Title

Field Deployment and Operational Test of an Agent-based, Multi-Jurisdictional Traffic Management System

Authors

Rindt, Craig R.
McNally, Michael G.

Publication Date

2007-06-01

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Field Deployment and Operational Test of an Agent-based, Multi-Jurisdictional Traffic Management System

Craig R. Rindt, Michael G. McNally
University of California, Irvine

California PATH Working Paper
UCB-ITS-PWP-2007-1

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation, and the United States Department Transportation, Federal Highway Administration.

Final Report for Task Order 5313

June 2007

ISSN 1055-1417

Field Deployment and Operational Test of an Agent-based, Multi-Jurisdictional Traffic Management System

Task Order 5313: Final Report

Craig R. Rindt and Michael G. M^{c}Nally
Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697
949-824-6544 - tel
crindt@uci.edu
mgmcnally@uci.edu

Abstract

This report describes a reinterpretation of how the philosophy underlying the Cartesius multi-jurisdictional incident management prototype can be used as an organizing principle for real-world multi-jurisdictional systems. This interpretation focuses on the power of the Distributed Problem Solving (DPS) approach Cartesius uses to partition analysis and optimization functions in the system across jurisdictions. This partitioning minimizes the amount of local information that must be shared between jurisdictions and paves the way for defining a collection of TMC-to-TMC messages that support the Cartesius DPS perspective in a manner that respects existing deployments.

Based on this interpretation, the report recommends building a new TMC software agent that provides operators with a view of the system from Cartesius DPS perspective. This tool will initially be advisory in nature, providing operators with guidance regarding how local actions are likely to conflict with the actions of neighboring jurisdictions (or lack thereof). Where it is appropriate, and where local policy permits, the new management agent could also be connected to available control subsystems to provide operational or tactical control in response to problems in the system.

Keywords: Advanced Traffic Management Systems, Incident Management, Artificial Intelligence

Executive Summary

This report details the results of research carried out under PATH Task Order 5313: *Field Deployment and Operational Test of an Agent-based, Multi-Jurisdictional Traffic Management System*. This project was funded to perform a field operational test of the CARTESIUS incident management system prototype with the ultimate goal of priming the prototype for eventual deployment in California TMCs.

During the course of the research, however, it became clear that the original project goals were unrealistic for a variety of reasons. The core reasons involve limitations of the existing CARTESIUS prototype that stem from its history as a research tool. Because of these identified limitations, and because concurrent PATH research related to CARTESIUS added additional changes to the expectations regarding CARTESIUS, the scope of Task Order 5313 was revised to perform a technical evaluation of the existing prototype, revise the functional role of CARTESIUS to be in-line with Caltrans expectations, and to develop a set of recommendations for making a product that implements the CARTESIUS philosophy in a manner consistent with real-world deployment.

The evaluation of the prototype led to a set of recommendations for moving the CARTESIUS project forward including:

- scrapping the current proprietary G2-based implementation in favor of one using general purpose languages and communications protocols,
- the use of an event-driven, service-oriented messaging architecture to provide communications between jurisdictional agents,
- a redesign of the software to better separate analytical functionality (business logic) from interface functionality,
- a revision of CARTESIUS data model to leverage data standards and relational datastores as well as available libraries for manipulating such data,
- refocusing CARTESIUS on characterizing the problem state and the implications of potential control actions rather than a focus on including CARTESIUS in a closed control loop,
- restating and focusing the set of functional requirements that define CARTESIUS role in the traffic management system

Based on these findings, the report describes a reinterpretation of how the CARTESIUS philosophy can be used as an organizing principle for real-world multi-jurisdictional systems. This interpretation focuses on the power of the Distributed Problem Solving (DPS) approach CARTESIUS uses to partition analysis and optimization functions in the system across jurisdictions. This partitioning minimizes the amount of local information that must be shared between jurisdictions and paves the way for defining a collection of TMC-to-TMC messages that support the CARTESIUS DPS perspective in a manner that respects existing deployments.

Finally, we recommend building a new TMC software agent that provides operators with a view of the system from CARTESIUS DPS perspective. This tool will initially be advisory

in nature, providing operators with guidance regarding how local actions are likely to conflict with the actions of neighboring jurisdictions (or lack thereof). Where it is appropriate, and where local policy permits, the new management agent could also be connected to available control subsystems to provide operational or tactical control in response to problems in the system.

Contents

List of Figures

Chapter 1

Introduction

1.1 A Brief History of TRICEPS and CARTESIUS

The TRICEPS and CARTESIUS arose from the California ATMIS Testbed program at the University of California, Irvine during the 1990s that developed a unique proving ground for traffic management technology. The resulting ATMS Testbed links a real-world, multi-jurisdictional transportation system with research laboratories at University of California, Irvine (UC Irvine) to facilitate the testing and evaluation of cutting edge Intelligent Transportation Systems (ITS) technologies.

TRICEPS—the Testbed Real-time Integrated Control and Evaluation Prototype System—was developed as an implementation platform to provide "plug and play" capabilities for the testing and evaluation of a range of Advanced Transportation Management and Information System (ATMIS) modules. Testbed Real-time Integrated Control and Evaluation Prototype System (TRICEPS) permits modules to be configured for various ATMIS applications and run in simulated, real-time, and integrated (real-time backed by simulation) modes. TRICEPS was motivated by the belief that traffic management systems will ultimately be created from a collection of available tools that could be recombined as needed to measure, analyze, model, and control transportation systems. The continued development of the National Intelligent Transportation Systems Architecture (NITSA) as a collection of processes and data flows has vindicated this original viewpoint as one that meets the needs of transportation system managers. TRICEPS was ultimately implemented as a set of interfaces and data types defined using Common Object Request Broker Architecture (CORBA). Implementations of those interfaces and data types were built for a number of functional ATMIS components, including CARTESIUS.

CARTESIUS—the Coordinated Adaptive Real-Time Expert System for Incident management in Urban Systems—is arguably the most important, and most complete module implemented for TRICEPS. It was developed with the realization that traffic management in urban corridors is complicated by jurisdictional as well as operational problems. Traffic in a freeway/arterial corridor requires coordinated management that optimizes corridor traffic while preserving the various levels of authority, data control, and decision-making power inherent in a multi-jurisdictional environment. CARTESIUS approaches this problem by employing advanced cooperation and conflict resolution methodologies for coordinated

traffic management operations among multiple jurisdictions using a multi-agent architecture to represent the diverse interests and needs of each agency. On the whole, CARTESIUS refers to both a *philosophy* regarding how to manage traffic under the constraints inherent to multi-jurisdictional transportation systems using a Distributed Problem Solving (DPS) approach and an *implementation* of that philosophy in a working two-agent prototype that demonstrates the efficacy of the philosophy.

1.2 Original Project Scope

The original purpose of this TO 5313 was to test CARTESIUS, the *implementation*, as a precursor to developing a deployable version of CARTESIUS for general use in California. The research plan originally identified three broad issues for this test:

- ***Architectural issues*** related to moving the software from the relatively homogeneous laboratory setting at UCI to a the heterogeneous field environment and usability issues associated with the use of CARTESIUS by Transportation Management Center (TMC) operators in practical, mission-critical settings.
- ***Verification and validation*** of embedded knowledge and agency policy regarding control and interagency-coordination strategies.
- ***Protocols for testing developing technology in the field*** to resolve legal and institutional barriers to field testing a comprehensive traffic management system

To address these issues, the research plan called for a test of CARTESIUS in two evaluation modes. In the first mode, the system was to process real-time data coming from sensors in the field and was to provide advisory management strategies and control actions for the consideration of Caltrans and local traffic management center personnel. This mode of evaluation was to provide on-line, risk-free verification and limited validation of CARTESIUS.

The second evaluation mode was to involve usability and stress testing of the CARTESIUS system with the CARTESIUS agents remotely connected to the Paramics traffic simulator at the UCI laboratories. In this mode, TMC operators/personnel were to be asked to respond to the scenarios using CARTESIUS to implement control strategies in Paramics.

1.3 Revised Project Scope

During the course of this project, numerous problems arose related to the deployment of CARTESIUS even in the limited on-line roles envisioned for the tests (see section 3). As a result, the scope of the project was revised several times to accommodate these problems.

At the same time, work on PATH Task Order 5324/6324—intended to explore integrating CARTESIUS with Caltrans CTNET traffic signal management system—provided additional incentive to revise the functional role of CARTESIUS to be more compatible with existing technology and standards. Coupled with the identified difficulties with the existing prototype, the management team of both this research effort and the CTNET effort collaborated

to develop a revised plan for TO 5313 with a focus on enhancing the deployability of Cartesius with a first effort being the Cartesius/CTNET integration effort in TO 5324/6324.

