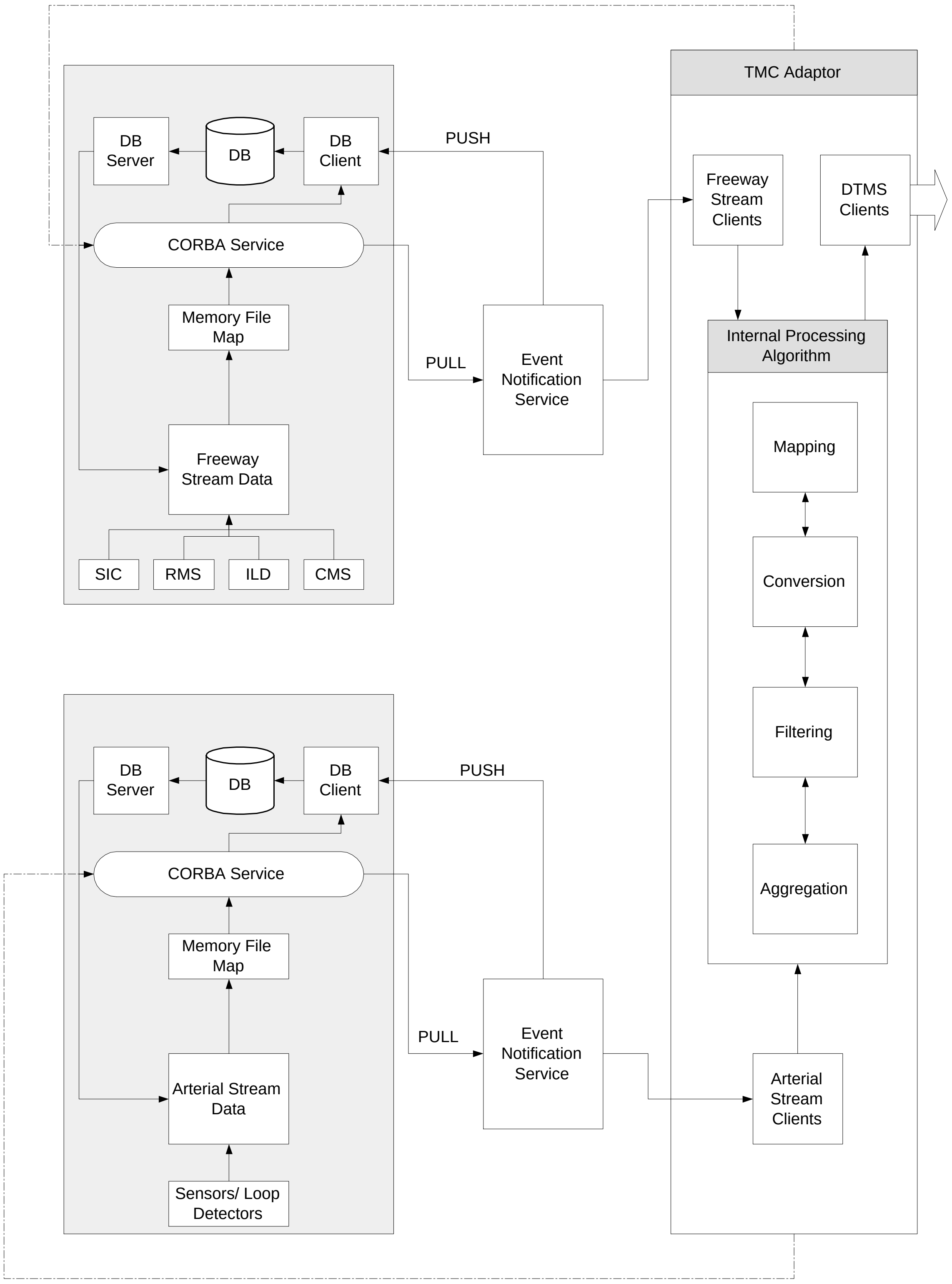


Figure 3-9: Integrated TMC architecture (Phase I)

subsystem's CORBA service receives a request for data from the notification service, it looks into the memory file and if there is new data it broadcasts it to the TMC adaptor client bound to the notification service. The data is communicated through a pull mechanism. The event notification service also pushes the data to a Database client, which publishes it to the database. The data is still unprocessed as obtained from the sensor stations.

The TMC adaptor once it receives the data, sends it to the internal algorithms that process it further (explained in detail in section 4.2) as per the requirements of DynaMIT. The adaptor then sends them to the respective DTMS servers, which further broadcast it to the DynaMIT communicator in much the same way as explained earlier in this chapter.

When the TMC adaptor receives the guidance information form DynaMIT, it sends it through the CORBA services to the individual TMCs or guidance and control subsystems.

The second stage of interfacing involves adding filters to the event notification service so that clients only receive the specific data they request, rather than the complete raw data stream. This will reduce the traffic between the TMC adaptor clients and the data servers considerably. The second stage will also provide interfacing with the off-line database being maintained at the TMCs. Figure 3-10 illustrates the TMC architecture in the second stage of implementation.

The TMCA clients send specific requests to the notification service. An example of a request would be - for example to get sensor data in VDS (Vehicle Detection Station) format (explained in section 4.2) for every 5 minute interval. The raw data is received from the TMC and is pushed to the three intermediate algorithm implementations, which then pushes it to the processed data notification service and ultimately to the client making the request.

The TMC adaptor will still have internal processing algorithms (of the types shown in Figure 3-9), in case any further processing is required. This will add additional flexibility in testing DynaMIT under different scenarios. Under the most basic application many of these internal algorithms will not be needed, as the clients

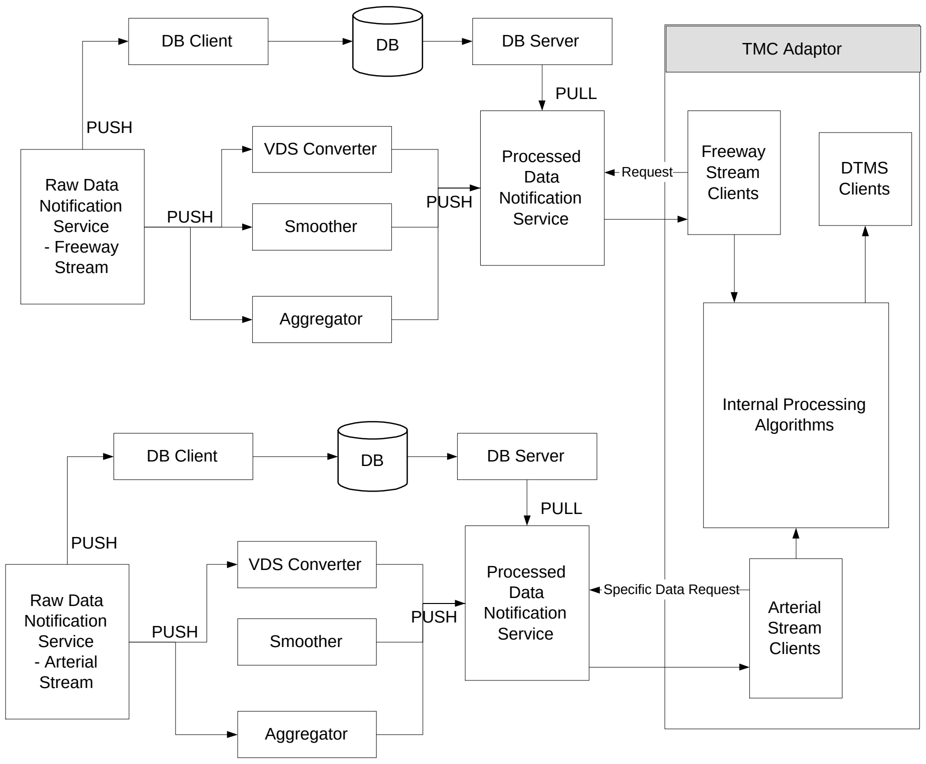


Figure 3-10: Integrated TMC architecture (Phase II)

can directly request the data format they want. However, some of these algorithms will still be required. For example, we may still require the sensor data mapping algorithm to convert the sensor location into corresponding DynaMIT sensor ID.

Chapter 4

Implementation Framework

This chapter discusses in detail how the architecture we adopted in the last chapter has been implemented for the off-line evaluation of DynaMIT, using MITSIMLab, and for the TMC-DynaMIT integration. We also discuss the relay of relevant information and data between the various subsystems, in order to achieve the requirements we identified in chapter 2. More specifically, we first discuss the application framework for the off-line evaluation version. Within the off-line evaluation description, we present both the open-loop and closed-loop applications. The second part of the chapter discusses the proposed interface implementation for integrating DynaMIT with the Irvine Traffic Management Center. To make the implementation more generic, we also discuss how this framework would comply with the requirements of the Interface Compliance Document(ICD).

4.1 Off-line Evaluation

The off-line evaluation of DynaMIT is intended to serve as a guideline for the more expensive field-operational tests. As mentioned in section 2.1.1, MITSIMLab can be an excellent tool for evaluating the performance of DynaMIT. Using MITSIMLab as an evaluation tool requires interfacing of DynaMIT with the Traffic Management Simulator (which resides within MITSIMLab and emulates the operation of a TMC). Section 3.3, had discussed the architectural design that was used for establishing the integration of DynaMIT with the TMS.

Off-line evaluation aims at addressing the issues related to the system functionality, accuracy, robustness, and applicability. It can also be used to evaluate the quality of estimated and predicted traffic conditions generated by DynaMIT, and how well they compare with the real data. Off-line evaluation of DynaMIT with MITSIMLab therfore, requires calibration of DynaMIT models using data from MITSIMLab. These include the route-choice, on the demand side and speed-density relationships and capacities on the supply side. This thesis does not address the calibration details. It focusses on the applicability and implementation of the system to perform the required tests.

4.1.1 Off-line, Open-loop Implementation

The off-line, open-loop implementation of DynaMIT requires real-time sensor and incident information from MITSIMLab (acting as the ground-truth simulator). Figure 4-1 shows the open-loop evaluation framework. In addition to the sensor counts and incident data, DynaMIT also requires (in order to perform the estimation and prediction) the following information, which would be provided separately through data files:

(a) Network topology, specifying the links, nodes, segments, lane groups, lanes and their connections.

(b) Calibrated supply parameter file that includes free flow speed, jam density, alpha [1], beta [2], capacity, and minimum speed, for each segment.

(c) Historical Demand file specifying the hourly flows between all the origin-destination pairs.

(d) Socio-Economic data file specifying the characteristics of the population using the network under investigation.

(e) A Behavioral Parameter file specifying the calibrated route-choice coefficients.

[1]Inner coefficient in the speed-density function

[2]Outer Coefficient in the speed-density function

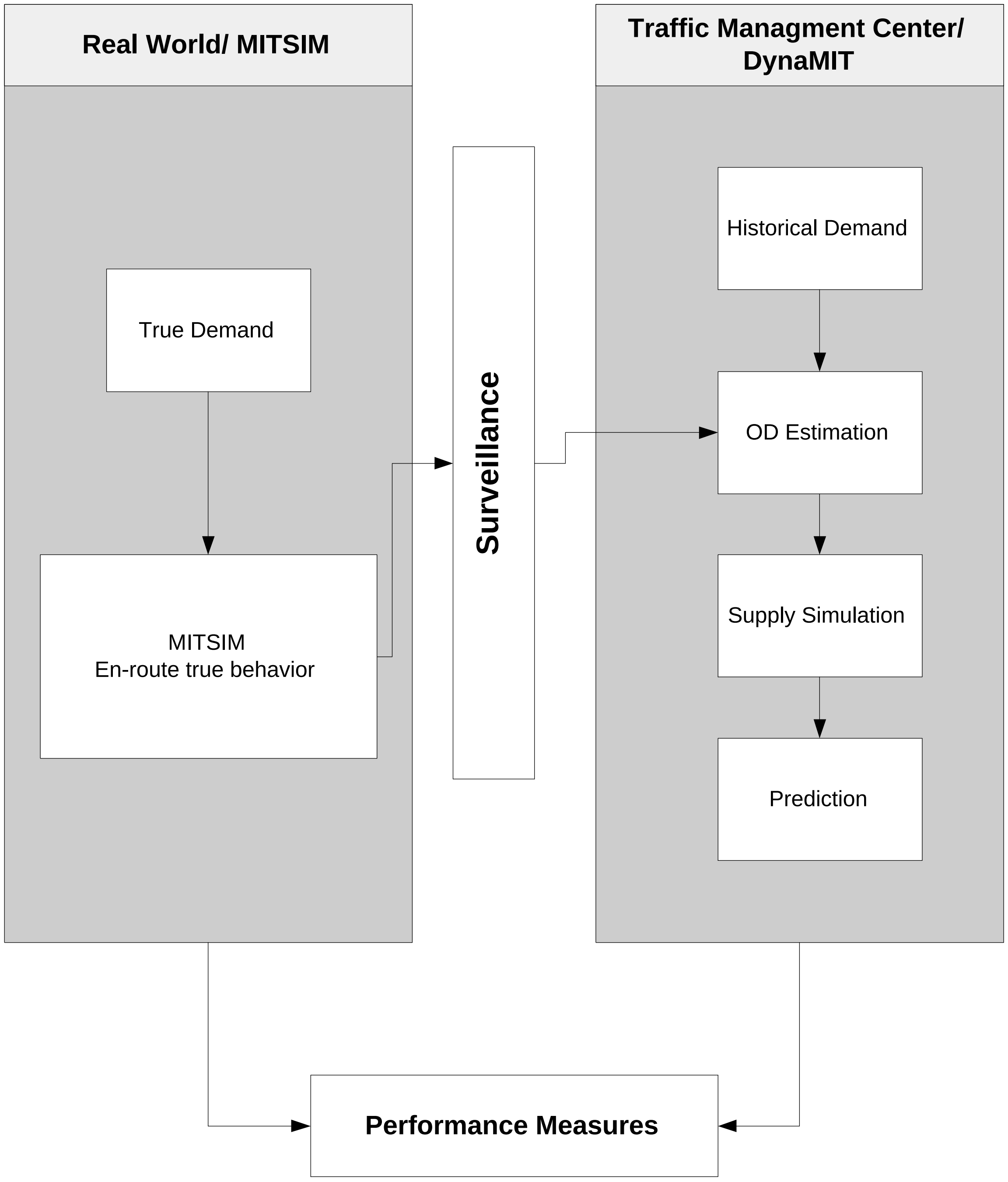


Figure 4-1: Framework for Open-loop Evaluation

(f) A Historical link time file, specifying the normal travel times on each link as a function of time (used by the drivers to make habitual route-choice decisions).

DynaMIT estimates the traffic conditions based on the sensor counts and incident data it receives from MITSIMLab. The control being used by MITSIMLab would remain fixed and DynaMIT would be emulating the same control as being used by MITSIMLab (or TMC). In other words, the traffic control would not be based on the prediction provided by DynaMIT. The DynaMIT-predicted traffic conditions can then be compared with the actual traffic conditions on the network (or in MITSIMLab).

The open-loop implementation thus, involves concurrent transfer of sensor counts and incident information data to DynaMIT. Figure 4-2 shows the overall framework with which the open-loop, MITSIMLab-DynaMIT interface has been implemented on the MITSIMLab side. Figure 4-3 shows a similar diagram of implementation on the DynaMIT side. Both these diagrams show the interfacing of the surveillance subsystem. The incident information subsystem interfacing is similar to the surveillance subsystem interfacing, except for a few details which would be discussed later in this section.

The system is instantiated by first starting the relevant servers and clients. The DTMS servers started are the, (i) Registry, (ii) Surveillance Server, (iii) Time Message Server, and the (iv) Incident Information server, Each of the servers is started on a specific host (as shown in Appendix A).

MITSIMLab and the TMC Adaptor are initiated using the Simlab Master Controller (SMC)[3]. The master controller initiates MITSIM, TMS and TMCA concurrently. All the three applications can be located on separate hosts. All the three subsystems read their master files at initiation. The master file of the TMC adaptor mentions the server names and the host on which it would look for the specific servers. The TMC Adaptor then initiates the clients corresponding to each of the DTMS servers mentioned above.

[3]A controller implementation that runs the parallel processes through PVM. Referred to as SMC in short.

