Multi-Modal Intelligent Traffic Signal System

System Design

University of Arizona (Lead)
University of California PATH Program
Savari Networks, Inc.
Econolite

Version 1.1

6/16/2015

RECORD OF CHANGES

A – Added, M- Modified, D - Deleted

Version Number	Date	Identification of Figure, Table, or Paragraph	Title or Brief Description	Change Request Number
1.0	2/25/2014	N/A	Initial Draft for Team Review/Development	
1.1	5/26/2014	All Sections	Integrated individual component design descriptions from team	
1.2	6/16/2015	All Sections	Updated design to match software that was developed. Software submitted to the OSADP.	

Table of Contents

List of Figures

List of Tables

1 Purpose of Document

This document contains the detail level system and software design for the Multi-Modal Intelligent Traffic Signal Systems (MMITSS). The approach taken to the design has been to create a high level component based design that ensures all of the requirements are addressed by one or more components. The component based design approach supports both reuse and customization in the software implementation. Each component can be developed by the best suited team member such that they can use any desired detailed design approach available to them and define the interfaces that enable communications with other components. Custom components can be developed where needed, such as at the controller interface that is different in California and in Arizona. The common components can be reused in each test network.

Version 1.2 of the Detail Design document has been modified to match the software that has been implemented by the University of Arizona portion of the MMITSS Team. The University of California, Berkeley PATH portion of the team elected to pursue an alternative architecture as well as implementing different interfaces to the traffic signal controller, different traffic and priority control methods. The University of California, Berkeley PATH team will submit a separate Detail Design document.

The MMITSS software is a prototype and as such is not intended to be fully ready for field deployment. However, the software design is such that additional error handling, management, and monitoring could be added to make it a fully deployable system.

2 Scope of Project

The Multi-Modal Intelligent Traffic Signal System (MMITSS) project is part of the Cooperative Transportation System Pooled Fund Study (CTS PFS) entitled “Program to Support the Development and Deployment of Cooperative Transportation System Applications.” The CTS PFS was developed by a group of state and local transportation agencies and the Federal Highway Administration (FHWA). The Virginia Department of Transportation (VDOT) serves as the lead agency and is assisted by the University of Virginia’s Center for Transportation Studies, which serves as the technical and administrative lead for the PFS.

The United States Department of Transportation (US DOT) has identified ten high-priority mobility applications under the Dynamic Mobility Applications (DMA) program for the connected vehicle environment where high-fidelity data from vehicles, infrastructure, pedestrians, etc. can be shared through wireless communications. Three of the applications (Intelligent Traffic Signal System, Transit Signal Priority, and Mobile Accessible Pedestrian Signal System) are related to transformative traffic signal operations. Since a major focus of the CTS PFS members – who are the actual owners

and operators of transportation infrastructure – lies in traffic signal related applications, the CTS PFS team is leading the project entitled “Multi-Modal Intelligent Traffic Signal System” in cooperation with US DOT’s Dynamic Mobility Applications Program.

The MMITSS Phase II project is divided into ten (10) tasks that include Detailed Design, System Development, System Integration and Laboratory Testing, Field Integration, Testing, and Evaluation, and finally a demonstration of the system in each the Arizona and the California test beds. The detailed design effort is described in this report.

3 Detailed Design Approach

The detailed design approach is based on a revised version of the high-level design definition of system architecture and component responsibilities. Each of the physical nodes has a defined set of system components that may be realized as programs (processes), configurations, or special physical components. Each component design is specified as follows:

1. Component Name: node_component
2. Overview of Functionality
3. Requirements
    a. Functional Requirements (reference)
    b. Derived Requirements
        i. Timing
        ii. Priority
4. Interfaces
    a. Required (input)
    b. Provided (output)
5. Functional Specification/Description
    a. (activity, state, class, etc. diagrams)
6. Example Applications (as appropriate)
    a. (sequence diagram)
7. Unit Test Descriptions (as appropriate - debugging and testing)
8. References (as appropriate)

Item 1 identifies the component name (see Section 4.2 below). Item 2 provides an overview of the functionality (responsibility) of the component. Item 3 is a reference to the requirements being satisfied, or partially satisfied, by the component and any additional derived requirements such as timing (processing time, latency time, etc.) or process priority requirements (e.g. high, medium, low priority). Item 4 identifies the required and provided interfaces for the requirement. Item 5 describes the functional specification of the component including any logic, states, etc. that may be expressed

as activity, state, or class diagrams. Item 5 provides an example interaction of the component with others (e.g. a sequence diagram) or an example if it helps explain the operation of the component. Item 7 provides a description of how the components can be tested. This includes laboratory and field testing. Item 8 provides any references to documents, papers, etc. that are relevant to the design.

4 High Level System Design (Revised)

The high level system design is defined by the physical and software components. The high level design in this document has been updated to reflect development and changes since the Phase I High Level design was completed.

The primary changes are in the Physical Architecture shown in Figure 1. The US DOT provided Security Certificate Message Server (SCMS) has been added. This server is interfaced to the RSE using IPv6 and is used to provide security certificates to trusted OBE's. There are two radio channels between an RSE and OBE. One radio is assigned to channel 172 (HALL – High Availability Low Latency) and is dedicated to broadcast of basic safety messages (BSM), signal phase and timing messages (SPaT), and MAP messages. The other radio is used for transactions, such as an OBE requesting priority (using a Signal Request Message – SRM) or and RSE providing signal status to a priority eligible vehicle (using a Signal Status Message – SSM) as well as providing Security Certificates. This radio is assigned to two channels (split assignment) including channel 178 (Control Channel) and channel 182 (selected channel for priority transactions). The transaction channel(s) could be used for other applications, such as Traffic Information Messages (TIM) and others. Ideally, a Wave Service Announcement (WSA) would be broadcast on channel 178 (Control Channel) notifying vehicles of services that are available on the service channels (174, 176, 180, and 182). [Implementation Note: only channel 182 is used for priority control transaction. The Wave Service Announcement and channel switching were not implemented].

There are two additional communications interfaces in the revised architecture. The first is to support wireless communication between the nomadic device and the RSE using wifi (802.11a,b,n) or DSRC (based on recent developments towards a DSRC capable smartphone device). The second is roadside communications between the sensor system and the MMITSS Roadside Processor. This capability was identified in the stakeholder interaction process and includes interface to a sensor system that might produce data that exceeds the traditional vehicle detection (e.g. loop) data that is used by the traffic controller. This data might be a loop count value, or vehicle trajectories. There are no standards for this type of data, but the capability might prove useful for MMITSS. [Implementation Note: No data was collected from sensor systems in this implementation, but the capability was considered in the design.]

The other architecture change is that the MMITSS Roadside Processor might be integrated with the RSE. In the original high-level design, there was a desire to support “thin” RSE radio communications. This implied that the actual RSE device would not have sufficient processing capacity to host the MMITSS intersection level applications. However, the Savari RSE (version 3.0) device has sufficient processing capacity to support the applications. The Caltrans selected MMITSS Roadside Processor was based on a device that might not be environmentally rugged and hence, not suitable for deployment in the Arizona test bed. Making the MMITSS Roadside Processor optional allows flexibility for the field test and deployment. It is assumed that both devices are based on Linux so that portability of the applications can be accomplished with minimal effort. [Implementation Note: The Arizona implementation runs on the Savari RSE version 3.0, but the interface between components was designed using sockets so that they could easily be relocated to an MRP.]

4.1 Physical Architecture

The Physical MMITSS architecture is shown in Figure 1 as a UML Deployment Diagram. This architecture is based on the Conceptual Architecture identified in the MMITSS Concept of Operations and MMITSS Systems Requirements documents. In the Unified Modeling Language (UML), nodes are shown as 3D blocks and represent physical devices that have a processor (at least one), memory, and physical interfaces (e.g. Ethernet, RS-232, or wireless – 3G/4G, 5.9GHz DSRC, or other such as CAN-bus). The basic architecture is applicable at each MMITSS controlled intersection. In a system that consists of several intersections, each intersection would have an RSE radio and traffic control equipment as shown in the deployment diagram.

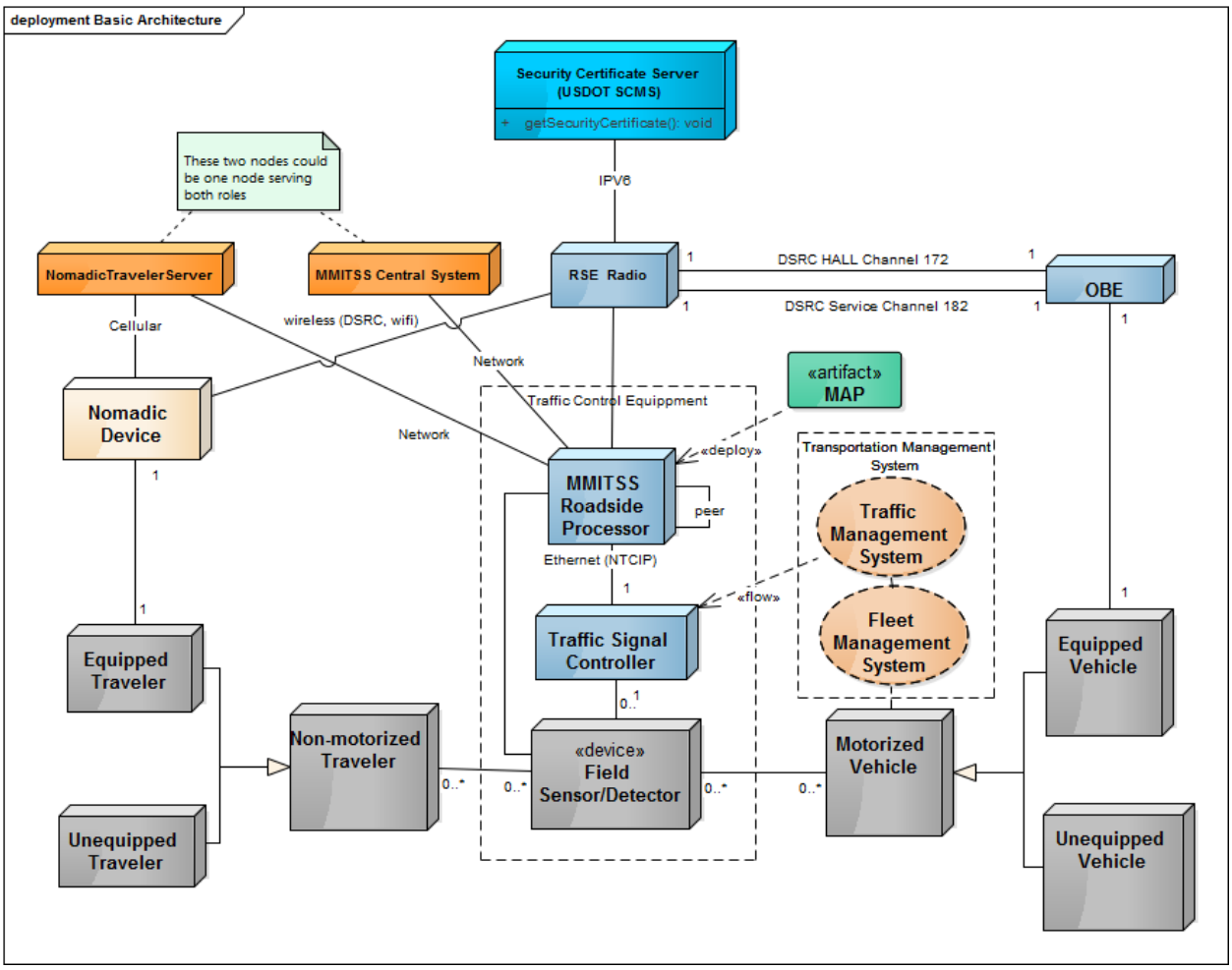


Figure 1 MMITSS Physical Architecture (updated).

The nodes have been shaded such that the light colored nodes are part of the connected vehicle system. Traffic Management and Fleet Management systems (or nodes that can be modified or assigned MMITSS responsibilities) and the gray colored nodes represent the vehicles and travelers. The orange colored nodes are the MMITSS Central System and Nomadic Traveler Server as described below. These two nodes may be realized as a single node for the test bed implementations.

In this view of the system, there are two types of travelers – motorized vehicles and non-motorized travelers. Motorized vehicles consist of passenger vehicles, trucks, transit vehicles, emergency vehicles, and motorcycles. This type of traveler includes any vehicle that must be licensed to operate on the public roadway. Non-motorized travelers include pedestrians, bicyclists, and other modes such as equestrians that are not required to be licensed to operate on the public roadway. These travelers are either unequipped or equipped, meaning that they have some type of OBE (On-Board

Equipment) or nomadic device that is connected vehicle (or MMITSS) aware and can operate as part of the traffic control system.

Motorized vehicles can be part of a fleet management system such as a transit management system, commercial freight management system, emergency vehicle dispatch system, or taxi dispatch, which is shown as a UML collaboration (oval in Figure 1) meaning that a collection of entities work together to perform the traffic management functions, but there may be many different systems involved in this collaboration.

The infrastructure based traffic signal control equipment consists of the traffic signal controller, field sensors/detectors, an optional MMITSS Roadside Processor (MRP). There are two traffic signal controllers models that will be utilized in the field installation: Econolite ASC/3 or Cobalt (NTCIP) (AZ) controllers and Type 2070 (Caltrans – AB3418) (CA). Each of these controllers offers different signal timing logic (software) and require different communications interfaces. The Econolite ASC/3 and the Cobalt controllers are based on NEMA standards and support NTCIP over Ethernet communications. The Caltrans Type 2070 controllers are based on a Caltrans standard and support AB3418 over serial RS-232 communications. Both networks utilize loop detectors for vehicle detection. The MMITSS Roadside Processor (MRP) is a Linux based general-purpose computer (see Section 4.1.1 below for details) [Note: this devices was specified for the California test bed].

The RSE Radio is the hardware device that is responsible for managing all of the 5.9GHz DSRC communications between the vehicles and the infrastructure. The Arizona network utilizes Savari RSE units.

The OBE is a hardware device deployed on the vehicle. MMITSS will be developed and tested using Savari MobilWave units (called aftermarket safety devices - ASD) for the OBE. These units are general purpose and provide a powerful and flexible platform for development and testing. The product data sheet for the OBE is in Appendix A: Savari OBE Product Specification Sheets.

The larger traffic management system is shown as an UML collaboration in Figure 1. The RSE is a general communications processing node that coordinates messages from the various modes of travelers. The MMITSS Roadside Processor (MRP) is a general purpose computer that can host the core intersection level infrastructure applications for MMITSS. The RSE contains (deploys) the MAP artifact, which is the digital description of the intersection geometry and associated traffic control definitions.

Both motorized and non-motorized travelers can be detected by the Field Sensor/Detector node at the intersections using a variety of detection technologies, including inductive loop detectors, video detection, microwave, radar, pedestrian push

button, etc. The detection system at an intersection provides information to the traffic signal controller that stimulates the control algorithms. For example, a vehicle that triggers a detector will call a signal control phase for service or extension. A pedestrian may activate a pedestrian push button to request the traffic signal pedestrian interval associated with a crosswalk movement. The direct communications path from the field sensor to the MRP allows the communication of information from the sensor. This information might include vehicle count from presence detectors or possibly vehicle trajectories from radar based sensors in the future. [Implementation Note: No direct sensor data was used in the MMITSS implementation.]

Each of the systems that can be active participants in the MMITSS (e.g., connected vehicle, Traffic Management, and Fleet Management) can have different responsibilities, and in alternative system designs some of these responsibilities can be assigned to different components. In the discussion presented here, the basic operating functions will be reviewed and the alternative assignments will be explored in the detail design effort.

4.1.1 MMITSS Roadside Processor

The MMITSS Roadside Processor (MRP) selected for the California test network is described by the following specification:

```
Jetway NF9E-Q77 Mini-ITX Motherboard, Q77 Express vPro iAMT,
   LGA1155, Ivy Bridge
Enclosure, power supply: Super Case MI-100BK Mini-ITX Case
Digital I/O board: PCIe-IIRO-8
CPU chipset, fan: Intel Core i5-3570 Ivy Bridge 3.4GHz
   (3.8GHz Turbo Boost) LGA 1155 77W Quad-Core Desktop
   Processor Intel HD Graphics 2500 BX80637i53570
Disk drive: Seagate Constellation ES ST500NM0011 500GB 7200
   RPM 64MB Cache SATA 6.0Gb/s 3.5" Enterprise Internal Hard
   Drive -Bare Drive
System memory: Transcend 8GB (2 x 4GB) 240-Pin DDR3 SDRAM
   DDR3 1333 Desktop Memory Model JM1333KLN-8GK
Operating System: Ubuntu Server 12.04.2 LTS
```

These components will be assembled into a field deployable roadside device. [Note: this processor is planned for deployment in the California test bed. The temperature range of these units has a maximum range of 140F. This processor was not selected for use in the Arizona test bed and the Savari StreetWave RSE had sufficient processing power to run the MMITSS algorithms].

4.1.2 RSE Radio

The Arizona test bed uses the Savari StreetWave version 3.0 RSE units. The units support two 5.9GHz DSRC radios and an Ethernet interface for communications. The product spec sheets are available in Appendix A: Savari RSE Product Specification Sheets.

4.2 Software Components

Figure 2 shows the primary software components of MMITSS. The software is grouped according to the deployment nodes. The OBE (on-board equipment) is deployed on each equipped vehicle. The OBE_*name* software components provide the functionality required of the OBE. The OBE communicates with the RSE (roadside equipment) over one or more of the DSRC WAVE channels. The RSE_*name* software components provide the key wireless communication behaviors including sending and receiving messages, as well as the Service Advertisement and Security Certificate services. The MMITSS Roadside Processor communicates with the RSE using local Ethernet network, and with the MMITSS Central System and the Nomadic Traveler Service using a combination of field and backhaul communications. In the Maricopa County SMARTDrive network a fiber optic local network is provided between all of the test intersections and a leased T-1 backhaul connection is provided from the field to the MCDOT traffic operations center (TOC). In the Arizona Test Network, the MMITSS Roadside Processor is the Savari StreetWave processor and communicates to the intersection traffic signal controller through NTCIP over Ethernet communications. The MMITSS Roadside Processor hosts software components that perform the core intersection level functions of MMITSS. These components include the MAP and SPaT broadcast manager, the equipped vehicle tracker, the priority request server, traffic control logic, the interface to the traffic signal controller (NTCIP), and the components that communicate status to the Nomadic Traveler including status. The MMITSS Roadside Processor also hosts the performance observer component.

MMITSS Detail Design

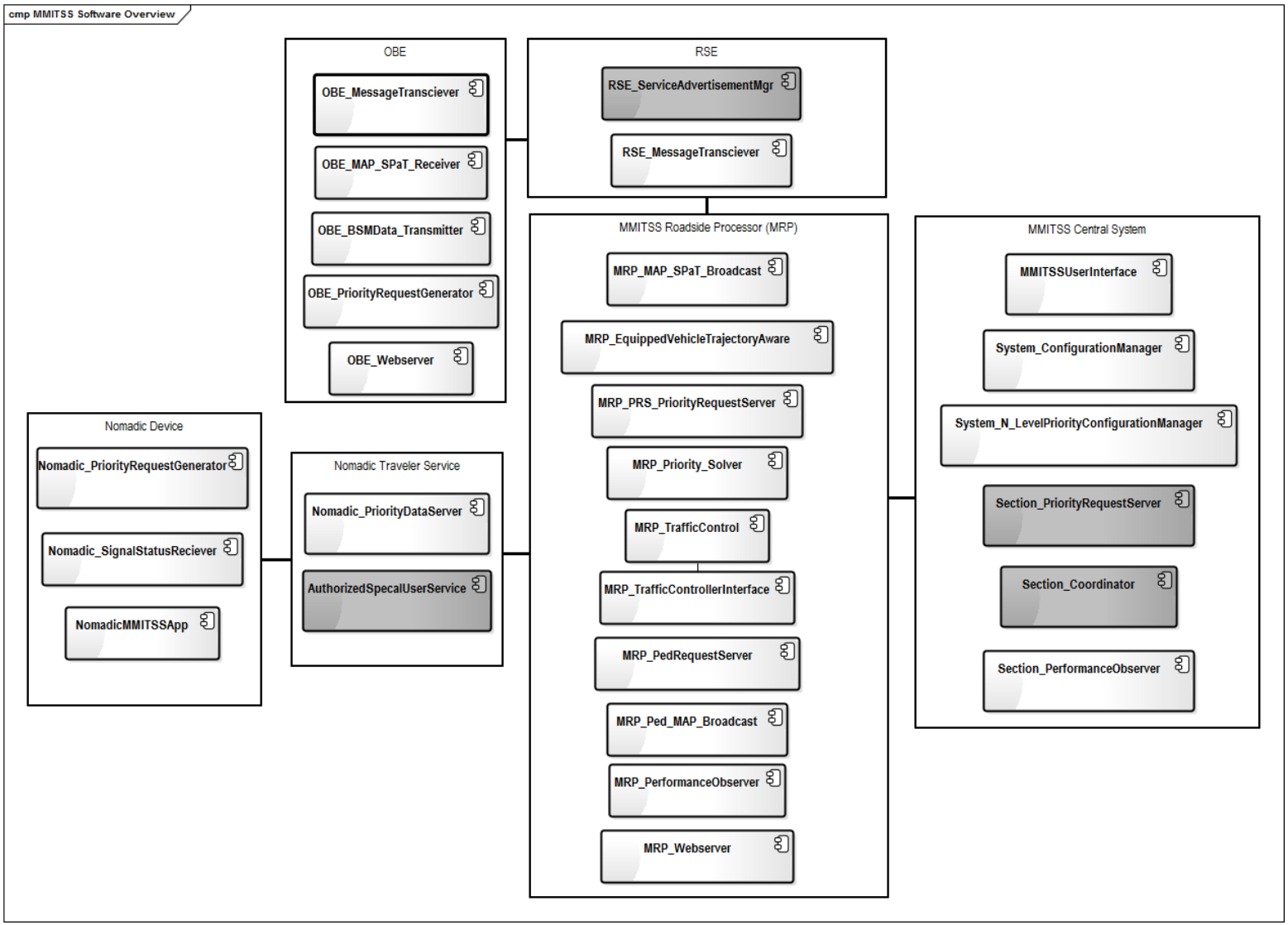


Figure 2 MMITSS Software Components.