As a result, the scope of TO 5313 has been revised to achieve the following:

- a technical evaluation of the existing Cartesius prototype,
- a study of functional roles in inter-jurisdictional traffic management, and
- recommendations for a new multi-agent software system design for implementing the Cartesius philosophy.

This report details these findings and, in particular, describes the design of the next generation Cartesius implementation.

Chapter 2

CARTESIUS

2.1 CARTESIUS as a theoretical system

The core goal of the original CARTESIUS research was to develop a new Distributed Problem Solving (DPS) approach for the provision of real-time decision-support to TMC personnel in multi-jurisdictional, network-wide management of non-recurring congestion. The approach is based on the Functionally Accurate/Cooperative (FA/C) approach to DPS (Lesser and Corkill, 1981).

2.1.1 Traffic Management Through Distributed Problem Solving

The core features of the CARTESIUS philosophy permit conforming distributed systems to:

- reflect the jurisdictional distribution of traffic problem-solving expertise;
- provide local management of data sources;
- reduce synchronization delay and communication overhead.

The main goals of the architecture are to allow:

> "a set of agents to interact in order to produce an efficient and conflict-free global solution in an environment that is inherently distributed. Their interaction is constrained by the lack of complete information, that results from the data and control knowledge being distributed according to geographical and administrative criteria. Input data available to the TMC operator, such as detector data describing traffic conditions, visual data coming from CCTV cameras installed at key locations, and incident reports or confirmations, are normally partitioned according to the institutional organization of the management agencies, which must administer the information in their possession as efficiently as possible, avoiding the transmission of irrelevant or ill-timed data. Another constraining factor is the need of each agency to preserve dedicated control over its jurisdiction, maintaining its decisional power, and applying local criteria and guidelines. At the same time, the agencies, as interacting components of a regional, integrated

> management organization, must attempt to converge towards a common goal, that of ameliorating network-wide traffic conditions, and can do so by effective collaboration and resolution of potential conflicts (Logi, 1999)."

To meet these goals, CARTESIUS uses the FA/C DPS framework. This paradigm organizes effective cooperation among DPS agents in order to produce an acceptable solution in reasonable time by assuming that individual agents need not always have all the information about the global problem to completely and accurately solve their subproblems. In particular, in some applications

> "there exist inter-agent constraints among subproblems that can be exploited to resolve the inconsistencies that may arise due to the lack of accurate, complete and up-to-date information. Agents can produce tentative, partial results based on local information and then exchange them with other agents to resolve local uncertainties and global inconsistencies (Logi, 1999)."

The CARTESIUS formulation maps these concepts to the multi-jurisdictional traffic management domain by describing a problem solving agent for each jurisdiction that is solely responsible for controlling its subnetwork. Each jurisdictional agent can first focus on its local problems, then exchange high-level information about its local candidate solutions with other agents. This exchanged information allows each agent to identify and remove local solutions that are at odds with the collection of management actions proposed by remote agents.

A key theoretical contribution provided by CARTESIUS is its definition of the features of the traffic management problem domain in a manner that is consistent with the FA/C DPS formulation. Toward this end, CARTESIUS leveraged an existing problem solving approach for traffic management, the Traffic Congestion Manager (TCM) (Logi, 1995), which used methodologies similar to those employed in existing FA/C systems.

Figure 2.1 shows the event processing that occurs in a three-agent CARTESIUS system when an incident is reported somewhere in the managed network. The event notification causes each agent to begin a two-stage analysis. During the ***Problem Analysis*** stage each agent characterizes all problems in its jurisdiction and attempts to identify their causes. Information is shared between agents to allow causation to be determined across jurisdictional boundaries. During the ***Problem Response*** stage, each agent searches for strategies and the local control actions that implement them. The assumptions underlying these local control actions are propagated to other agents as constraints that those agents are requested to consider in a global solver step. At this point, each agent has its own copy of all feasible local solutions that conform to globally propagated constraints. A given agent's copy of this solution set contains detailed information about local actions to be taken for each solution and high-level information about the remote plans that are consistent with these local actions.

The following sections describe in more detail the local problem solving approach and the limited information that must be shared to identify compatible global solutions.

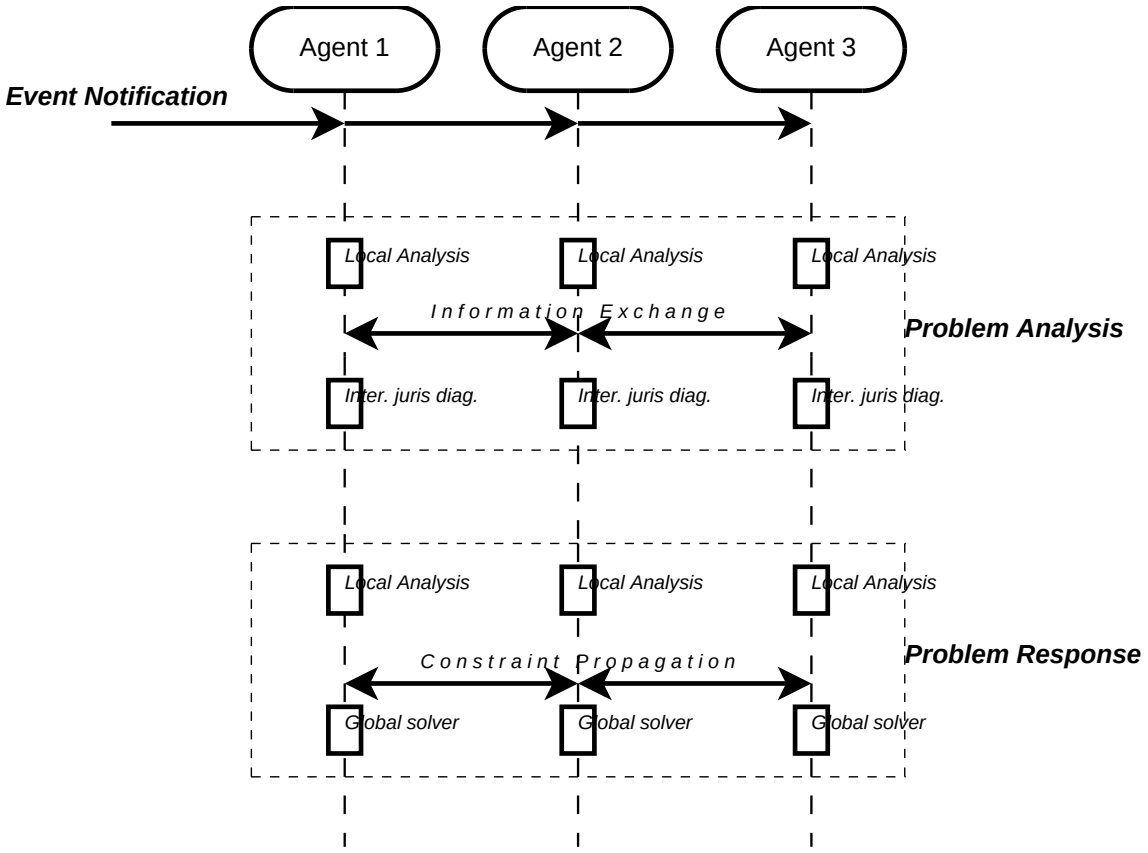

Figure 2.1: Event processing and information sharing in a hypothetical three-agent CARTESIUS system.

2.1.2 Local problem solving

The CARTESIUS approach is based upon the concept that each agent solves any problems local to its jurisdiction subject to limited information shared between the agents, which may include new potential problems caused by actions taken in neighboring jurisdictions. Thus, a problem solving agent can simply focus on mitigating a given set of problems using a local algorithm. The core features of the local problem solving algorithm that makes up the CARTESIUS philosophy are discussed below.

Knowledge Representation

CARTESIUS characterizes problems in the system using problem-specific frame-driven recognition techniques (Charniak and McDermott, 1985). At their core, these techniques depend on a "hierarchical database of knowledge packets," or *frames* (Minsky, 1974; Winston, 1975), that represent expert knowledge regarding the causal origins of problems in the transportation system as *problem frames* defined in terms of network topology and qualitative characterizations of sensor data. The database of problem frames are systematically compared to individual *network objects*—discrete partitions of the traffic network with particular operational characteristics (e.g., freeway on-ramp, freeway off-ramp, intersection approach,

etc.). Each jurisdictional agent must therefore create and maintain this database of problem frames and network objects representing the network it manages. Specific knowledge about neighboring jurisdictions networks, control systems, particular problem types, and real-time data is not required. This partitioning satisfies the need for the system to preserve local autonomy and greatly simplifies data management.