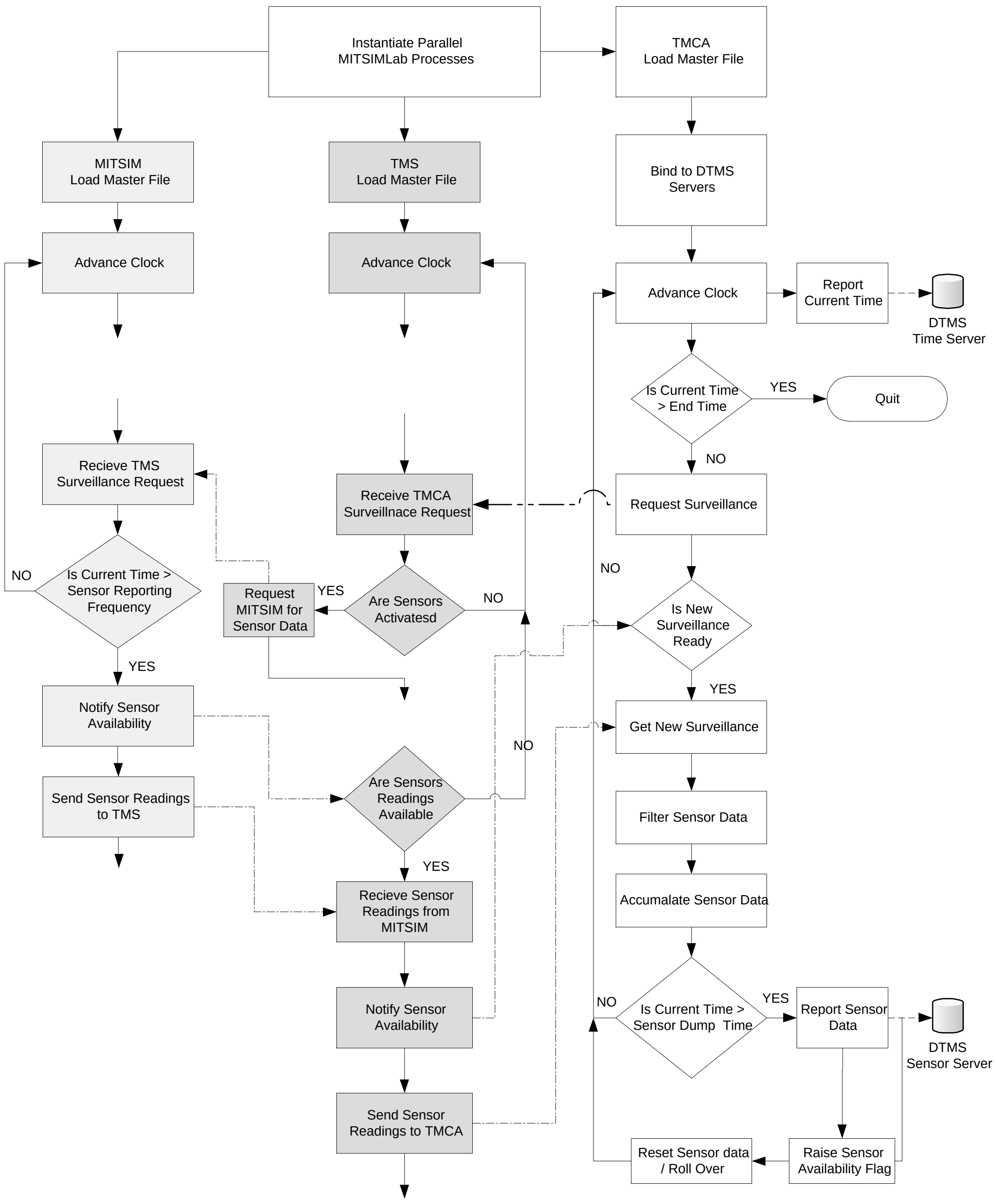

Instantiate Parallel MITSIMLab Processes
TMCA Load Master File
MITSIM Load Master File
TMS Load Master File
Bind to DTMS Servers
Advance Clock
Advance Clock
Advance Clock
Report Current Time
DTMS Time Server
Is Current Time > End Time
YES
Quit
NO
Recieve TMS Surveillance Request
Receive TMCA Surveillnace Request
Request Surveillance
NO
Is Current Time > Sensor Reporting Frequency
NO
Request MITSIM for Sensor Data
YES
Are Sensors Activatesd
NO
Is New Surveillance Ready
YES
YES
Notify Sensor Availability
Get New Surveillance
NO
Are Sensors Readings Available
Send Sensor Readings to TMS
Filter Sensor Data
YES
Recieve Sensor Readings from MITSIM
Accumalate Sensor Data
Notify Sensor Availability
NO
Is Current Time > Sensor Dump Time
YES
Report Sensor Data
DTMS Sensor Server
Send Sensor Readings to TMCA
Reset Sensor data / Roll Over
Raise Sensor Availability Flag

Figure 4-2: Framework for Off-line, Open-loop, Implementation (MITSIMLab-side)

MITSIM is a time-based microscopic simulation model. The step sizes are managed through an internal clock, which manages the time within all MITSIMLab components. The car-following, lane changing, and event and signal response functions are invoked for each vehicle at a specified interval, based on these step sizes (which can be as small as 0.1 seconds). The simulation clock is also advanced based on these step sizes. The clock time is transmitted to TMS and TMCA through PVM. Every time the clock is advanced the TMC adaptor sends the new time message to the DTMS Time Server.

The TMC adaptor, once it gets the new time, checks if it is within its intended time limits. If not, it signals the various operations to close down and kills all the instantiated servers. If it is with the time limits, it sends a message to TMS to request surveillance. TMS looks up if sensors are activated. If they are activated, it sends a message to MITSIM to start recording sesnor data.

Speeds and positions of the vehicles and the states of the sensors are updated at a frequency specified by the user and they are accumulated internally at another specified rate. This accumulation rate is generally based on the frequency at which the controller module expects the sensor counts. Sensors of the same type at the same longitudinal position in the segment are generally grouped into sensor stations. Each sensor station has its sensor type, sensor task, length of detection zone, longitudinal position, and the number of lanes that are equipped. Sensors also have a working probability - to simulate malfunctions and errors (Yang (1997)).

Once the sensor data is accumulated at each reporting step size, TMS is notified about the sensor data availability. TMS in turn notifies the sensor availability to TMC adaptor, which then gets the data from MITSIM. The TMC adaptor then performs the following operations on the data it receives from MITSIM:

(a) *Filters* the data based on the type of information that is required by DynaMIT. DynaMIT only uses information from link-wide sensors whereas MITSIM reports data from different sensor types (e.g. lane-wide, area-wide, link-wide, etc.). The Filtering process extracts information only from the link-wide sensor data. Further MITSIM sensors report information like the vehicle counts, speed, etc. DynaMIT at this time only uses the vehicle counts data. Thus the filter can extract only a specific attribute to transfer to the DTMS server.

(b) *Accumulates* the data based on the estimation time period. As previously mentioned, MITSIM accumulates and reports data based on its internal requirements. But DynaMIT requires aggregated sensor counts for its estimation time period length. The TMC adaptor accomplishes this by accumulating the data reported by MITSIM according to the estimation period length.

Once the processed sensor count data is available to the TMC adaptor, it sends it to the DTMS Surveillance server. It continues sending the data until all the network sensors have reported. It keeps track of all the network sensors through an internal counter mechanism. Once this cycle finishes, it resets all the counters and data values to zero and starts a new cycle.

The DynaMIT communicator is instantiated as one of the DynaMIT server objects (see section, 3.1.4 for more details). At initiation the DynaMIT communicator launches the graphical user interface (Xdta). The DynaMIT communicator reads the data file which mentions the name of the servers it wants to get the surveillance data from and the server it would be providing the guidance to. The data file would also provide the host location of the DTMS servers.

The communicator keeps pinging [4] the DTMS Time server and enquires if a new time message is available. It updates its internal time whenever it finds a new time message in the server. Similarly it continuously keeps enquiring if there is new data on the DTMS surveillance server. It keeps a memory of the last data it had received from the surveillance server and it compares it against available data on the server at each time step. If there is no new data it advances the time clock and keeps waiting for the next available data. If it gets the new data, it appends it to a data file which it will send to DynaMIT at a later time. It continues to get the sensor data until it covers all the registered sensors. Once it has covered all the sensors it initiates the

[4]A standard protocol to test whether the server is alive and available.

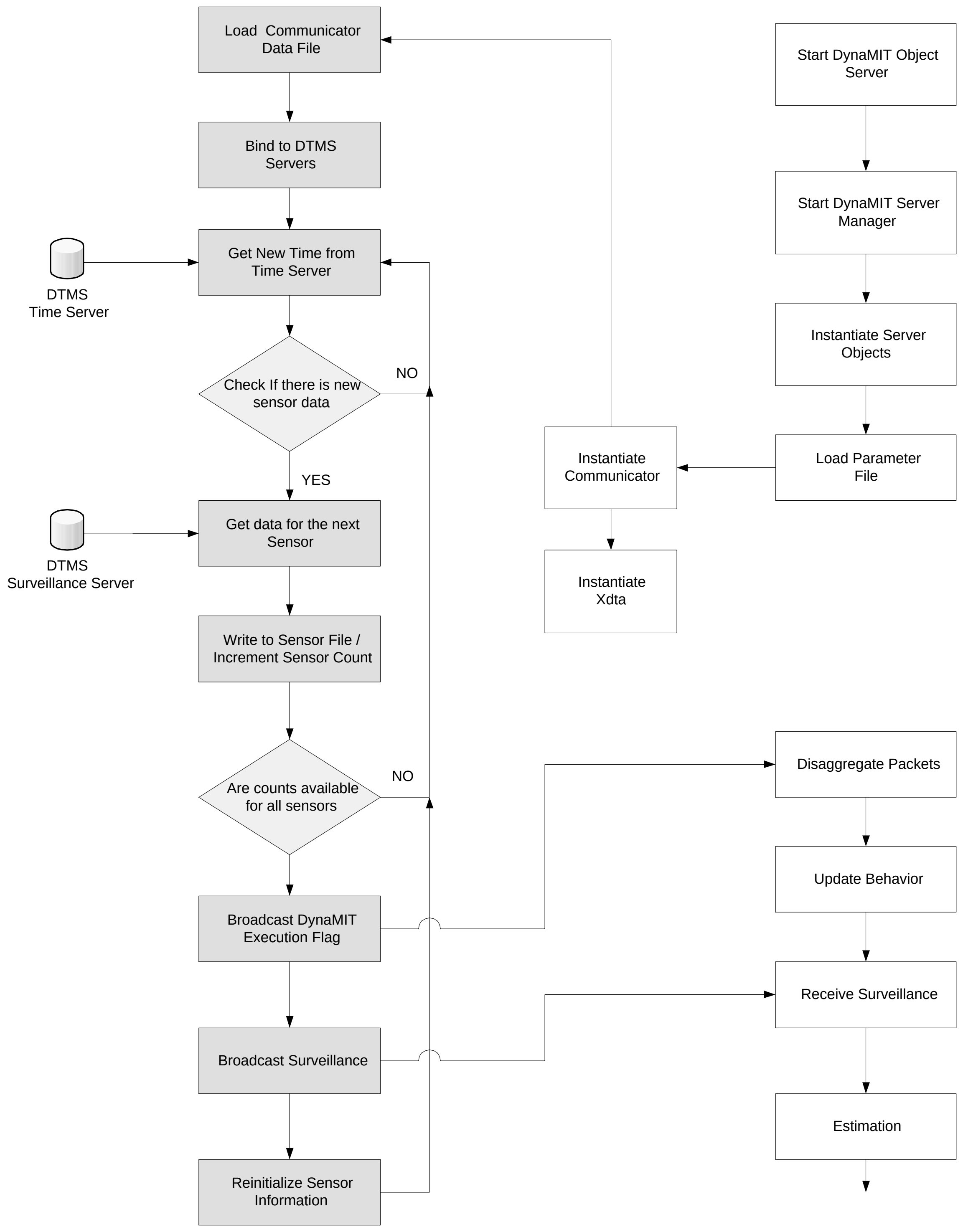

Load Communicator Data File
Bind to DTMS Servers
DTMS Time Server
Get New Time from Time Server
Check If there is new sensor data
NO
YES
DTMS Surveillance Server
Get data for the next Sensor
Write to Sensor File / Increment Sensor Count
Are counts available for all sensors
NO
Broadcast DynaMIT Execution Flag
Broadcast Surveillance
Reinitialize Sensor Information
Start DynaMIT Object Server
Start DynaMIT Server Manager
Instantiate Server Objects
Load Parameter File
Instantiate Communicator
Instantiate Xdta
Disaggregate Packets
Update Behavior
Receive Surveillance
Estimation

Figure 4-3: Framework for Off-line, Open-loop,Implementation (DynaMIT-side)

estimation and prediction processes in DynaMIT and sends the recorded sensor data to the surveillance component. It resets the internal counters once the data has been sent to DyanMIT.

The incident detection and communication proceeds in much the same way as sensor counts. MITSIM detects an incident (which is generally read through a data file) and conveys it to TMS. Each incident may affect one of more lanes, and the MITSIM incident information includes location, number of lanes affected, visibility and equivalent reduction in capacity. The reduction in capacity has to be pre-calibrated. Start time, expected duration and maximum speed with which the vehicles can pass by on adjacent lanes are also reported. The start time of the incident may differ from the time incident was detected by the traffic management system. The clearance time of the incident would generally be the start time plus duration, but it can be changed by the TMS. The incident information is conveyed to the TMC adaptor at the detection time. The TMC adaptor converts the incident into the DynaMIT format that contains, (i) start time, (ii) end time or expected end time, (iii) location, and (iv) capacity reduction factor. As the simulation proceeds this information can get modified (expected end time can change or the severity can be different than anticipated). Thus, the TMC adaptor does not send the incident message to the DTMS Incident Server, until the time DynaMIT is scheduled to start estimation. At this time, it sends the latest available information on the incident to the DTMS Incident server. DynaMIT looks up for this information just before the scheduled estimation start time, and sends it to the Supply module to effectively modify the capacity at the affected locations.

4.1.2 Off-line, Closed-Loop Implementation

The off-line closed-loop evaluation is the extension of the previously described open-loop evaluation. Figure 4-4 shows the closed-loop evaluation framework. This implementation supports the transfer of incident information from MITSIMLab to DynaMIT and the provision of predictive guidance from DynaMIT to MITSIMLab.

Guidance generation in DynaMIT is an iterative process. An important

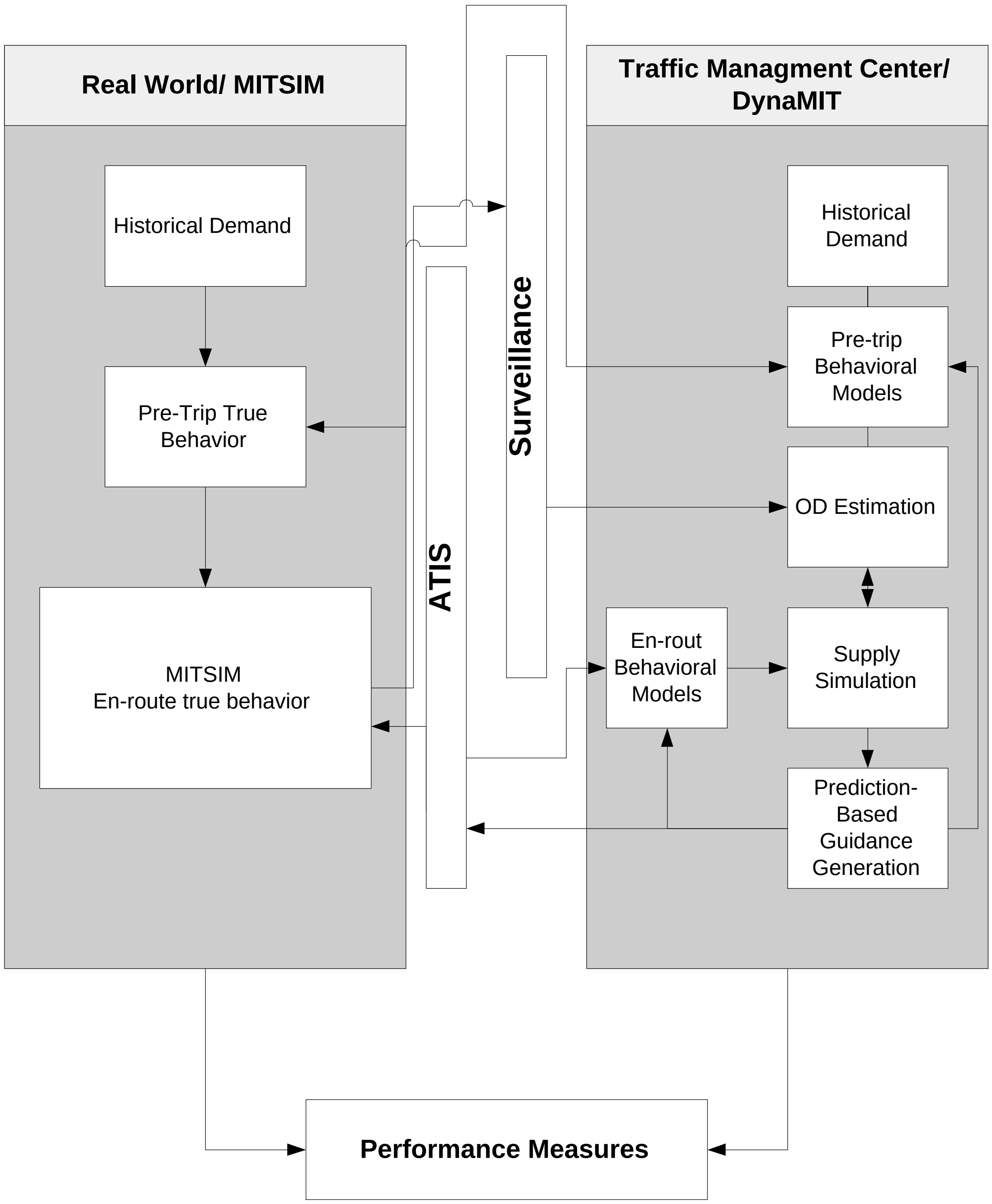


Figure 4-4: Framework for Closed-loop Evaluation

consideration while using DynaMIT for predictive route guidance generation is the issue of consistency. The guidance generated should take into account the response to information of the drivers using the guidance. The control and route-guidance modules in DynaMIT have to be calibrated against the corresponding models in MITSIMLab, to anticipate the response to information.