MMITSS Detail Design

The MMITSS Central system hosts software components that provide a user interface, configuration manager, N-level priority policy configuration manager, and section level priority server and signal coordination [Implementation Note: The section level priority server and the signal coordination components were not implemented]. The MMITSS Central system also hosts the section level performance observers.

The Nomadic Traveler Service hosts two software components that provide the priority data service that relays intersection data (e.g. MAP and SPaT data) to nomadic devices and receives requests for service/priority from nomadic devices. The Nomadic Traveler Service also hosts the security and authorization service for nomadic devices that ensures these devices are associated with valid users and are authorized to receive special service if the user is a disabled pedestrian. [Implementation Note: The Savari SmartCross SBIR research project was leveraged to achieve most of the pedestrian service. A simple pedestrian application, called the MMITSS Pedestrian App, was developed separately from the SmartCross application. The AuthorizedSpecialUserService was not implemented].

The Nomadic device is assumed to be a smart phone, Android based, and hosts an applications (app) that uses two components. One component requests service/priority and the other collects and makes status information available for display on the app.

The System_PerformanceObserver component was removed from the component design. It was decided that section level performance was more meaningful at this stage in the development of MMITSS.

4.3 WAVE Communication

4.3.1 Overview of Functionality

WAVE messaging on the Road-side Equipment (RSE) is used for communication with On-board Equipment (OBE) or After-market Safety Device (ASD), either of which is present in vehicles. It uses the DSRC protocol, originally standardized as IEEE 802.11p-2010, and is now incorporated in IEEE 802.11-2012. The applications running on the RSE gather different kinds of data from OBE's and ASDs, process them, and make decisions such as signal timing plans and priority resolution. The RSE sends status (SPaT and SSM) data to the vehicles.

The MMITSS component based design has several components that are involved in the DSRC communications. The OBE contains two components that transmit DSRC messages (OBE_BSMTransmitter and OBE_PriorityRequestGenerator) and one component that receives DSRC messages (OBE_MAP_SPaT_Receiver).

Similarly, the RSE has the following DSRC components: RSE_ServiceAdvertisementMgr which is responsible to advertise the services provided by the RSE, RSE_MessageRX which receives the information sent out by OBEs and sends it to the components that process it, RSE_MessageTX which transmits data over DSRC and RSE_SecurityCertificateService which manages trust between OBEs and the RSEs. [Implementation Note: The RSE_ServiceAdvertisementMgr was not implemented as a software component, but rather is part of the Savari IEEE 1609 stack that is a standard part of the RSE operating system. At the time of development, the IEEE 1609 technical committee was communicating with the SAE DSRC technical committee to determine the proper operating of the Wave Service Advertisement. A request has been made for a MMITSS PSID, but the request was not granted. The RSE_MessageRX and RSE_MessageTX components were implemented at single components that perform both the transmission (TX) and the receipt (RX) functions. Several instances of this component is implemented for each message type, e.g. one for BSM, one for SRM, etc.]

Figure 3 shows a state diagram describing the activities carried out through WAVE messaging.

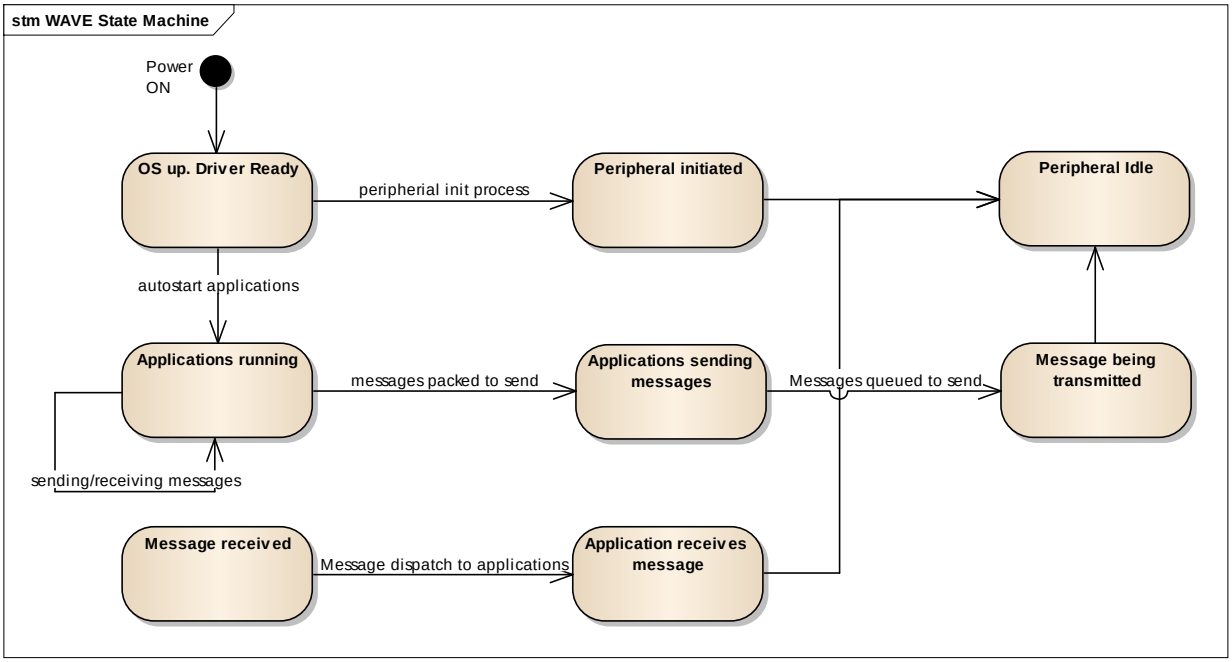


Figure 3 State diagram for WAVE Communication

The RSEs and OBEs/ASDs have to support communication between the applications running on them and external world. To ensure this, the WAVE messaging architecture should be capable of providing message agnostic wireless data transmission or reception services to other applications on it. A high level architecture description is shown in Figure 4.

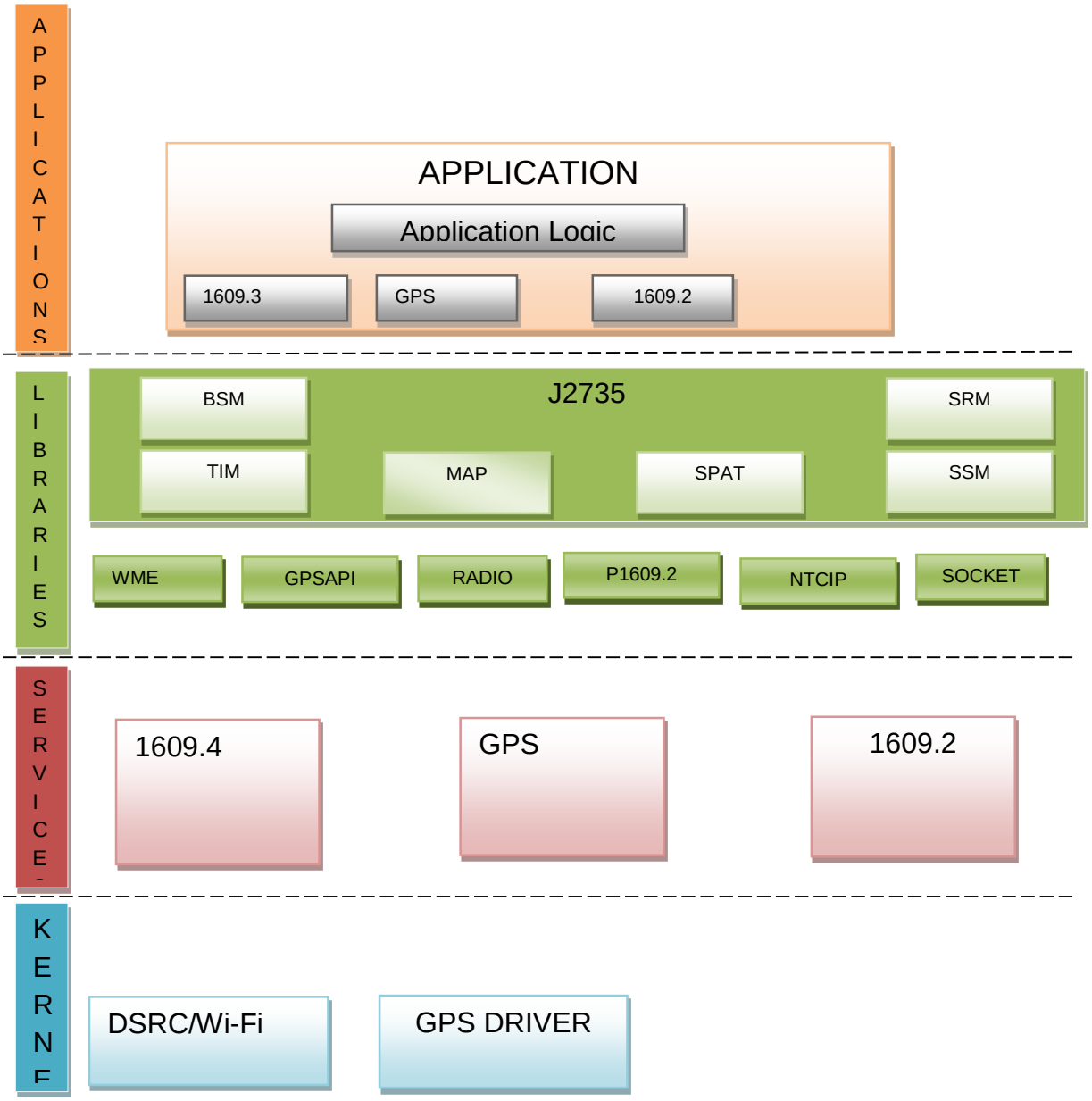


Figure 4 WAVE Architecture Diagram (Savari)

The wireless communication between devices is handled by the architecture presented in the Figure 4 above. It consists of 4 layers. The layers, excluding the application layer, are common for all applications and based on a set of services defined in software libraries. The Kernel layer handles the hardware level device drivers for the radios for wireless devices such as DSRC, WiFi and GPS. All messages on the device are sent/received via these devices. The services layer handles the messaging services such as security. Messages are processed by individual applications. These messages are handled by libraries based on the message type and the appropriate protocols. The application layer consists of applications that make use of the libraries to get the

messages that they want to process further.

Communications between MMITSS components and most modern transportation management system components (e.g. controllers, transit management systems, and fleet management systems) will be supported by IP based communications, such as Ethernet.

The DSRC and the WAVE protocol were designed to allow communications to take place without incurring latency penalties of over 200 microseconds. Since it is a short range communication protocol, typically less than 1000m, it can function well within this latency limit and provide a data rate up to 27 Mbps.

The data that needs to be broadcast is provided by the Service Advertisement Manager and SPAT/MAP broadcaster. The summary of messages that are transmitted from the RSE are as in Table 1.

Table 1 RSE Broadcast Messages.

Message	Abbr	From	Frequency	To	Radio	Channel
Signal Status Message	SSM	RSE	1 Hz	OBE/ASD	Ath0	174,176, 180 or 182
WAVE Service Announcement	WSA	RSE	1 Hz	OBE/ASD	Ath0	178
MAP	MAP	RSE	1 Hz	OBE/ASD	Ath1	172
Signal Phase and Timing	SPAT	RSE	10 Hz	OBE/ASD	Ath1	172
Basic Safety Message	BSM	OBE/ASD	10 Hz per Vehicle	RSE	Ath1	172
Signal Request Message	SRM	OBE/ASD	ASYNC	RSE	Ath0	174,176, 180 or 182
Security		Both	ASYNC	Both	Ath0	174,176, 180 or 182

The WAVE communication takes place with the help of 2 on-board DSRC radios, whose interfaces are named ath0 and ath1. Of these, ath1 will operate on the HALL (High Availability, Low Latency) channel 172 at a frequency of 5.86 GHz. As the name indicates, this channel is used to exchange high volume and/or high frequency data such as BSMs, MAP and SPAT.

The interface ath0 will be configured in an alternating mode where it can switch between 2 different channels giving a 50ms time slice to each of them. Of these two alternating channels, one is configured to operate on channel 178 at 5.89 GHz which operates as a control channel. This channel is used to relay information to devices such

as the channels on which they have to listen for requesting signal priority, security certificates, etc. Based on the services advertised as available, the RSE can transmit or listen on the 2nd alternating channel.

The other channel used by the ath0 interface is used to bind to different applications operating on the other available channels. However, the application with the highest priority will bind to it, not releasing the channel to other applications until it is finished transmitting. The messages and their channel configurations are as shown in Table 1. For the MMITSS prototype, channel 182 has been selected as the operating channel for sending SRM and SSM messages.

Each of the components defined in Section 5 are based on this WAVE messaging structure.

4.4 Software Component Interfaces

The interfaces between software components allows data to be transferred between suppliers and consumers. Two different architectures were considered for the flow of data between components: distributed loosely coupled components and a central data manager. The distributed loosely coupled components was selected for the MMITSS system since most data is shared between two or three peers and not required by all of the peer components. [Implementation Note: The University of California, Berkeley PATH team decided to use a central data manager approach late in the development process. The interfaces are intended to be the same, but the data will be brokered by the central data manager. PATH felt that their experience with this architecture would allow it to support a broader range of functions that just MMITSS.]

The data transmission between each component is implemented by UDP sockets. The initialization, sending and receiving functions of a UDP socket are described below.

There are two types of interfaces: push (streaming) and request-response. The push interface is used when there is a stream of data, such as BSM messages that need to be processed as they arrive. The request-response interface allows the consumer to request data as need for processing. Figure 5 shows an example of how the interfaces are used. The RSE_MessageTransceiver is responsible for listening to the DSRC HALL channel and collecting BSM messages. It pushes the messages to the MRP_EquippedVehicleTrajectoryAware components (using the iBSM interface) where sequences of BSM’s are used to construct vehicle trajectories.

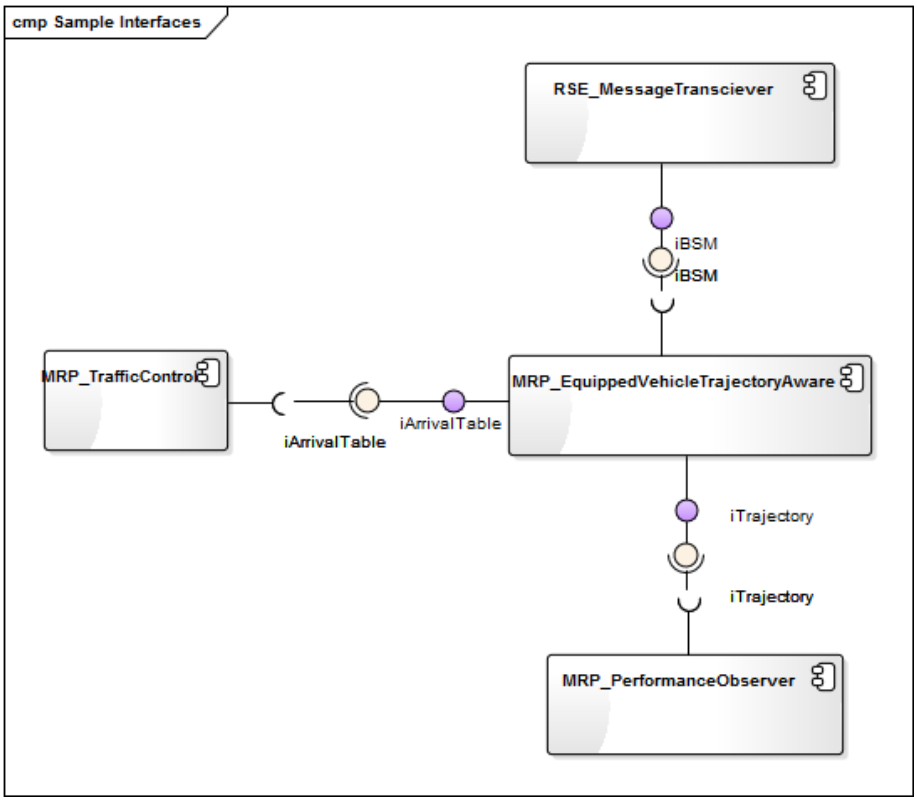


Figure 5 Examples of MMITSS component interfaces.

The MRP_PerformanceObserver will request the current collection of trajectories (using the iTrajectory interface) from the MRP_EquippedVehicleTrajectoryAware component. The MRP_PerformanceObserver will use the trajectory data, along with traditional vehicle detector data, to estimate performance measures over a defined period of time (e.g. 1 seconds for queue estimates, 15 seconds for stops, delay, travel time, etc., or longer periods of time as in is configured). Similarly, MRP_TrafficControl will request current collection of trajectories (using the same interface) as needed for the phase allocation algorithm.

4.4.1 Socket Initialization

This function initializes the UDP socket for sending and receiving messages. The sender address is set to INADDR_ANY which means the function will receive UDP packets from all IP addresses. In this example, the receiver's address is set to 127.0.0.1 which means the function will send the packets to local host, but this could be a remote host. Both ports are set to 30000 for the example, but this is specified for each interface. Users can define other IP addresses and ports.

Supporting Code:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <getopt.h>
#include <unistd.h>

void socket_init(int sockfd)
{
//Setup the socket, type: udp
       struct sockaddr_in sendaddr;
       struct sockaddr_in recvaddr;
       int numbytes, addr_len;
       if((sockfd = socket(AF_INET,SOCK_DGRAM,0)) == -1)
       {      perror("sockfd");
              exit(1);      }
       sendaddr.sin_family = AF_INET;
       sendaddr.sin_port = htons(30000);  //*** IMPORTANT: the provider should also
have this port. VISSIM in the lab and the RSE_MessageRX in the depoloyment ***//
       sendaddr.sin_addr.s_addr = INADDR_ANY;// any address using the same port
       memset(sendaddr.sin_zero,'\0',sizeof sendaddr.sin_zero);
       if(bind(sockfd, (struct sockaddr*) &sendaddr, sizeof sendaddr) == -1)
       {
              perror("bind");        exit(1);
       }

    recvaddr.sin_family = AF_INET;
    recvaddr.sin_port = htons(30000);
    recvaddr.sin_addr.s_addr = inet_addr("127.0.0.1");


    memset(recvaddr.sin_zero,'\0',sizeof recvaddr.sin_zero);
    int addr_length = sizeof (sendaddr);
}
```

4.4.1.1 Receive Data through UDP socket

This function receives data through UDP socket.

Supporting Code:

```
char Recvbuf[256];  //receive buffer
int Socket_Receive(int sockfd)
{
       int recv_data_len;
       recv_data_len = recvfrom(sockfd, Recvbuf, sizeof(Recvbuf), 0,
                  (struct sockaddr *)&sendaddr, (socklen_t *)&addr_length);
       return recv_data_len;
}
```

4.4.1.2 Send Data through UDP socket

This function sends data through UDP socket.

Supporting Code:

```
char Sendingbuf[102400]; //at most 100k
int Socket_Send(int sockfd)
{
    int numbytes;
    numbytes = sendto(sockfd, Sendingbuf, sizeof(Sendingbuf)+1,0,(struct sockaddr
*)&recvaddr, addr_length);
    return numbytes;
}
```

5 Detailed Component Designs

This section contains a detailed discussion of each of the software components in terms of their responsibility as defined by the associated requirements and description in the concept of operations document. Each component is given a name, traceability information to requirements, concept of operations, and/or stakeholder requirements, a brief description of the components responsibility, and any additional text useful for understanding the components role in MMITSS.

There are three documents that contain the key traceability information:

1. MMITSS Stakeholder Input Document, 8/3/2012, http://www.cts.virginia.edu/wp-content/uploads/2014/05/PFS_MMITSS02_Task2.2.pdf
2. MMITSS Concept of Operations, 12/4/2012, http://www.cts.virginia.edu/wp-content/uploads/2014/05/Task2.3._CONOPS_6_Final_Revised.pdf
3. MMITSS System Requirements Document, 3/7/2012, http://www.cts.virginia.edu/wp-content/uploads/2014/05/Task3._SyRS_4_PostSubmittal_V3.pdf

Section 7.6 of the MMITSS System Requirements Document provides a detailed traceability of each requirement to the Stakeholder inputs and Use Cases identified in the Concept of Operations.

5.1 Road Side Equipment (RSE) Components

5.1.1 RSE_SecurityCertificateService

5.1.1.1 Functional Requirements

Node: RSE	**Component Name:** RSE_SecurityCertificateService
	Traceability: System Requirements Section 6.6.3
Description of Responsibility: This component is responsible for providing security services on the RSE. It is responsible for ensuring trusted, eligible equipped vehicles (OBE) can send BSM and SRM messages to the RSE.	
Supporting Text: *The RSE_SecurityCertificateService will have a connection to a Certificate Server (SCMS) that will provide the security certificates and a revocation list.*	

5.1.1.2 Functional Description

This component is responsible for providing security services on the RSE. It is responsible for ensuring trusted, eligible equipped vehicles (OBE) can send BSM and SRM messages to the RSE and to deny services to untrusted equipped vehicles whose certificates appear on the revocation list.

The RSE_SecurityCertificateService is not a traditional software component. The security services are implemented as part of the IEEE 1609.2 stack that is in the Savari operating system services.

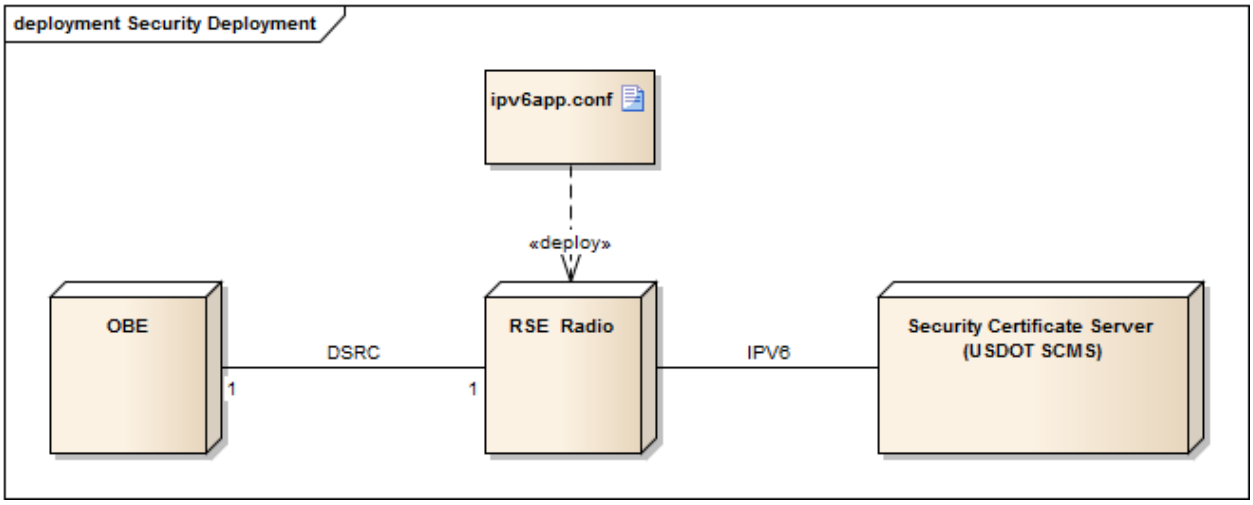


Figure 6 Security Service Architecture

The requirement that communication between the RSE and the SCMS be IPv6 is a design challenge for the field deployment. The current backbone communication in the Arizona Test Bed is based on the IPv4 protocol and existing equipment (field hardened Ethernet switches and the existing T1) does not support IPv6. Several options were

consider, including the use of a commercial Tunnel Broker service (commercial web service) will be used to transport security certificate requests from the RSE to the SCMS. This functionality requires the RSE_SecurityCertificateService component to support IPv6 to IPv4 mapping (e.g. similar to network address translation – NAT, but between different protocols). However, at the time of this document development, the option of upgrading the field network (Ethernet switches and existing T1) to IPv6 is being pursued. The development team was not able to test the Tunnel Broker service on the UA campus due to restrictions by the university IT services.

5.1.2 RSE_ServiceAdvertisementMgr

5.1.2.1 Functional Requirements

Node: RSE	**Component Name:** RSE_ServiceAdvertisementMgr
	Traceability: System Requirements Section 6.6.2
Description of Responsibility: This component is responsible for broadcasting a service advertisement message over a WAVE control channel that include information about the intersection ID, the MAP (GID) version number, and the service channel used for sending SRM and receiving SSM.	
Supporting Text: *The RSE_ServiceAdvertisementMgr broadcasts a service advertisement so that approaching vehicles will be aware of the MMITSS Traffic Control application(s) that are available. Vehicles that receive the advertisement will be instructed to switch communication channels to receive application specific information.*	

5.1.2.2 Functional Description

This component is responsible for broadcasting a service advertisement message over a WAVE control channel that includes information about the intersection ID, the MAP (GID) version number, and the service channel used for broadcasting the full MAP and GPS corrections as well as sending and receiving SRM and SSM. The RSE_ServiceAdvertisementMgr broadcasts a service advertisement so that approaching vehicles will be aware of the MMITSS Traffic Control application(s) that are available. Vehicles that receive the advertisement will be instructed to switch communication channels to receive application specific information.

Currently, there are no SAE J2735 message specifications for the WSA message. WSA is defined as part of the IEEE1609.4 standard specification, but it is not clear how Dynamic Mobility Applications, such as MMITSS, should implement the WSA. [Note: There was a teleconference organized by the SAE DSRC Message Framework subcommittee on May 23, 2014 to discuss this issue. No specific implementation issues were resolved, but there was very constructive discussion from a variety of experts].

Vehicles that are eligible for priority need to be aware that a signalized intersection provides special timing services, e.g. the MMITSS application. It is envisioned that they will become aware of the MMITSS application through a WAVE Service Announcement (WSA) that is broadcast on the DSRC Control Channel (178). The WSA will instruct eligible vehicles to switch to a specified service channel (e.g. 182) where the vehicles can send SAE J2735 specified Signal Request Messages (SRM) and receive SAE J2735 Signal Status Messages (SSM). It is assume that the vehicles will receive SAE J2735 MAP and SPaT data on the DSRC High Availability Low Latency (HALL) channel 172 and that these vehicles will be broadcasting Basic Safety Messages on the HALL channel.
The WSA will include a specified PSID and well as provider service context (PSC) information. The PSID notifies priority eligible vehicles that MMITSS is available. The PSC can contain information about the vehicle class and types that the operating agency permits to send priority requests. In addition, the security certificates must have matching credentials that specify the vehicle is authorized to send SRM messages.

The WSA use is currently being debated by the IEEE 1609 Technical Committee and the SAE DSRC Technical Committee. Future developments will clarify the use and protocol for WSA use.

5.1.3 RSE_MessageTransceiver

The RSE_MessageTransceiver and the OBE_MessageTransceiver are identical software components.

5.1.3.1 Functional Requirements

Node: RSE	**Component Name:** RSE_MessageTransceiver
	Traceability: C2001.001 (All vehicle data acquired by RSE through the WAVE communications)
Description of Responsibility: This component is responsible for sending/receiving properly formatted messages over a specified WAVE channel.	
Supporting Text: *The RSE_MessageTransceiver component is responsible for sending/receiving properly formatted SAE J2735 messages over a specified (or appropriate) WAVE channel. Example messages would include the Basic Safety Message (BSM), Signal Request Message (SRM), etc.*	

5.1.3.2 Provided Interfaces

The interfaces provided by this application are a Remote IP and a Remote Port. Through this, other applications can subscribe to send or receive WAVE messages.

5.1.3.3 Required Interfaces

This component requires interfaces to the DSRC library which can be called to communicate data over DSRC channel.

5.1.3.4 Functional Description

This component is responsible for sending and receiving properly formatted messages over a specified WAVE channel. The RSE_MessageTransceiver component is responsible for communicating properly formatted SAE J2735 messages over a specified (or appropriate) WAVE channel. Example messages would include the Basic Safety Message (BSM), Signal Request Message (SRM), MAP, Signal Phase and Timing (SPaT), Signal Status Message (SSM), etc.

One instance of this component is required for each radio/channel/message combination. For example, if the RSE is receiving Signal Request Messages (SRM) on channel 182 and sending Signal Phase and Timing messages (SPaT) on channel 172, this will require two instances of the RSE_MessageTransceiver application to be running with the appropriate configuration files specified as command line arguments. Each of these instances will support the required interfaces for communicating the SRM and SPaT messages to the subscribing components.

Each instance of this application will read the configuration parameters such as radio interface, channel number, message length, remote IP, remote Port and PSID from the config file. The radio interface and channel number are used to open a WAVE messaging handle with the underlying driver. This handle is then used to implement an event-based mechanism to handle incoming/outgoing messages.

The remote IP and Port is used to create a connection to the subscribers using regular UDP socket communication. The subscribers can use this connection to send/receive SAE-J2735 messages to/from the RSE_MessageTransceiver, which essentially broadcasts local UDP messages as WAVE packets or receives WAVE packets and sends them out to subscribers.

In sending mode, the RSE_MessageTransceiver will wait for the subscribers to send messages to it and upon receiving them, will broadcast out as WAVE packets on the specified radio interface and channel. In receiving mode, the application will be event-triggered by an incoming packet received on the specified radio interface and channel. This packet is then sent out to the subscriber as a UDP message over the specified IP and Port.

5.1.3.5 Configuration Data

Sample Configuration file is as follows. Inline comments explain the role of each option:

```
########################################
###   Sample config file for WME app   ###
########################################

### Configuration enabled: 0 -> Disabled; 1-> Enabled; Default -> 1
Enabled = 1

### Name of the Message: Default -> WME_MSG
MsgName = BSM

### Direction: 0 -> Outgoing; 1 -> Incoming; Default -> 1
Direction = 1

### Channel number to use for WME: Default -> 172
ChannelNumber = 172

### Interface to use for WME: Default -> ath0
InterfaceName = ath1

### PSID of the message which uses this config file: Default -> 0x10
PSID = 0x20

### Channel Mode: 0 -> Alternating; 1 -> Continuous; Default -> 1
ChannelMode = 1

### Remote IP to which socket is opened: Default -> 127.0.0.1
RemoteIP = 127.0.0.1

### Remote port to which socket is opened: Default -> 9999
RemotePort = 3333

### Protocol to use for socket: Default -> UDP
RemoteProtocol = UDP

### Provider Service Context: Default -> "PSC"
PSC = MMITSS

### Message Length: Default -> 128 bytes
MsgLength = 75
```

5.2 On-Board Equipment (OBE) Components

5.2.1 OBE_MessageTransceiver

The OBE_MessageTransceiver and the RSE_MessageTransceiver are identical software components.

5.2.1.1 Functional Requirements

Node: OBE	**Component Name:** OBE_MessageTransceiver
	Traceability:
Description of Responsibility: This component is responsible for sending/receiving properly formatted messages over a specified WAVE channel.	
Supporting Text: *The OBE_MessageTransceiver component is responsible for sending/receiving properly formatted SAE J2735 messages over a specified (or appropriate) WAVE channel. Example messages would include the Basic Safety Message (BSM), Signal Request Message (SRM), etc.*	

5.2.1.2 Provided Interfaces

The interfaces provided by this application are a Remote IP and a Remote Port. Through this, other applications can subscribe to send or receive WAVE messages.

5.2.1.3 Required Interfaces

This component requires interfaces to the DSRC library which can be called to communicate data over DSRC channel.

5.2.1.4 Functional Description

This component is responsible for sending and receiving properly formatted messages over a specified WAVE channel. The OBE_MessageTransceiver component is responsible for communicating properly formatted SAE J2735 messages over a specified (or appropriate) WAVE channel. Example messages would include the Basic Safety Message (BSM), Signal Request Message (SRM), etc.

One instance of this component is required for each radio/channel/message combination. For example, if the RSE is receiving Signal Request Messages (SRM) on channel 182 and sending Signal Phase and Timing messages (SPaT) on channel 172, this will require two instances of the RSE_MessageTransceiver application to be running with the appropriate configuration files specified as command line arguments. Each of these instances will support the required interfaces for communicating the SRM and SPaT messages to the subscribing components.

Each instance of this application will read the configuration parameters such as radio interface, channel number, message length, remote IP, remote Port and PSID from the config file. The radio interface and channel number are used to open a WAVE

messaging handle with the underlying driver. This handle is then used to implement an event-based mechanism to handle incoming/outgoing messages.

The remote IP and Port is used to create a connection to the subscribers using regular UDP socket communication. The subscribers can use this connection to send/receive SAE-J2735 messages to/from the OBE_MessageTransceiver, which essentially broadcasts local UDP messages as WAVE packets or receives WAVE packets and sends them out to subscribers.

In sending mode, the OBE_MessageTransceiver will wait for the subscribers to send messages to it and upon receiving them, will broadcast out as WAVE packets on the specified radio interface and channel. In receiving mode, the application will be event-triggered by an incoming packet received on the specified radio interface and channel. This packet is then sent out to the subscriber as a UDP message over the specified IP and Port.

5.2.2 OBE_BSMData_Transmitter

Node: OBE	**Component Name:** OBE_BSMData_Transmitter
	Traceability: C2001.001, 13.3.1, 13.3.2, 13.3.4, 13.3.5; *§5, §8, §11.1.1, §11.1.3, §11.2.1, §11.4.1, §11.5.1*
Description of Responsibility: The OBE_BSMData_Transmitter is responsible for collecting vehicle data from the GPS receiver and vehicle data systems (CAN bus) and formatting the data into a J2735 BSM Message.	
Supporting Text: *This component is responsible for getting vehicle data, GPS data, and forming the BSM message. (Part 1 and Part 2 as data is available). The OBE_BSMData_Transmitter component is responsible for broadcasting the data.*	

5.2.2.1 Provided Interfaces

This component does not provide any interfaces as it is a standalone application which does not provide any data to other components.

5.2.2.2 Required Interfaces

This component requires interfaces to the library to

- Receive GPS data to calculate position, speed and direction
- Transmit the captured data as a Basic Safety Message (BSM) over DSRC channel using the OBE_MessageTransceiver.

5.2.2.3 Functional Description

The OBE_BSMData_Transmitter is responsible for collecting vehicle data from the GPS receiver and, if available, vehicle data systems (CAN bus) and formatting the data into a

J2735 BSM Message. This component is responsible for getting vehicle data, GPS data, and forming the BSM message. (Part 1 and Part 2 as data is available). The OBE_BSMData_Transmitter component is responsible for broadcasting the data over the specified radio and channel (ath1 channel 172).

5.2.1 OBE_MAP_SPaT_Receiver

5.2.1.1 Functional Requirements

Node: OBE	**Component Name:** OBE_MAP_SPaT_Receiver
	Traceability: A1002, 13.3.1, 13.3.2, 13.3.4, 13.3.5; §5, §8, §11.1
Description of Responsibility: The OBE_MAP_SPaT_Receiver is responsible for listening for the MAP and SPaT messages and making the received data available to the other OBE components.	
Supporting Text: *The MAP and SPaT data are critical to the vehicle components related to requesting priority.*	

5.2.1.2 Provided Interfaces

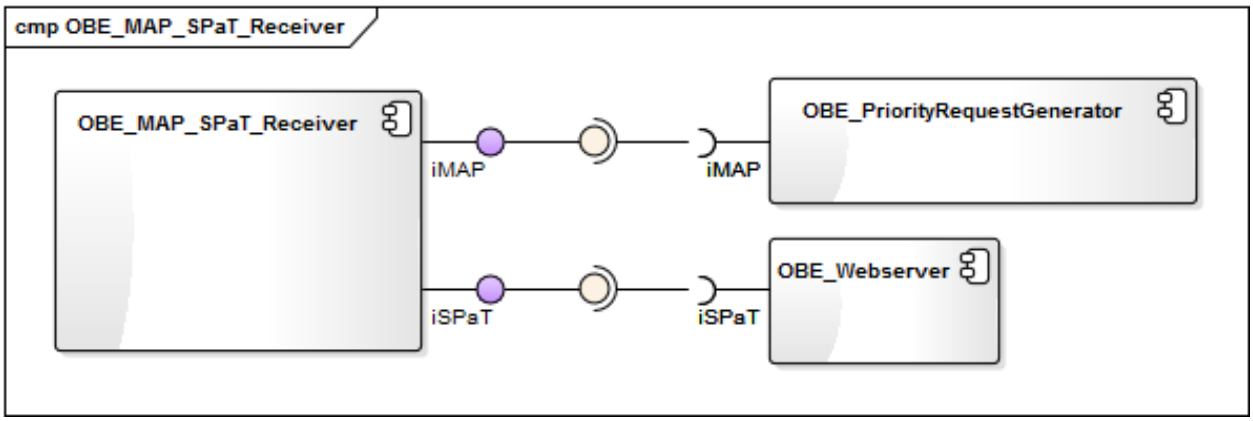


Figure 7 OBE_MAP_SPaT_Receiver Interface Diagram

5.2.1.3 Required Interfaces

This component requires interfaces to the library which can be called to receive data over DSRC channel.

5.2.1.4 Functional Description

The OBE_MAP_SPaT_Receiver is responsible for listening for the MAP and SPaT messages and making the received data available to the other OBE components. The MAP and SPaT data are critical to the vehicle component related to requesting priority.

5.2.2 OBE_PriorityRequestGenerator

5.2.2.1 Overview of Functionality

The OBE_PriorityRequestGenerator is a critical MMITSS component that is responsible for determining a vehicle's eligibility for priority, the level of priority to request, estimating the desired service time (e.g. estimated time of arrival at the stop bar). It is also responsible for updating any of the request information if there is a change in status or data.

An activity diagram showing the basic operation of the OBE_PriorityRequestGenerator component is shown in Figure 8. The process starts when a priority vehicle enters the range of a DSRC radio. The process gets the MAP data from the OBE_MAP_SPaT_Receiver. It then acquires the GPS data and determines its position on the MAP and calculates its expected time of arrival (ETA) and determines the desired inLane and outLane (if available). The vehicle determines if it is approaching or leaving the intersection of interest.

The process checks three conditions in order to decide if it should send a signal request message (SRM) or not. These conditions are:

- Has the vehicle speed changed out of a specific threshold? (vehicle is accelerating or decelerating)
- Is vehicle speed less than a specific threshold? (vehicle is approaching the queue)
- Is the speed of the vehicle zero? (vehicle is in the queue)
- Was the previous SRM delivered successfully?

If the vehicle speed has changed or the vehicle speed is less than a predefined threshold or if the vehicle stopped for the first time, the process decides send and updated SRM. The process waits to listen if the SRM has been delivered to RSE (by noting the receipt of a Signal Status Message (SSM). If it is not delivered, the process acquires the GPS data and sends another SRM.

If the vehicle is leaving the intersection, the value of isCanceled in the SRM is set to "True". The process broadcasts this new SRM to let the RSE knows that the vehicle has left the intersection.

5.2.2.2 Requirements

Node: OBE	**Component Name:** OBE_PriorityRequestGenerator
	Traceability: C1003.201, C1003.402, C1003.503, C1004.201, C1004.402, C1004.503, A2501, Use Case 13.3.2; Use Case 13.3.4; Use Case 13.3.5; *§4, §4.1.2, §4.1.4, §4.1.5, §5, §9.3.4, §11.0, §11.0.1, §11.0.2, §11.2, §11.4, §11.5*; 11.5.1;
Description of Responsibility: The OBE_PriorityRequestGenerator is a critical MMITSS component that is responsible for determining a vehicles eligibility for priority, the level of priority to request, determining the response mode of emergency vehicles, estimating the desired service time (e.g. estimated time of arrival at the stop bar). It is also responsible for updating any of the request information if there is a change in status or data.	
Supporting Text: *The OBE_PriorityRequestGenerator is an important and active component on the vehicle that is responsible for determining eligibility, level, desired time, and service phase or approach. The OBE may require communications with a fleet management system (external to MMITSS) or may have the priority policy information pre-programmed. All vehicles that are active in the requesting of priority are required to update the status of their priority request, especially when they are leaving the intersection and priority is no longer required.*	

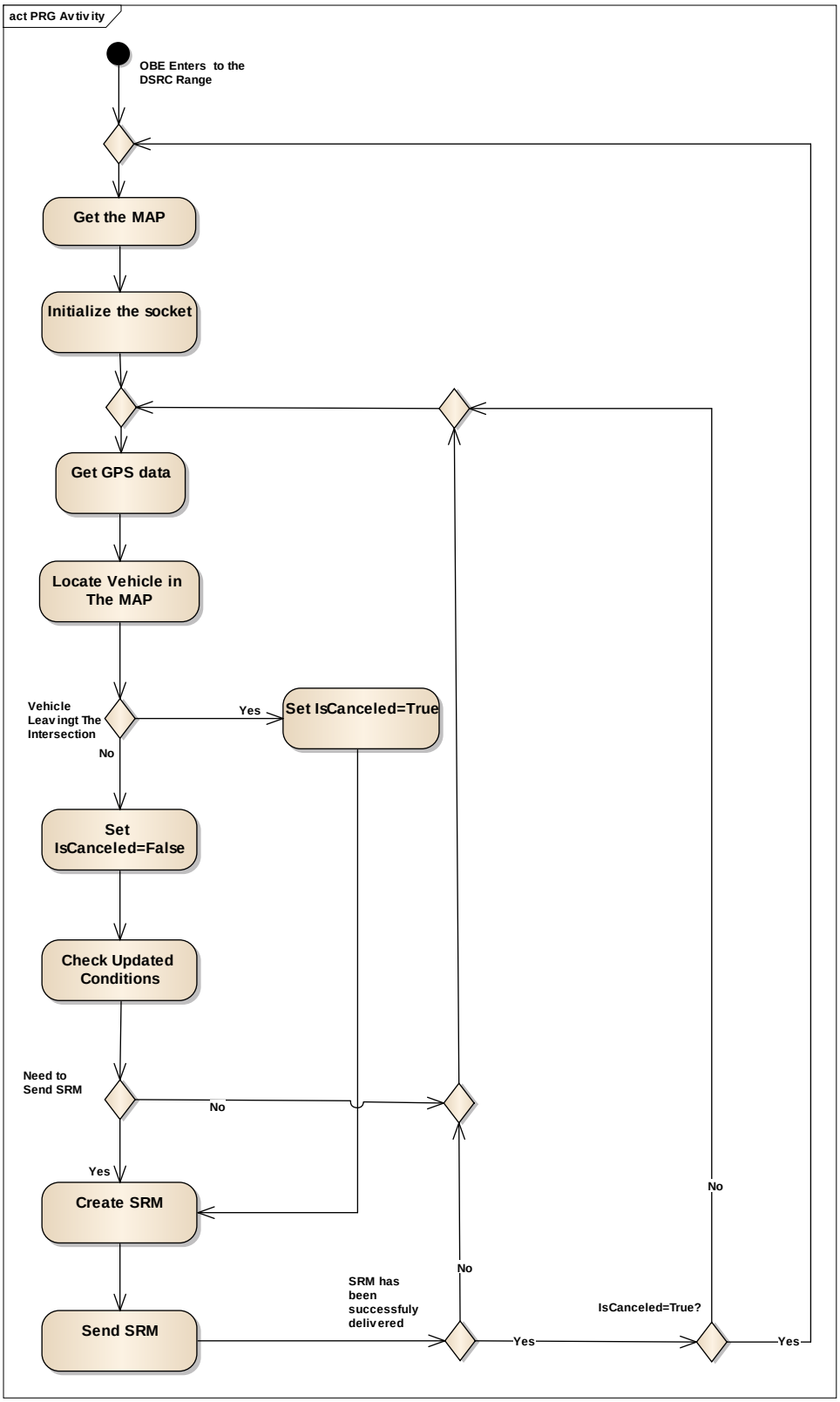


Figure 8 Activity diagram of OBE_PriorityRequestGenerator

5.2.2.3 Interfaces

The OBE_PriorityRequestGenerator interface is shown in Figure 9.

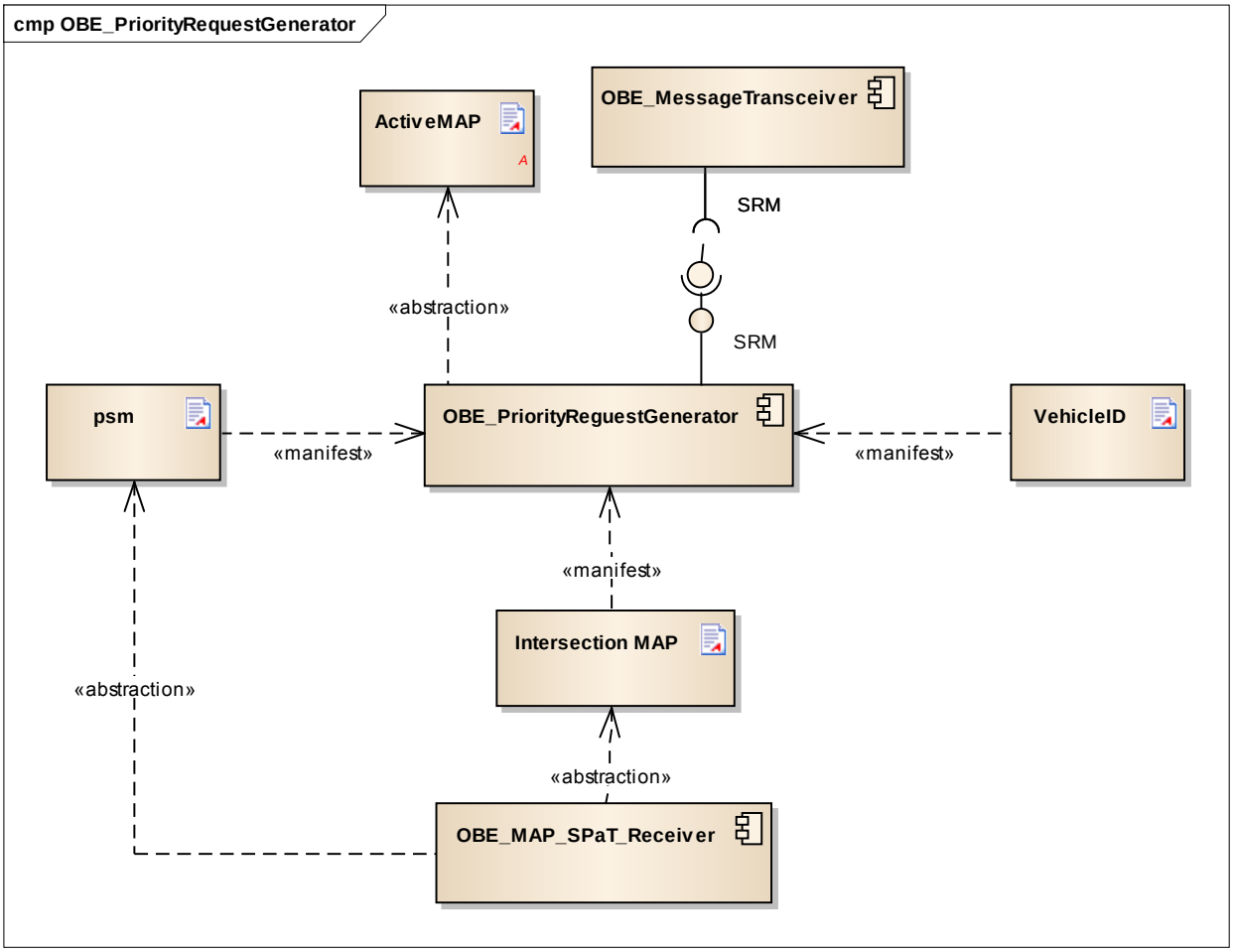


Figure 9 Interfaces of the OBE_PriorityRequestGenerator.

Required (input): MAP description file, psm file, vehicleID file

Read MAP description file

The OBE_PriorityRequestGenerator reads the MAP description from MAP_File_Name which is received by OBE_MAP_SPaT_Receiver and saved to a MAP data structure which is explained in the MRP_TrajectoryAwared component design. The MAP data structure is used to associate the lanes of each approach with the corresponding traffic

signal control phase. The OBE_PriorityRequestGenerator needs the map to know the vehicle relation to the intersection and create the SRM. (see MRP_EquippedVehicleTrajectoryAware for the definition.)

Read VehicleID description file

The OBE_PriorityRequestGenerator reads the specification of the vehicle from "vehicleID.txt" file. The vehicle ID, vehicle type and the name of the vehicle is described in this file. An example of this file is as follow:

```
1234 2 OBE1234
```

Type 1 is associated to EV, type 2 is associated to transit, and type 3 is associated to truck.

Read psm file

The "psm.txt" file is written by OBE_MAP_SPaT_Receiver and includes all of the acknowledged SRMs. The OBE_PriorityRequestGenerator reads the "psm.txt" file and determines if the sent SRM is acknowledged or not. If the OBE_PriorityRequestGenerator cannot find its vehicle ID in the psm, or if the MsgCnt in the psm is not the same as the MsgCnt in SRM, the OBE_PriorityRequestGenerator will send a new SRM with updated msgCnt. The OBE_PriorityRequestGenerator keeps sending the SRM until the MsgCnt in the psm matches the MsgCnt in SRM.

Provided (output): Signal Request Message (SRM), ActiveMAP file

Signal Request Message (SRM)

The OBE_PriorityRequestGenerator sends a SRM from a UDP socket through channel 182 from ath0. The following ASN.1 represents shows the structure of SRM message based on the SAE J2735 definition.

ASN.1 Representation:

```
SignalRequestMsg ::= SEQUENCE {
   msgID           DSRCmsgID,
   msgCnt          MsgCount,

   -- Request Data
   request         SignalRequest,
                   -- the specific request to the intersection
                   -- contains IntersectionID, cancel flags,
                   -- requested action, optional lanes data

   timeOfService   DTime  OPTIONAL,
                   -- the time in the near future when service is
```

```
                -- requested to start

   endOfService    DTime  OPTIONAL,
                   -- the time in the near future when service is
                   -- requested to end

   transitStatus   TransitStatus OPTIONAL,
                   -- additional information pertaining
                   -- to transit events

   -- User Data
   vehicleVIN      VehicleIdent  OPTIONAL,
                   -- a set of unique strings to identify the requesting vehicle

   vehicleData     BSMblob,
                   -- current position data about the vehicle

   status          VehicleRequestStatus  OPTIONAL,
                   -- current status data about the vehicle

   ...
           }
```

The following ASN.1 representation shows definition of the SAE J2735 SignalRequest data element.

ASN.1 Representation:

```
SignalRequest ::= SEQUENCE {
   -- the regionally unique ID of the target intersection
   id     IntersectionID,  --  intersection ID

   -- Below present only when canceling a prior request
   isCancel   SignalReqScheme OPTIONAL,

   -- In typical use either a SignalReqScheme
   -- or a lane number would be given, this
   -- indicates the scheme to use or the
   -- path through the intersection
   -- to the degree it is known.
   -- Note that SignalReqScheme can hold either
   -- a preempt or a priority value.
   requestedAction  SignalReqScheme OPTIONAL,
                          --  preempt ID or the
                          --  priority ID
                          --  (and strategy)
   inLane       LaneNumber OPTIONAL,
                          --  approach Lane
   outLane      LaneNumber OPTIONAL,
                          --  egress Lane
   type         NTCIPVehicleclass,
                          --  Two 4 bit nibbles as:
                          --  NTCIP vehicle class type
                          --  NTCIP vehicle class level

   -- any validation string used by the system
   codeWord     CodeWord OPTIONAL,
   ...
  }
```

Active MAP file
It is possible that the vehicle receives the MAP of two or more closely spaced intersections. The OBE_PriorityRequestGenerator writes the intersection ID of the closest approaching or leaving intersection into “ActiveMAP.txt”. This file will be used in OBE_GUI to show the vehicle is approaching to intersection or leaving.

5.2.2.4 Functional Specification/Description
The OBE_PriorityRequestGenerator is an important and active component on the vehicle that is responsible for determining vehicle type, desired time, and requested phase. Based on SAE J2735 terminology, the OBE_PriorityRequestGenerator is responsible for generating a message that contains the *vehicleType* which is an attribute of *VehicleIdent* class, a *request* of type *SignalRequest, and timeofservice* of type *DTime*.

An Emergency vehicle, a transit vehicle or a freight vehicle can acquire an active response from a center system outside of MMITSS. For example, consider an equipped transit vehicle that has communications with a fleet management system (external to MMITSS). The transit vehicle can acquire an active response status from the fleet management system. If the transit vehicle is ahead of its schedule it would not receive permission to request priority, hence it would not send a signal request message. If it is behind schedule, by a transit agency defined time, it would receive permission to request priority and would send a signal request message. For the MMITSS project, it is assumed that all of the priority eligible vehicles have permission and have acquired an active response status. Emergency vehicles are assumed to have an active response status

The OBE_PriorityRequestGenerator acquires the MAP from the OBE_MAP_SPaT_Receiver component. The MAP provides the intersection geometry and ID. The OBE can acquire its position using the OBE’s GPS unit. Based on the received MAP and the current GPS position, the OBE_PriorityRequestGenerator can determine the vehicles position with respect to the intersection. The OBE_PriorityRequestGenerator uses this information to determine its current lane (inLane) and to estimate the expected arrival time to stop bar. The OBE_PriorityRequestGenerator generates a SRM as a result. The OBE_PriorityRequestGenerator sends an SRM cancel message when OBE passes the stop bar.

Signal Request Message (SRM)
The OBE_PriorityRequestGenerator will broadcast Signal Request Messages (SRM) from priority eligible vehicles.

This is a type of vehicle to infrastructure message. SRM data including message ID (*MsgID*) and message count (*MsgCount*). The message ID is a data element used in each message to define which type of message follows from the message set defined by J2735 standard. Based on this standard, the message ID for SRM is 14. The message count is the number of time the SRM has been sent. The *MsgCount* data element is used to provide a sequence number within a stream of messages with the same *MsgID* and from the same sender. A sender may initialize this element to any value in the range 0-127 when sending the first message with a given DSRCmsgID. When the rest of the message content to be sent changes (e.g. service time), the *MsgCount* shall be set equal to one greater than the value used in the most recent message sent with the same *MsgID*. When the message content has not changed, the *MsgCount* is not changed. For this element the value after 127 is zero
The SRM contains other information such as signal request (*SignalRequest)*, requested time for service (TimeofService), end of time for service (EndofService), vehicle identification (vehicleIdent), and vehicle data (BSMblob). In the following paragraphs, these parts of the SRM are explained.

The *SignalRequest* is a data element that has several parts. It contains the intersection ID, vehicle approaching lane (*inLane*), vehicle egressing lane (*outLane*), and vehicle type. It also has information that allows a vehicle to cancel the request (*isCancled*).
The intersection ID is a globally and uniquely ID that defines an intersection within a country or region in a 32 bit field. Assignment rules for this value should be established based on regional assignment schemas that vary.
The expected Time of Arrival (ETA) is calculated in the OBE based on its speed and distance to the stop bar. The OBE_PriorityRequestGenerator uses ETA to set time for service (TimeofService) and end of time for service (EndofService) which is the desired time of service to support the movement of the equipped vehicle through the intersection.
The vehicle type is also part of SRM. Here is the NTCIP 1211 (version 7a) definition of Vehicle Class Type .

Priority Request Vehicle Class Type

```
priorityRequestVehicleClassType OBJECT-TYPE
        SYNTAX          INTEGER (1..10)
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
                "This object is the 'PRG requested' class type (relative priority of a request).
                The order of precedence is by class type with
                1                                                       highest;
                …
                10                                                      lowest
                A request with a higher class type will override a lower class type."
        DEFVAL { 10 }
::= { priorityRequestTableEntry 4 }
```

The SRM has 197 bytes of data. Each data element has specific space. The following ASN.1 representation shows the memory space for different SRM elements.

ASN.1 Representation:

```
SignalRequestMsg ::= SEQUENCE {

   -- Sent as a single octet blob
   --   blob1       SRMblob

   -- The blob consists of the following 197 packed bytes:
   --
   --
   -- msgID              MsgID                  -x-  1 byte
   -- msgCnt             MsgCount,              -x-  1 byte
   -- request            SignalRequest,         -x-  25 bytes
      -- id              IntersectionID           -x-  4 bytes
      -- isCancel        SignalReqSchem           -x-  1 byte
      -- requestedAction SignalReqScheme          -x-  1 byte
      -- inLane          LaneNumber               -x-  1 byte
      -- outLane         LaneNumber               -x-  1 byte
      -- type            NTSIPVehicleClass        -x-  1 byte
      -- codeWord        CodeWord                 -x-  16 byte
   -- timeOfService      Dtime                  -x-  4 byte
      -- hour             Dhour                    -x-  1 byte
      -- minute           Dminute                  -x-  1 byte
      -- second           Dsecond                  -x-  2 byte
   -- endOfService       Dtime                  -x-  4 byte
      -- hour             Dhour                    -x-  1 byte
      -- minute           Dminute                  -x-  1 byte
      -- second           Dsecond                  -x-  2 byte
   -- transitStatus      TransitStatus          -x-  6 byte
   -- vehicleVin         VehicleIdent           -x-  118 byte
      -- name             Descriptivename          -x-  63 byte
      -- vin              VINString                -x-  17 byte
      -- ownerCode        OwnerCode                -x-  32 byte
      -- ID               TemporaryID              -x-  4  byte
      -- vehicleType      VehicleType              -x-  1  byte
      -- vehcileClass     VehicleClass             -x-  1 byte
   -- vehicleData        BSMblob                -x-  38 byte
         }
```

5.2.2.5 Sequence Diagrams

Figure 10 shows the sequence diagram of OBE_PriorityRequestGenerator and its interactions with other software components in MMITSS.

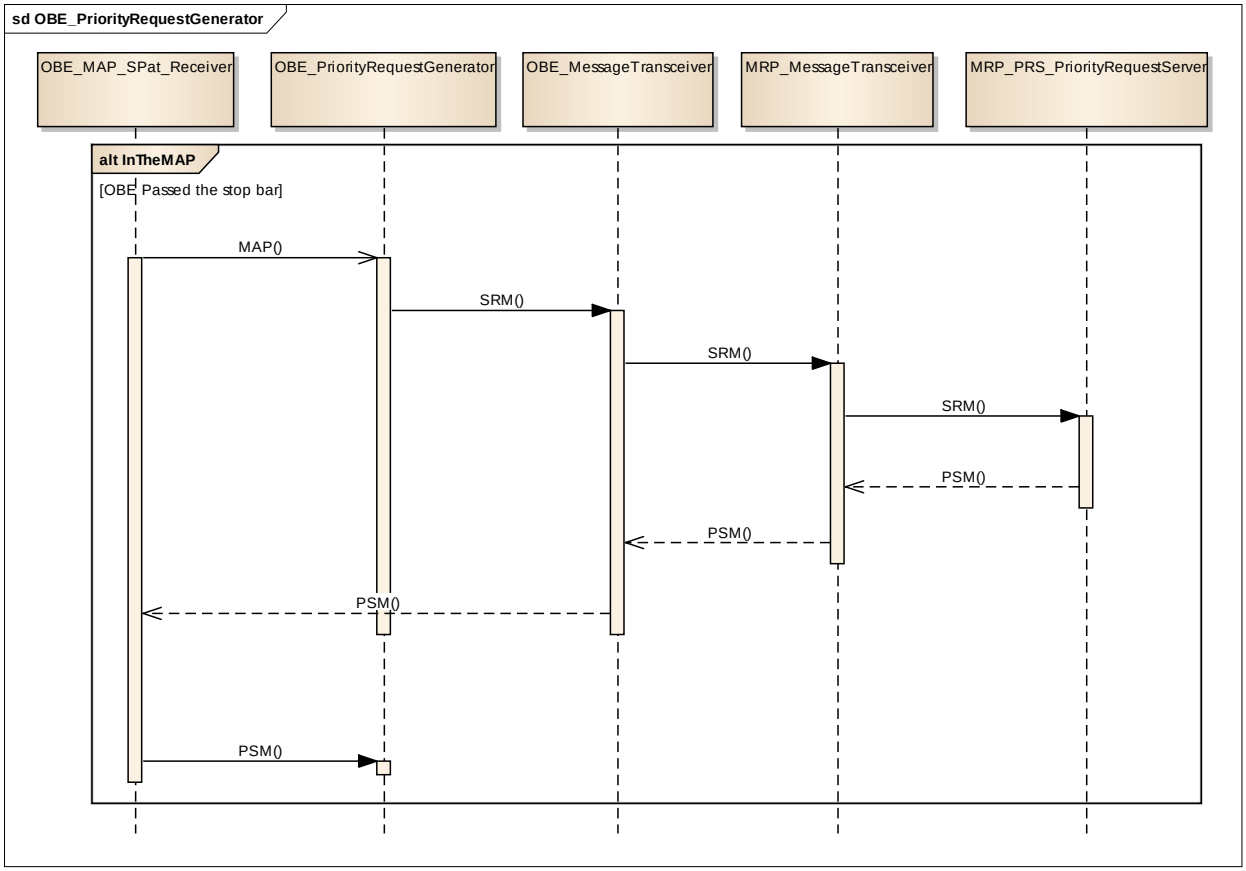


Figure 10 Sequence Diagram of the OBE_PriorityRequestGenerator.

5.2.1 OBE_Webserver

The OBE_GUI is an important component of the in-vehicle system for the MMITSS project. It allows developers and interested observers to visualize the status of the vehicle, the communication system, traffic signals, and priority control services.

5.2.1.1 Requirements

Node: OBE	**Component Name:** OBE_Webserver
	Traceability: System Requirements Section 6.3 (General Interface Requirements)
Description of Responsibility: OBE_ Webserver component is responsible for providing a human interface to the OBE so that the developers can visualize the data used by the OBE processes including vehicle data and priority request data.	
Supporting Text: *The ability to visualize the status of the OBE components is important in the development, testing, and demonstration of the MMITSS system. The use of a webserver and custom web pages allows the developers to easily visualize data. It is acknowledged that there are some delays or latencies in using web pages, but they have been valuable tools.*	

5.2.1.2 Interfaces

The OBE_ Webserver interfaces are shown in Figure 11

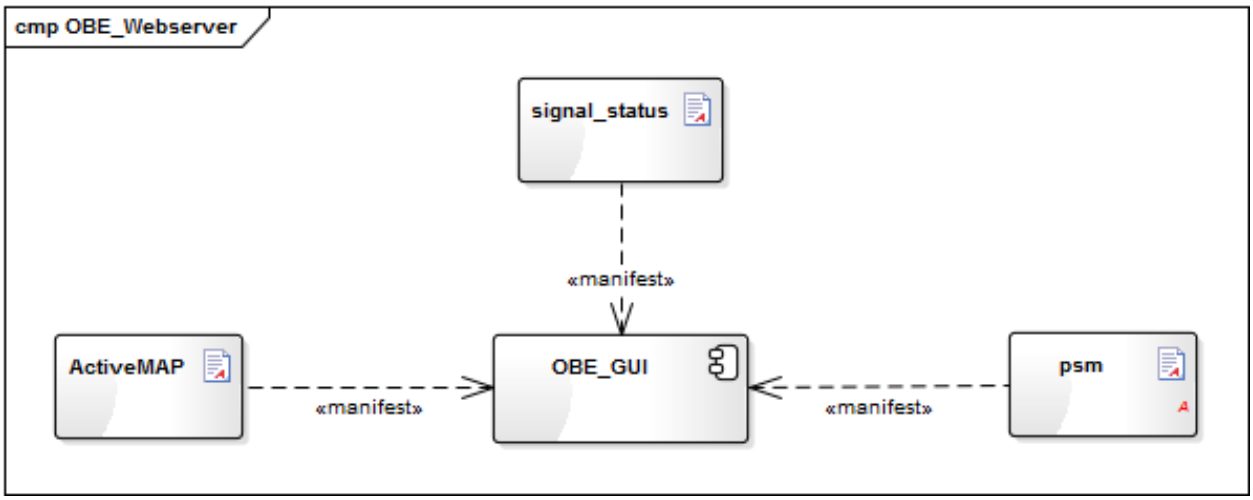


Figure 11 Interfaces to OBE_ Webserver

5.2.1.3 Required (input): Signal Status description file, Priority Status Message (PSM) description file, Active MAP description file

Read signal status file

The OBE_ Webserver component reads signal_status.txt file. This file contains intersection ID as well as signal status and pedestrian status. This file is generated by OBE_MAP_SPaT_Receiver component. The OBE_ Webserver shows signal_status.txt content in Signal Status table on the webpage. An example of this file is shown as follow:

```
301 signal_status 1 3 1 1 1 3 1 1   1 1 1 1 1 1 1 1
```

The intersection ID in this example is 301. The first 8 digits after `signal_status` show the signal status and the rest show pedestrian status. Digit 3 indicates the phase is green, digit 1 indicates it is red, and digit 4 indicates it is yellow. The signal is currently on green for phase 2 and 6 and there is no ped call.

Read priority status file

The OBE_ Webserver reads psm.txt file. The OBE_MAP_SPaT_Receiver component receives PSM from OBE_MessageTranceiver and writes the content of PSM into

psm.txt file. The file includes all of the acknowledged SRMs. The OBE_ Webserver shows the psm.txt content in the Active Request Table on the webpage.

Read active MAP file

The OBE_ Webserver reads ActiveMAP.txt file. This file is generated by OBE_PriorityRequestGenerator and includes the intersection ID of the RSEs that the vehicle is in the range of them. The OBE_GUI shows the ID of the closest approaching intersection in the MAP table on the webpage. An example of ActiveMAP.txt is shown as follow:

```
301 1428888550 1

302 1428888550 0
```

Each row in this file shows the intersection ID of the received MAP, the last time that MAP is received, and an indicator that shows the ID of the approaching intersection. If this indicator is 1, the vehicle is approaching the intersection; if the indicator is -1, the vehicle is leaving the intersection; if the indicator is 0, the intersection is not the closest approaching intersection for the vehicle. In this example, the vehicle is in the range of two intersection 301 and 302. But the closest approaching intersection is intersection 301.

5.2.1.4 Functional Specification/Description

The OBE_ Webserver provides a web page interface to display data to developers and interested observers for the purpose of visualizing the status of the vehicle. The OBE_ Webserver utilizes an open-source web server called Lighttpd (pronounced ”lightly”) that is a light performance web server package. This package has enhanced supports for implementing scalable CGI web applications and focuses on delivering dynamic content with less overhead than traditionally methods like Apache.

Figure 12 shows an example of the user interface that is located on a transit vehicle. As soon as the transit vehicle enters the DSRC range of RSE, the interface shows the intersection ID that the vehicle is approaching to. The green box “On Map: intersectionID” shows that the vehicle is on the MAP of the intersection. The web page shows the signal status of the approaching intersection in the “Current Signal Status” table. The “Current Active Request Table” shows all of the requests that are at the intersection. In the example shown in Figure 12, there are 2 requests that are approaching the intersection. In the “Active” column of the request table, there is a little green check mark that shows the EV vehicle has override the transit.

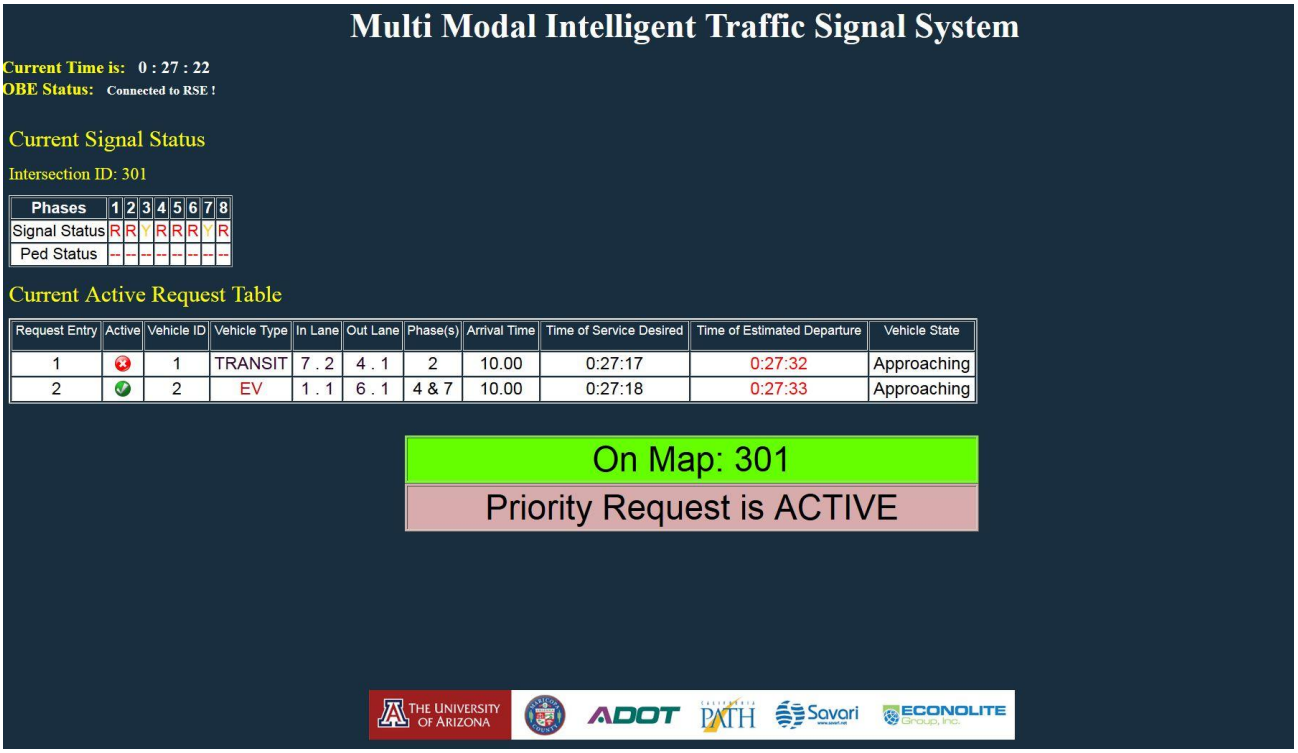


Figure 12 Example of the web page based user interface

5.3 MMITSS Roadside Processor (MRP) Components

5.3.1 MRP_MAP_SPaT_Broadcast

5.3.1.1 Overview of functionality

The MRP_MAP_SPaT_Broadcast component is responsible for constructing and broadcasting J2735 SPAT and MAPData messages. This component receives SPaT data pushed out from signal controller and pack the data into a standard SPaT message. This component reads the map configuration file and packs the data into a standard MAPdata message.

5.3.1.2 Functional Requirements

Node: MRP	**Component Name:** MRP_MAP_SPaT_Broadcast
	Traceability: C2004.001, C2005.001, C2005.302, 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; *§4.1.3, §5, §8, §9.1, §11.0, §11.2.1, §11.3, §11.4.1, §11.5.1*
Description of Responsibility: The MRP_MAP_SPaT_Broadcast is responsible for constructing and broadcasting J2735 SPAT and MAPData messages.	

Supporting Text: *The MCDOT SMARTDrive network uses Econolite ASC3 controller to push out SPaT data at each intersection.*

5.3.1.3 Derived requirements

Timing
The SPaT message is broadcast every 0.1 seconds and the MAPData message is broadcast every 1 second.

5.3.1.4 Interface
Figure 13 shows the required and provided interface.

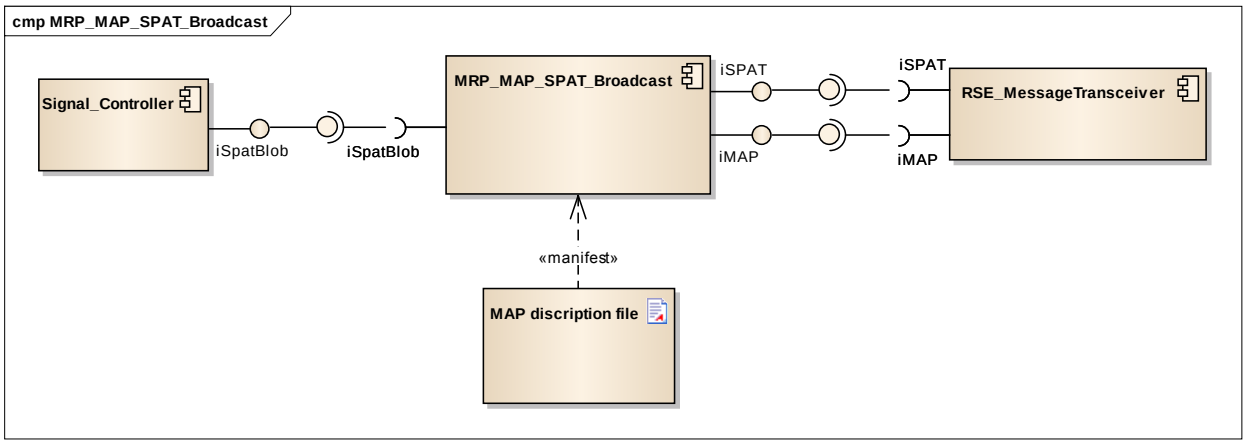


Figure 13 MRP_MAP_SPAT_Broadcast Interface Diagram

Required (Input)
Receive SPaT data blob from traffic controller

This function receives SPaT data blob from traffic controller. The SPaT data object includes data elements defined by Battelle SPaT standard (DTFH61-06-D-00007) and is sent to the *MRP_MAP_SPaT_Broadcast* for broadcast. Table 2 shows the structure of the SPaT Blob per Battelle SPaT definition (DTFH61-06-D-00007).

Table 2 SPaT Blob Structure (per Battelle Definition DTFH61-06-D-00007)

Object Identifier	Object Type
0x01	Intersection ID
0x02	Intersection Status
0x03	Message Timestamp
0x04	Movement
0x05	Lane Set
0x06	Current State

0x07	Minimum Time Remaining
0x08	Maximum Time Remaining
0x09	Yellow State
0x0A	Yellow Time
0x0B	Pedestrian Detected
0x0C	Vehicle or Pedestrian Count
0xFF	End of Blob

For the MCDOT SMARTDrive network, the Econolite ASC/3 controller provides a single message that contains all SPaT data by phase (Econolite, 2012).

In order to generate the SPaT data object by movement and lane set, the signal status messages received from traffic signal controller (Econolite ASC/3) need to be mapped to the lane movement status for each lane at the intersection. The mapping structure between signal phase and lane movements will follow Battelle definition (see DTFH61-06-D-00007).

Supporting Code:

```
#ifndef SPATBLOB_MSG_SIZE
    #define SPATBLOB_MSG_SIZE  (500)
#endif
char Recvbuf[SPATBLOB_MSG_SIZE];

int GetSPATBLOB (int sockfd)
{
       int recv_data_len;
       recv_data_len=Socket_Receive(sockfd);
       return recv_data_len;
}
```

Read Map from configuration file

This function reads the MAP description from MAP_File_Name and saves to a MAP data structure which is defined in MRP_EquippedVehicleTrajectoryAware.

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>
```

```
#include "NMAP.h"
int ReadMap (char * MAP_File_Name)
{
       MAP NewMap;
       NewMap.ID=1;
       NewMap.Version=1;
       // Parse the MAP file
       NewMap.ParseIntersection(MAP_File_Name);
       sprintf(temp_log,"Read the map successfully At (%d).\n",time(NULL));
       return 1;
}
```

Provided (output)

J2735 SPAT Message

This function encodes the SPAT data blob into a standard J2735 SPAT message and broadcasts through UDP socket.

Supporting Code:

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
char Sendingbuf[500]; //at most 500 bytes
int SendSPAT(int sockfd, SPAT_DATA Datablob)
{

    //Encode SPAT data blob to a J2735 Message before sending
    FillSpat(Sendingbuf, Datablob)
    //Send the SPAT
    int numbytes = sendto(sockfd, Sendingbuf,sizeof(Sendingbuf)+1 , 0,(struct
sockaddr *)&recvaddr, addr_length);
    return numbytes;
}
```

J2735 MAPdata Message

This function encodes the MAP data structure into a standard J2735 MAPdata message and broadcasts through UDP socket.

Supporting Code:

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include "NMAP.h"
```

```
char Sendingbuf[500]; //at most 500 bytes
int SendMAP(int sockfd, NMAP Newmap)
{

    //Encode MAP data structure to a J2735 Message before sending
    FillMAP(Sendingbuf, Newmap)
    //Send the MAP
    int numbytes = sendto(sockfd, Sendingbuf,sizeof(Sendingbuf)+1 , 0,(struct
sockaddr *)&recvaddr, addr_length);
    return numbytes;
}
```

5.3.1.5 Functional Specification/Description

Message Encoding/Decoding

Two methods are used to encode and decode the messages. Method one use Savari's J2735 message encoder and decoder to generate the message. Applications that call Savari tools can only be run on a Savari device. Method two uses a generic open source ASN.1 encoder/decoder tool (http://lionet.info/asn1c/asn1c.cgi) to encoder the SPaT and MAPdata messages. Applications that call the open source tool can be run on a generic Linux machine including Savari devices.

5.3.1.6 Reference

[1] Signal Phase and Timing and Related Messages for V-I Applications: Concept of Operation Document. Contract No. DTFH61-06-D-00007. July 2011.
[2] Signal Phase and Timing and Related Message Binary Format (Blob) Details. February 2012.
[3] SPaT MIB Support Document. Econolite. July 2012.

5.3.2 MRP_EquippedVehicleTrajectoryAware

5.3.2.1 Overview of Functionality

The MRP_EquippedVehicleTrajectoryAware component is responsible for constructing and storing equipped vehicle trajectories. This component is also responsible for filter received BSMs based on geo-fencing areas. Based on received BSMs and the intersection map, the MRP_EquippedVehicleTrajectoryAware component locates vehicles on the map and stores corresponding information including position, speed, heading, Estimated Time of Arrival (ETA) and requested phase. The trajectory data is used by two other MMITSS components including the MRP_PerformanceObserver and the MRP_TrafficControl component. These components will request trajectory data.

An activity diagram showing the basic operation of the MRP_EquippedVehicleTrajectoryAware component is shown in Figure 14. The process starts with UDP socket initialization. When a new message is available on the socket server, the component first determines the message type by unpacking the data string as a BSM. If the unpacking is successful which means received message is a BSM, then the component extracts the vehicle information from the BSM. Otherwise, the component further determines whether it is a trajectory request message from the MRP_TrafficControl component or MRP_PerformanceObserver component. If it is a trajectory request message, the component will vehicle trajectories of all the active vehicles in the list.

If the received message is a BSM, the temporary vehicle ID of the new BSM will be compared to the temporary vehicle IDs in the current active vehicle list. If the BSM comes from a new vehicle, a new vehicle trajectory is added to the list provided the vehicle is in the geo-fencing area 1. If the BSM comes from an existing vehicle, a timer will be used to decide whether to update the trajectory. The timer makes sure the components to update the trajectory every 0.5s to save computation effort. Then the position of the vehicle will be compared to geo-fencing area 2. If the vehicle is in the area, the component then locates the vehicle on the MAP and calculates its ETA and requested phase. The ETA is calculated by the current distance to the stop bar divided by the current speed. If the vehicle is in queue (speed equals to 0), the ETA is considered as 0. For the requested phase calculation, because intersections may have different phase assignment with geometry, a lane-phase mapping table is created for each intersection. Once the vehicle is located in a certain lane, the corresponding requested phase can be obtained. The vehicle’s active flag is then renewed which means its trajectory has been updated. Every 1 second, the process checks for the active flag of all vehicles in the active vehicle list. If a vehicle’s active flag hasn’t been renewed for 15 seconds, the vehicle will be deleted from the active vehicle list which means either the vehicle has left the DSRC range or the vehicle is not within the geo-fencing area. The cycle repeats indefinitely while there are messages available on the Socket to read.

5.3.2.2 Requirements

Node: MRP	**Component Name:** MRP_EquippedVehicleTrajectoryAware
	Traceability: C2006.001, C2007.001, C2007.202, C2007.404, C2007.505, C2008.001, C2008.202, C2008.303, C2008.404, C2008.505, F2019.001, F2019.202, F2019.403, A2103, 13.3.1, 13.3.2, 13.3.4, 13.3.5,11.0, §11.0.1, §11.1,§11.1.1, §11.2, §11.4, §11.1.4, §11.2.1, §11.2.2, §11.4.1, §11.5, §11.5.1, §5, §8
Description of Responsibility: The MRP_EquippedVehicleTrajectoryAware component is responsible for maintaining a collection of vehicles that are reporting status (BSM). Vehicle trajectories are persisted as long as a vehicle is actively reporting status.	
Supporting Text: *This is a critical component of MMITSS that maintains a collection of all equipped vehicles, of each mode/class, that are actively reporting status within the communication range of an RSE. It is possible that a vehicle will change id numbers (to maintain anonymity). The MRP_TrafficControl, MRP_PriortyRequestServer, and the MRP_PerformaneObserver will use information about the collection of equipped vehicles. Equipped vehicles should be associated with an intersection approach, lane, and traffic signal movement (phase) in a manner that is consistent with the current MAP.*	

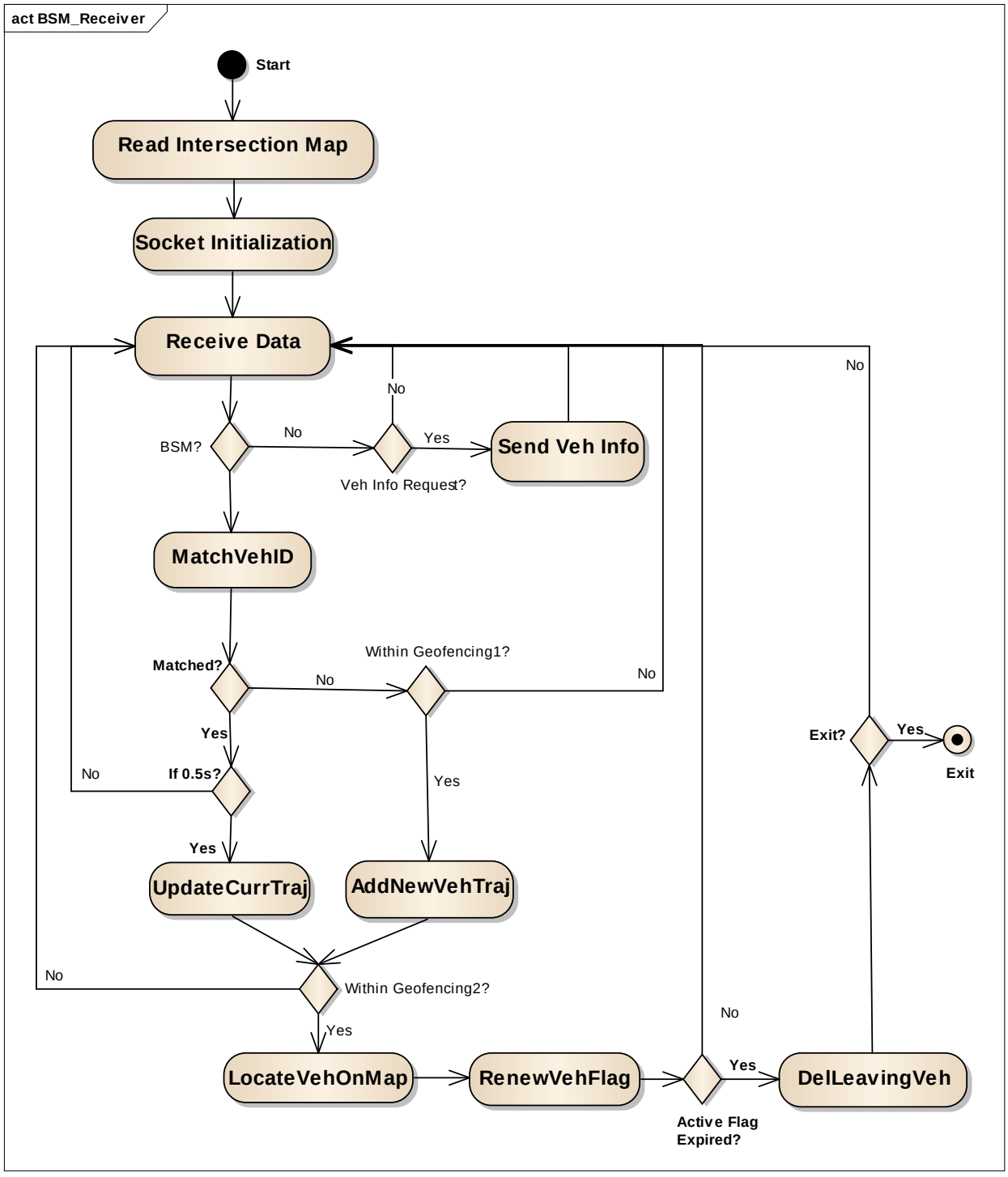


Figure 14 Activity Diagram of MRP_EquippedVehicleTrajectoryAware Component

5.3.2.3 Derived Requirements

Timing

The BSMs from each equipped vehicle are received every 0.1 second. The trajectory data for each vehicle is updated every 0.5 second. For privacy issue, a vehicle's trajectory will only be stored for a short period of time when the vehicle is "active". An active vehicle list is maintained and updated when a vehicle enters and when a vehicle leaves the range of the DSRC radio. A vehicle will be added to the list once its first BSM is received. A vehicle will be deleted from the list five seconds after it leaves the range of the DSRC radio and the trajectory has been requested by the subscribing client, e.g. MRP_PerformanceObserver component. If the vehicle changes its ID, it will be considered as a new vehicle. Its partial trajectory along with the old ID will be deleted after being requested by MRP_PerformanceObserver component.

Figure 15 illustrates the idea of maintaining only active trajectories. The figure shows trajectories of several vehicles on one approach to an intersection. At time t1, four vehicle trajectories are active (as shown as red highlighted trajectories). These trajectories become active as they enter the DSRC radio range and are still active at time t1. At time t2, the same vehicles, plus several others, are active (shown in red). At time t3, some of the previous trajectories are no longer active (now shown as grey) and new trajectories have become active.

The MRP_EquippedVehicleTrajectoryAware component shall have a capacity of processing BSMs from 100 vehicles simultaneously.

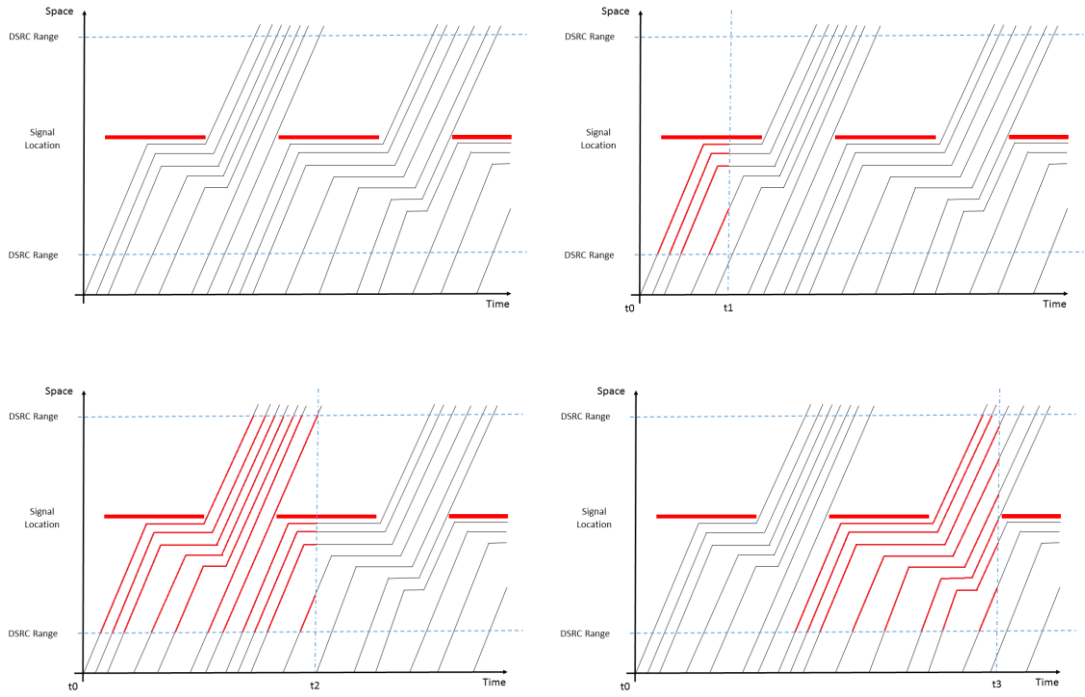


Figure 15 Request for Trajectory at Different Time Point

5.3.2.4 Interfaces

The interface of MRP_EquippedVehicleTrajectoryAware component takes BSM and MAP description file as the input and sends trajectory data to MRP_PerformanceObserver component and MRP_TrafficControl component. The structure of the interface is shown in Figure 16. The details of each interface are described below.