Problem Characterization

The identified problems output from the local frame matching procedures, using real-time measurements from the system, collectively form a *problem state*. Each problem in the problem state identifies a *critical section* in the network for which the estimated demand exceeds the estimated capacity and, depending on the problem frame that matched to the current conditions, provides an assessment of the cause and general characteristics of the disruption based upon local expert knowledge embedded in the system's problem frame database.

The characterization process also recognizes the fact that a problem at a particular location in the network can lead to secondary congestion at other locations in the network (e.g., on-ramp spillback onto the arterial network). The local problem characterization works to identify *derives from* relationships between problems. These relationships are used in the logic that determines responses recommended by the CARTESIUS system.

Problem Response

To mitigate the identified problem state, the CARTESIUS is structured around a *planning* approach, in the Artificial Intelligence (AI) sense, to generate a sequence of local control actions that can change the estimated state of the transportation system to a more favorable outcome. The approach uses a heuristic Breadth First Search (BFS) algorithm that starts from the identified problem state, identifies subgoals that aim to move the system to a more desired state (e.g., a state in which demand and capacity are balanced), and finds concrete control actions that can achieve these subgoals. This general formulation is implemented for the traffic management problem by defining subgoals (with corresponding strategies) as follows:

- *increase capacity* at a congested location by *signalization*,
- *decrease flow* at a congested location by *traffic diversion*, and
- *decrease flow* at a congested location by upstream *metering*.

Defining concrete actions to implement these strategies, and algorithms to estimate those effects is a location and implementation-specific task because the available strategies and associated control actions will vary with each deployment—every jurisdiction is likely to employ different control subsystems that have distinct characteristics that are difficult to generalize.

Problem Monitoring

The type of high-level traffic management tackled by CARTESIUS inherently must deal with large degrees of uncertainty regarding knowledge of the state of the system and the effectiveness of implemented control strategies. Toward this end, CARTESIUS includes problem monitoring functions whose role is to track the performance of given control solution—in an absolute sense as well as whether the evolution of the system under that solution is consistent with expectation. The secondary function of the problem monitor is to filter out redundant information (such as the re-reporting of existing problems) to reduce the need for unnecessary analysis of problems that have already been considered.

2.1.3 Information Sharing for Distributed Problem Solving

The jurisdictional partitioning of traffic management described above can effectively isolate the management of problems to a single jurisdiction in the system as long the effects of those problems are contained in one jurisdiction. The existence of spillback effects across jurisdictional boundaries, and the fact that some control strategies (such as diversion) can place operational burdens on portions of the network outside the jurisdiction of the diverting agency, require a means for sharing information between jurisdictions regarding such effects. It is here that CARTESIUS DPS philosophy provides benefits by defining the specific types of information that need to be shared across jurisdictions in order to ensure that the collective of local responses to a given problem state produce a global solution that is acceptable to all jurisdictions.

The CARTESIUS adaptation of FA/C methods to this problem defines three distinct stages during problem solving when distributed agents should exchange information, and also what type of high-level information should be exchanged at those points. In each case, the general approach is to locally perform detailed analysis, then to create and post an abstraction of that analysis that can be used by other agents that then leverage the abstraction to refine their analysis. The process can be iterative and each instance must be carefully analyzed to ensure tractability and the avoidance of race conditions in the distributed system.

Problem Analysis: Spillbacks and derived problems

The characterization of traffic problems is complicated in a distributed system because of the potential relationship between problems as discussed above. Since such propagation can cross jurisdictional boundaries, the local problem characterization process described above must be augmented to handle information regarding the possible relationship between problems across jurisdictional boundaries.

CARTESIUS handles this problem by identifying all spillback effects and posting their characteristics to other agents for them to identify possible links between them. This information sharing allows agents whose jurisdictions are involved to compute *derives from* relationships between problems that cross these boundaries and to share those determinations with the agent collective.

Problem Response

The nature of partially decoupled DPS for complex, large-scale systems invariably leads to uncertainty and imperfect knowledge that makes the use of heuristic algorithms the only tractable approach. Any exact or heuristic optimization algorithm requires specification of an objective that can be evaluated in reasonable time (as dictated by the constraints of the algorithm and available computing power). Furthermore, the algorithm needs a means for determining how to reach the objective (i.e., a search direction).

CARTESIUS uses a problem solving heuristic that seeks to both avoid the spread of congestion that may lead to oversaturation and secondarily, to achieve a balanced ratio between network capacity and traffic demand. The CARTESIUS traffic management agent uses problem solving algorithm that incrementally searches a space of problem mitigation strategies, each of which can be *translated* by a corresponding control action or set of control actions. Thus, a strategy to *reduce critical section demand through diversion* might be realized by setting a collection of Changeable Message Sign (CMS) to reroute traffic around the problem. The expansion of the solution search tree through strategy and control actions generates new nodes representing an estimate of the state of the system if the control actions on the path from the root node to the target node are applied.

The selection of given strategy or the specific control actions that translate that strategy may require the satisfaction of particular conditions that define dynamic constraints that must be met for those strategies or actions to achieve their anticipated effects. These constraints may only apply to the local jurisdiction, but often, as in the case of diversion strategies requiring sufficient network capacity, they may propagate to remote jurisdictions.

CARTESIUS propagates such constraints as conditions that the remote agent must meet. The remote agent translates these conditions into strategies that have the goal of meeting the condition. These strategies are used to expand the search tree by branching from existing nodes to generate new portions of the search tree that will satisfy the conditions broadcast by other agents.

The process continues until no more conditional strategies are being broadcast by any agents. This indicates that the global search has been exhausted and the existing candidate solutions in the tree represent global collective of candidate solutions under consideration, including the extent to which each solution meets any broadcast constraints. The task of selecting a single global solution from among the set of candidates is one of removing solutions that are inconsistent with local or remote constraints. Since each agent has the same list of the available feasible solutions, it is now possible for the agent collective to jointly select a single global solution consisting of a combination of locally acceptable control actions.

2.2 CARTESIUS as a prototype

The original CARTESIUS prototype was realized as a distributed software system composed of two interacting real-time decision support systems representing the California Department of Transportation (Caltrans), District 12–Orange County (Caltrans–D12) and the City of Irvine traffic management centers as shown in figure 2.2. The agents were designed to perform cooperative reasoning and resolve conflicts for the analysis of congestion phenomena

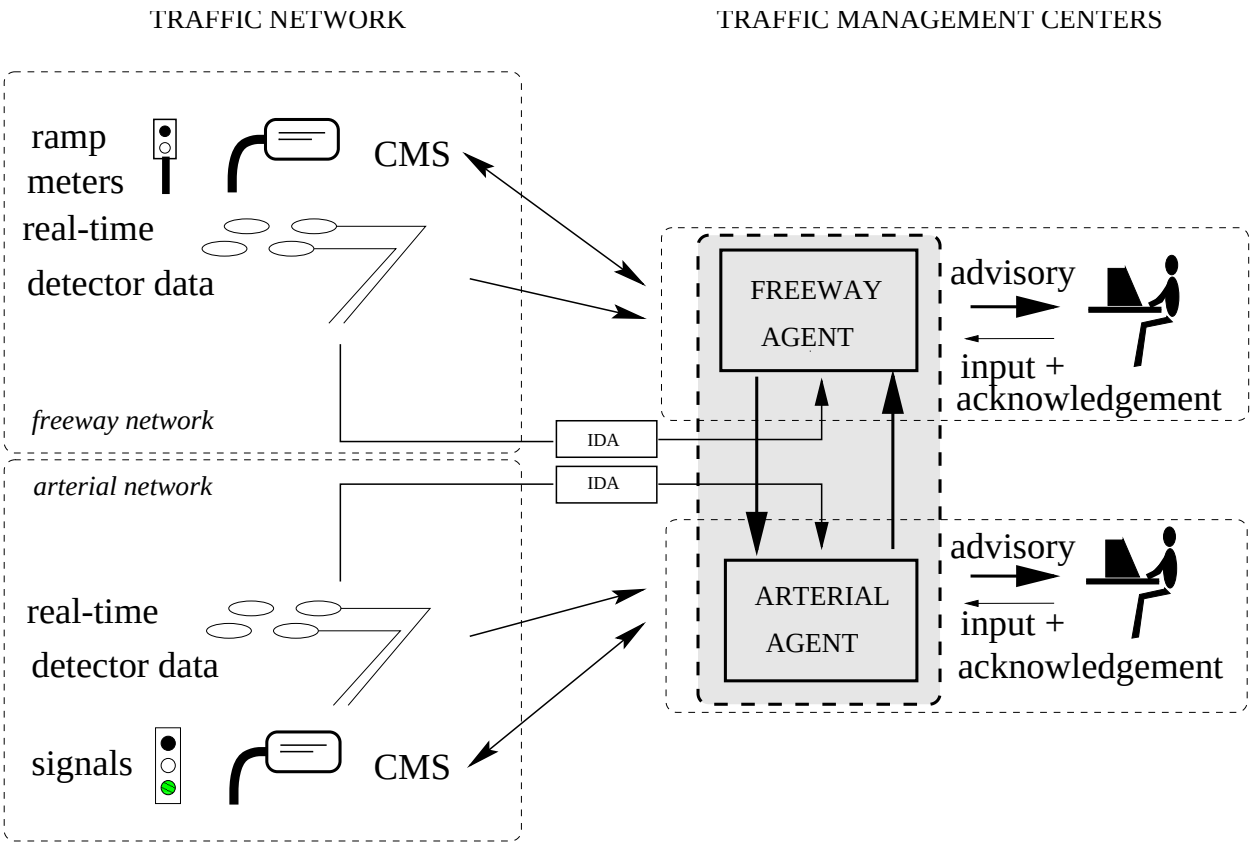

Figure 2.2: The prototype CARTESIUS multi-agent incident management system, actually a tightly integrated two-agent system involving an agent for Caltrans, District 12–Orange County and the City of Irvine Transportation Management Centers.