Closed-loop evaluation can be a very powerful tool for evaluating the system performance before its actual deployment. This is because in many way it replicates the intended end-use of the system. The system can be directly tested for its ability in saving travel times and relieving congestion. Different scenarios can be examined and the analyst can address issues of real-time operability, deployability and effectiveness of guidance strategies.

The issues related to transfer of the surveillance and incident information have already been discussed in section 4.1.1. This section discusses the issues related to the transfer of guidance data from DynaMIT to MITSIMLab, and its deployability. Figure 4-5, shows the overall framework with which the closed-loop, MITSIMLab-DynaMIT interface has been implemented on the DynaMIT side. Figure 4-6, shows a similar diagram of implementation on the MITSIMLab side. Both these diagrams show the interfacing of the guidance subsystem.

After the network state is estimated (using the sensor and incident information sent from MITSIMLab), DynaMIT begins its predictive cycle. Guidance is generated using a iterative algorithm. At each iteration guidance is updated using *time smoothing*:

$$g_{ij}^{k} = \lambda * g_{ij}^{k-1} + (1 - \lambda) * s_{ij}^{k} \qquad (4.1)$$

where:

g_{ij}^{k} = current guidance travel times for iteration k, link i, and link entry time-interval j;

s_{ij}^{k} = simulator output travel times for iteration k, link i, and link entry time-interval j;

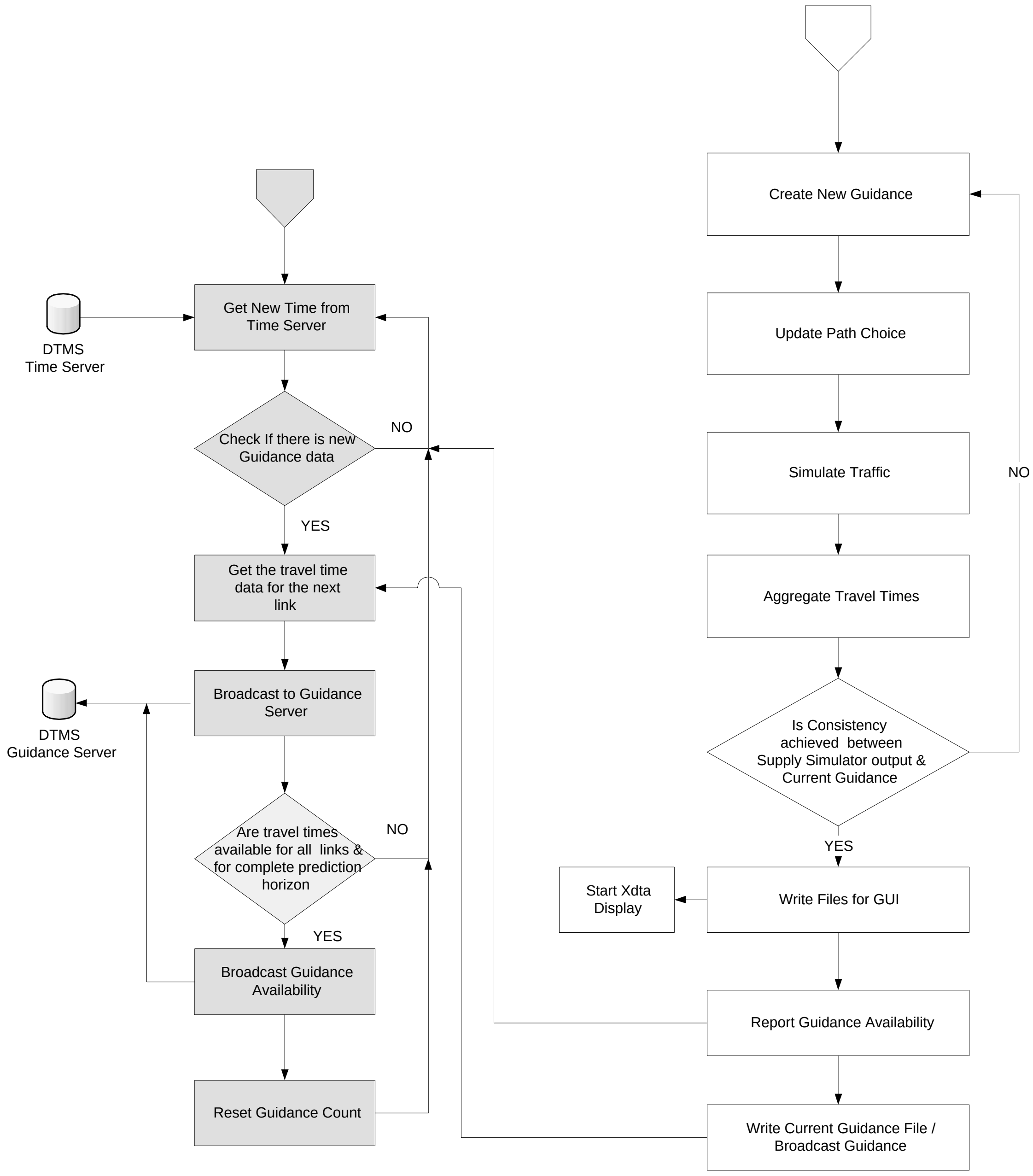

Create New Guidance
Update Path Choice
Simulate Traffic
Aggregate Travel Times
Is Consistency achieved between Supply Simulator output & Current Guidance
NO
YES
Write Files for GUI
Start Xdta Display
Report Guidance Availability
Write Current Guidance File / Broadcast Guidance
DTMS Time Server
Get New Time from Time Server
Check If there is new Guidance data
NO
YES
Get the travel time data for the next link
Broadcast to Guidance Server
DTMS Guidance Server
Are travel times available for all links & for complete prediction horizon
NO
YES
Broadcast Guidance Availability
Reset Guidance Count

Figure 4-5: Framework for Off-line, Closed-loop, Implementation (DynaMIT-side)

$h^0_{ij} =$ historical travel times for link i, and link entry time-interval j;

$\lambda =$ Smoothing Coefficient;

In the first iteration, historical travel times are used. At each iteration, DynaMIT uses the current guidance for an en-route route choice, and simulates the traffic again to obtain new link travel times. It compares these link travel times with the corresponding travel times in the current guidance table. If the difference falls within a pre-specified tolerance limit (or if the pre-specified number of maximum iterations are met), it reports the guidance availability and sends the latest current guidance to the DynaMIT communicator. Equation 4.2, shows the calculation of the consistency measurement.

$$\varepsilon = \frac{1}{nLinks}\sqrt{\sum_i \sum_j (g^k_{ij} - s^k_{ij})^2} \tag{4.2}$$

where:

$\varepsilon =$ Consistency measurement;

$nLinks =$ number of links;

If the consistency is not met, the same process is repeated again. Once the consistency is met, DynaMIT writes speed, flow and density to files and sends them to Xdta for display.

The DynaMIT communicator receives the travel time based guidance data and sends it to the DTMS guidance information server. It keeps a counter on all the links and the prediction horizon time intervals. Once all the links and time intervals are processed, DTMS notifies TMC adaptor about guidance availability and the communicator resets the internal counters.

The TMC adaptor starts the guidance dissemination process by broadcasting a pre-specified *computational delay factor*. DynaMIT uses various computationally intensive algorithm for OD-Estimation and guidance generation. Moreover these models use multiple iterations to generate congruent estimates and consistent predictions. Therefore, there might be significant computational delay before guidance is generated and transmitted to MITSIMLab. Furthermore, DynaMIT

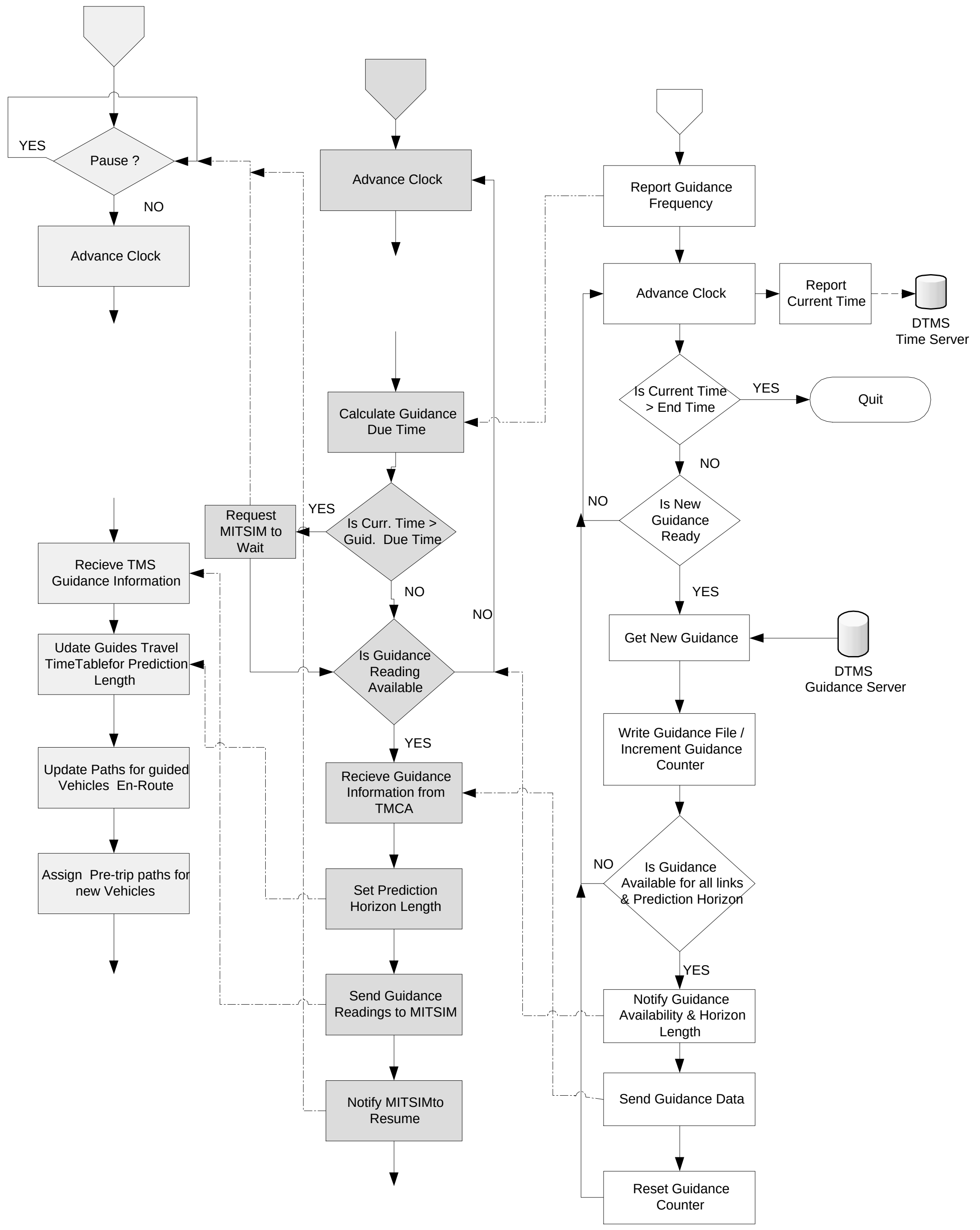

YES
Pause ?
NO
Advance Clock
Advance Clock
Report Guidance Frequency
Advance Clock
Report Current Time
DTMS Time Server
Is Current Time > End Time
YES
Quit
NO
Calculate Guidance Due Time
YES
Is Curr. Time > Guid. Due Time
Request MITSIM to Wait
NO
Is New Guidance Ready
NO
YES
Recieve TMS Guidance Information
NO
Is Guidance Reading Available
NO
Get New Guidance
DTMS Guidance Server
Udate Guides Travel TimeTablefor Prediction Length
YES
Write Guidance File / Increment Guidance Counter
Update Paths for guided Vehicles En-Route
Recieve Guidance Information from TMCA
NO
Is Guidance Available for all links & Prediction Horizon
Assign Pre-trip paths for new Vehicles
Set Prediction Horizon Length
YES
Send Guidance Readings to MITSIM
Notify Guidance Availability & Horizon Length
Send Guidance Data
Notify MITSIMto Resume
Reset Guidance Counter

Figure 4-6: Framework for Off-line, Closed-loop, Implementation (MITSIMLab-side)

has various lever, to control computational time. The computational delay factor facilitates the assessment of trade-offs for using various models and convergence criterion for the various algorithm. The system has the option of defining either the actual delay or a pre-specified delay.

Once the computational delay factor is conveyed to TMS, it calculates an internal variable, *guidance due time*, which is equal to the time at which surveillance was sent to DynaMIT and expected computational delay. As TMS advances its clock, it continuously checks if the current time has exceeded the guidance due time. If it has it directs MITSIM to pause and wait for the guidance. The TMC adaptor continuously checks for new guidance availability on the DTMS guidance server. If there is new data, it gets the new data and sends it to the TMS module. It continues to seek data for all the links and for the entire prediction horizon length. Once it gets the data for all the links, it notifies the TMS of guidance availability and resets its own guidance recording counters. TMS gets the message of guidance availability from TMCA and if MITSIM is paused, it directs it to resume. TMS further sends the new guidance to MITSIM along with the prediction horizon length.

MITSIM uses to two distinct ways to provide route guidance. It can use a *Static Route Guidance* methodology where it calculates an average of the travel times over the entire prediction horizon for each link and then adds all the link travel times on all the available paths.

The other model is the *Dynamic Route Guidance*, which takes into account the drivers current position, destination and projected time-variant link travel times on the alternate paths. The travel time for each link is equal to the travel time when the driver is expected to arrive at that particular link. So the cost experienced by a driver for travelling on a particular path would be:

$$C_i(t) = c_{i0}(t) + c_{i1}(t + c_{i0}(t)) + \dots \tag{4.3}$$

where:

$C_i(t) =$ Travel time on path i, for a vehicle departing at time t;

$c_{ij}(t) =$ link travel time on link j, on path i, for a vehicle coming up at the upstream node of that link at time t;

If the arrival time of a vehicle at a link does not coincide with the discrete intervals which are used to store the time-dependent travel times, interpolation is used to approximate the actual travel time. MITSIM can update the paths in between the reported guidance step size. The update step in that case would be based on *Guidance Resolution*- an input parameter that is provided in the master file Yang (1997). Shortest paths can be recalculated if specified in the master file. Appendix A provides a sample MITSIM master file used for the closed-loop.

MITSIM has two sets of drivers - *Guided* and *Unguided.* The information provided by DynaMIT is only used on guided drivers. The unguided drivers continue to use the historical travel times.

It then uses a *Route Choice Model* or a *Route Switching Model* to generate the probability of a vehicle to chose a particular path (see Yang (1997)).

4.2 On-line Implementation

The on-line implementation of DynaMIT would be in congruence with its intended role in the Traffic Management Centers, with the requirements as laid down in Chapter 2. The integration with the UC Irvine TMCs would provide an excellent opportunity to evaluate the system against real-field data and against real-time implementation.

The overall system architecture for the integration of DynaMIT with the TMCs and with the UC Irvine Testbed has already been specified in section 3.4. This section would discuss the implementation of this architecture in order to achieve our requirements as specified in section 2.3. The discussion would be divided into two parts. The first part will discuss the on-line, open-loop implementation. At the current state of infrastructure and guidance and control subsystem configurations, at the test site, it would not be possible to exactly spell out the implementation framework for the closed-loop. In the absence of a clear-cut knowledge and implementation plan for the closed-loop, we would discuss the adherence of our system configuration with the norms set out in the Interface Compliance Document (ICD). ICD provides a set of common rules and protocols for the design of applications, in order to ease their interfacing with other subsystems operating in the TMC environment. An ICD compliance would ensure convenience in interfacing with the guidance and control subsystems that may finally be installed in the TMCs. The second part will discuss the compliance of the DynaMIT interface with the specification of the ICD. We will discuss the levels on which the designed interface would comply with the features spelled out in ICD. ICD spells out features and requirements that go much beyond the scope of this work. WE would only be concentrating with the issues that concern our study. We would also point out, in the second part, what a possible closed loop implementation may look like.