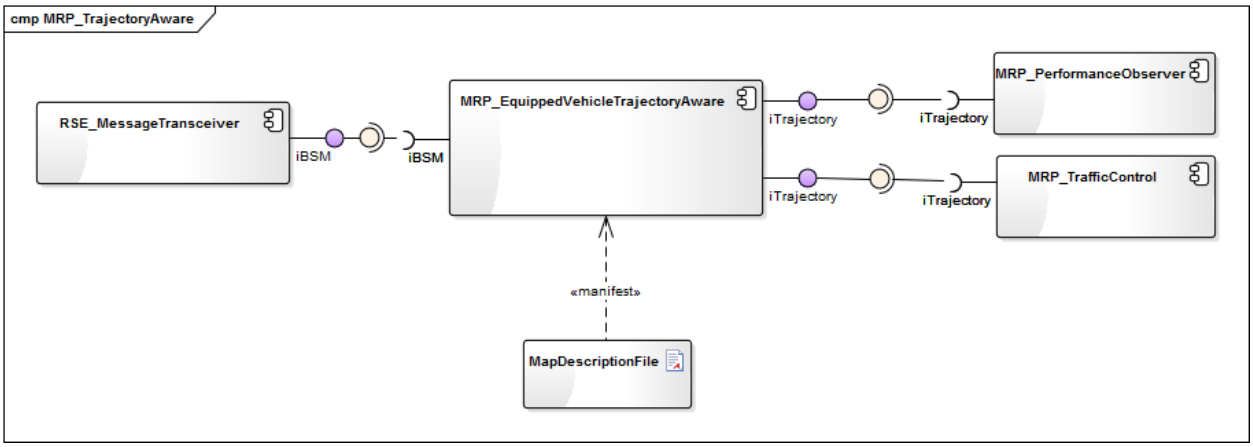


Figure 16 MRP_EquippedVehicleTrajectoryAware Component Interface

Required (input): BSM, MAP description file

Receive BSM

This function reads BSM from a UDP socket and returns the number of bytes received. The function `socket_init` needs to be called first to initialize the socket. The structure of BSM message is shown as follows:

```
BSMMessage :: = Sequence {
   Header       InternalMsgHeader        -- two bytes (0xFFFF)
   msgID        BSMmsgID                 -- one byte (0x01)
   timeStamp    CurEpochTime             -- four bytes (0.01s)
   datablob     BSMblob                  -- BSM message payload}
```

Note: the BSMblob contains the complete J2735 BSM.

Supporting Code:

```
#ifndef BSM_MSG_SIZE
    #define BSM_MSG_SIZE  (45)
#endif
char Recvbuf[BSM_MSG_SIZE];

int GetBSM(int sockfd)
{
       int recv_data_len;
       recv_data_len=Socket_Receive(sockfd);
       return recv_data_len;
}
```

Read MAP description file

This function reads the MAP description from MAP_File_Name and save to a MAP data structure which is defined in the later section.

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>
#include "NMAP.h"
int ReadMap (char * MAP_File_Name)
{
	MAP NewMap;
	NewMap.ID=1;
	NewMap.Version=1;
	// Parse the MAP file
	NewMap.ParseIntersection(MAP_File_Name);
	sprintf(temp_log,"Read the map successfully At (%d).\n",time(NULL));
	return 1;
}
```

Provided (output): Vehicle Trajectory, Arrival Table

Send Vehicle trajectory

This function takes the tracked vehicle list data and pack it to a trajectory message (octet stream) and send to MRP_PerformanceObserver through UDP socket. It returns the number of bytes sent. The function `socket_init` needs to be called first to initialize the socket. The structure of trajectory message is shown as follows:

```
TrajMessage :: = Sequence {
   Header      InternalMsgHeader        -- two bytes (0xFFFF)
   msgID       TRAJmsgID                -- one byte (0x02)
   timeStamp   CurEpochTime             --four bytes (0.01s)
   datablob    TRAJblob                 -- trajectory message payload}
```

The TRAJ blob is a compilation of objects in binary form that provides the vehicle trajectory within the DSRC range. Table 3 describes each object identifier.

Table 3 Object Identifiers of Trajectory Blob Data

Object Identifier	Object Type

0x01	Vehicle ID
0x02	nFrame (Number of trajectory points)
0x03	Vehicle Trajectory

Object Format – Vehicle ID

Field	Note
Object Identifier	0x01
Size	0x02
Vehicle ID	Unsigned 16-bit Integer

Object Format – nFrame

Field	Note
Object Identifier	0x02
Size	0x02
nFrame	Unsigned 16-bit Integer

Object Format – Vehicle Trajectory

Field	Note
Object Identifier	0x03
Size	0x23
msgCnt	Signed 8-bit Char, from BSM
secMark	Signed 16-bit Integer, milliseconds within a minutes, from BSM
Latitude	Signed 32-bit Integer
Longitude	Signed 32-bit Integer
Elevation	Signed 16-bit Integer - optional
Speed	Signed 16-bit Integer
Heading	Signed 16-bit Integer
IntersectionID	Signed 32-bit Integer
N_Offset	Signed 32-bit Integer
E_Offset	Signed 32-bit Integer
LanId	Signed 8-bit Char
VehStatus	Signed 8-bit Char
ETA	Signed 32-bit Integer

Note: N_Offset and E_Offset are local coordinates to the center of the intersection

Supporting Code:

```
#include "ConnectedVehicle.h"
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include "LinkedList.h”
```

```
char Sendingbuf[102400]; //at most 100k
int SendTrajectory(int sockfd, LinkedList <ConnectedVehicle> trackedveh)
{


    //Process the vehicle trajectory to an octet stream before sending
    TrajectoryProcessing (Sendingbuf, trackedveh);
    //Send the trajectory message
    int numbytes = Socket_Send(sockfd);
    return numbytes;
}
```

5.3.2.5 Functional Specification/Description

The MRP_EquippedVehicleTrajectoryAware component is used to locate vehicles within the road network as specified by the MAP description and to temporary save vehicle trajectory information including vehicle position, speed, heading, ETA and requested phase to a database based on received BSMs and the current MAP. A MAP is a lane level geometric intersection description (GID) of the road network. The MAP is provided by the infrastructure management system.

The BSMData_Transmitter component, on the OBE, broadcasts Basic Safety Messages (BSM) through the DSRC channel. When the MRP_EquippedVehicleTrajectoryAware receives the BSMs from non-priority connected vehicles, it locates each vehicle based on the MAP. The vehicle locations are used to form trajectories. These trajectories are used to generate an arrival table for MRP_TrafficControl component. The trajectory of each connected vehicle is maintained and passed to MRP_PerformaneObserver component for performance measurement.

The MAP

The MMITSS project is using SAE J2735 standard to construct the MAP. The MAP includes the intersection ID, attributes of the intersection, intersection reference point, approach, lane, lane attributes, lane width, lane nodes, lane connections, reference lane and an end indicator. The MAP coding supports GPS location accuracy up to 10^{-7} degrees (1.1 cm). The location of each lane node is presented as the northern and eastern offset to the reference point which is nominally selected as the center of the intersection.

To determine the locations of the lane nodes, a GoogleEarth map is built and each lane node is added manually to characterize the geometry of the intersection. A reference point which is the center of an intersection should be added for the lane node offset calculation. Traffic control applications do not require highly accurate MAP data and

testing has shown that the Google earth map is sufficiently accurate in the Arizona Testbed.

The MAP description file

A MAP description file includes all information needed to describe the geometry of the intersection being controlled. The MMITSS MAP description file should follow the following format. For the meaning of each bit in intersection attributes and lane attributes, see reference 2 (The map spec) for details.

```
MAP_Name    Intersection_Name.nmap
RSU_ID      Intersection_Name
IntersectionID xxx
Intersection_attributes          xxxxxxxx    /* Attributes of the intersection, 8 bits */
Reference_point    xx.xxxxxxx  xxx.xxxxxxx xxxx   /*lat, long, evelation(decimeter)*/
No_Approach x /*number of approaches*/
Approach    1
Approach_type        x              /*1: approach, 2: engress*/
No_lane      x
Lane   1.x
Lane_ID      1
Lane_type    x      /* Veh Lane, Computed Lane, Ped Lane, Special Lane, Barrier*/
Lane_attributes        xxxxxxxxxxxxxxxx    /*Attributes of the lane, 16 bits*/
Lane_width    xxx     /*in centimeters */
No_nodes      x   /*number of lane nodes*/
1.1.1   xx.xxxxxxx    xxx.xxxxxxx
1.1.2   xx.xxxxxxx    xxx.xxxxxxx
.
No_Conn_lane        x
x.x     x
x.x     x
end_lane
Lane   1.2
.
end_lane
end_approach
Approach    2
.
end_approach
end_map
```

The MRP_EquippedVehicleTrajectoryAware reads the MAP description file and creates a MAP data structure based on the class structure shown in Figure 17.

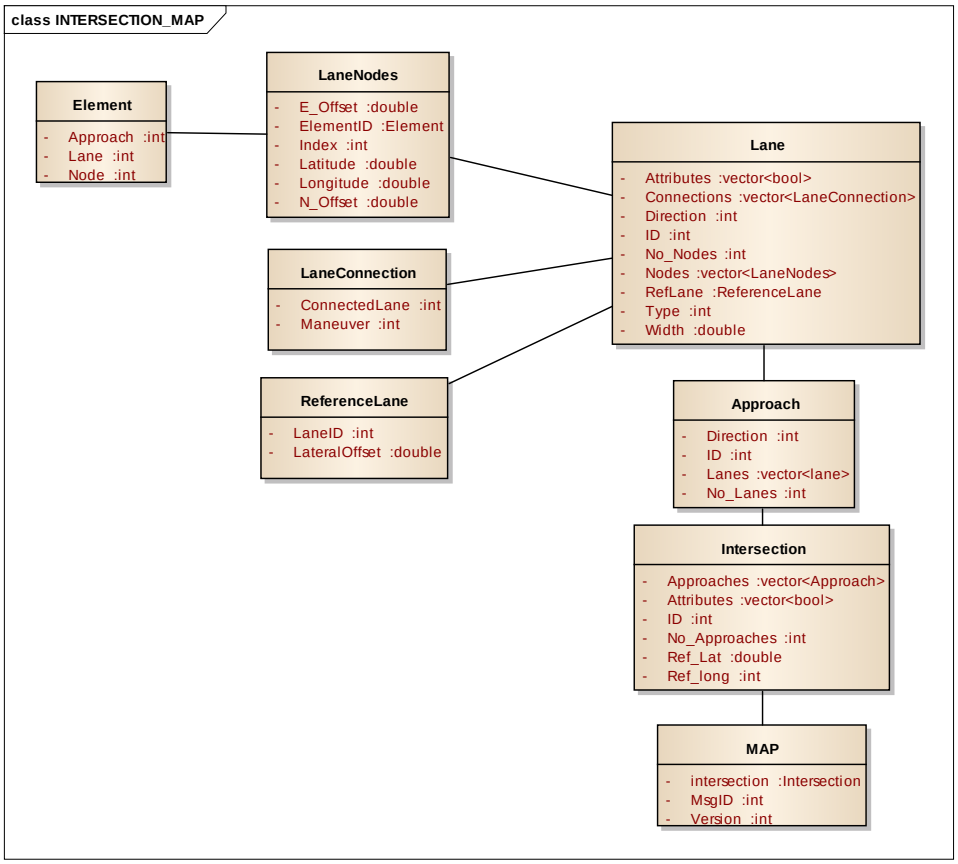


Figure 17 MAP Data Structure

5.3.2.6 Lane Phase Mapping Table

There are two types of lanes on the MAP: egress lane and approaching lane (first bit of the lane attribute). Egress lanes don't have requested phase. For approaching lanes, based on the map specification, the second, third and fourth bits of the lane attributes includes whether this is a right turn lane, a through movement lane or a left turn lane. If the bit is set to 1, that mean the corresponding movement is permitted. Therefore combined with the traffic signal installation plans, the lane phase mapping table can be constructed. The table is stored in the memory and is used by locating a vehicle on the MAP algorithm. An example lane phase mapping table can be seen in the application example section.

5.3.2.7 Locating a vehicle on the MAP

After a BSM is received and unpacked, the vehicle's location and status in the MAP is estimated. The vehicle's location, speed and heading can be directly obtained from the

BSM. An algorithm is used to estimate the vehicle's estimated time of arrival (ETA) at the stop bar and the vehicle's requested phase.

The flow of the algorithm is described below and shown in Figure 19:

1. Extract BSM data to obtain temporary ID, speed, heading and GPS coordinates.
2. Match the trajectory by temporary ID.
3. Transform GPS coordinates of the vehicle position to local offset based on the reference point (Farrell and Barth, 1998).
4. Find the nearest lane node to the vehicle and determine the current approach and lane.
5. Calculate the lane heading based on the nearest node and its preceding node.
6. Compare the lane heading with vehicle heading to determine whether the vehicle is on the map.
7. Based on vehicle speed and the comparison of distances to the stop bar between two time steps (further to the stop bar or closer to the stop bar), determine the vehicle status (approaching, in queue or departing). Figure 18 shows the transitions between each state. When the vehicle is in "Not on the map" state, the algorithm will not be called.

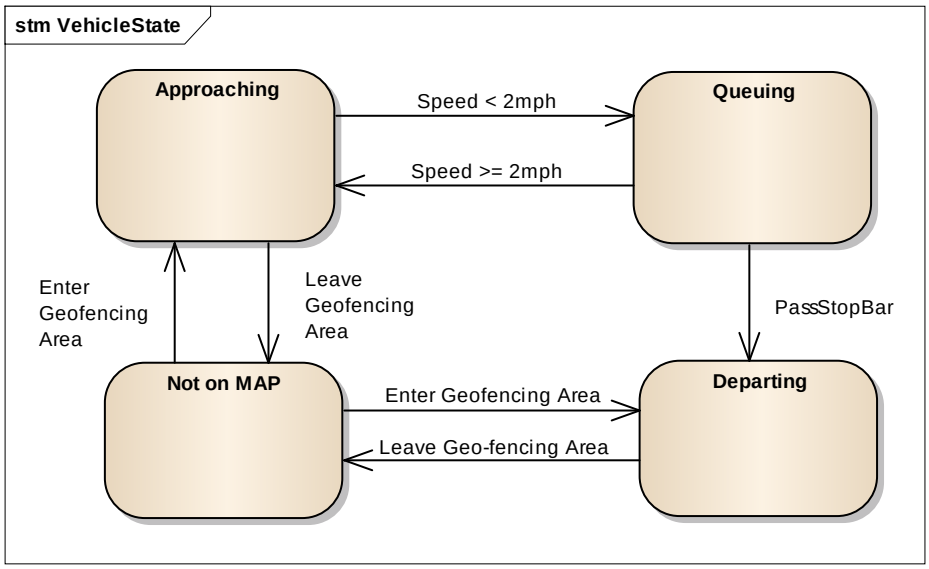


Figure 18 Vehicle State Transition

8. If the vehicle is in the Approaching state, the ETA is calculated as the distance to the stop bar divided by current vehicle speed. The requested phase is calculated based on the lane attributes (through lane or left turn lane) and lane-phase mapping table.

9. If the vehicle is in queuing state, the ETA is set to 0 since the vehicle is delayed. The requested phase is calculated as in Step 6.
10. If the vehicle is in departing state, no ETA and requested phase are estimated.

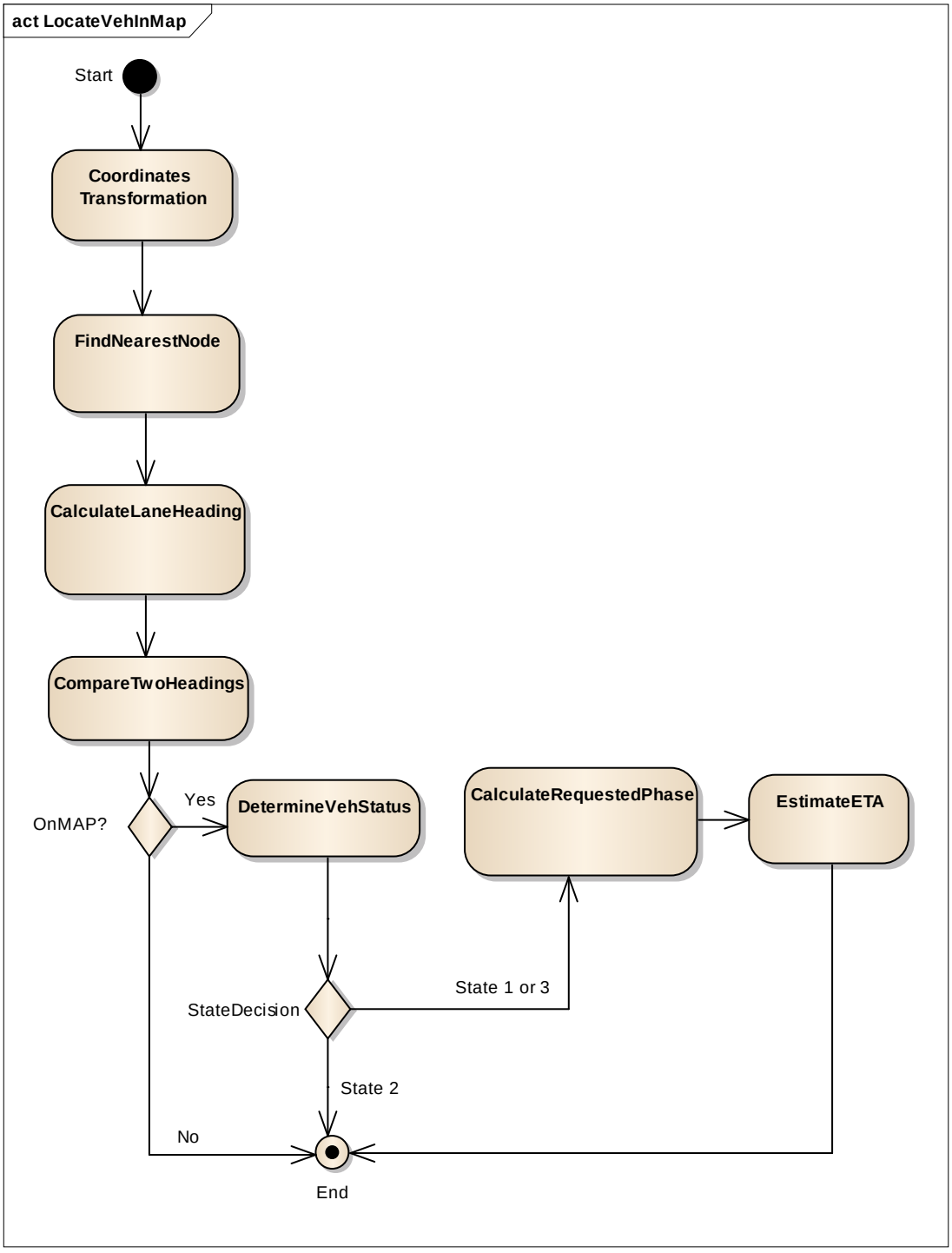


Figure 19 Flow of the Locating Vehicle in Map Algorithm

5.3.2.8 Vehicle trajectory info database

An active vehicle list is constructed and maintained to save vehicle trajectory information. A vehicle will be added to the list when it first appears within the geo-fencing area. A vehicle will be deleted from the list when it leaves the geo-fencing area

and the trajectory information has been requested by other components. As long as the vehicle is within the geo-fencing area, the following vehicle information will be saved every 0.5s: vehicle position (both GPS coordinates and offsets to the reference point), vehicle speed, vehicle heading, vehicle distance to the stop bar, vehicle estimated time of arrival to the stop bar, vehicle status (approaching, departing, or in queue), and vehicle requested phase. If the vehicle temporary ID remains the same, the new vehicle position will be matched to an existing trajectory by ID. If the temporary ID is changed for security reason, the vehicle will be considered as a new vehicle.

5.3.2.9 Geo-fencing

Vehicles are not added to the list of active vehicles unless they are inside the geo fence area. GeoFence_Range.txt file specifies the distance of the MAP lane nodes from the reference point on each approach. It also contains a variable called “Desired Distance to the Nearest Line” that is helpful to decide whether the vehicle is in a parking lot or stopped on side of the street.

Supporting Code:

```
//Distance to the Reference Point
dis_to_ref = sqrt(pow(local_x,2) + pow(local_y,2));

if (dis_to_ref <= GeoFence_Range[N_App-1] && distanceLine <= distance_nearest_line)
   {
       trackedveh.InsertRear(TempVeh);   //add the new vehicle to the tracked list
       sprintf(temp_log,"Add Vehicle No. %d at %lf, the list size is
%d\n",TempVeh.TempID, GetSeconds(),trackedveh.ListSize());
       outputlog(temp_log); cout<<temp_log;
       sprintf(temp_log,"The added location is %lf
%lf\n",TempVeh.traj[0].latitude,TempVeh.traj[0].longitude);
       outputlog(temp_log);
       cout<<temp_log;
       Veh_pos_inlist=trackedveh.ListSize()-1;  //start from 0
   }
else
    cout<<"Vehicle ID of "<<TempVeh.TempID<<" is Out of Range!!! The vehicle is not
considered!"<<endl;
```

Example Applications

The sequence diagram of the component is shown in Figure 20.

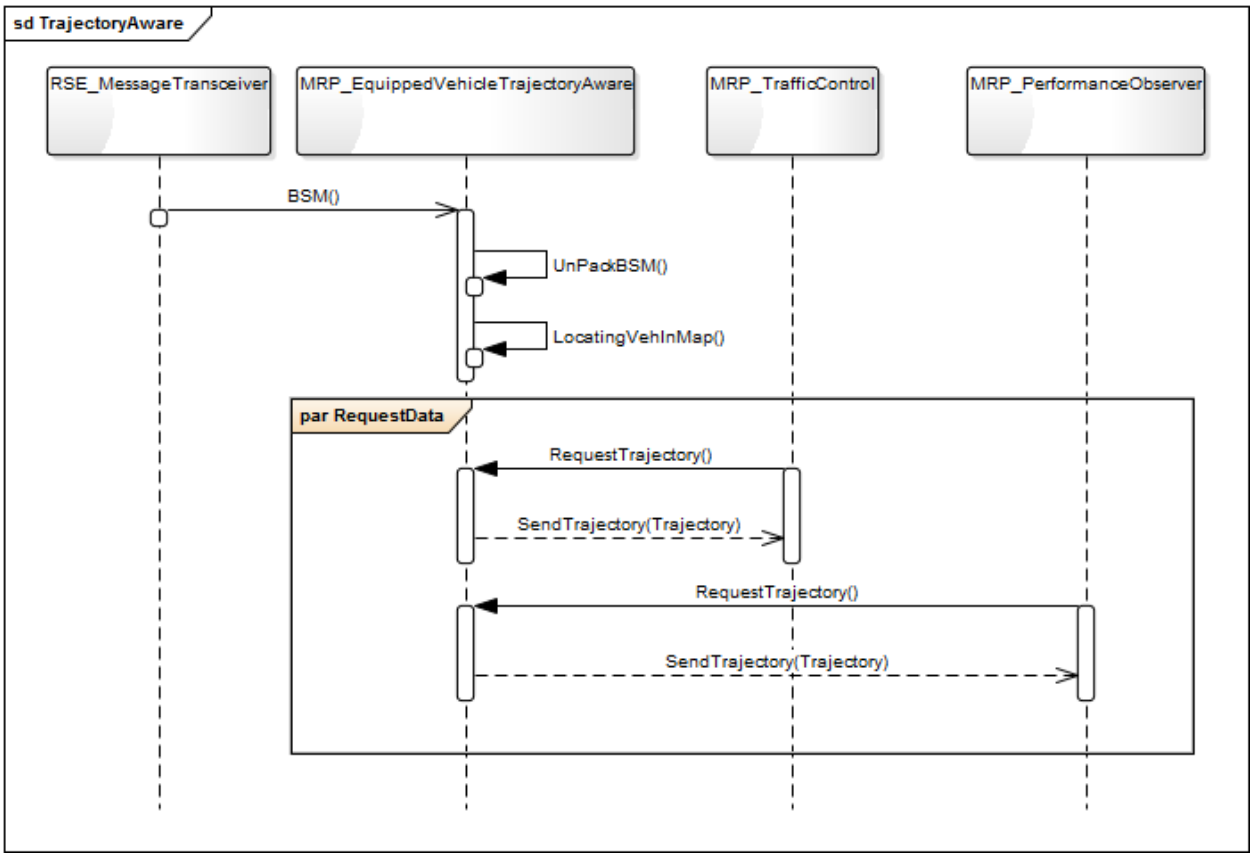


Figure 20 Sequence Diagram of MRP_EquippedVehicleTrajectoryAware Component

5.3.2.10 Example: Daisy Mountain Dr and Gavilan Peak, Anthem Test Bed

An example MAP of Dsiay Mountain Dr and Gavilan Peak at Anthem test bed is shown in Figure 21 and Figure 22.

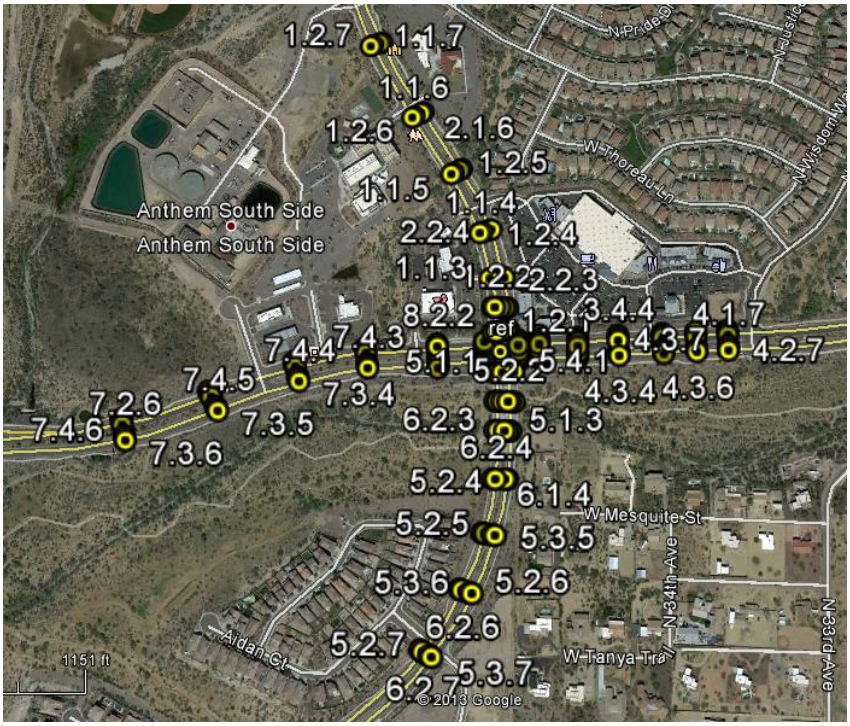


Figure 21 MAP - Daisy Mountain Dr and Gavilan Peak (area)

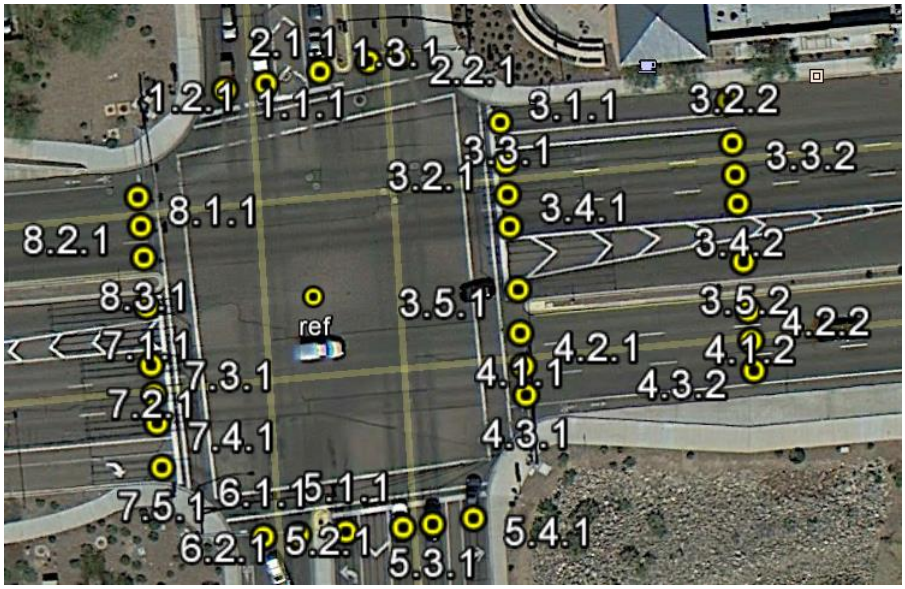


Figure 22 MAP - Daisy Mountain Dr and Gavilan Peak (near)

- MAP description file example: Daisy Mountain Dr and Gavilan Peak

An example MAP description file of Daisy Mountain Dr and Gavilan Peak is shown in Section 5.3.2.13

5.3.2.11 Example: Lane-phase Mapping Table of Daisy Mountain Dr. and Gavilan Peak Intersection

Lane ID	Lane Type	Through	Left Turn	Right Turn	Phase
1.1	0	1	0	1	4
1.2	0	1	0	0	4
1.3	0	0	1	0	7
2.1	1	1	0	0	0
2.2	1	1	0	0	0
3.1	0	0	0	1	0
3.2	0	1	0	0	6
3.3	0	1	0	0	6
3.4	0	1	0	0	6
3.5	0	0	1	0	1
4.1	1	1	0	0	0
4.2	1	1	0	0	0
4.3	1	1	0	0	0
5.1	0	0	1	0	3
5.2	0	1	0	0	8
5.3	0	1	0	0	8
5.4	0	0	0	1	0
6.1	1	1	0	0	0
6.2	1	1	0	0	0
7.1	0	0	1	0	5
7.2	0	1	0	1	2
7.3	0	1	0	1	2
7.4	0	1	0	1	2
7.5	0	0	0	1	0
8.1	1	1	0	0	0
8.2	1	1	0	0	0
8.3	1	1	0	0	0

Notes:

Lane Type: 0 means approaching lane and 1 means egress lane.

Through, left turn, and right turn: 1 means this movement is permitted and 0 means not permitted.

Phase: 1-8 is corresponding phase for the lane. If the value is 0, either this is a right turn only lane (right turn movement is assumed not to request any phase) or an egress lane.

5.3.2.12 References

1. Farrell J. A. and Barth M., The Global Positioning System & Inertial Navigation, McGraw-Hill Professional, 1998.
2. Battelle Memorial Institute, SIGNAL PHASE AND TIMING AND RELATED MESSAGE Binary Format (BLOB) Details, DRAFT, Rev.c, 2012.

5.3.2.13 MAP description file of Daisy Mountain Dr and Gavilan Peak

```
MAP_Name    Daisy_Gav.nmap
RSU_ID      Daisy_Gav
IntersectionID 1
Intersection_attributes 00110011    /*elevation: Yes, lane width: Yes, Node date 16 bits, node offset solution: cm, geometry: Yes, navigation: Yes*/
Reference_point     33.8429408  -112.1352055 5350   /*lat, long, evelation(decimeter)*/
No_Approach 8
Approach    1
Approach_type       1           /*1: approach, 2: engress*/
No_lane     3
Lane    1.1
Lane_ID     1
Lane_type   1   /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes     0000000000101010    /*not egress path, straight and right turn permitted, turn on red, no u-turn */
Lane_width  365     /* in centimeter =12 feet
No_nodes    8
1.1.1   33.8431696  -112.1353224
1.1.2   33.8434774  -112.1353437
1.1.3   33.8438323  -112.1353991
1.1.4   33.8444039  -112.1355749
1.1.5   33.8451492  -112.1360386
1.1.6   33.8458580  -112.1366679
1.1.7   33.8467824  -112.1373606
1.1.8   33.8477167  -112.1378525
No_Conn_lane        2
6.1     4   /*Lane 3.1, Straight head */
8.1     3   /*Lane 4.1, Right Turn */
end_lane
Lane    1.2
Lane_ID     2
Lane_type   1
Lane_attributes     0000000001100010    /*not egress path, straight permitted, no u-turn */
Lane_width  365
No_nodes    8
1.2.1   33.8431764  -112.1352686
```

1.2.2 33.8434765 -112.1353073
1.2.3 33.8438387 -112.1353569
1.2.4 33.8444167 -112.1355321
1.2.5 33.8451668 -112.1360031
1.2.6 33.8458773 -112.1366308
1.2.7 33.8467969 -112.1373230
1.2.8 33.8477313 -112.1378162
No_Conn_lane 1
6.2 4 /*Lane 3.2, Straight head */
end_lane
Lane 1.3
Lane_ID 3
Lane_type 1
Lane_attributes 0000000001100100 /*not egress path, left turn permitted, no u-turn */
Lane_width 365
No_nodes 3
1.3.1 33.8431898 -112.1351964
1.3.2 33.8434776 -112.1352271
1.3.3 33.8438435 -112.1353141
No_Conn_lane 1
4.1 2 /*Lane 4.1, Left turn */
end_lane
end_approach
Approach 2
Approach_type 2 /*1: approach, 2: engress*/
No_lane 2
Lane 2.1
Lane_ID 4
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100011 /*egress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet*/
No_nodes 8
2.1.1 33.8432011 -112.1351319
2.1.2 33.8434776 -112.1351605
2.1.3 33.8438497 -112.1352225
2.1.4 33.8444425 -112.1354401
2.1.5 33.8452071 -112.1359128
2.1.6 33.8459196 -112.1365366
2.1.7 33.8468263 -112.1372213

2.1.8 33.8477607 -112.1377204
No_Conn_lane 0 /* no connected lane because it is a engress lane*/
end_lane
Lane 2.2
Lane_ID 5
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100011 /*egress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 8
2.2.1 33.8432066 -112.1350907
2.2.2 33.8434768 -112.1351143
2.2.3 33.8438515 -112.1351818
2.2.4 33.8444532 -112.1354009
2.2.5 33.8452233 -112.1358807
2.2.6 33.8459371 -112.1365005
2.2.7 33.8468394 -112.1371844
2.2.8 33.8477741 -112.1376817
No_Conn_lane 0 /* no connected lane because it is a engress lane*/
end_lane
end_approach
Approach 3
Approach_type 1 /*1: approach, 2: engress*/
No_lane 5
Lane 3.1
Lane_ID 6
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000000111000 /*not engress path, right-turn permitted, yield, turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 2
3.1.1 33.8431382 -112.1349523
3.1.2 33.8431582 -112.1346514
No_Conn_lane 1
2.2 3 /*Lane 2.2, Right turn */
end_lane
Lane 3.2
Lane_ID 7
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */

Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
3.2.1 33.8430894 -112.1349488
3.2.2 33.8431114 -112.1346463
3.2.3 33.8431517 -112.1340837
3.2.4 33.8431997 -112.1334985
3.2.5 33.8432431 -112.1328550
3.2.6 33.8432802 -112.1323663
3.2.7 33.8433159 -112.1319029
No_Conn_lane 1
8.1 4 /*Lane 8.1, Straight */
end_lane
Lane 3.3
Lane_ID 8
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
3.3.1 33.8430540 -112.1349466
3.3.2 33.8430763 -112.1346428
3.3.3 33.8431171 -112.1340816
3.3.4 33.8431633 -112.1334942
3.3.5 33.8432142 -112.1328517
3.3.6 33.8432489 -112.1323636
3.3.7 33.8432820 -112.1319002
No_Conn_lane 1
8.2 4 /*Lane 8.2, Straight */
end_lane
Lane 3.4
Lane_ID 9
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
3.4.1 33.8430187 -112.1349437
3.4.2 33.8430442 -112.1346394

```
3.4.3   33.8430848    -112.1340795
3.4.4   33.8431309    -112.1334922
3.4.5   33.8431820    -112.1328476
3.4.6   33.8432166    -112.1323595
3.4.7   33.8432498    -112.1318975
No_Conn_lane        1
8.3     4    /*Lane 8.3, Straight */
end_lane
Lane   3.5
Lane_ID       10
Lane_type     1      /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes       0000000001110100    /*not engress path, left permitted, no turn on red, no
u-turn */
Lane_width    365    /* in centimeter =12 feet
No_nodes      3
3.5.1   33.8429492    -112.1349325
3.5.2   33.8429799    -112.1346320
3.5.3   33.8430372    -112.1340776
No_Conn_lane        1
6.2 2   /*Lane 6.2, Left turn */
end_lane
end_approach
Approach      4
Approach_type       2              /*1: approach, 2: engress*/
No_lane       3
Lane   4.1
Lane_ID       11
Lane_type     1      /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes       0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width    365    /* in centimeter =12 feet
No_nodes      7
4.1.1   33.8429010    -112.1349296
4.1.2   33.8429257    -112.1346255
4.1.3   33.8429644    -112.1340767
4.1.4   33.8430094    -112.1334783
4.1.5   33.8430560    -112.1328307
4.1.6   33.8430959    -112.1323493
4.1.7   33.8431286    -112.1318916
No_Conn_lane        0   /* no connected lane because it is a engress lane*/
```

```
end_lane
Lane    4.2
Lane_ID         12
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        7
4.2.1   33.8428642      -112.1349256
4.2.2   33.8428925      -112.1346211
4.2.3   33.8429315      -112.1340754
4.2.4   33.8429781      -112.1334732
4.2.5   33.8430235      -112.1328263
4.2.6   33.8430629      -112.1323462
4.2.7   33.8430950      -112.1318885
No_Conn_lane            0    /* no connected lane because it is a engress lane*/
end_lane
Lane    4.3
Lane_ID         13
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        7
4.3.1   33.8428323      -112.1349218
4.3.2   33.8428591      -112.1346168
4.3.3   33.8429002      -112.1340742
4.3.4   33.8429459      -112.1334691
4.3.5   33.8429909      -112.1328221
4.3.6   33.8430303      -112.1323432
4.3.7   33.8430634      -112.1318859
No_Conn_lane            0    /* no connected lane because it is a engress lane*/
end_lane
end_approach
Approach        5
Approach_type           1               /*1: approach, 2: engress*/
No_lane         4
Lane    5.1
Lane_ID         14
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
```

Lane_attributes 0000000001100100 /*not egress path, left permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 3
5.1.1 33.8426804 -112.1351622
5.1.2 33.8423323 -112.1351466
5.1.3 33.8419788 -112.1350981
No_Conn_lane 1
8.3 2 /*Lane 8.3, Left turn */
end_lane
Lane 5.2
Lane_ID 15
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not egress path, left permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 9
5.2.1 33.8426850 -112.1350861
5.2.2 33.8423323 -112.1350643
5.2.3 33.8419797 -112.1350482
5.2.4 33.8414066 -112.1350652
5.2.5 33.8407512 -112.1352137
5.2.6 33.8400770 -112.1355026
5.2.7 33.8393485 -112.1360278
5.2.8 33.8387132 -112.1367453
5.2.9 33.8380366 -112.1377007
No_Conn_lane 1
2.1 4 /*Lane 2.1, Straight */
end_lane
Lane 5.3
Lane_ID 16
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not egress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 9
5.3.1 33.8426886 -112.1350463
5.3.2 33.8423317 -112.1350253
5.3.3 33.8419801 -112.1350044
5.3.4 33.8414081 -112.1350264

5.3.5 33.8407461 -112.1351733
5.3.6 33.8400680 -112.1354663
5.3.7 33.8393328 -112.1359922
5.3.8 33.8386928 -112.1367145
5.3.9 33.8380175 -112.1376693
No_Conn_lane 1
2.2 4 /*Lane 2.2, Straight */
end_lane
Lane 5.4
Lane_ID 17
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000000111000 /*not egress path, right permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 2
5.4.1 33.8426951 -112.1349918
5.4.2 33.8423306 -112.1349676
No_Conn_lane 1
4.3 3 /*Lane 4.3, right turn */
end_lane
end_approach
Approach 6
Approach_type 2 /*1: approach, 2: engress*/
No_lane 2
Lane 6.1
Lane_ID 18
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100011 /*egress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 9
6.1.1 33.8426741 -112.1352729
6.1.2 33.8423321 -112.1352513
6.1.3 33.8419801 -112.1352359
6.1.4 33.8414022 -112.1352401
6.1.5 33.8407680 -112.1353608
6.1.6 33.8401121 -112.1356476
6.1.7 33.8394079 -112.1361541
6.1.8 33.8387952 -112.1368640
6.1.9 33.8381254 -112.1378031

```
No_Conn_lane        0    /* no connected lane because it is a engress lane*/
end_lane
Lane    6.2
Lane_ID         19
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        9
6.2.1   33.8426767    -112.1352262
6.2.2   33.8423322    -112.1352121
6.2.3   33.8419800    -112.1351883
6.2.4   33.8414018    -112.1351989
6.2.5   33.8407625    -112.1353226
6.2.6   33.8401018    -112.1356098
6.2.7   33.8393883    -112.1361182
6.2.8   33.8387725    -112.1368310
6.2.9   33.8381035    -112.1377706
No_Conn_lane        0    /* no connected lane because it is a engress lane*/
end_lane
end_approach
Approach        7
Approach_type           1               /*1: approach, 2: engress*/
No_lane         5
Lane    7.1
Lane_ID         20
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001110100    /*not engress path, left-turn permitted, yield, no turn
on red, no u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        2
7.1.1   33.8429302    -112.1354272
7.1.2   33.8428636    -112.1361080
No_Conn_lane        1
2.1     2    /*Lane 2.1, Left turn */
end_lane
Lane    7.2
Lane_ID         21
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
```

Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
7.2.1 33.8428654 -112.1354214
7.2.2 33.8428135 -112.1361033
7.2.3 33.8427284 -112.1371296
7.2.4 33.8425319 -112.1381276
7.2.5 33.8421312 -112.1392849
7.2.6 33.8417334 -112.1405642
7.2.7 33.8415676 -112.1416605
No_Conn_lane 1
4.1 4 /*Lane 4.1, Straight */
end_lane
Lane 7.3
Lane_ID 22
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
7.3.1 33.8428326 -112.1354178
7.3.2 33.8427817 -112.1361002
7.3.3 33.8426974 -112.1371231
7.3.4 33.8425013 -112.1381152
7.3.5 33.8421005 -112.1392649
7.3.6 33.8417018 -112.1405498
7.3.7 33.8415360 -112.1416550
No_Conn_lane 1
4.2 4 /*Lane 4.2, Straight */
end_lane
Lane 7.4
Lane_ID 23
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100010 /*not engress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
7.4.1 33.8428010 -112.1354142
7.4.2 33.8427477 -112.1360977

7.4.3 33.8426653 -112.1371154
7.4.4 33.8424691 -112.1381003
7.4.5 33.8420734 -112.1392453
7.4.6 33.8416709 -112.1405380
7.4.7 33.8415027 -112.1416507
No_Conn_lane 1
4.3 4 /*Lane 4.3, Straight */
end_lane
Lane 7.5
Lane_ID 24
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001111000 /*not engress path, right turn permitted, yeild, turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 2
7.5.1 33.8427514 -112.1354085
7.5.2 33.8427004 -112.1360939
No_Conn_lane 1
6.1 3 /*Lane 6.1, Straight */
end_lane
end_approach
Approach 8
Approach_type 2 /*1: approach, 2: engress*/
No_lane 3
Lane 8.1
Lane_ID 25
Lane_type 1 /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes 0000000001100011 /*egress path, straight permitted, no turn on red, no u-turn */
Lane_width 365 /* in centimeter =12 feet
No_nodes 7
8.1.1 33.8430543 -112.1354386
8.1.2 33.8429994 -112.1361246
8.1.3 33.8429109 -112.1371694
8.1.4 33.8427083 -112.1381982
8.1.5 33.8423043 -112.1394037
8.1.6 33.8419244 -112.1406316
8.1.7 33.8417790 -112.1416925
No_Conn_lane 0 /* no connected lane because it is a engress lane*/
end_lane