occurring in the Caltrans/UCI ATMS Testbed (Testbed) network and the generation of suitable network-wide, responses that integrate control strategies involving TMCs from both agencies.

As noted above, the CARTESIUS prototype was implemented from an existing single-jurisdiction problem solving prototype known as TCM. TCM itself was the product of research that evolved from earlier work into the use of expert systems for traffic management. Largely for this reason, TCM, and subsequently CARTESIUS after it, were implemented using the proprietary G2 expert system shell, which had been used in the earlier work.

The prototype focused on the *integration* of existing techniques for real-time generation and assessment of appropriate settings for traffic control devices, with emphasis on the delicate problem of coordinating the efforts of the agencies responsible for operations and control in urban freeway and arterial networks. As such, it did not propose *new* control algorithms *per se*, but used the CARTESIUS philosophy of seeking a compatible combination of local control actions in each jurisdiction that produced a globally efficient solution to incidents in the system.

A more detailed overview of the prototype is available elsewhere (Logi, 1999; Logi and Ritchie, 2002; Logi et al., 2001). These references include specifics regarding the algorithms for computing suitable control actions to realize particular strategies during the problem

response algorithm. The research included a detailed simulation-based evaluation of CARTESIUS that showed an 8% reduction in total delay experienced in the system over all incident scenarios evaluated. The magnitude of the delay savings varied, as expected, depending on the location of the simulated incident and the demand levels. Nonetheless, the broad conclusions regarding the potential of the system were unequivocally positive.

Probably the most important point about the existing implementation is that it was developed as a *research* prototype for demonstrating the validity of the CARTESIUS philosophy—a task that it performed admirably. The resulting software implementation, however, has shortcomings with respect to its deployability ranging from far-reaching assumptions necessary to complete the research to an inflexible and fragile software implementation that is prone to faults. These shortcomings precipitated the scope changes made during this project and we discuss them in detail in chapter 3.

Chapter 3

Findings Regarding the Deployability of the CARTESIUS prototype

This chapter details the findings of the research team with respect to the deployability of the CARTESIUS prototype. These findings are the result of detailed analysis of the prototype's code, communications structure, data representation, and general usability. Together, these findings provide a justification for the altered scope of TO 5313 from a field operational test to a re-imagining of the CARTESIUS philosophy into a deployable set of concepts for modern ITS architectures—with a particular eye on potential compatibility with the burgeoning NITSA.

3.1 Barriers to deploying the prototype

CARTESIUS has proven difficult to deploy in even very limited capacities involving in-laboratory testing. The reasons for this difficulty are numerous and cross cutting. The following sections detail characteristics of the existing CARTESIUS system that limit its potential for real-world deployment in the near to medium term. Though some of these limitations are more burdensome than others, they are listed in no particular order.

3.1.1 Proprietary lock-in using a niche development platform

The CARTESIUS prototype is built atop the proprietary G2 expert system shell that uses a niche language with a steep learning curve and lacks the flexibility of a general purpose language. G2 uses a natural language programming dialect that is structured specifically for designing knowledge-based systems. The language is expressive and powerful for certain tasks, but is invariably counterintuitive to developers familiar with more common general purpose procedural and object-oriented languages such as C++, Java, or C#.

The vendor lock-in also limits the availability of third party tools that might be used in analysis. For instance, the lack of an available G2 library for analyzing graphs required construction of graph theoretical routines using G2. This increases the developer time and propensity for faults in the system since widely used general purpose libraries generally receive substantial testing that reduces the prevalence of bugs. Purpose built, sole-use libraries,

on the other hand, require in-house testing that by its nature cannot be as thorough as that received by general purpose libraries.

A move away from G2 to a general purpose language would require a complete reimplementation of CARTESIUS—a non-trivial undertaking. On the other hand, it would also:

- permit a refocusing and redesign of the implementation that will make it more suitable for deployment,
- allow the use of general purpose libraries for core analytical components, and
- open the door to transitioning from restrictive and costly commercial licensing to an open-source implementation CARTESIUS that will have fewer barriers to deployment.
- make it easier for new developers to understand the design of the system (assuming a well-known general purpose language is used in the redeployment).

.

We recommend reimplementing the CARTESIUS traffic management prototype using appropriate general purpose languages and communications protocols to free it from costly vendor lock-in that ultimately restricts its development.

3.1.2 Strictly a two-agent system

A major barrier to deploying the existing CARTESIUS implementation is the fact that it is, in reality, a two-agent system that cannot be trivially converted to a multi-agent system consistent with the CARTESIUS philosophy. Looking at the details of the implementation, one finds that the knowledge sharing and even the core algorithmic structure is hard coded using Remote Procedure Callss (RPCs) for a two-agent implementation. While this limitation was not an issue for the original research, it has far reaching implications for the adaptability of the existing prototype to real-world scenarios in which it is basic requirement that the system be able to handle more than two jurisdictions.

With a few manageable exceptions, there is nothing inherent in the CARTESIUS philosophy that limits the approach to two agents. Indeed, the core DPS architecture upon which CARTESIUS is based is specifically designed for systems with an arbitrary number of interacting agents. The two-agent design of the existing prototype is therefore strictly a limitation of the implementation.

We recommend exploring modern, service-oriented messaging architectures that allow individual components to be loosely coupled using event-driven methodologies to achieve coordination as necessary.

3.1.3 Heavy intermingling of business logic and user interface code

The software architecture used for the CARTESIUS traffic management agent prototype has a tight integration between algorithmic code (the business logic) and the code implementing the Graphical User Interface (GUI) used by the operator. Some of this design is a result of selecting the G2 platform for development—a system that generally encourages such intermingling. The consequences of this design choice are that it restricts the modularity of the existing CARTESIUS implementation. Removal or modification of core analytical components necessitates modification of the interface that, in turn, adds to the difficulty of improving the system through upgrades.

A preferred design decouples the business logic from the interface. This approach is in line with standard software engineering practice and permits the design of multiple interfaces to the system. Adopting a standard software engineering pattern, such as Model View Controller (MVC), will offer much greater flexibility for adapting CARTESIUS for use with external systems by potentially exposing core analytical processes to existing or future interfaces.

> **We recommend a reworking of the CARTESIUS traffic management agent design that separates analytical functionality from interface functionality.**

3.1.4 Limitations and complexity of TRICEPS

The plug-and-play TRICEPS architecture is a flexible and powerful system for deploying distributed algorithms that relies on CORBA for communications and cross-platform data representation. The implementation, however, is showing its age. Since its original conception, the NITSA has defined numerous standards for information sharing and data representation in traffic management systems. Many of the original problems solved by TRICEPS are now treated by new standards that will likely to move into standard practice.

The cost of modernizing TRICEPS to support these standards likely outweighs the cost of simply migrating Testbed architectural components to use standards, whether these are NITSA or California standards (such as the AB3418E standard used by CTNET for managing signal subsystems). Furthermore, with the development of such standards, the need for a purpose-built, encompassing distributed computing platform is diminished. Finally, Caltrans personnel have expressed a desire to move away from CORBA-based solutions due to their complexity[1]. Ultimately, the existence of mature alternative messaging architectures—in particular various service-oriented architectural solutions that are easier to deploy that CORBA solutions—makes a move from the legacy TRICEPS architecture attractive.

> **We recommend adoption of an out-of-the box service-oriented messaging architecture for information sharing between CARTESIUS agents and between those agents and external data sources and control subsystems.**

[1]This finding is based upon personal communication between the authors and management personnel at Caltrans D12 and D11 TMCs.