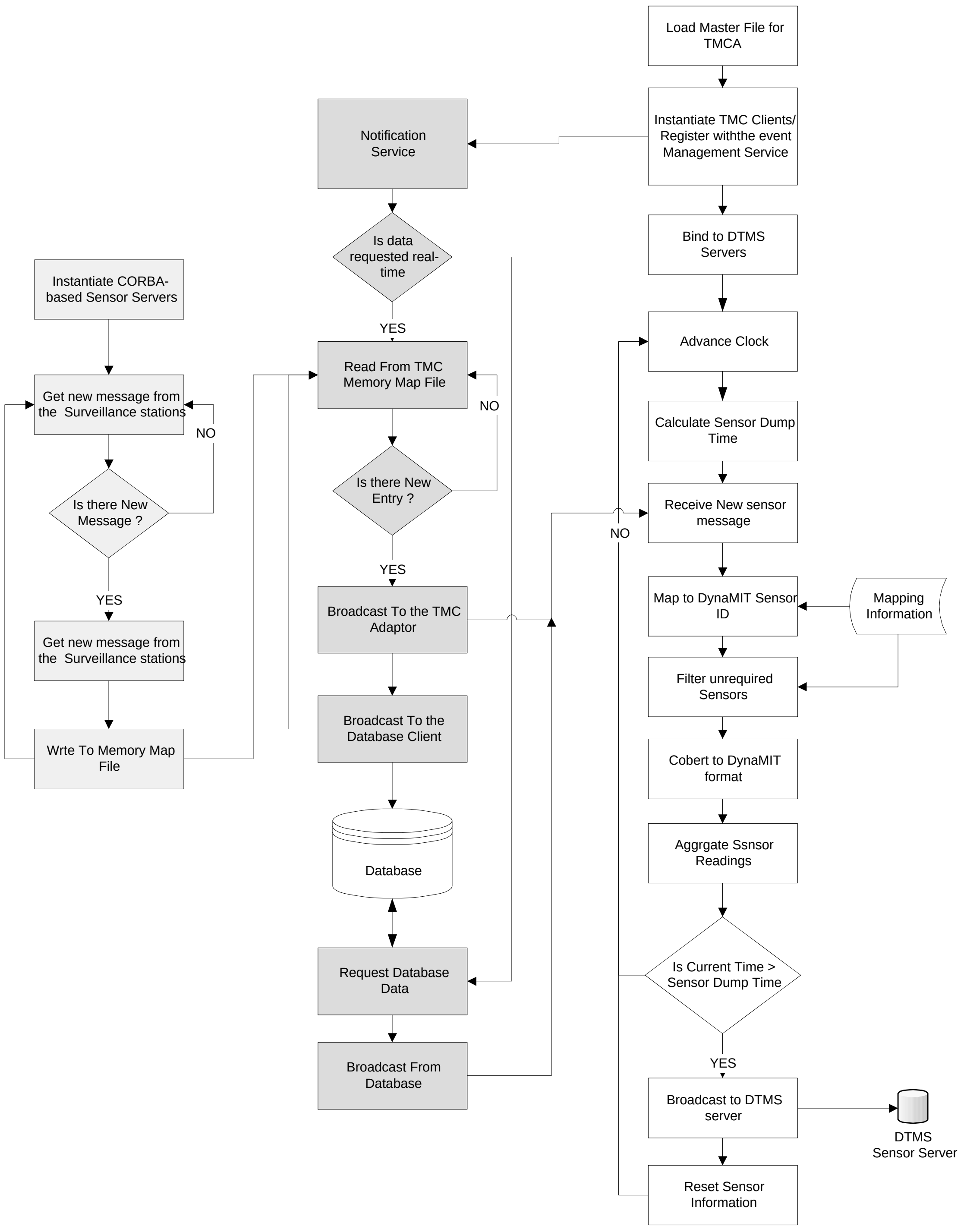
4.2.1 On-line, Open-loop Implementation

Figure 4-7 shows the algorithmic implementation of the designed interface at the Irvine TMC testbed. The figure represents the interface between the TMC and the TMC adaptor.

The TMC adaptor at initiation registers its clients with the Event Notification service at the Traffic Management Centers or at the UCI ATMS laboratories [5].

The real-time data is obtained from the physical devices (sensors) on the field through an interface implemented on the TMCs Front End Processors (FEP). The FEP system includes a dedicated computer that collects data from the traffic controllers in the real world, via modems and interfaces embedded on the TCP/IP network. The FEP acts as a small buffer to store the traffic data collected temporarily. The data is continuously overwritten at regular intervals (depending on the storage space in FEP). The FEP can continuously send out the data stored, if requested by another client host. There is a RECEIVER program that runs on a remote host and continuously requests for the data stored in FEP's RAM. The RECEIVER program converts the data into a human-readable form and dumps it's memory into a disk file.

[5]The UCI ATMS laboratories have access to the TMCs through Orbix or other commercial or public domain CORBA objects



Load Master File for TMCA
Instantiate TMC Clients/ Register withthe event Management Service
Notification Service
Bind to DTMS Servers
Is data requested real-time
Instantiate CORBA-based Sensor Servers
Advance Clock
YES
Read From TMC Memory Map File
Get new message from the Surveillance stations
NO
NO
Calculate Sensor Dump Time
Is there New Entry ?
Is there New Message ?
Receive New sensor message
NO
YES
YES
Map to DynaMIT Sensor ID
Mapping Information
Broadcast To the TMC Adaptor
Get new message from the Surveillance stations
Filter unrequired Sensors
Broadcast To the Database Client
Wrte To Memory Map File
Cobert to DynaMIT format
Aggrgate Ssnsor Readings
Database
Is Current Time > Sensor Dump Time
Request Database Data
Broadcast From Database
YES
Broadcast to DTMS server
DTMS Sensor Server
Reset Sensor Information

Figure 4-7: Framework for On-line, Open-loop,Implementation of DynaMIT)

The CORBA EVENT Notification Service continuously pulls out the data written to the file and pushes it to a Database Client and the TMCA clients registered to it. the database clients puts it in a permanently maintained database (Microsoft Access or Oracle).

The TMC Adaptor Clients can work on two modes. The clients can either be initiated for the real-time data or for the data stored in the database. They bind to the Database Server if they have to get the data from the database. Otherwise they retrieve the data directly through the notification service (which gets it form the memory dump file). The new data received is sent through a set of refining and transformation algorithms.

Firstly the data is *mapped* from their physical location identifications DynaMIT representations. A disk file keeps the mapping between the physical identifications of each sensor with their corresponding IDs in DynaMIT notation. At the start of the TMC adaptor this file is parsed and the one-to-one mappings are stored in the memory. As the sensor readings come in, the DynaMIT ID for each sensor is identified and the original identification of each sensor is replaced by this ID.

A converter algorithm, then *converts* the sensor data format into DynaMIT notations. the sensor data would usually come in a VDS (Vehicle Detection Station) format containing the following fields: (i) date time, (ii) VDS ID, (ii) Loop count, (iv) lane count, (v) Volume, (vi) occupancy, and (vii) status. The converter puts these data elements into the representative DynaMIT format for each data type. In case the data is provided through some source other than the VDS (e.g. raw data or Ramp Metering Station Data), the converter works in similarly to generate a unique format for use by DynaMIT.

DynaMIT does not require all the information transmitted here for doing state estimation. The *filter* algorithm only keeps the information that is relevant and required by DynaMIT.

The vehicle detection station send in data at different intervals (usually 30 seconds, but can be 5 minute, 15 minute or more). The *aggregation* algorithm cumulates these data into DynaMIT-required intervals. For example if DynaMIT would be doing a 15-

minute state estimation, the data is aggregated for fifteen minutes. Once the fifteen minute aggregation is complete, the data is sent to the DTMS sensor servers through previously instantiated DTMS clients.

The DTMS and DynaMIT communicator from there onwards work in the same manner as described in section 4.1.1.

4.2.2 Interface Compliance

The previous section described the open-loop implementation of DynaMIT within the Traffic Management Centers. DynaMIT is principally envisioned as an ATMS support system, residing within the TMCs. But the consistent guidance provided by DynaMIT can be used by many ITS technologies currently being used by the service providers. With such wide scope of applicability of the output of DynaMIT, it would more useful to design a more generic interface for output-consumption (closed-loop) of DynaMIT, rather than a more specific implementation designed to cater only to a particular guidance and control subsystem.

FHWA produced a program plan for evaluation of DTA systems in 1997. This plan contained a provision for an ICD (Interface Compliance Document). Oak Ridge National Laboratories produced a version of this document (Summers and Crutchfield (1999)) that spelled out the various functions that would need to be implemented in a DynaMIT interface that would make the system compliant with the a protocol, making different ITS applications able to access the results produced by DtynaMIT in real-time. This section would discuss the provision of these functionalities in DynaMIT.

Figure 4-8 represents the DTA system interfaces as provided by Summers and Crutchfield (1999) in the ICD. The system architecture provided for interfacing DynaMIT (shown in figure 3-1) has a very similar structure to what has been required in the ICD (shown in figure 4-8).

The DynaMIT Communicator (described in section 3.1.4) would embed all the functions mentioned in ICD, that would be required to achieve the compliance and standardization. DynaMIT Communicator acts as a channel for retrieving and

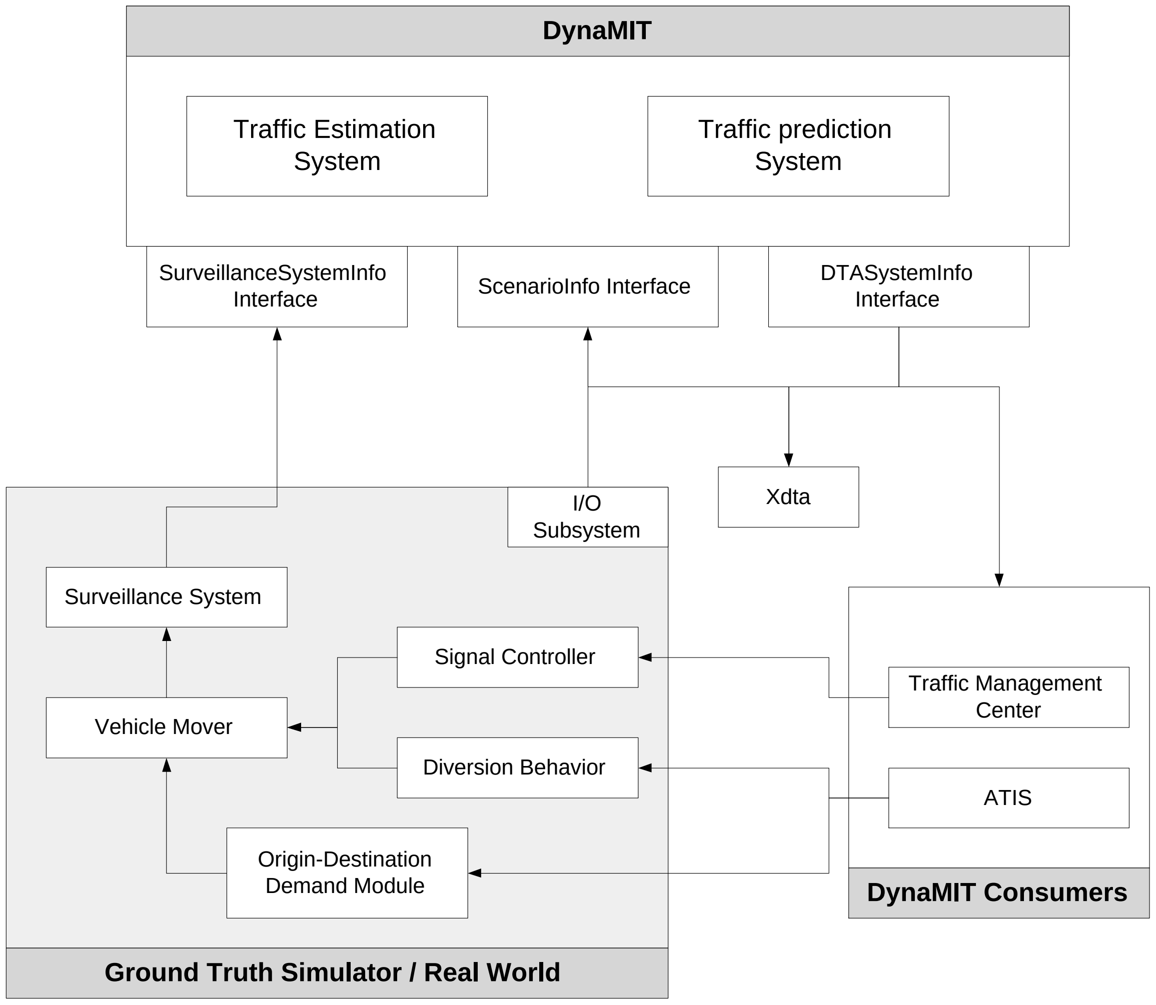

DynaMIT
Traffic Estimation System
Traffic prediction System
SurveillanceSystemInfo Interface
ScenarioInfo Interface
DTASystemInfo Interface
Xdta
I/O Subsystem
Surveillance System
Signal Controller
Vehicle Mover
Diversion Behavior
Origin-Destination Demand Module
Ground Truth Simulator / Real World
Traffic Management Center
ATIS
DynaMIT Consumers

Figure 4-8: DynaMIT system interfaces as specified in ICD (from Summers and Crutchfield (1999))

sending data to all the components and modules within DynaMIT. It has been implemented as a CORBA object and hence has easy access to all the CORBA-based services in the outside world. It will act as a server to which all the clients can bind to retrieve all the required data through standard written functions. We would describe here the standard functions that have been written (as per ICD) in the DynaMIT communicator to retrieve the guidance information through the ICD's DTASystemInfo Interface.

(a) `getComputation` function gets the guidance information for the prediction horizon specified. The calculated guidance information for each prediction horizon is stored in a sequential order. The reference to a horizon would return the dynamic (time interval based) travel times for each link in the network.

(b) `getComputationInfo` function would return the start time, horizon length, interval durations and number of intervals for each specific computation.

(c) `getEarliestRollingHorizonComputationID` function would return the ID for the first set of predicted guidance available.

(d) `getLatestRollingHorizonComputationID` function would return the ID for the latest set of predicted guidance available.

(e) `getODTravelTimeDataByInterval` function would return the static travel time for all the OD pairs in the network as recorded by a particular prediction horizon.

These functional calls can also be used by the guidance and control subsystems to enable a closed-loop implementation.

Chapter 5

Case Study

We have so far discussed the design and implementation of the interface that was required to integrate DynaMIT with the Traffic Management Center and MITSIMLab. We develop a set of requirements necessary for interfacing in Chapter 2 while in Chapter 3 we proposed a system architecture for the interface between DynaMIT and MITSIMLab, and with the TMCs. Chapter 4 discussed the implementation framework for the proposed design. In this chapter we demonstrate the application of our interface through a couple of case studies. The case studies present the interface between DynaMIT and MITSIMLab.

In this section we present two case studies. The first is based on a relatively smaller network for which the models in DynaMIT have been calibrated against the corresponding models in MITSIMLab. This case study gives us the opportunity to demonstrate the strength and viability of DynaMIT-MITSIMLab as a research and a pre-installation evaluation tool. The second case study is on the much bigger Irvine network, demonstrating the robustness of the interface in handling the exchange of larger volumes of data and its scalability to larger applications. This case study is also the first step in DynaMIT's final installation at the Irvine TMC testbed.

5.1 Case Study I - Central Artery/Tunnel Network

The first case study is a MITSIMLab-DynaMIT closed-loop implementation of the Boston's Central Artery/Tunnel (CA/T) Network.

5.1.1 The Network

The Central Artery/Tunnel (see Figure 5-2) is one of the largest construction projects in the U.S. The objective is to provide a replacement for the highly congested elevated Central Artery (I-93). The six-lane elevated highway is being replaced by an eight-to-ten lane underground expressway with a new tunnel beneath the inner harbor to improve access to Logan Airport. I-90 (Massachusetts Turnpike) is also being extended from its current terminal position (south of downtown Boston). The project spans 7.5 miles of highway, 161 lanes miles in all, half of which will be in tunnels. Even with the expected doubling of capacity, the Central Artery is expected to remain fairly congested because of projected high traffic volumes.

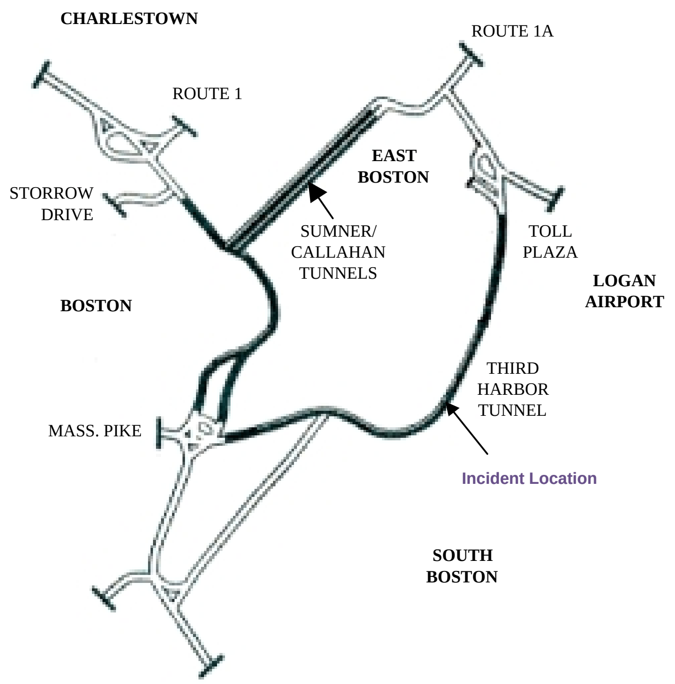


Figure 5-1: The CA/T Network - Incident in Tunnel

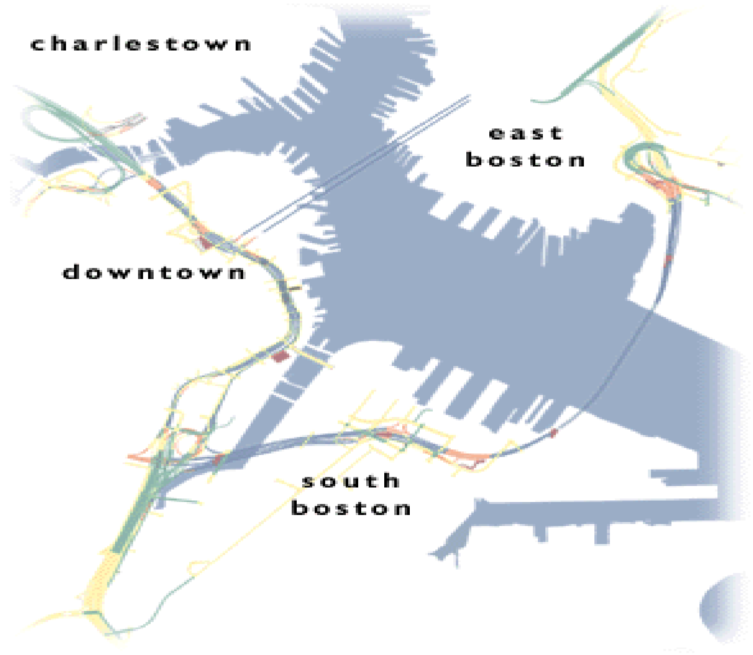


Figure 5-2: The Central Artery / Tunnel Network (source: http://www.bigdig.com)

The network also has four major highway interchanges connecting new roads to the existing highway systems. The multiple interchanges and the circular geometry with many on and off ramps, provide more than one route choices for almost all the OD traffic on the network. The fact that all the latest ITS technologies and elaborate surveillance and control devices are scheduled to be deployed on the network. Hence, the CA/T network is a very useful choice for studying and evaluating dynamic traffic assignment systems.

5.1.2 The Scenarios - Incident in Third Harbor/ Ted Williams Tunnel

The Third Harbor/ Ted Williams tunnel segment of the network is a two-way, four-lane, controlled access toll highway, approximately four miles in length connecting the Massachusetts Turnpike (I-90), Southeast Expressway (I-93), and South Station on the south of harbor, with Logan Airport and Route 1A on the north side. There exist alternate routes for most of the O-D pairs served by the tunnel. This makes this tunnel an ideal candidate to study the effectiveness and applicability of dynamic traffic management systems.

Ben-Akiva et al. (1995) previously used a partial lane-blockage scenario inside the Third Harbor/Ted Williams tunnel to study CO buildup inside the tunnel and conditions that will require the its closure. In this section, we will generate and study two similar scenarios of partial lane-blockage within the Tunnel (Figure 5-1), in order to study the effectiveness of the guidance generation capabilities of DynaMIT, and test its interface with MITSIMLab.

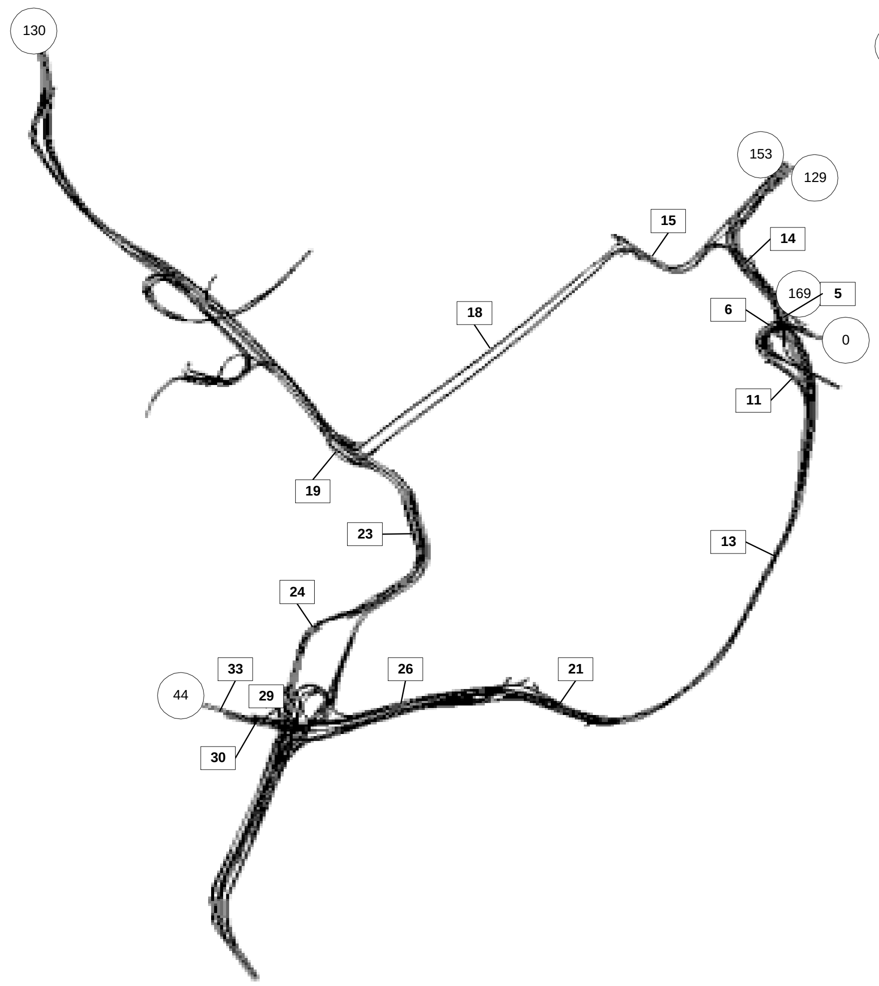


Figure 5-3: The CA/T Network - Representative Sensor Locations

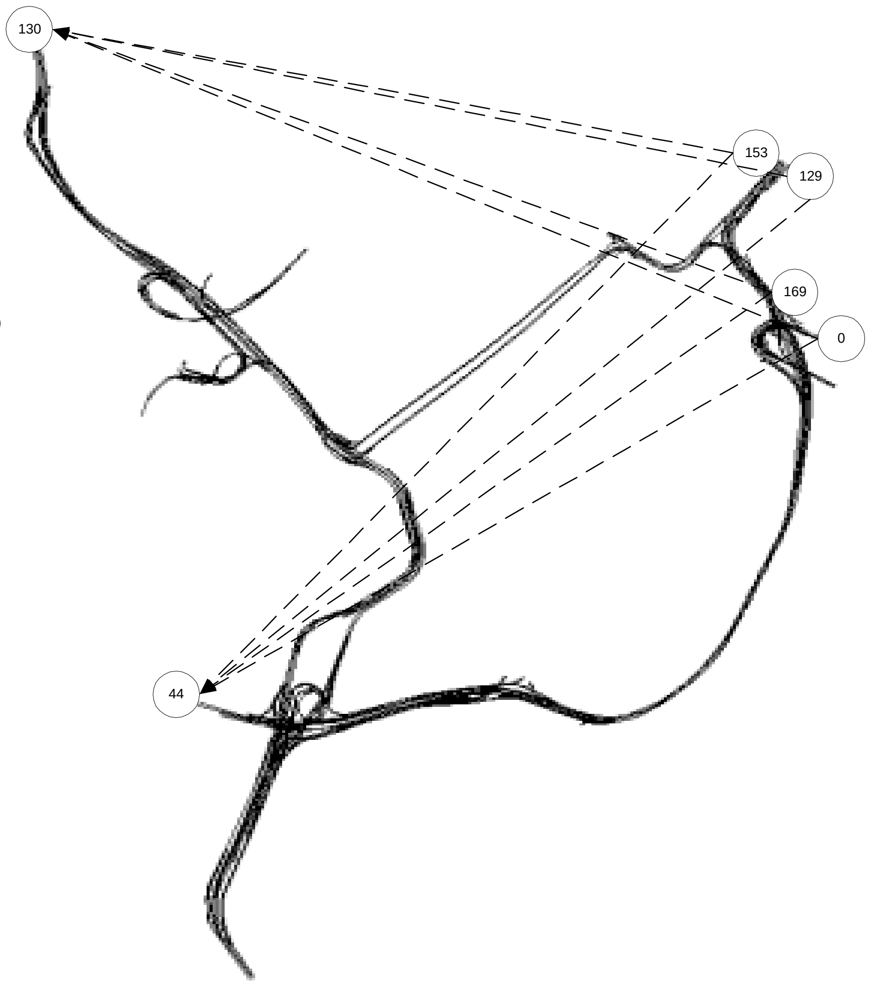


Figure 5-4: OD pairs for the CA/T Network

The incident begins at 7:25 and lasts for 20 minutes. It blocks completely one of the lanes in the tunnel and reduces the speed for the other from 55 mph to 10 mph. We study the following two scenarios:

(a) *No Guidance:* This would be the base scenario when the drivers on the network are provided with no real-tine information or guidance and they follow their habitual routes based on the historical travel times.

(b) *DynaMIT-generated Predictive Guidance:* DynaMIT generates and provides guidance in the form of predicted travel times. The guided drivers chose their routes based on these predicted travel times. The route choice can be both pre-trip as well as enroute. The unguided drivers continue to use their historical routes. The guidance being provided by DynaMIT is in a rolling horizon form, with the horizon length of 30 minutes and guidance updated every 15 minutes. The computational delay is set to 2 minutes.

The simulation starts at 7:15 am and continues in a rolling horizon manner with 15 minute roll-overs (new prediction arrive every 15 minutes). All the other data files and inputs are the same for both the scenarios. Ninety percent of drivers are assumed to be guided and provided with the information (predicted travel times). The high number of guided drivers provides a good test for the importance of prediction. Other studies, usually not based on predictive guidance, indicate that benefits due to ATIS are greatest when informed drivers are around 50% of the total driver population.

5.1.3 Measures of Effectiveness (MOE)

We use the average trip travel times as the measure of effectiveness in this case study. The average trip travel times are compared for a fixed number of vehicles in the network that complete their trips. The travel times are also compared in accordance with the departure times of vehicles. The average trip travel times are also compared between the guidance and no-guidance case for all the OD pairs in the network.

5.1.4 Historical Data and Inputs

The two scenarios had a common set of historical data and common parameter files. The data files used are explained below.

Network File: The network had been coded using the RNE (Road Network Editor), based on the data obtained from the CA/T project management office. The network had 184 nodes, 209 links, 636 segments, 1259 lanes, and 35 sensors.

Historical Link Travel Times: Both MITSIMLab and DynaMIT use time-dependent link travel time to implement the route-choice. The historical travel times are used by the unguided drivers to generate their habitual routes and by the guided drivers for the period in which there is no guidance available. Sam historical travel times were used by both MITSIMLab and DynaMIT in this case study. The historical travel times were generated by running MITSIM in a stand alone mode and recording the travel times experienced by the users.

OD Flow: MITSIMLab, which acts as a representation of the "reality" requires time-dependent OD trip tables file, based on which the vehicles are generated. DynaMIT uses a seed OD-flow file which may be different from the actual OD-flow in the network at the time of simulation. DynaMIT would use the sensor counts generated by the surveillance components in MITSIMLab (which would have the true OD-flow) and the seed OD-matrix to estimate the true OD flow at the time it is called to do the prediction. The historical OD file used in MITSIMLab was an abstract approximation (derived from informal discussions with the researchers and preliminary data assumed by traffic planners) of what might be the true OD flow. The OD trip table used is shown in table 5.1. Figure 5-4 shows the locations of these OD pairs in the network. The correctness of the historical OD-flow would not matter in this study. The closed loop implementation and the respective roles of DynaMIT and MITSIMLab would remain the same. The numerical values of savings in travel times would be more if the ODs which have paths passing through through the blocked link have higher values and the savings would be less if the OD-flow is less than what has been assumed.

Table 5.1: Origin-Destination Flows and Paths for CA/T Network

Origin Node	Destination Node	Hourly Flows	Paths
0	44	1950	S, T
0	130	300	S, T
129	44	135	S, T
129	130	135	S, T
153	44	120	S
153	130	120	S
169	44	15	T
169	130	15	T

S represents a path that goes through the Sumner/Callahan Tunnel;
T represents a path that goes through the Ted Williams Tunnel;

Path Table: It is important that both DynaMIT and MITSIMLab have the same set of paths between all the OD pairs. This would ensure consistency between MITSIMLab and DynaMIT on the number of vehicles choosing a particular path, given the same route choice parameters and link travel times. DynaMIT path topology module was used to generate all the paths between all the simulated OD pairs. The same path table file was used by MITSIMLab.

Supply Parameter File: The DynaMIT uses a mesoscopic supply simulator which needs to be calibrated against MITSIM. The calibrated parameters are stored in the supply parameter file that is used by DynaMIT at execution. The main parameters of interest are capacities and the parameters in the speed-density relationships that capture traffic dynamics. The speed-density relationship used by DynaMIT is of the form:

$$\nu = \min\left[\nu_{min}, \nu_{max}[1 - (\frac{k}{k_{jam}})^{\beta}]^{\alpha}\right] \tag{5.1}$$

where:

$\nu =$ the speed in the segment;

$\nu_{max} =$ the calibrated free flow speed;

$k =$ the segment density;

k_{max} = the calibrated jam density of the segment;

α and β = the calibrated coefficients;

ν_{max} = the calibrated minimum speed;

The parameters were calibrated by recording the speeds, densities and flows from MITSIM for each segment. Segments were grouped into the following categories: (i) Single-lane ramp, (ii) Multiple-lane ramp, (iii) Freeway with on-ramp, (iv) Freeway with off-ramp, (v) Mainline, and (vi) Weaving section.

Socio-Economic Characteristics File: This file stores the characteristics of drivers on the network which are used by the route-choice models. This file does not play any role of importance in our evaluation process.

5.1.5 Route-Choice and Control

Drivers in both DynaMIT and MITSIM have the same set of paths to choose from. The route choice models in DynaMIT and MITSIMLab are discussed next.

The vehicles with paths in MITSIM use a *route switching model* to make route choice. The utility, $V_i(t)$, for a vehicle of choosing path i at time t is given by a multinomial logit model:

$$V_i(t) = \beta \frac{C_i(t) + Z_i}{C_0(t) + Z_0} + \gamma l_i \tag{5.2}$$

where:

β = parameter;

$C_i(t)$ = the expected travel time on path i, at time t ;

Z_i = diversion penalty ;

γ = the commonality factor parameter;

l_i = the commonality factor;

The subscript 0 represents the corresponding values for the shortest path.

The diversion penalty is the additional cost imposed for switching from the current route to an alternate route. The freeway bias is a penalty against taking the off and on-ramp paths. The commonality factor captures the bias generated as a result of overlap between alternate paths. DynaMIT, at this time, only considers the travel times for making route-choice decisions. Thus the parameter for commonality factor, freeway bias and diversion penalty in MITSIM were set to minimize their influence in route-choice decisions.

5.1.6 Results

The results in this section intend to serve two purposes. Firstly they are used to demonstrate the successful working of the interface between MITSIMLab and DynaMIT with real-time exchange of data in a rolling-horizon. Then they show the potential benefits that can be experienced by the integration of dynamic traffic assignment systems like DynaMIT, in Traffic Management Centers.

DynaMIT operates in a step of 15 minutes. It receives the first set of sensor readings at 7:30 and performs a **state estimation** from 7:15 to 7:30. The sensor readings are from 35 link-wide sensors on the CA/T network. Based on the sensor readings, DynaMIT performs (i) an estimate of congruent OD flows, and (ii) an estimate of the network state at 7:30. Based on the network state, it predicts the traffic conditions from 7:30 to 8:00, and generates link travel times (guidance) which takes into account drivers response. The best guidance (in terms of consistency) are sent to MITSIMLab as time-dependent link travel times. Figure 5-5 shows the measure of consistency as a function of the number of iterations. As shown the consistency generally increases with the number of iterations.