3.1.5 Difficulty managing the knowledge base

The problem-solving algorithm employed by the CARTESIUS traffic management agent requires a significant amount of site-specific knowledge. While *design* of the knowledge representation used in the CARTESIUS prototype is elegant and efficient, it's implementation is hampered by the G2 platform. As already noted, the lack of general purpose analytical libraries for G2 required the development of purpose-built libraries for representing the managed transportation network, for representing analytical graphs used in the search process, and for representing a number of logic-based concepts beyond the capabilities of the G2 core. These structures are hindered by G2's data representation capabilities which has led to a knowledge management problem for CARTESIUS.

It is possible that more recent versions of G2 offer more flexible data representation and/or that a more experienced G2 developer could produce better knowledge representation. But this just illustrates the vendor lock-in issues already discussed in section 3.1.1—use of a niche platform limits flexibility and/or increases development costs.

Porting the CARTESIUS agent to a general purpose language, as discussed in section 3.1.1, will simplify management of significant portions of the knowledge base by allowing use of standard data libraries for representing many data types required by CARTESIUS algorithms. Furthermore, since most standards are ultimately supported by libraries using general purpose languages, a port should improve CARTESIUS' ability to conform to industry standards to increase its portability.

Finally, a move from G2 will allow the development of multiple interfaces to CARTESIUS datasets. This will enable purpose-built interfaces for individual portions of the CARTESIUS knowledge base using technology ranging from thin web clients to rich native clients.

> **We recommend a redefinition of the CARTESIUS data model that focuses on leveraging data standards and software tools and libraries that operate on those standards. Furthermore, to facilitate broader use of CARTESIUS-related data, we recommend the use of a general-purpose relational database that can expose core data used by CARTESIUS to multiple sources.**

3.1.6 Oversimplified representation of real-world systems

Recall that the CARTESIUS prototype seeks to resolve identified problems in the system by searching sets of *strategies* that can be implemented using available *control actions*. In an effort to demonstrate its potential, the research prototype was constructed to consider a full set of such control actions, essentially seeking to actively manage all control subsystems in a given jurisdiction, including traffic signals, ramp meters, and CMSs in order to mitigate congestion.

This ambitious goal meant that the CARTESIUS agent include optimization algorithms for each of these subsystems in order to determine appropriate control settings. However, many simplifying assumptions were made in order to reduce the complexity of the implementation and thus allow a complete evaluation of the system's potential. While these assumptions were acceptable in a research context, they have proven to be far to restrictive for actual real-world deployment.

We recommend first focusing CARTESIUS on the problem of estimating the implications of anticipated control actions given currently deployed control algorithms. The problem of actively adjusting control (or advising local controllers) to mitigate non-recurrent congestion using customized control algorithms can be investigated where necessary, but preference should be given to existing research focusing on those problems independently.

For completeness, the following sections detail various oversimplifications and their implications for the deployability of CARTESIUS.

Intersection Signalization

The CARTESIUS prototype uses algorithms developed by Akcelik (1981) and Gazis (1974) to analyze and optimize signalization effects in the face of anticipated changes demand patterns caused by problems in the system and control actions (such as diversion) taken to mitigate those problems. In particular, the CARTESIUS agent needs to:

- estimate the capacity of particular intersection movements to support potential diversion, and
- optimize intersection control settings to handle that possible demand.

The major shortcoming of this algorithm, from a deployment perspective, is that it does not consider coordination of signals, which is a major feature of virtually all deployed signal systems that manage even moderately congested corridors. This fact in itself prevents the existing CARTESIUS prototype from any meaningful field deployment until it can be reworked to either properly determine control settings for real-world configurations, or to leverage existing control subsystems to access available timing plans.

Ramp Metering

As with intersection signalization, CARTESIUS uses internal algorithms to determine control settings for ramp meters in the field. While ramp metering field controllers generally operate independently of other controllers and thus have simpler constraints, they are subject to administrative policy that is currently not represented in the CARTESIUS prototype. Furthermore, the nature of the ramp metering problem is less complex and is therefore more amenable to strictly local control algorithms. It is likely, therefore, that CARTESIUS needn't actively optimize these systems either, instead focusing on estimating their capacity to identify potential strategy conflicts.

CMS

The CARTESIUS prototype's sole means of effecting diversion strategies is the use of CMS messaging. CARTESIUS identifies diversion control actions using a knowledge base of permitted CMS messages that have expected diversion effects. Estimation of potential diversion effects for possible real-world message sets, however, will require substantial additional research

to calibrate and validate. Because of the potential for contributing to traffic disruption, CMS messages are tightly controlled by Caltrans, making the type of messaging originally envisioned by CARTESIUS potentially undeployable. Thus, while the general framework used by CARTESIUS is generally still applicable, it is likely that the available message sets will be both difficult to estimate effects for and significantly less effective that those demonstrated in the original CARTESIUS research.

3.2 Aging Priorities: Functional Redundancy in CARTESIUS

The shortcomings discussed in section 3.1.6 can be considered as a failure to properly bound the functional role of the CARTESIUS management agent. In a practical sense, however, they are primarily the result of shifting the CARTESIUS project's priorities. The history of the original CARTESIUS prototype led to its implementation as a "Swiss Army knife" of traffic management that ultimately sought to optimize all components of the transportation system on the basis of estimation models contained within CARTESIUS itself. This development was largely driven by the need to create a functional prototype that could demonstrate the efficacy of the core CARTESIUS algorithms for finding solutions to non-recurrent congestion for a given set of potential control actions. Unfortunately, this development path also resulted in CARTESIUS assuming responsibility for functions outside its mandate. Thus, the CARTESIUS prototype became a system for optimizing traffic signals, ramp meters, and variable message signs using its own internal algorithms.

Assuming responsibility for all of these functions makes maintenance of the system challenging and makes a particular CARTESIUS traffic management agent's implementation susceptible to obsolescence since it is difficult to incorporate newly available technologies and algorithms into a monolithic agent. Furthermore, such feature concentration limits the scalability and fault-tolerance of the system. Finally, developing standards in the industry (Iteris, Inc, 2002) put the CARTESIUS prototype at odds with current and likely future of traffic management systems.

In an age when the deployment of research projects is a primary goal of sponsors, researchers would be well served to plan accordingly from the early stages of research. This is particularly true for research that ultimately targets one or more functional roles that exist in mature, complex operating environments such as ITS.

> **We recommend that continuing CARTESIUS research be guided by a well defined set of functional requirements derived from the known capabilities of the CARTESIUS approach and from the realities of the real-world technical and institutional landscape. For more details, see section 4.**

3.3 Finding Synergy with Existing and Developing Technologies

As has been suggested in earlier sections, the existing CARTESIUS prototype suffers from a lack of focus that can be alleviated by clearly defining CARTESIUS's role in traffic management. A side effect of proper tasking of CARTESIUS is that it will make it easier for CARTESIUS to find synergy with existing traffic management technologies, and more importantly, with developing research that has the potential to broadly impact the quality of traffic management systems.

3.3.1 Dynamic Demand Estimation

CARTESIUS algorithms ranging from incident diagnosis to control action selection are heavily dependent on an estimate of the current demand on each path in the managed network. In the existing prototype, the system uses a demand estimate obtained via a multi-step process involving a broad planning-based estimate of prevailing demand that is fed into a dynamic traffic simulator. The simulator iteratively estimates time-dependent path flows in the system based upon a user-equilibrium principle. This multi-step process is subject to significant estimation errors to the point that the demand estimate obtained is dubious at best.

Indeed, the results of the original evaluation assume that CARTESIUS has a very accurate estimate of actual demand in the system. For a CARTESIUS system to actually meet the promise indicated by the original research, a true real-time path-based dynamic demand estimate must be available to the system. UC Irvine is currently involved in the development of two tools that can provide such an estimate, the most notable being work to develop a dynamic path-flow estimator (Nie et al., 2005). Integration of such an algorithm into the CARTESIUS framework is critical to a successful deployment of CARTESIUS.

3.3.2 Dynamic Capacity Estimation

Because CARTESIUS defines a "problem" in the system as an imbalance between the prevailing demand and the available capacity at a given roadway section, knowledge of a section's capacity is critical to CARTESIUS's functioning. The existing prototype estimates capacity statically based upon the number of lanes at a section and general performance characteristics for each particular type of roadway. In reality, the effective capacity of a section is a dynamic quantity that changes over time depending on the operational status of the roadway. Such capacity can be estimated by examining available loop detector data to identify the various regimes of flow occurring on each section of roadway. CARTESIUS could potentially leverage such a dynamic model to compute available capacity dynamically. Such a model would be a natural extension to existing work to improve the analytical capabilities of the Testbed website.