MITSIMLab moves vehicles based on this guidance and reports another set of link sensor counts from 7:30 to 7:45. Table 5.2 shows the sensor counts as reported by MITSIMLab for the two scenarios. Under the no-guidance scenario drivers make route choices based on historical travel times. Under the guided scenario and during the period 7:30-7:45, when the guidance that is generated takes into account the occurrence of the incident, the guided vehicles make route choice based on the predicted travel times obtained from DynaMIT. The sensor counts observed are now very different. For example, we observe a decrease in sensor counts as reported by

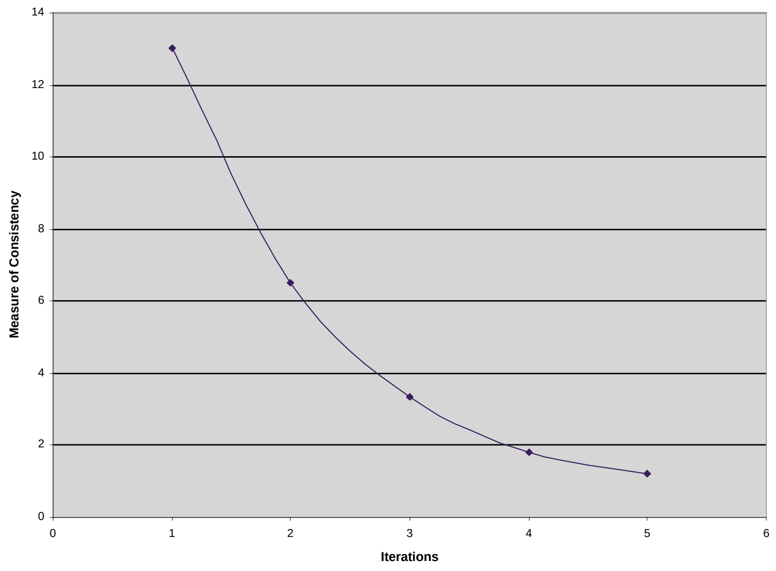


Figure 5-5: Measure of Consistency for Route-Guidance

Sensor ID 6 (placed on the Ted Williams route) and increase in the count on the Sumner/ Callahan tunnel route, as reported by sensor ID 5.

For the first estimation time interval (between 7:15 and 7:30) very little difference between the sensor counts of the two scenarios is observed. This is because during this time conditions are are similar to the historical conditions. The differences are due to the randomly generated departure times and random allocation of paths to the vehicles and the impact of the incident that started at 7:25. The columns of table 5.2 show the observed sensor counts for representative sensors for time interval 7:30 to 7:45. The representative sensor locations are marked in Figure 5-3.

Table 5.2: Comparison of Sensor Counts for No-Guidance and Guidance Scenarios

Sensor ID	No Guidance	Guidance	Change (%)
0	565	559	-1.06
2	65	68	4.62
3	58	59	1.72
4	8	7	-12.50
5	47	184	291.49
6	519	377	-27.36
7	60	95	58.33
8	62	30	-51.61
9	62	95	53.23
11	507	375	-26.04
12	61	32	-47.54
13	265	114	-56.98
14	55	190	245.45
15	121	280	131.40
16	123	284	130.89
17	123	282	129.27
18	123	275	123.58
19	81	160	97.53
20	54	96	77.78
21	219	81	-63.01
22	37	13	-64.86
23	85	161	89.41
24	87	146	67.82
25	53	98	84.91
26	229	95	-58.52
27	40	15	-62.50
28	42	22	-47.62
29	85	143	68.24
30	239	104	-56.49
31	52	89	71.15
32	41	26	-36.59
33	323	251	-22.29
34	96	110	14.58

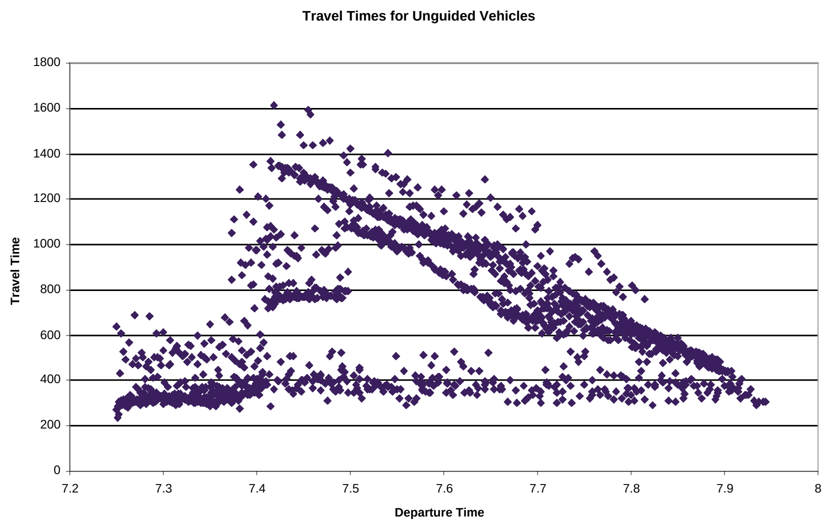


Figure 5-6: Scatter Plot of Travel Times for UnGuided Vehicles

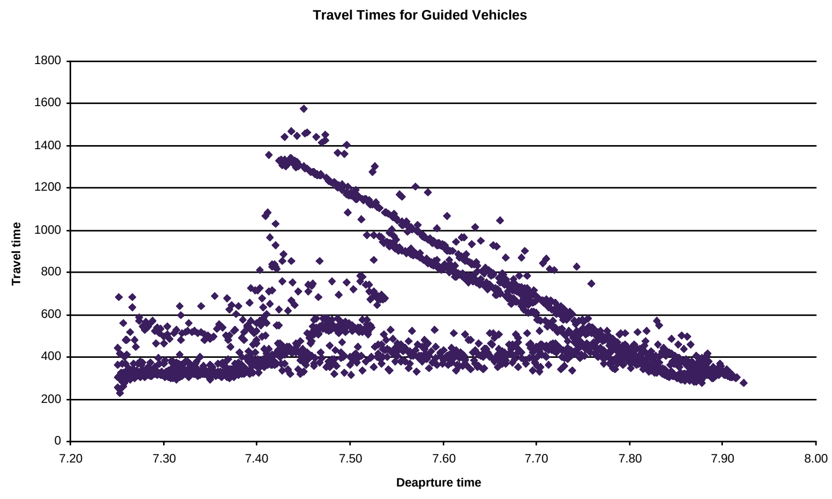


Figure 5-7: Scatter Plot of Travel Times for Guided Vehicles

DynaMIT generates guidance assuming a 30 minutes time horizon. Since the time resolution of the predicted travel times is 1 minute, DynaMIT, through the interface, sends to MITSIMLab a travel-time table of size 208links x 30 minutes. Figures 5-6 and 5-7 illustrate the travel times for vehicles departing between 7:15 and 8:00 for the base case (no-guidance) and guidance scenarios. The travel times increase at 7:25 because of the 20-minute incident. We can see that the unguided scenario has more vehicles which are excessively delayed because of the incident. On the other hand, vehicles the guided scenario experience lower travel times in comparison.

The average travel times for all vehicles (guided and unguided) for the base case and guidance scenarios are shown in figure 5-8. The travel times are compared with the no-incident case. The drivers on the CA/T network take on an average of 6.15 minutes to travel to their destinations without the incident. The travel time increases by 81% to 11.14 minutes because of the 20 minute incident in the Ted Williams Tunnel, if no guidance is provided to the drivers. However, with the predictive guidance from DynaMIT the travel time only increases by 45% to 8.9 minutes. The predictive guidance from DynaMIT produces a savings of about 20%.

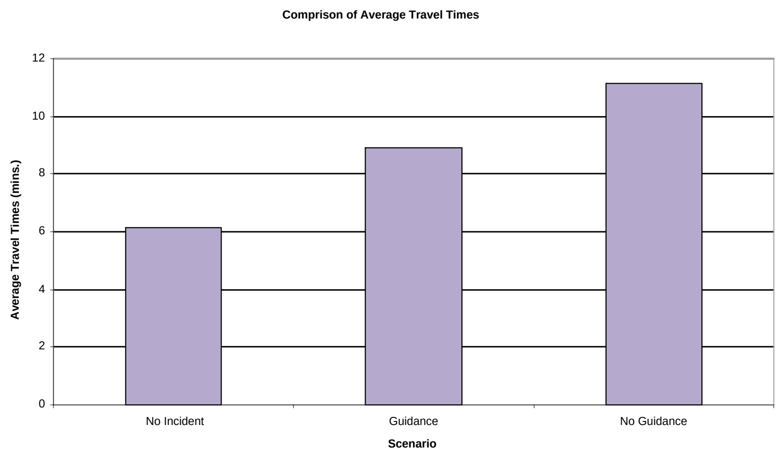


Figure 5-8: Comparison of Average Travel Times for All Vehicles

Figure 5-9 compares the average travel times for the guided and no-guidance scenarios, as a function of departure times. The incident begins at 7:25 and ends at 7:45. We see that travel times are comparable for vehicles departing before 7:20 (note

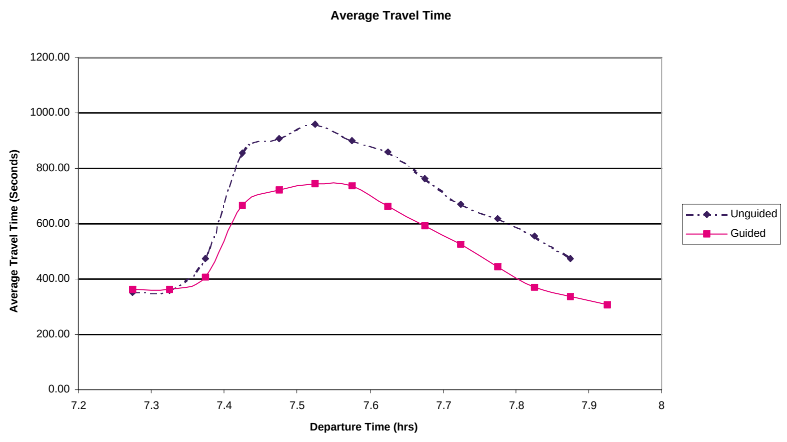


Figure 5-9: Comparison of Average Travel Times as a function of Departure Times

Table 5.3: Comparison of Average Travel Times for OD Pairs

OD Pair	No Guidance	Guidance	Change (%)
0 - 44	684.97	536.3	-21.6
0 - 130	783.76	609.58	-22.2
129 - 44	638.9	590.32	-7.6
129 - 130	862.2	551.35	-36.05
153 - 44	362.02	385.51	6.4
153 - 130	369.8	382.6	3.4
169 - 44	384.1	371.5	-3.2
169 - 130	403.37	401.37	-0.49

that it takes about 5 minutes for vehicles to reach the incident location from the main origin). The vehicles departing later than this time get caught in the incident before they reach their destinations. The no-guidance scenario shows a steeper slope than the guidance scenario. Also the average travel times remain higher for all departure times during the incident time. Even after the incident is cleared (at 7:45), the vehicles, caught up on the Ted Williams tunnel route, delay the vehicles departing at later time periods and hence average travel times in the base case are higher.

Table 5.3 gives the average travel times for all the OD pairs, for the two scenarios. Although, on average there are travel time savings when guidance is provided, the savings are not uniform across all OD pairs. For some of the OD pairs actually the travel time increases. OD pairs 153 - 44 and 153- 130, for example, only have one path that goes through the Sumner/ Callahan Tunnel. We observe an increase in their travel times (see Table 5.3). This can be explained by the increase in congestion on the Callahan Tunnel route because a number of guided drivers, who previously used the Ted Williams Tunnel route, are now diverted to this route as a result of the predictive guidance from DynaMIT. OD pairs 0 - 130 and 129 - 130 experience the biggest gains, 22% and 36% respectively. These pairs have paths that use the Ted Williams tunnel and the incident directly affects them. As a result, the guided drivers gain significant savings by taking the other routes. Unguided drivers and those of the guided drivers who remain on their original paths through the tunnel also benefit, since due to the reduced demand, the impact of the incident is minimized.

5.2 Case Study II - The Irvine Network

This case study demonstrates the successful applicability of the interface and its robustness for larger networks. The Irvine network is currently being calibrated, and hence it would be futile to evaluate state estimation and traffic prediction results. Nevertheless, this demonstration shows the applicability and ability of the system to tackle large-scale, real-life problems.

5.2.1 The Network

The Irvine network is shown in figure 5-11. It consists of two major interstates I-5 and I -405, which intersect on the south end and diverge towards the north. The two interstates are crossed by three state highways - 55, 261 and 133. Anaheim is located further north. The network also includes numerous arterials and local streets, some of which (like Baranca) carry significant amount of traffic. The network contains many traffic actuated signals and is equipped with multiple surveillance stations and information sources. The network experiences significant traffic delays during the morning and evening peak hours as people travel to the central business districts of Irvine and Anaheim. The University of California, Irvine which coordinates the surveillance collection and dispersion efforts for the testbed, is located in close proximity towards the west side.

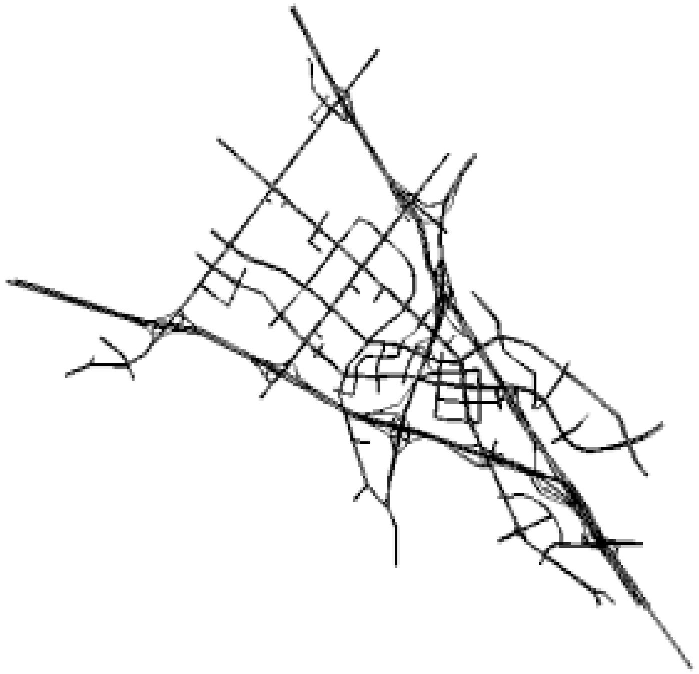


Figure 5-10: The Coded Irvine Network

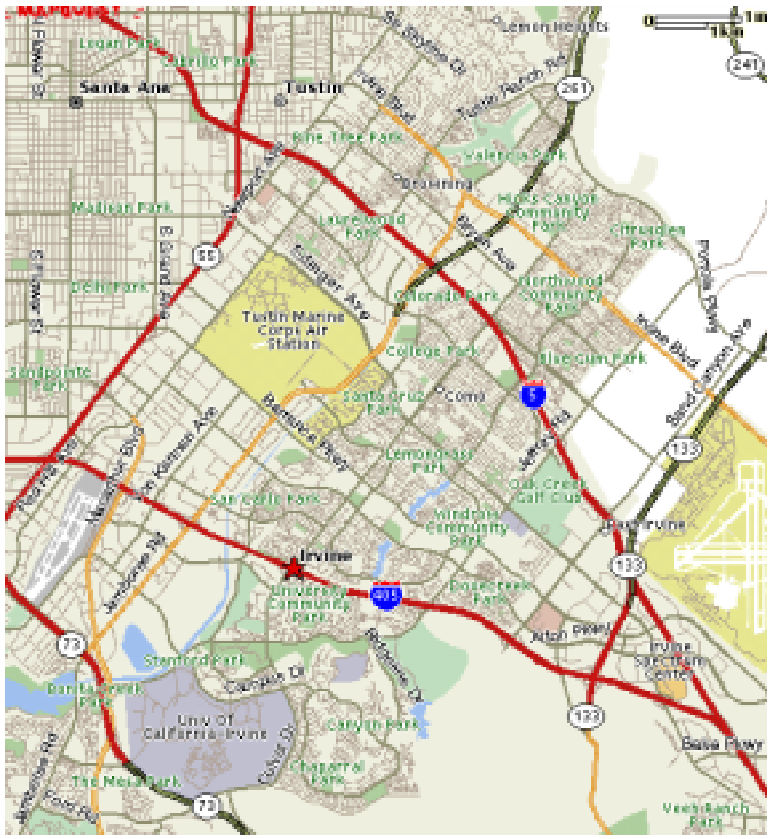


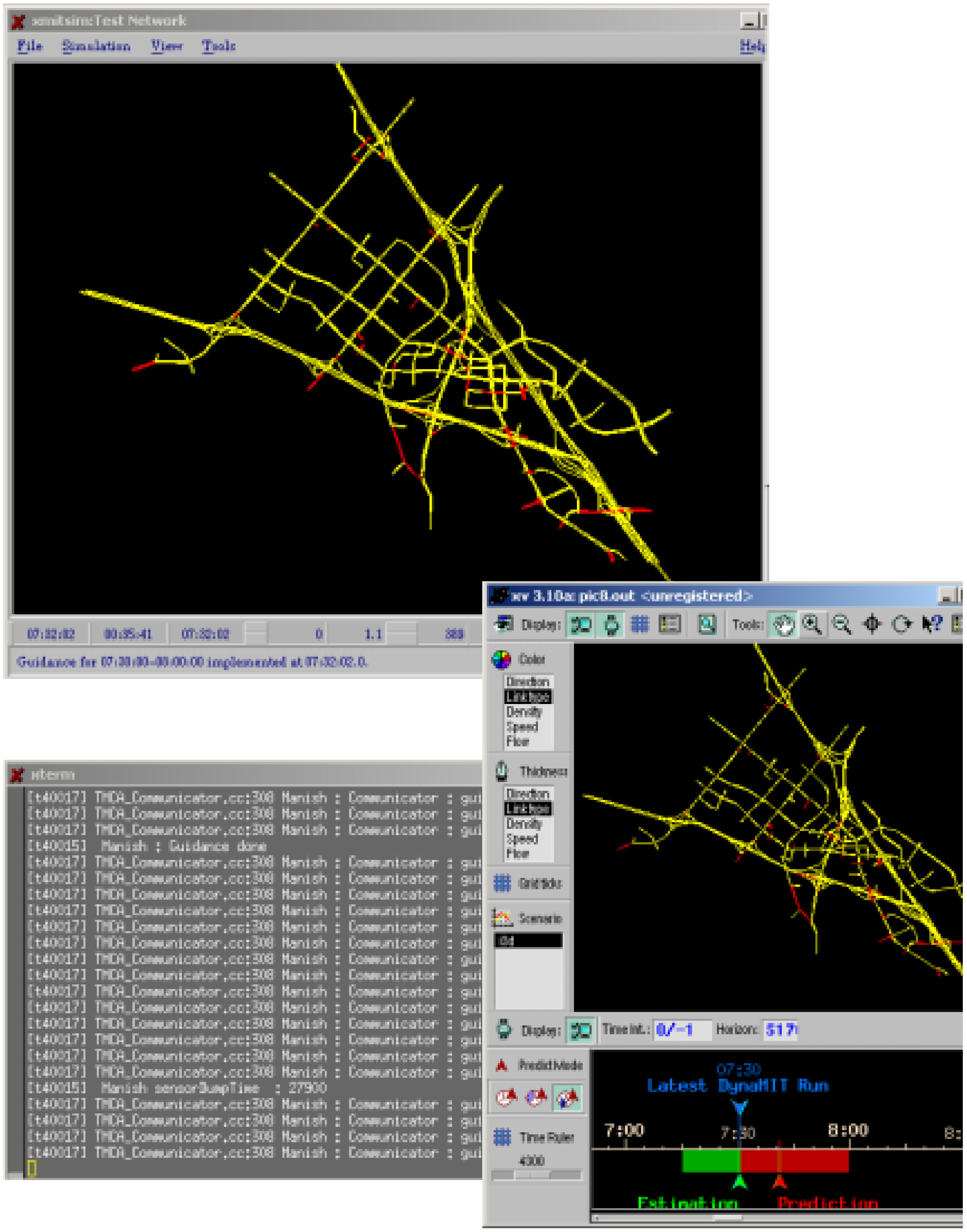
Figure 5-11: The Irvine Network (source: http://www.mapquest.com)

The network consists of 296 nodes, 618 links, 1373 segments, and 3524 lanes. There are about 219 link-wide sensors which are used for the collection of traffic counts, besides other lane specific sensors that are used to time the signals at the intersections. The coded network is shown in figure 5-10. The network consists of 622 OD pairs.