3.3.3 Adaptive Local Control Methods

As previously mentioned, the existing CARTESIUS prototype computes all control actions internally. This approach ignores the potential existence of well-tuned and/or adaptive control systems that may be operating in the field. Ideally, CARTESIUS could leave the actual optimization to local controllers and instead focus on estimating the ability of the system to meet local and remote goals that are determined during the problem response algorithm. In this way, CARTESIUS could continue to evolve as new control algorithms come on-line, eventually serving as strictly a high-level tactical manager that acts to prioritize flows in the system such that individual local control systems find solutions that provide necessary capacity for short-term demand shifts dictated by the incident response.

Chapter 4

Envisioning the Way Forward for Cartesius

This chapter offers a plan for moving the Cartesius project forward based on the findings outlined in chapter 3. This plan places a priority on increasing the deployability of the concept across California's diverse jurisdictional landscape. In general, we recommend that Cartesius be reframed to serve primarily as a TMC-to-TMC information sharing protocol that supports the Cartesius view of traffic management as a DPS algorithm.

This plan places an emphasis on the details of the information sharing protocol instead of a top-level TMC management agent. Such a focus makes Cartesius a more inclusive paradigm that can play nicely with and augment existing systems. The initial role of the TMC management agent (what heretofore has been called the Cartesius implementation) will be reduced somewhat to be an advisory system that gives TMC operators an interface to qualitative and quantitative characterizations of the state of the system and offers a view of local and control actions in a distributed global problem solving context.

4.1 Simplification and Openness: The Case for Reimplementation

The core Cartesius philosophy is a powerful conceptualization of the multi-jurisdictional traffic management problem. However, the findings of chapter 3 argue that the current prototype is a dated realization of this philosophy that must be revised. This begs the question of whether the revision should entail a modification to the existing prototype, or whether a complete reimplementation is warranted.

The benefits of revision include:

- incremental changes can be tracked through testing so that a working prototype is always available and
- new work can leverage the existing code base. directly

The drawbacks of revision include:

- accepting core design decisions that may now be dated and

- accepting a costly development and deployment platform without the need for many of its features.

The benefits of reimplementation include:

- the ability to generate a modern design for implementation of the CARTESIUS philosophy—including a re-scoping of CARTESIUS functionality to be more in-line with industry standards and trends,
- an opportunity to form a synergy with complementary projects,
- the shift to a modern, open development platforms without the need for vendor lock-in, and
- an opportunity to optimize internal algorithms where possible.

The drawbacks of reimplementation include:

- the loss of a working prototype,
- one-time developer cost (offset by a more accessible code base), and
- the potential to introduce new bugs and inconsistencies in the system.

In our judgment these points support reimplementation primarily because it is clear that the existing CARTESIUS prototype attempts to do too much in incident management given its capabilities. This makes the benefits of reimplementation greatly outweigh the benefits of revision—particularly since the latter most revolve around the value of an existing codebase that solves problems outside its appropriate functional role.

As a result, we recommend focusing on CARTESIUS core strength as a system for managing conflicts between the potential actions of different jurisdictions and negotiating alternative strategies to mitigate those conflicts. This does not preclude using a CARTESIUS management agent to perform optimization, but the most feasible course of action for deployment is to emphasize CARTESIUS monitoring and general advisory role regarding multi-jurisdictional conflicts.

4.2 CARTESIUS as a organizing principle

That the CARTESIUS approach to multi-jurisdictional traffic management via DPS is a promising management strategy should not be in question; the results of the original research demonstrate the power and potential of the underlying concepts. The challenge lies in transforming these concepts into incrementally deployable components that are compatible with existing systems.

The real-world landscape is one of existing traffic management subsystems, each using a variety of variably standards-based approaches to managing traffic. For most jurisdictions, the emphasis is on *strategic optimization of control settings* (on roughly annual timescales) to match prevailing demands. Top-down management of non-recurrent congestion is generally limited to rapid deployment of emergency and operations services (to clear disruptions) and

information provision (to notify travelers of disruptions). Locally adaptive control schemes that obviate the need for centralized re-optimization are in limited deployment but are likely to increase in the future.

Whether more aggressive top-down management is warranted in operational contexts remains an open question. CARTESIUS research has certainly shown promise for centralized management through active re-optimization of field controllers and diversion-oriented information provision under varying assumptions and a variety of system failures. But no comparisons have been made to truly adaptive systems that rapidly respond to changing demand patterns.

Assuming such systems continue in development and eventually see broader deployment, what is the future role of top-down management approaches such as that used by the current CARTESIUS agent prototype? We believe that the answer lies in the strength of viewing multi-jurisdictional traffic management as a DPS process. Regardless of the local algorithms employed, multi-jurisdictional traffic management is inherently an instance of DPS. Each controller, or control subsystem takes action on the basis of local knowledge to achieve some local goal (e.g., minimizing delay for the managed subsystem). The extent to which these local goals assist or hinder system-wide goals is ignored for the most part.

For instance, virtually all adaptive traffic signal algorithms are based on dynamic programming approaches to estimating prevailing demands and adapting the control parameters of local controllers to best service those demands. These demand estimation procedures, however, will have difficulty responding rapidly to discontinuous changes is demand caused, for instance, by massive re-routing of freeway traffic to the arterial system. Foreknowledge of such rapid demand shifts could reasonably be added to local adaptive control algorithms to rapidly respond to demand shifts and prevent the oversaturation and queuing that can cripple arterial control systems.

The CARTESIUS philosophy offers an organizing principle for traffic management systems that lends itself to unobtrusive information sharing that can be leveraged by control subsystems, when feasible and/or desired. It also provides a mechanism for distributed diagnosis of interconnected problems in the system. In short, CARTESIUS becomes an organizing principle about how to conceptualize control resources in the transportation system.

The degree to which a given jurisdiction participates in the CARTESIUS system could vary from the most abstract level of information sharing down to detailed participation of each controller in the global problem solution. This flexibility makes CARTESIUS simpler to deploy incrementally by allowing managers to minimize the risk they take in participating with other CARTESIUS components.

4.3 CARTESIUS as an information sharing protocol

Adopting the CARTESIUS DPS philosophy requires a protocol for sharing high-level information related to traffic management between jurisdictions. We propose transforming the information sharing techniques used in the original CARTESIUS two-agent implementation into a general-purpose collection of messages that can be leveraged by any CARTESIUS aware component as it sees fit. We propose that this can be a model for general purpose TMC to TMC traffic management communication that may have relevance to developing standards

in the NITSA.

4.3.1 Assumptions regarding the partitioning of knowledge

As discussed in section 2.1.2, CARTESIUS was originally designed using a hierarchical frame-based representation of possible problems in the system that relied upon representing the transportation network as a collection of *network objects* whose dynamic state can be compared with the problem frame database. This is a somewhat unconventional approach to network representation that may or may not be convenient to a particular jurisdiction.

This highlights the broader problem of defining data representations for sharing information across jurisdictions. Any collection of jurisdictions that agree to participate in a collaborative management system must ultimately agree on how common knowledge across jurisdictions is represented. If CARTESIUS is implemented as a monolithic agent that acts to control all resources, this problem is mitigated to the extent that a custom, CARTESIUS-compatible network representation can be built for each implementation. This creates a higher cost for participation, however, that may not be palatable to potential participants. Ideally, a deployable CARTESIUS system would play nicely with a broad range of network representations that may be in use by existing management systems. One approach is to simply perform data transformations on-the-fly as information is exchanged between CARTESIUS components and external management systems.

With the above caveats in mind, the knowledge partitioning approach endorsed by the CARTESIUS philosophy is a good one that minimizes the need for a given jurisdiction to maintain datasets representing the assets of neighboring jurisdictions. This, in turn, supports the goal of allowing jurisdictions to control access to possibly sensitive data about their systems. The level of common knowledge needed is ultimately dictated by the types of information sharing required for CARTESIUS to perform its function, which we discuss in detail in the following section.

4.3.2 Required information sharing

The following sections broadly describe the message sets envisioned for information sharing. These message sets are generalized translations of the information sharing implemented in the original CARTESIUS prototype.

Messages related to the problem state

According to CARTESIUS's DPS approach, the characteristics (location, severity, etc) of problems in a particular jurisdiction's subnetwork are generally only the concern of that jurisdiction. The only exception to this rule is the occurrence of *derived* problems in which a primary cause of congestion leads to secondary problems elsewhere in the network. An example is freeway congestion that spills back onto the surface street network. The fact that surface street congestion is being caused by an incident on the freeway may be relevant to the mitigation efforts taken by the agency in charge of the surface street network. For instance, such knowledge might place a priority on diversion strategies over a strategy like

surface street metering[1]. This is of particular importance in a DPS framework as problems may be related across jurisdictional boundaries, therefore requiring problem information to be shared.