5.2.2 The Interface

The MITSIMLab-DynaMIT interface was tested on the Irvine network. A 15-minute estimation period and a 30-minute prediction horizon were used for the test. In addition to the various functionalities tested with the case study using the CA/T network, several additional features of the interface were tested using the Irvine network. The most important of these are related to the sensors and their operations. Sensors in the Irvine network are used, either for counts, or for activating traffic lights at intersections. Because of the traffic actuated signals, MITSIM needs sensor readings every 1-second. MITSIMLab was thus reporting signal readings for about 500 sensors every 1-second. Hence, the *aggregation* algorithm, in the TMC adaptor was used to accumulate the sensor data for 15-minute intervals, as needed by DynaMIT. Also DynaMIT, only used information from the sensors providing link-wide counts. Thus the filtering algorithm was used to extract information from link-wide sensors. In this case study, there was no need to use the sensor *mapping* or *conversion* utilities, since both DynaMIT and MITSIMLab use the same sensor IDs and data format.

DynaMIT supplied the guidance information for a period of 30 minutes. The guidance consisted of the dynamic link travel times in one minute intervals. The guidance data thus had a size of 618 x 30 which was transmitted through the CORBA-based communication architecture to the TMC adaptor which further sent it to the appropriate MITSIMLab module using the PVM communication infrastructure of MITSIMLab. Figure 5-12 gives a snapshot of the operation of DynaMIT within the Traffic Management Center Simulator of MITSIMLab, for the Irvine case study.



```
07:32:02    00:35:41    07:32:02        0     1.1     388
Guidance for 07:30:00-08:00:00 implemented at 07:32:02.0.
```

```
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40015]  Manish : Guidance done
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40015]  Manish sensorDumpTime  : 27900
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
[t40017] TMCA_Communicator.cc:308 Manish : Communicator : gui
```

Figure 5-12: A snapshot of the DynaMIT-MITSIMLab closed-loop implementation on the Irvine Network

Chapter 6

Conclusions

6.1 Research Contribution

The contribution of this research spans across four basic areas: (a) design of a generic interface that integrates DynaMIT in Traffic Management Centers, (b) implementation of the interface design for integrating DynaMIT with the Traffic Management Simulator (TMS) within MITSIMLab and the application of the design for integrating DynaMIT with the Irvine TMC, (c) design and implementation of DynaMIT-Communicator, an open Communication Interface for DynaMIT, making it accessible to various ITS applications and making it ICD compliant, and (d)demonstration of the functionality of the interface and the associated system with two different transportation networks.

The interface design extends, modifies and improves the architecture suggested by Ruiz (2000). It has been designed as a client-server application, using a basic push/pull model. The exchange of data takes place between two subsystem clients which bind to the same server. The application has been distributed using the Common Object Request Broker Architecture. The interface has been designed in multiple modules in order to provide a more generic design that is flexible enough to be adopted to different TMCs with minimal changes which would be concentrated on only one of the modules. The other modules remain unaltered. The design supports application and execution of multiple systems in parallel, and allows concurrent execution of different processes in DynaMIT as well TMC subsystems, without stopping or waiting for the completion of the other. Finally, the interface design leaves ample flexibility to further enhance the functionality of the system by adding more utilities and applications.

The proposed architecture was used to implement the interface for the offline evaluation system (DynaMIT-MITSIMLab) as well as for the online integration of DynMIT with the Traffic Management Center at Irvine. The offline evaluation system can be a very powerful tool for analyzing the effectiveness of DynaMIT and other dynamic traffic assignment systems. The evaluation tool also provides numerous methods for studying the different characteristics of traffic networks and observing results form applying different routing and control strategies. The offline closed-loop system also provides a tool for evaluating and observing the computational performance of various algorithms and models that have been applied for the first time in DynaMIT. The implementation framework for integrating DynaMIT with the Irvine TMC provides opportunity to evaluate the system with data received from the network in real-time. Moreover, it provides useful insights into the final deployability of the system within TMCs.

Since the deployment of systems like DynaMIT can potentially be useful to many other ITS technologies and functions within TMCs, FHWA has proposed some form of standardization in the interface implementation of DynaMIT, through the Interface Compliance Document (ICD). ICD provides a common set of function calls that various technologies can use to obtain prediction data from DynaMIT. The DynaMIT communicator interface acts as a channel for any outside communication to and from DynaMIT. These function calls have been implemented in the DynaMIT communicator which makes DynaMIT ICD compliant from the DTASystemInfo (Summers and Crutchfield (1999)) point of view.

Finally, the MITSIMLab-DynaMIT system was applied on two case studies involving two different networks - the CA/T network and the Irvine network. In the CA/T case study, we obtained results for two scenarios under incident conditions - no-guidance and predictive guidance. The results indicated the potential benefits of real-time application of DynaMIT within a TMC, and the use of the interface in evaluating the performance of DynaMIT. The second case study on the Irvine network showed the robustness of the system to operate with large networks, under realistic operating conditions.

6.2 Future Work

The research presented in this thesis can in future be extended in the following directions:

Detailed Methodology for the Evaluation of DynaMIT

Ben-Akiva et al. (1995) had previously presented a framework for the evaluation of dynamic traffic management systems. The interface implementation provides an opportunity to extend this framework into a detailed evaluation methodology for on-line and off-line evaluation of DynaMIT. The evaluation methodology can make use of the several features and attributes of the interface, as presented in this thesis.

Design Refinement and optimization of DynaMIT algorithms and models

DynaMIT models can be tested for their efficacy and efficiency using alternate scenarios (one example of which was presented in case study I in the last chapter). The interface provides an opportunity to immediately observe (through MITSIMLab) the results of the application of these scenarios. Researchers can analyze the relationships between the systems input parameters and the observed outputs or response from drivers in MITSIMLab. The input-response relationships can be studied to identify optimal design parameters. These relationships can also give the sensitivity of the results to change in the value of these parameters. Various design refinements and model optimization can be pursued in DynaMIT based on these results.

Application of the Interface to multiple TMC configurations

The interface can be further applied to different TMC configurations. The TMC configuration can be in the form of a centralized system covering single region or a centralized system covering multiple regions.It can also be of the form of a decentralized region (Ruiz (2000); Ben-Akiva et al. (1994b)). The interface can be extended to the different TMC designs by implementing the appropriate refinements in the TMC adaptor.

Interfacing with other ITS technologies in the TMCs

The ICD-compliance of DynaMIT achieved by implementing the appropriate functions in the DynaMIT communicator part of the interface, can allow DynaMIT integration with other ITS technologies and functions within the TMCs, which may require information on the predicted traffic conditions. Therefore, integration of DynaMIT with these technologies can be a logical next step in the near future.

Interfacing for Parallel Simulation

Various researchers (Jha et al. (1995); Yang (1997); Junchaya et al. (1992)) have talked about the possibility of parallelizing traffic simulation. Parallel simulation would decompose a large network into several smaller subnetworks, and simulate simultaneously on multiple processors. This can decrease the computational time tremendously, when simulating larger networks. The interfacing of DynaMIT would have to modified to incorporate parallel simulation. The exchange of data in this case would be between numerous subsystems and small-network simulations. The interface would be require to associate each data-type with the subsystem and simulation it belongs to. The interfacing of for parallel simulation, though, can still be built on the concepts introduced and implemented in this thesis.

Bibliography

Jaime Barcelo and Jaime L. Ferrer. A simulation study for an area of Dublin using the AIMSUN2 traffic simulator. Technical report, Department of Statistics and Operation Research, Universitat Politecnica de Catalunya, Spain, 1995.

M. Ben-Akiva and S. R. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, 1985.

M. E. Ben-Akiva, M. Bierlaire, J. Bottom, H. N. Koutsopoulos, and R. G. Mishalani. Development of a route guidance generation system for real-time application, 1997. Presented at 8th IFAC Symposium on Transportation Systems.

M. E. Ben-Akiva, M. Bierlaire, H. N. Koutsopoulos, and R. G. Mishalani. DynaMIT: Dynamic network assignment for the management of information to travelers. In *Proceedings of the 4th Meeting of the EURO Working Group on Transportation*, Newcastle, 1996a.

M. E. Ben-Akiva, H. N. Koutsopoulos, R. G. Mishalani, and Q. Yang. Integrated simulation framework for evaluating dynamic traffic management systems. In *Proceedings of the First World Congress on Applications of Transport Telematics and Intelligent Transportation Systems*, Paris, 1994a.

Moshe E. Ben-Akiva, Anthony F. Hotz, Rabi. G. Mishalani, and Nageswar R. V. Jonnalagadda. A design-evaluation framework for dynamic traffic management systems using simulation. In *WCTR Proceedings*, Sydney, Australia, 1995.

Moshe E. Ben-Akiva, Haris N. Koutsopoulos, Rabi G. Mishalani, and Qi Yang. Simulation laboratory for evaluating dynamic traffic management systems. *ASCE Journal in Transportation Engineering*, 1996b. Accepted for publication.

Moshe E. Ben-Akiva, Haris N. Koutsopoulos, and Anil Mukandan. A dynamic traffic model system for ATMS/ATIS operations. *IVHS Journal*, 1(4), 1994b.

J. Bottom, M. Ben-Akiva, M. Bierlaire, and Chabini I. Generation of consistent anticipatory route guidance, 1998. Presented at TRISTAN III Symposium.

Syd Bowcott. The ADVANCE project. In Sam Yagar and Alberto Santiago, editors, *Large Urban Systems – Proceedings of the Advanced Traffic Management Conference*, pages 59–70, St. Peterburg, Florida, 1993.

Ennio Cascetta, Domenico Inaudi, and Gerald Marquis. Dynamic estimators of origin-destination matrices using traffic counts. *Transportation Science*, 27(4):363–373, 1993.

Huey Kuo Chen and Che Fu Hsueh. Combining signal timing plan and dynamic traffic assignment. 76th Transportation Research Board Annual Meeting, 1997.

Charles Donnelly and Richard Stallman. *Bison – The YACC-compatible parser generator*. the Free Software Foundation, Cambridge, MA, 1992.

DYNA. DYNA – A dynamic traffic model for real-time applications – DRIVE II project. Annual review reports and deliverables, Commision of the European Communities - R&D programme telematics system in the area of transport, 1992-1995.

FHWA. DTA RFP. Technical report, Federal Highway Administration, US-DOT, McLean, Virginia, 1995.

FHWA. CORSIM User Guide. Technical Report Version 1.0, Federal Highway Administration, US-DOT, McLean, Virginia, 1996.

FHWA. A roadmap for the research, development and deployment of traffic estimation and prediction systems for real-time and off-line applications. Technical report, Federal Highway Administration, US-DOT, McLean, Virginia, 2000.

Loral AeroSys FHWA. Traffic management center - the state-of-the-practice. Task A: Final Working Paper for Design of Support Systems for ATMS DTFH61-92C-00073, U.S. Department of Transportation - Federal Highway Administration, 1993.

Nathan H. Gartner and Chronis Stamatiadis. Integration of dynamic traffic assignment with real time traffic adaptive control. 76th Transportation Research Board Annual Meeting, 1997.

A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine – A users' guide and tutorial for networked parallel computing*. The MIT Press, Cambridge, MA, 1994.

Hague Consulting Group. What happens: European trials of anticipatory traffic control. *Traffic Technology International*, 8:64–68, 1997.

Kevin A. Haboian. A case for freeway mainline metering. 74th Transportation Research Board Annual Meeting, Washington, D.C., 1995.

Masroor Hasan. Evaluation of ramp control algorithms using a microscopic traffic simulation laboratory, mitsim. Master's thesis, Massachusetts Institute of Technology, 1999.

Mithilesh Jha, Anupam Joshi, and Kumares Sinha. A framework for dynamic traffic simulation on distributed systems. In *Proceedings of applications of advanced technologies in transportation*, Capri, Italy, 1995.

T. Junchaya, G.L. Chang, and A. Santiago. ATMS: Real-time network traffic simulation methodology with a massively parallel computing architecture. In *Proceedings of 71st TRB Annual Meeting*, Washington, D.C., 1992.

Ashok Kalidas. *Estimation and prediction of time-dependent origin-destination flows.* PhD thesis, Massachusetts Inst. of Tech., Cambridge, MA, 1996.

Ashok Kalidas, M. Ben-Akiva, M. Bierlaire, A. Chachich, A. Hotz, H.N. Koutsopoulos, R. Mishalani, and Q. Yang. Tools for design and operation of dynamic traffic management systems. *Third Annual World Congress on Intelligent Transport Systems*, 1997.

I. Kaysi, Moshe E. Ben-Akiva, and Haris N. Koutsopoulos. Integrated approach to vehicle routing and congestion prediction for real-time driver guidance. *Transporation Research Record*, 1408, 1993.

H.N. Koutsopoulos, T. Lotan, and Q. Yang. A driver simulator for data collection and ITS application to the route choice problem. *Transportation Research C*, 2C (2), 1994.

John R. Levine, Tony Mason, and Doug Brown. *Lex & Yacc.* O'Reilly & Assoc, Cambridge, MA, 2nd edition, 1992.

Hani S. Mahmassani, Ta-Yin Hu, Sriniva Peeta, and Athanasios Ziliaskopoulos. Development and testing of dynamic traffic assignment and simulation procedures for ATIS/ATMS applications. Report DTFH61-90-R-00074-FG, U.S. DOT, Federal Highway Administration, McLean, Virgina, 1994.

M. McNally. Evaluation of the anaheim advanced traffic control system field operational test. Technical Report UCB-ITS-PRR-99-18, University of California, Berkeley, 1999.

F. Middelham, W. J. Schouten, J. Chrisoulakis, M. Papageorgiou, and H. Haj-Salem. Eurocor and a10-west – coordinated ramp metering near amsterdam. In *Proceedings of the First World Congress on Applications of Transport Telematics & Intelligent Vehicle-Highway Systems*, pages 1158–1165, Paris, France, 1994.

M. Miller. Travinfo evaluation traveler information ceneter study. Technical Report UCB-ITS-PWP-98-21, University of California, Berkeley, 1998.

MIT. Development of a deployable real-time dynamic traffic assignment system. Technical Report Task B-C, Massachusetts Inst. of Tech., Intelligent Transportation Systems Program and Center for Transportation Studies, Cambridge, MA, 1996. Interim reports submitted to the Oak Ridge National Laboratory.

Robert Orfali and Dan Harkey. *Client/Server Programming with Java and CORBA*. Wiley, John & Sons, 1998.

Harold J. Payne. Freeway traffic control and surveillance model. *Trasportation Engineering Journal*, 99(TE4):767–783, 1973.