The approach to this problem taken by the original CARTESIUS implementation involves both how the system identifies and classifies problems in the network and how information is partitioned between agents. The CARTESIUS implementation currently uses a frame-matching process to match the measured state of discrete "network objects" to pre-defined "problem frames" that assign an expert-defined cause to perceived failures in the system. The particular cause assigned to a problem is fundamental to the determination of mitigation strategies by the problem solving algorithm. In addition to core operational problems such as those caused by incidents in particular locations, CARTESIUS includes a set of spillback problems that are inherently related to another problem in the system that is causing the spillback.

CARTESIUS uses straightforward logic based upon knowledge of the network topology to identify the relationship between a secondary spillback problem and the problem that is its primary cause. This logic, however, requires direct or indirect knowledge of all potential primary problems—some of which be in neighboring jurisdictions. The identification of inter-jurisdictional problem relationships, therefore requires information sharing between agents.

The CARTESIUS implementation solved this problem using a series of RPCs made from the agent analyzing the spillback problem to the agent analyzing potential primary problems. The first call asked the neighboring agent to determine whether the spillback problem derived from any of its local primary problems. The second call imports more detailed information about the remote problem(s) that was/were related to the local spillback problem(s).

As noted earlier, a solution based upon embedded RPCs does not provide a general purpose solution to the information sharing problem involving n agents. Although in this case, the possible relationship between two problems is always a binary relationship that can involve at most two jurisdictions, managing explicit connections between agents inside of core business logic is fault prone and difficult to scale. More recent communications patterns are available to facilitate this type of information sharing.

In particular, a publish/subscribe pattern may solve this problem while permitting all problem derivation computations to occur in the local agent (the one that needs to know the root cause of a spillback problem). In this revised implementation all necessary information about problems in a given jurisdiction is published to authenticated subscribers so that subscribers to that information (other neighboring jurisdictions) can use that data to determine relationships between problems.

In this revised approach, the local jurisdiction (the subscriber) is solely responsible for determining the relevance of information (i.e., the relationships between problems). This actually permits a greater decoupling of agents than the current RPC approach. Using this pattern will also remove the need for multistage communication between agents that is more prone to race conditions or locking in a truly multi-agent system. And while the proposed

[1]Surface street metering, or gating (Quinn, 1992), is a relatively recent concept by which excess capacity at surface street intersection approaches is used to store demand that could oversaturate portions of the network downstream (such as a ramp meter or other signalized intersection). The theory is that spreading queuing across the network will prevent oversaturation and the resulting failure of any particular location in the system.

approach alters some of the strict knowledge partitioning described in section 4.3.1, it does so in a manner consistent with the spirit of the original partitioning: namely, it does not violate a particular jurisdiction's control of its resources or sensitive data.

Possible approaches to this sort of publish/subscribe model include various types of event-driven messaging solutions, such as an Enterprise Service Bus (ESB) architecture, and well-known AI approaches such as a blackboard architecture. The best solution will depend on design choices made for other components in the system.

The problem state messages place certain requirements on participating jurisdictions to support data representation standards that include the following.

- Communicating jurisdictions must use an agreed-upon dialect regarding the characterization of problems and their causes. Ultimately, this demands some authoritative standard for characterizing incidents. We recommend initially using the approach used in the original CARTESIUS implementation.

- A problem dialect also requires common knowledge regarding network representation (e.g., a common set of network objects, a shared network graph, or at least a commonly agreed upon set of critical sections).

Messages related to strategy-based problem solving

There are essentially two functions in problem solving that must be supported by communications. The first is the broadcasting of conditions to other agents deriving from local strategies or control actions. The role of these messages in DPS is essentially to ask a remote jurisdiction to consider the impact of potential actions under consideration by a neighboring agency. In the original implementation, this information was used to allow remote agents to augment their solution search tree with an analysis any necessary local actions that are needed to support the remote agent's proposed action. The mutual exchange of information results in every traffic management agent developing an equivalent list of solutions (in which matching solutions between agents contain the same list of control actions).

The first function can easily be supported using a publish/subscribe communications pattern. Agents that wish to be notified of the assumptions (constraints) being made by other jurisdictions in the formulation of response strategies can simply subscribe to the authenticated event channel upon which that agency will broadcast constraints. When such events are received, a local agent can incorporate that information into its mitigation algorithm as appropriate.

Using this event-based model means that a particular agency needn't actively participate in global problem solving beyond simply notifying its neighbors when particular assumptions (and their associated constraints) will not be supported. In this way, an agency with more or less predetermined time-of-day control of its subsystems can still participate in CARTESIUS-based global problem solving by simply determining whether proposed assumptions can be met. Note that such a determination could be based upon a simple look-up table regarding, for instance, the capacity of the subnetwork to handle added demand.

The second function supported by communications is the mutual selection of a particular globally acceptable solution to the current problem state on the basis of local problem solving (using reduced global information). In our example, during the mutual selection of global

strategies, the subordinate agency can advise neighbors which mitigation solutions will be more likely to perform as expected based on the belief that their associated constraints will be met.

The case described above essentially corresponds to a situation in which a particular agency doesn't have an ability to take active mitigation steps, but they do have the ability to advise on the core performance of their system (in terms of its available capacity at a given time of day). This case is more likely to apply to small municipal jurisdictions that use conventional time-of-day traffic control systems but is illustrative of the fact that they can still offer important information to the formulation of global mitigation solutions.

Note that in the case when an agency cannot support another's proposed actions, it is likely not in the interest of the requesting party to unilaterally select that set of actions since their performance estimate will be based upon the assumptions that are known to be false.

Messages related to problem monitoring

These messages ultimately drive processing amongst the traffic management agents. In the existing two-agent implementation, processing is started when one agent determines that there is a change in the problem state. At this point, the agent notifies the other agent that it has detected a change in the system state that requires new analysis (again, via explicit RPCs). The change in problem state may be due to changes in real-time data received from sensors, external event-notification systems—such as the CHP's Computer-Aided Dispatch (CAD) system, or by internal events triggered by CARTESIUS internal models of congestion propagation.

The problem monitoring messages used in the current CARTESIUS implementation are simple; they simply notify the other agent that collaborative processing should begin. Necessary changes in the problem monitoring message set will be driven by the communications architecture adopted. Again, a publish/subscribe pattern is likely to be sufficient for broadcasting a single agent's determination that the problem state has changed. In modifying the protocol, however, care will need to be taken to ensure that participating jurisdictions can participate in any analysis in a manner that maintains the synchronization points in the information processing and sharing process discussed earlier and shown in figure 2.1.

4.4 CARTESIUS as a TMC operator's tool

From the perspective of the TMC, a CARTESIUS agent should be viewed as a high-level advisory system that listens to CARTESIUS-related messages and offers operators insights as to

- the state of the transportation system as a whole as it relates to the local jurisdiction,
- the role of a particular agency's jurisdiction in the performance of that system, and
- the collection of possible mitigation actions, given available resources, that are consistent with (do not conflict with) neighboring jurisdictions actions

The advisory role reduces the core operational responsibility of the system, taking it out of the direct control loop. The CARTESIUS agent's role as a direct controller of the system would be a jurisdiction-dependent decision and would require CARTESIUS to become an authenticated client to existing control subsystems. An example of such an implementation is one of the goals of PATH Task Order 5324/6324.

In this perspective, the role of the CARTESIUS agent is to offer view of the system from a DPS perspective. Its role would be to highlight potential conflicts given that perspective and, where mechanisms have been installed, broker solutions to those conflicts. What this means in a practical sense is that the entire system need not be re-architected to adopt the DPS view of traffic management for CARTESIUS to be useful. Furthermore, it would remove significant burdens from the CARTESIUS prototype to take active management of the system and instead would offer monitoring, advisory, and post analysis capabilities that leverage the features of the DPS viewpoint to offer a high-level view of conflicts in the global management of disruptive events in the transportation system.

This is a reworking of the original CARTESIUS agent's role, which sought to actively manage non-recurrent congestion in a jurisdiction from the top down. In place of the former "heavy-controller" approach adopted for the original agent, the reimplementation would be a lighter-weight tool that could more easily interoperate with existing or future systems. The CARTESIUS approach to monitoring problems could also make the agent useful for evaluating the performance of various TMC strategies and different levels of inter-jurisdictional coordination used during incident management.

Chapter 5

Conclusion

This report details the findings of research carried out under PATH Task Order 5313. This project was originally intended to perform a limited Field Operational Test (FOT) of the CARTESIUS multi-jurisdictional traffic management system prototype. During the course of the research, however, it was determined that the existing prototype was unsuited to a realistic field deployment and required some re-working for that purpose.

The shortcomings of the prototype are cross-cutting and range from architectural problems associated with software design decisions to the impracticality of core assumptions made to generate suitable control actions for incident management. These problems aside, we judge the core philosophy of CARTESIUS to provide a promising framework for inter-jurisdictional traffic management. In particular, the power of DPS approach proposed by CARTESIUS nicely maps to the multi-jurisdictional traffic management domain.

Given these findings, we recommend that a new implementation of CARTESIUS be developed using general purpose languages and communications protocols. Such a re-implementation has significant costs, but it also permits the implementation to be streamlined to take advantage of the most powerful features of the CARTESIUS philosophy while simultaneously adjusting the functional role of CARTESIUS from a top-level monolithic traffic management system to a flexible advisory tool that provides each participating jurisdiction with a DPS perspective on managing the transportation system.

We recommend the following steps for the continued development of CARTESIUS to meet the needs of Caltrans and other stakeholders.

- Development of a set of functional requirements for the new CARTESIUS software implementation that meets the needs of all stakeholders and, in particular, seeks to put the CARTESIUS project on a path toward eventual deployment in California's TMCs.
- Development of a software design that efficiently satisfies these requirements
- Implementation of that software design for the Testbed sub-network involving Caltrans–D12 and the City of Irvine.

These tasks will be incorporated into PATH Task Order 5324/6324, which is integrating the functions of CARTESIUS and the Caltrans Traffic Signal Management and Surveillance System (CTNET).

Bibliography

Akcelik, R. (1981). Traffic signals: capacity and timing analysis. Research Report ARR123, Australian Road Research Board, Victoria, Australia.

Charniak, E. and McDermott, D. (1985). *Introduction to artificial intelligence*. Addison-Wesley, Reading, MA.

Gazis, D. C. (1974). *Traffic control - theory and application*, chapter 3. John Wiley and Sons, New York.

Iteris, Inc (2002). National ITS architecture. Technical report, US Department of Transportation. URL `http://itsarch.iteris.com/itsarch/`.

Lesser, V. R. and Corkill, D. D. (1981). Functionally Accurate, Cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–96.

Logi, F. (1995). A software architecture for the integration of Advanced Transportation Management and Information Systems. Master of Science in Civil Engineering, University of California, Irvine.

Logi, F. (1999). *CARTESIUS: A Cooperative Approach to Real-time Decision Support for Multijurisdictional Traffic Congestion Management*. PhD thesis, University of California, Irvine.

Logi, F., Rindt, C. R., McNally, M. G., and Ritchie, S. G. (2001). Advancd transportation management system test bed for evaluation of interjurisdictional traffic management strategies. *Transportation Research Record*, 1748:125–131.

Logi, F. and Ritchie, S. G. (2002). A multi-agent architecture for cooperative interjurisdictional traffic congestion management. *Transportation Research, Part C: Emerging Technologies*, 10C(5–6):507–527.

Minsky, M. (1974). A framework for representing knowledge. Artificial Intelligence Memo 306, MIT AI Lab, Massachussets Insititute of Technology.

Nie, Y., Zhang, H. M., and Recker, W. W. (2005). Inferring origin-destination trip matrices with a decoupled gls path flow estimator. *Transportation Research*, 39B:497–518.

Quinn, D. J. (1992). A review of queue management strategies. *Traffic Engineering and Control*, 33(11):600–605.

Winston, P. (1975). *The psychology of computer vision*. McGraw Hill, New York, NY.

Appendix A

Additional Completed Work

During the course of this project, a considerable effort was undertaken to support the original project plan—a limited field operational test that included using Paramics as a proxy for the real-world in tests with TMC operators. The changed project scope means that this work is not directly relevant to the findings in chapter 3. This appendix summarizes the work completed, however, because these contributions will be useful in any future work with Cartesius.

A.1 Paramics Plugin Development

Work on several new Paramics plugins was completed during this project. These plugins include both anticipated and unanticipated improvements to the Paramics model. The primary improvement was the creation of a new driver route choice model under the influence of CMS information. A significant portion of Cartesius's control impact comes from diverting demand around incidents in the system. As a result, the simulator used to evaluate network performance under Cartesius management must include a CMS diversion model.

At the outset of this project, we intended to use the Paradyn plugin—essentially an overlay of the Dynasmart model atop Paramics—to model CMS effects. The capabilities and design of the Paradyn plugin, however, have proven to be restrictive and difficult to upgrade for the purposes of this project. As such, a new collection of plugins were developed to model driver routing. To speed development, we chose to implement these plugins using the `C++` language, for which a collection of data types and algorithms are available via the standard template library (STL) and other freely available libraries. The development of these plugins was carried out on a GNU/Linux system using a `gcc` toolchain. Though Quadstone Ltd. recently dropped Linux support for Paramics, these plugins use portable, well-known standard tools should be adaptable to a MS-Windows-based toolchain without undue difficulty

The new plugins developed are summarized below.

`multi-userdata` This plugin extends the standard paramics API userdata functions to allow for multiple userdata pointers to be associated with a paramics object. This prevents multiple, independent plugins from overwriting the data the store in the "userdata" slots

provided by the Paramics API for each datatype in the model.

`paramics-api++` This plugin provides a collection of C++ wrapper classes for the Paramics API. These wrapper classes encapsulate each portion of the paramics API in a single class (the API divisions are indicated by three letter character codes that differentiate between object types including classes of objects with multiple instantiations: nodes (NDE), links (LNK), vehicles (VHC), etc. and classes of "object" with a single, global instantiation: the network (NET), the model configuration (CFG), the graphical user interface (GUI), etc. The qp[sg] functions associated with each of these types are included as methods to the class.

`route-names` This plugin allows the definition of named routes (e.g., streets) through a paramics network. The use of named routes instead of lists of Paramics network nodes can provide a measure of flexibility in design for plugins that work with paths through the network. Modifications to a Paramics network can change the node numbering in the network, causing such plugins to require their path listings to be updated as well. By using named routes instead of Paramics nodes, these plugins can insulate themselves from minor to moderate modifications in the Paramics network. Furthermore, path storage in the form of routes can be significantly more efficient than node listings.

`path-routing` A plugin adds the ability to perform path-based routing of vehicles traversing the network. This is not a route choice model in itself, but a plugin that supports a large class of route choice models. In a nutshell, vehicles operate on the basis of a planned path, which is expected to come from a path-based driver decision model. If a planned path is defined for a vehicle, and the vehicle is making a decision at a point in the network that is included in the planned path, the vehicle will follow the path specified in the plan. If there is no planned path that informs about the current decision, then the paramics default path is used.

`cms` This plugin models CMS and their effects on driver behavior. CMS are modeled by defining a series of possible messages in the `cmsdata` file in the Paramics network directory. Each possible message contains a list of effects and the rules for applying them to individual drivers. When the simulation is running, these rules are checked for each driver passing the CMS and the resulting effects are applied.

`path-recorder` To use paramics in an iterated assignment context, the ability to archive used paths during a given simulation run is necessary. This plugin can also store historical link travel times to be used with a time-dependent shortest-path algorithm. An initial version of the path-recorder plugin was completed during the course of this project. It allows the analyst to have all vehicles save their paths to a text file, including departure time, departure link. This capability is necessary to be able to use Paramics for iterated assignment.

`path-based-historical-routing` Also needed for iterated assignment is the ability to have vehicle select paths based upon a historical travel history database (i.e., experience

from prior iterations). This plugin provides this functionality leveraging the `path-routing` plugin and the data stored by the `path-recorder` plugin.

`iterated-assignment` The iterated assignment plugin allows a the analyst to specify which path history database drivers should use during a particular simulation run. This plugin facilitates using Paramics for iterated assignment by allowing a series of batch simulation jobs to be executed automatically in sequence, with each job using the results from the prior run and updated the path database accordingly.

Coding this plugin identified some shortcomings of Paramics routing API have been identified that require workarounds. The most significant is the program's routing logic that plans two links ahead of time in an effort to allow vehicles to make needed lane changes. If the network contains short links, this two-link rule may not provide sufficient time for vehicles to make their movements. This can, in turn, cause secondary congestion as vehicles queue in through lanes waiting for a spot in turning lanes. The two workarounds are (a) code customized routing and lane changing routines (a difficult task) or (b) modify coded network geometry to do avoid the problem.

A.2 Updated Datasets and Knowledge

Demand Matrix for Testbed Area The existing demand matrix was out of date with respect to the current state of the real world system, and did not match the network updates made to the model last quarter. During this project, a new demand matrix using a windowed version of the OCTAM model was developed using the Tranplan package,