Karl Petty, Hisham Noeimi, Kumud Sanwal, Dan Rydzewski, Alexander Skabardonis, Pravin Varaiya, and Haitham Al-Deek. The freeway service patrol evaluation project: database, support programs, and accessibility. http://www.path.berkeley.edu/FSP/, 1996.

Robert A Reiss and Nathan H. Gartner. Dynamic control and traffic performance in a freeway corridor: A simulation study. *Transportation Research*, Vol. 25A(5): 267–276, 1991.

Bruno M. Fernandez Ruiz. Architecture for the integration of dynamic traffic management systems. Master's thesis, Massachusetts Institute of Technology, 2000.

Christopher L. Saricks, Joseph L. Schofer, Siim Soot, and Paul A. Belella. Evaluating effectiveness of a real-time ATIS using a small test vehicle fleet. In *Proceeding TRB 76th Annual Meeting*, 1997. Paper No: 970585.

Joseph L. Schofer, Frank S. Koppelman, Regina G. Webster, Stanislaw Berka, and Tsia shiou Peng. Field test of the effectiveness of ADVANCE dynamic route guidance on a suburban arterial street network. ADVANCE Project Document #8463.01, Northwest University, Transportation Center, 1996.

Scott W. Sibley. NETSIM for microcomputers – simulates microscopic traffic flow on urban streets). *Public Roads*, 49, 1985.

S. A. Smith and C. Perez. Evaluation of inform: Lessons learned and application to other systems. *Transportation Research Record*, TRR 1360, 1992.

Y. J. Stephanedes, E. Kwon, and P. G Michalopoulos. Demand diversion for vehicle guidance, simulation and control in freeway corridors. *Transportation Research Record*, TRR 1220, 1989.

Michael Summers and James Crutchfield. Treps interface version 0.1. Technical report, Oak Ridge National Laboratory, 1999.

C. Sun. Dynamic origin/destination estimation using true section densities. Technical Report D99-49, PATH, 1999.

Philip J. Tarnoff and Nathan Gartner. Real-time, traffic adaptive signal control. In Sam Yagar and Alberto Santiago, editors, *Large Urban Systems – Proceedings of the Advanced Traffic Management Conference*, pages 157–168, St. Peterburg, Florida, 1993.

Qi Yang. *A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1997.

Qi Yang and Haris N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research C*, 4(3):113–129, 1996.

Peter J. Yauch, James C. Gray, and William A. Lewis. Using NETSIM to evaluate the effects of drawbridge openings on adjacent signalized intersections. *ITE Journal*, 58(3):35–44, 1988.

Appendix A

Sample Data Files

— Start of `files/master.smc` —

```
# Simlab master file

[Title] = "An example of using simulation master controller"
[Input Directory] = "/users/manishm/cloop"
[Output Directory] = "/users/manishm/cloop/Output"

# Microscopic traffic simulator
[MITSIM] = {
"master.mitsim" # master file
"$HOST" # host
"$DISPLAY" # display
}

# Traffic management simulator
[TMS] = {
"master.tms" # master file
"$HOST" # host
"" # display
}
```

```
# Traffic Management Center Adaptor
[TMCA] = {
"master.tmca"   # master file
"$HOST" # host
"" # display
}

[Break Points] = {
}

# [Randomize] = 0
# [Verbose]   = 1
# [Nice]      = 1
```

— End of `files/master.smc` —

— Start of `files/master.mitsim` —

```
/*
 * MITSIM master file
 */

[Title] = "Local Ramp Control 100%"

[Default Parameter Directory]     = "/users/manishm/view/data/Common"
[Input Directory]                 = "/users/manishm/cloop"
[Output Directory]                = "/users/manishm/cloop/Output"
[Working Directory]               = "/users/manishm/cloop/Output"

[Parameter File]                  = "paralib.dat"
[Network Database File]           = "network_test.dat"
```

```
[Trip Table File]                 = "/users/manishm/cloop/demand0.dat"
[Vehicle Table File]              = ""
[State Dump File]                 = ""
[GDS Files]                       = {
% Filename  MinScale  MaxScale
}
[Link Travel Times Input File]    = {
"cloop_test3new.txt" # Historical travel time
"cloop_test3new.txt" # Updated travel time
0x105 # SP flags
% 0x001 Time variant path calculation
% 0x002 Calulate shortest path peridically
% 0x004 Update path table travel time peridically
% 0x008 Use existing (cached) shortest path table
% 0x100 Updated travel time used for pretrip plan
% 0x200 Receives updated travel time at beacon
}
[Incident File]                   = "incident.dat"
[Path Table File]                 = "newpath.out"
[MOE Specification File]          = ""
[MOE Output File]                 = "moe.out"
[Network State Tag]               = "i3d"
[Segment Statistics File]         = "segstats.out"
[Segment Travel Times File]       = "segtime.out"
[LinkFlowTravelTimes Output File] = "lft_i.out"
[Link Travel Times Output File]   = "linktime.out"
[Vehicle File]                    = "vehicle.out"
[Vehicle Trajectory File]         = "trajectory.out"
[Vehicle Path Record File]        = "pathrec.out"
[Departure Record File]           = "dep.out"
[Queue File]                      = "queue.out"
[Point Sensor File]               = "sensor.out"
```

```
[VRC Sensor File]                  = "vrc.out"
[Assignment Matrix File]           = "assignment_matrix.out"

[Start Time]                       = 07:15:00
[Stop Time]                        = 08:15:00
[Step Size]                        = 0.1

[Segment Data Sampling Step Size] = 30
[Segment Data Report Step Size] = 300
[Point Sensor Step Size]           = 900
[Sensor Dump Size] = 900
[Area Sensor Step Size]            = 60
[Animatiion Step Size]             = 0.1
[Segment Color Step Size]          = 15
[Console Message Step Size]        = 60
[MOE Step Size]                    = 60
[MOE OD Pairs]                     = {

}

[Output]   = 0x03351
%   0x00001 = Vehicle log
%   0x00002 = Sensor readings
%   0x00004 = VRC readings
%   0x00008 = Assignment matrix output
%   0x00010 = Link travel times
%   0x00020 = Segment travel times
%   0x00040 = Segment statistics
%   0x00080 = Queue statistics
%   0x00100 = Travel time tables
%   0x00200 = Vehicle path records
%   0x00400 = Vehicle departure record
%   0x00800 = Vehicle trajectories
%   0x01000 = Output rectangular text
%   0x02000 = No comments
%   0x10000 = State 3D

[Segments] = 2
%   0 = Link type
%   1 = Density
%   2 = Speed
%   3 = Flow

[Signals]  = 0x20
%   0x01 = Traffic signals
%   0x02 = Portal signals
%   0x04 = Variable speed limit signs
%   0x08 = Variable message signs
%   0x10 = Lane use signs
%   0x20 = Ramp meters

[Sensor Types] = 0x1
%   0x1 = Loop detectors
%   0x2 = VRC sensors
%   0x4 = Area sensors

[Sensor Color Code]  = 3
%   0 = Count
%   1 = Flow
%   2 = Speed
%   3 = Occupancy

[Vehicles] = 5
%   0 = None
%   1 = Vehicle type
%   2 = Information availability
%   3 = Turning movement
%   4 = Driver behavior group
%   5 = Lane use

[Vehicle Shade Params] = {
0 # Shade
86400 # Outstanding time in a segment
86400 # Outstanding time in the network
}

[View Markers] = {
  # Label Position Scale Angle Tool ViewType Segment
  # SensorType/Label/Color SignalType Map Legend
  "Map view" 0.5 0.5 0.112069 0 0 0 2 1 0 3 50 1 0
  "I90 / I93 Interchange" 0.332886 0.284409 1.34483 0 0 0 2 1 0 3 50 0 1
  "Incident" 0.400879 0.373999 1.34483 0 1 0 2 1 0 3 50 0 0
  "Sensors" 0.400879 0.373999 5.13012 0 1 0 2 1 2 3 50 0 0
  "Ramp Metring" 0.393233 0.356719 4.2751 5.59596 6 0 2 1 0 3 50 0 0
  "Toll Plaza" 0.948412 0.68546 1.92619 0 0 0 2 1 0 3 50 0 1
  "Ramp Metering 2" 0.347848 0.28101 4.1041 0 1 0 2 1 0 3 50 0 0
  "Ramp Metering 3" 0.263239 0.146937 4.1041 0 1 0 2 1 0 3 50 0 0
}



# [Verbose]   = 1
# [Nice]      = 1
```

— End of `files/master.mitsim` —

— Start of `files/master.tms` —

```
/*
```

```
 * TMS master file
 */

[Title] = "Local Ramp Design"

[Default Parameter Directory]     = "/users/manishm/view/data/Common"
[Input Directory]                 = "/users/manishm/cloop"
[Output Directory]                = "/users/manishm/cloop/Output"
[Working Directory]               = "/users/manishm/cloop/Output"

[Network Database File]           = "network_test.dat"
[GDS Files]                       = {
% Filename  MinScale  MaxScale
}
[Parameter File]                  = "ctrlpara.dat"
[Control Logic File]              = "ctrllogic.dat"
[Signal Plan File]                = ""
[Control Logic]                   = 0
%   0 = None
%   1 = A1 incident response
%   2 = Gating logic
[Information]                     = 2
%   0 = Historical data
%   1 = Real time measurement
%   2 = Prediction

[Start Time]                      = 07:15:00
[Stop Time]                       = 08:15:00
[Step Size]                       = 1

[Console Message Step Size]       = 60
[RollingStepSize]       = 120
```

```
[RollingLength]         = 1800
[NumOfDTAInterations]   = 2
[DTAComputationalDelay] = 110

[Segments] = 1
%   0 = Direction
%   1 = Link type
%   2 = Density
%   3 = Speed
%   4 = Flow

[Signals]  = 0x20
%   0x01 = Traffic signals
%   0x02 = Portal signals
%   0x04 = Variable speed limit signs
%   0x08 = Variable message signs
%   0x10 = Lane use signs
%   0x20 = Ramp meters

[Sensor Types] = 0x1
%   0x1 = Loop detectors
%   0x2 = AVI sensors
%   0x4 = Area sensors

[Sensor Color Code]  = 3
%   0 = Count
%   1 = Flow
%   2 = Speed
%   3 = Occupancy


# [Randomize] = 0
# [Verbose]   = 1
# [Nice]      = 1
```

— End of files/master.tms —

— Start of files/master.tmca —

```
/*
 * TMCA master file
 */

[Title] = "Local Ramp Design"

[Default Parameter Directory]     = "/users/manishm/view/data/Common"
[Input Directory]                 = "/users/manishm/cloop"
[Output Directory]                = "/users/manishm/cloop/Output"
[Working Directory]               = "/users/manishm/cloop/Output"





# The marker needs to be a Orbix marker:
# object:server
[Registry Marker]             = "Registry:PROXIMIT"
[Registry HostName]               = "xylophone.mit.edu"

# The names can be up to 64 characters
# By convention
# object_system
# Character ':' cannot be used in the names

[Factory Name]                  = "Factory_MITSIM"
[Time Message Factory Name]         = "TimeMessageFactory_MITSIM"
```

```
[Sensor Reading Message Factory Name]       = "SensorReadingMessageFactory_MITSIM"
[Incident Message Factory]         =    "IncidentMessageFactory_MITSIM"

[Guidance Listener Name]            = "GuidanceMessageFactory_MITSIM"

# Set the frequency at which TMCA will process the available guidance and
# send it to TMS. Set to zero for real-time broadcast.
# Value in seconds
[Guidance Frequency] = 120
```

— End of `files/master.tmca` —

— Start of `files/dtaparam.dat` —

```
[Files]

InputDirectory  =  "/users/manishm/cloop"
OutputDirectory =  "/users/manishm/cloop/output"
TmpDirectory    =  "/users/manishm/cloop/temp100p"
                                    // MITSIM is the only recognized value for
                                    // now
InputFormat     = "MITSIM"
                                    // If no path is provided, InputDirectory
                                    // is assumed
                                    // If the character '/' appears in the
                                    // file name, InputDirectory is ignored.

NetworkFile   = "network_test.dat"
HistODFile    = "demand0.dat"
SupplyParamFile = "newsupplyparam_test.txt"
HistTTFile    = "linktime-new.dat"
SocioEcoFile  = "socioEco_test0.dat"
BehParamFile  = "BehavioralParameters.dat"
```

```
IncidentFile  = "incdyna.dat"


MitsimOdFile   = "demand0.dat"

MitsimSensorsFile = "sensor_test.dat"




[Simulation]

StartSimulation = 07:15:00
StopSimulation  = 08:30:00

OdInterval      = 15                // in minutes
HorizonLength   = 30            // in minutes

UpdateInterval  = 60                // in seconds
AdvanceInterval = 5               // in seconds
SupplyEpsilon   = 0.01


[Default]

                                    // Default output capacity per lane
OutputCapacity = 0.55               // Unit: veh/lane . sec

FreeFlowSpeed  = 90.0               // Unit: km/hour

JamDensity     =  0.075             // Unit: vehicles/lane-group . meter
```

```
                                    // 0.075 ~= 120 veh/lane-mile

SpeedDensityAlpha = 1.1
SpeedDensityBeta  = 1.5

LoaderInputCapacity = 3.611         // Unit: veh/sec
                                    // 2200veh/hour

LoaderOutputCapacity = 3.611        // Unit: veh/sec
                                    // 2200 veh/hour
MaxEstIter = 3

MaxPredIter = 5
```

— End of `files/dtaparam.dat` —

— Start of `files/configuration.dat` —

```
PLATFORMS

xylophone.mit.edu

SERVERS

dtaPlanning xylophone.mit.edu



OBJECTS

GuidanceModule 2 dtaPlanning
StoppingCriteriaModule 3 dtaPlanning
PreTripDemand 4 dtaPlanning
SupplyModule 6 dtaPlanning
```

```
AssignmentMatrixList 8 dtaPlanning
ListOfImpTable  10 dtaPlanning
BehaviorModels 12 dtaPlanning
ListOfPackets 13 dtaPlanning
NetTopo 19 dtaPlanning
PathTopoTbl 20 dtaPlanning
Clock 21 dtaPlanning
EstimationProcess 22 dtaPlanning
PredictionAndGuidanceProcess 23 dtaPlanning
SimulatedDensity 24 dtaPlanning
SimulatedQueueLength 25 dtaPlanning
SimulatedSpeed 26 dtaPlanning
SimulatedSegmentSpeed           26 dtaPlanning
SimulatedFlow 27 dtaPlanning
SimulatedSegmentFlow 27 dtaPlanning
SimulatedTravelTime 28 dtaPlanning
SocioEcoData 29 dtaPlanning
ODFactory 30 dtaPlanning
Surveillance 31 dtaPlanning
Parameters 32 dtaPlanning
MitsimDemandProcess 33 dtaPlanning
StatusManager 34 dtaPlanning
AggregateOutput 35 dtaPlanning
Logger 36 dtaPlanning
Report                          37      dtaPlanning
Communicator                    38      dtaPlanning
```

— End of `files/configuration.dat` —

— Start of `files/runit.dat` —

```
#!/bin/sh
```

```
#Run orbixd

orbixd -u &

orbixd_pid=$!

sleep 4


#Run the Registry server

/users/manishm/DTMS/bin/DTMS_Registry Registry PROXIMIT &

registry_pid=$!

sleep 4


#Run the factory server

/users/manishm/DTMS/bin/DTMS_Factory Factory_MITSIM Registry:PROXIMIT

xylophone.mit.edu &

factory_pid=$!

sleep 2


#Run the time message factory server

/users/manishm/DTMS/bin/DTMS_TimeMessageFactory TimeMessageFactory_MITSIM

Registry:PROXIMIT xylophone.mit.edu &

tmfactory_pid=$!

sleep 2


#Run the Incident message factory server

/users/manishm/DTMS/bin/DTMS_IncidentMessageFactory

IncidentMessageFactory_MITSIM Registry:PROXIMIT xylophone.mit.edu

& infactory_pid=$! sleep 2


#Run the sensor reading factory server

/users/manishm/DTMS/bin/DTMS_SensorReadingMessageFactory

SensorReadingMessageFactory_MITSIM Registry:PROXIMIT xylophone.mit.edu &

srfactory_pid=$!

sleep 2
```

```
#Run the guidance factory server
/users/manishm/DTMS/bin/DTMS_GuidanceMessageFactory
GuidanceMessageFactory_MITSIM Registry:PROXIMIT xylophone.mit.edu &
gdfactory_pid=$!
sleep 2

trap "kill $orbixd_pid $registry_pid $factory_pid
$tmfactory_pid $infactory_pid $srfactory_pid $gdfactory_pid" 0

#smc -debugger 1 -m master.smc
smc -m master.smc
```

— End of `files/runit.dat` —

— Start of `files/communicator.dat` —

```
[HostInfo]

RegistryName = "Registry:PROXIMIT"

HostName = "xylophone.mit.edu"

FactoryName = "Factory_MITSIM"

[Factory]

GuidanceFactoryName = "GuidanceMessageFactory_MITSIM"

[Listener]

IncidentListener = "IncidentMessageFactory_MITSIM"
```

```
TimeListener = "TimeMessageFactory_MITSIM"


SensorListener = "SensorReadingMessageFactory_MITSIM"
```

— End of `files/communicator.dat` —