```
Lane    8.2
Lane_ID         26
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        7
8.2.1   33.8430190    -112.1354348
8.2.2   33.8429646    -112.1361202
8.2.3   33.8428790    -112.1371606
8.2.4   33.8426796    -112.1381817
8.2.5   33.8422668    -112.1393744
8.2.6   33.8418905    -112.1406170
8.2.7   33.8417423    -112.1416862
No_Conn_lane            0    /* no connected lane because it is a engress lane*/
end_lane
Lane    8.3
Lane_ID         27
Lane_type       1       /*1 to 5, for this intersection all 1: motorized vehicle lane */
Lane_attributes         0000000001100011    /*egress path, straight permitted, no turn on red, no
u-turn */
Lane_width      365     /* in centimeter =12 feet
No_nodes        7
8.3.1   33.8429844    -112.1354310
8.3.2   33.8429334    -112.1361169
8.3.3   33.8428492    -112.1371516
8.3.4   33.8426475    -112.1381667
8.3.5   33.8422380    -112.1393535
8.3.6   33.8418552    -112.1406033
8.3.7   33.8417100    -112.1416820
No_Conn_lane            0    /* no connected lane because it is a engress lane*/
end_lane
end_approach
end_map
```

5.3.3 MRP_PRS_PriorityRequestServer

5.3.3.1 Overview of Functionality

The MRP_PRS_PriorityRequestServer is responsible for managing and prioritizing the priority requests (SRM) from multiple priority eligible vehicles on the same or conflicting movements with the consideration of the prevailing traffic conditions and the requested level of priority (as determined by the vehicle and the established policy). The MRP_PRS_PriorityRequestServer constructs an Active Request Table (ART) based on received SRMs and passes the ART as input to MRP_Priority_Solver component.

5.3.3.2 Requirements

Node: MRP	**Component Name:** MRP_PRS_PriorityRequestServer
	Traceability: F2002.001, C2002.202, C2002.303, C2002.404, C2002.505, F2003, C2003.201, C2003.302, F2003.303, C2009.001, F2010.001, C2010.202, C2010.303, C2010.404, C2010.505, 13.3.1; 13.3.2; 13.3.3; 13.3.4; 13.3.5; §8, §11.0, §11.0.1, §11.0.2, §11.1, §11.2, §11.2.1, §11.2.2, §11.2.3, §11.3, §11.3.2, §11.3.3, §11.3.4, §11.4, 11.4.1, §11.4.2, §11.5, §11.5.1
Description of Responsibility: The MRP_PRS_PriorityRequestServer is responsible for managing all Requests for Priority that are received in terms of determining eligible requests and providing an Active Request Table for MRP_Priority_Solver.	
Supporting Text: *The MRP_PRS_PriorityRequestServer implements the server side of the general NTCIP 1211 use cases for providing priority at traffic signals, but doesn't contain the signal control algorithms. In the MMITSS (Arizona) implementation, the signal control algorithms are in the MRP_Priority_Solver component, but it is possible that the signal control algorithms could be existing logic on a traffic signal controller and the MRP_PRS_PriorityRequestServer would be modified to call the appropriate preemption or priority input.*	

5.3.3.3 Interfaces

The MRP_PRS_PriorityRequestServer interfaces are shown in Figure 23.

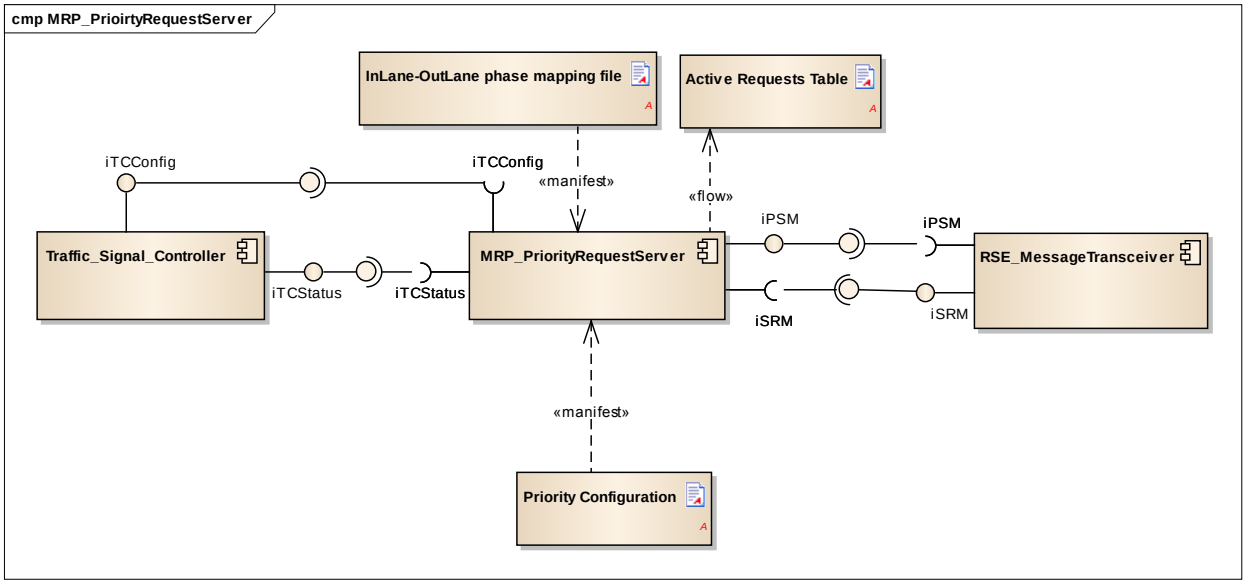


Figure 23 Interfaces to MRP_PriorityRequestServer

5.3.3.4 Required (input): SRM, TCStatus, TCConfig, PriorityPolicy, InLane OutLane Phase mapping description file

Receive SRM (iSRM)

This function receives the SRM from the RSE_MessageTransceiver that is configured to listen for SRM messages. The structure of SRM message is shown as follows:

Supporting Code:

```
#ifndef SRM_MSG_SIZE
    #define SRM_MSG_SIZE (197)
#endif
char Recvbuf[SRM_MSG_SIZE];
SRM isrm;
int GetSRM(char * buffer)
{
       int recv_data_len;
       recv_data_len=Socket_Receive(buffer);
       isrm=Decode_Buffer(buffer)
       return recv_data_len;
}
```

The MRP_PRS_PriorityRequestServer will acquire Signal Request Messages (SRM) from priority eligible vehicles from the RSE_MessageTransceiver component that is listening to the assigned service channel (182). The SRM data includes: intersection ID, vehicle ID, vehicle type, approaching lane of vehicle (inLane) to intersection and the egressing lane of vehicle (outLane), and the expected time of arrival (timeOfService,

endOfService) at the intersection of equipped vehicles, or nomadic devices within range of an intersection. The MRP_PRS_PriorityRequestServer component is responsible for filtering the SRMs that are related to the intersection. This component compares the Intersection ID element in SRMs with the ID of the intersection. If the two IDs are not matched, the SRM is ignored. After filtering the SRMs, the MRP_PRS_PriorityRequestServer processes the received SRMs to calculate Expected Time of Arrival (ETA). The ETA is calculated in the MRP_PRS_PriorityRequestServer based on the difference between timeOfService and current time. Also, priority vehicle requested phase is calculated in the MRP_PRS_PriorityRequestServer based on projecting the (inLane, OutLane) pair to intersection phase configuration table to support the movement of the equipped vehicle through the intersection. The requested phase and ETA are used in the Active Request Table (ART).

The MRP_PRS_PriorityRequestServer can also read the coordination plan and sets a virtual coordination request every cycle.

Receive TCStatus

The Traffic Control Status (TCStatus) is the real-time signal status obtained from Traffic Signal Controller. (Note: This information should come from MRP_TrafficControllerInterface component but in the current system, this information is directly obtained from the Traffic Signal Controller using NTCIP objects).

Receive TCConfig

The MRP_PRS_PriorityRequestServer obtains the Traffic Control Configuration (TCConfig) from the traffic signal controller (using NTCIP). The TCConfig includes phase in use, ring structure, minimum green time, maximum green time, yellow change interval, red clearance time, pedestrian walking time, and pedestrian clearance time. The TCConfig is stored in a local file. An example of TCConfig is shown as follows:

```
Phase_Num 8
Phase_Seq 1 2 3 4 5 6 7 8
Gmin   8      8      8      8      8      8      8      8
Yellow 3.5    4.3    3      4.9    3.5    4.3    3      4.9
Red    1.5    1.5    1      2      1      1.8    1      2
Gmax   16     36     15     29     16     36     15     29
PedWk  0      10     0      10     0      10     0      10
PedClr 0      15     0      15     0      15     0      15
```

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>
#include "NMAP.h"

class TCConfig
{
public:
	int No_phases;
	int Phase_Seq[8];   //Maximum allow 8 phases
	float Gmin[8];
	float Yellow[8];
	float RedClr[8];
	float Gmax[8];
	float PedWk[8];
	float PedClr[8];
};

int ReadTCConfig (char * Config_File_Name)
{
	TCConfig Plan;
	ReadTCConfig(Config_File_Name, Config);
	printf("Read the Signal Plan successfully At (%d).\n",time(NULL));
	return 1;
}
```

Read PriorityPolicy

The MRP_PRS_PriorityRequestServer obtains Priority Policy from “priorityConfiguration.txt” file. The Priority Policy includes modes type ID, mode type weight, mode type hierarchy. (see System_N_LevelPriorityConfigurationManager for details).

Read InLane OutLane phase mapping description file

This function reads the InLane, OutLane phase mapping description file. MPR_PRS_PriorityRequestServer needs this file to obtain the requested phase from the received SRM. This file is generated by MRP_MAP_SPaT_Broadcast component.

An example of “InLane_OutLane_Phase_Mapping.txt” is shown as follow:

```
IntersectionID  318
No_Approach     8
No_Phase        8
No_Ingress      4
Approach        1
No_Lane 4
1.1 6.1 2
1.2 6.2 2
1.3 6.3 2
```

```
1.4 4.1 5
end_Approach
Approach        3
No_Lane 3
3.1 6.3 3
3.2 6.2 3
3.3 2.2 3
end_Approach
Approach        5
No_Lane 3
5.1 8.1 1
5.2 2.1 6
5.3 2.2 6
end_Approach
Approach        7
No_Lane 2
7.1 2.1 4
7.2 2.2 4
end_Approach
end_file
```

5.3.3.5 Provided (output): Active Request Table file, Priority Status Message (PSM)

Active Request Table (ART)

The Active Request Table (ART) is a list of priority requests that are currently active. Also, ART is useful in for providing the Connected Vehicles information about all active priority requests[1].

MRP_PRS_PriorityRequestServer writes ART into a file called “requests_combined.txt”. Whenever a new SRM is received, MRP_PRS_PriorityRequestServer updates the contents of this file. Along with ART, MRP_PRS_PriorityRequestServer changes a flag in this file to a positive number whenever a new SRM is received and the ART is updated. The flag is an indicator for the MRP_Priority_Solver component. The MRP_Priority_Solver continuously reads this file and if the flag is positive, the MRP_Priority_Solver solves an optimization problem and changes the flag back to zero.

An example of request_combined.txt file is shown as follow:

```
Num_req 1 1
115 2 14 4 0 1428888550 11 61 1 21 53 1 22 7 1 0
```

[1] The Signal Status Message (SSM) as specified by SAE J2735 version 2009 only indicates that the traffic signal controller is serving a preemption or a priority input and does not currently contain information about the active requests. The MMITSS Team will recommend that this message be revised to include relevant information about active priority requests as well as the status of the signal controller. The MMITSS Team applies a new message called the Priority Status Message (PSM) instead of the SSM. This information has been shared with the SAE DSRC Technical Committee.

In this example, the first number in the first row shows the total number of requests. The second number is the flag that is 1 in this example. The second row shows the request information.

Priority Status Message (PSM)

MRP_PRS_PriorityRequestServer sends Priority Status Message (PSM) to an instance of the MRP_MessageTransceiver. The MRP_MessageTransceiver broadcasts the PSM to all priority eligible vehicles in the range of RSE so that the nearby connected vehicles know their status as well as the status of other active vehicles at the intersection. The PSM is used to provide a feedback mechanism for priority eligible vehicles that send SRMs to the RSE.

Figure 24 shows the activity diagram for MRP_PRS_PriorityRequestServer when a new SRM is received.

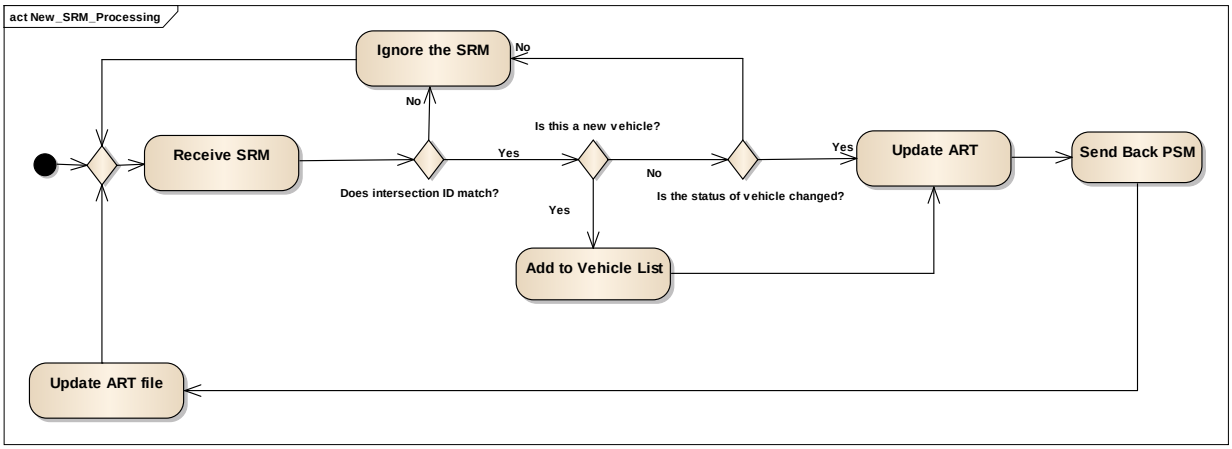


Figure 24 Activity Diagram for receiving new SRM

5.3.4 MRP_Priority_Solver

5.3.4.1 Overview of Functionality

The MRP_Priority_Solver is responsible for prioritizing the priority requests in Active Request Table (ART) from multiple priority eligible vehicles on the same or conflicting movements and providing the best signal timing to serve all of the requests simultaneously with the consideration of the prevailing traffic conditions and the requested level of priority (as determined by the vehicle and the established policy). In

addition, data representing the signal controller configuration, i.e., minimum green, maximum green, red and yellow clearance intervals is required to determine the optimal priority signal timing. The output of the optimization solver is an optimal phase timing schedule that accommodates all active requests in the ART. The optimal phase timing schedule is provided to the MRP_TrafficControllerInterface.

5.3.4.2 Requirements

Node: MRP	**Component Name:** MRP_Priority_Solver
	Traceability: This component was originally part of the MRP_PriorityRequestServer, but was developed as a separate software component that complements the MRP_PriorityRequestServer and includes the priority control logic. It is a component so that it can be replaced with alternate logic as desired.
Description of Responsibility: The MRP_Priority_Solver is responsible for solving an optimization model to obtain the best signal timing schedule for all active requests based on the N-Level priority policy, the prevailing traffic conditions, and signal controller capability.	
Supporting Text: *The MRP_Priority_Solver formulates an optimization problem (see references) and solves the problem for the optimal timing schedule. This algorithm is the result of nearly a decade of research on priority based traffic signal control.*	

5.3.4.3 Interfaces

The MRP_Priority_Solver interfaces are shown in Figure 25.

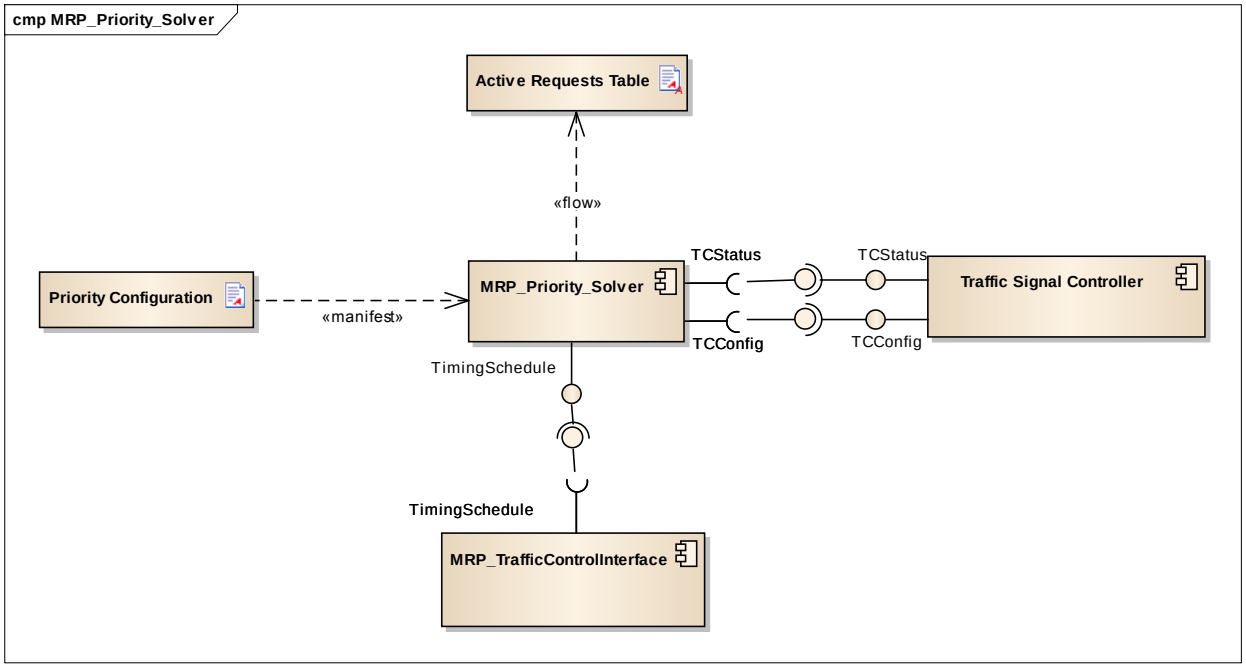


Figure 25 Interfaces to MRP_Priority_Solver

5.3.4.4 Required (input): Active Request Table, TCStatus, TCConfig, PriorityPolicy

Read Active Request Table (ART)
The Active Request Table (ART) is a list of priority requests that are currently active. MRP_Priority_Solver reads this table from “requests_combined.txt”. The ART is explained in MRP_PRS_PriorityRequestServer in detail.

Receive TCStatus
The Traffic Control Status (TCStatus) is the real-time signal status obtained from Traffic Signal Controller. (Note: This information should come from MRP_TrafficControlInterface component but in the current system this information is read directly from the Traffic Signal Controller).

Receive TCConfig
The MRP_Priority_Solver obtains Traffic Control Configuration (TCConfig) from Signal Controller. The structure of TCConfig is explained in detail in MRP_PRS_PriorityRequestServer section.

Read PriorityPolicy
The MRP_Priority_Solver obtains PriorityPolicy from “priorityConfiguration.txt” file. This file is explained in MRP_PRS_PriorityRequestServer section.

5.3.4.5 Provided (output) Signal Timing Schedule (TimingSchedule)

Send Signal Timing Schedule
The MRP_Priority_Solver sends the Signal Timing Schedule (TimingSchedule) message to MRP_TrafficControlInterface component. The structure of this message is explained in MRP_TrafficControl section.

5.3.4.6 Functional Specification/Description
The MRP_Priority_Solver is responsible for managing all Requests for Priority that are received in terms of determining eligible requests based on the N-Level priority policy. This component is also responsible for determining the best signal timing strategy based on the prevailing traffic conditions and signal controller capability for all priority eligible requests. The priority request server must choose the best method for serving the active priority requests under the current traffic conditions.

The MRP_Priority_Solver reads the priority strategy configuration “priorityconfiguration.txt” file. This strategy configuration establishes a framework and mechanism for implementing a priority policy where each mode can be given priority over other modes and within each mode vehicles can be given relative priority over other vehicles. For example, transit vehicles may be determined to have priority over freight vehicles. Emergency vehicles can be given override priority over transit and freight, meaning that any active transit or freight requests are cancelled by the policy. Generally, rail is given the highest level of priority due to the need for track clearance.

The priority strategy defines the modes that should be considered as eligible priority modes at the intersection level. Each mode has a weight that shows the level of importance of that mode. If coordination is considered as a type of priority in the N_level priority configuration, then the coordination weight and required coordination priority requests would be added to the priority policy.

A request can be served if there is not an overriding request active. An overriding request would occur if a higher priority request is configured to override lower priority requests, such as an emergency vehicle request could override a transit vehicle request.

Because the configuration of a signal controller may vary according to the geometry of the intersection, time of day, and traffic pattern, the MRP_Priority_Solver must acquire

the intersection traffic signal controller configuration data. The MRP_Priority_Solver is responsible for constructing a data structure that is applied in constructing the mathematical formulation. The data structure includes the signal controller configuration, i.e., phases in use, phase sequence ,minimum green, maximum green, red and yellow clearance intervals and phase sequence.

The MRP_Priority_Solver also receives Signal Status from Traffic Signal Controller. The Traffic Control Signal Status (TCStatus) information describes the controller's current phase status. The MRP_Priority_Solver uses the TCStatus information to construct the mathematical formulation. Then, based on the signal controller configuration and the TCStatus information, the MRP_Priority_Solver calculates parameters such as the elapsed green time of the current green phase and the remaining time to the start of the next phase if the current phase is in the yellow change interval or all phases are in the red clearance interval. If phase status is in the clearance interval, a countdown function is be called to calculate the remaining time to the start of the next phase.

MMITSS uses a priority timing routine that has the intelligence to apply sophisticated strategies rather than "first called, first served" for conflicting requests. The MRP_Priority_Solver is responsible for deciding when the priority control problem should be solved. Based on the priority policy, the MRP_Priority_Solver knows the weight of priority eligible modes that should be considered as priority eligible modes.

Optimizer

The MRP_Priority_Solver constructs a mathematical model that is the input data structure for the Optimizer based on the information described above. The mathematical model is built in the properly formatted data structure (which is stored in a file) for the Optimizer. The details of the mathematical optimization problem are described in the following papers:

- Zamanipour, M., K.L. Head, Feng, Y., Khoshmagham, S, "A Priority Optimization Control Framework for Multi-Modal Intelligent Traffic Signal Systems", Working Paper.
- He, Q., K.L. Head, and J. Ding, "Multi-Modal Traffic Signal Control with Priority, Signal Actuation and Coordination, Transportation Research Part C: Emerging Technologies, 46, 2014, pp 65-82.

The Optimizer uses the mathematical model to solve the priority control optimization problem. This model is solved using the GNU Linear Programming Kit (GLPK) package (GNU Project), which is open source and intended for solving large-scale linear

programming (LP), mixed integer programming (MIP), and other related problems. A problem with 10 requests can be solved in less than a second using the processor in the Savari StreetWave RSE.

When the GLPK solver is applied, there are two extra files: one is a required data file ("NewModelData.dat") including the input from the Mathematical Model Developer. The other is a mod file ("NewModel.mod") including constraints and an optimization goal. After obtaining an optimal solution (OPT) by solving the optimization problem, the OPT is used to construct a Signal Timing Schedule (TimingSchedule) that is passed to the MRP_TrafficControllerInterface component where the flexible green time can be allocated.

5.3.4.7 Example Applications

Figure 26 shows a snap shot of one priority control scenario modeled in the VISSIM simulation environment. There are three priority eligible vehicles involved: two Transit vehicles on the main street and one Truck on a conflicting approach to the transit vehicles. Table 4 describes the ART of this example.

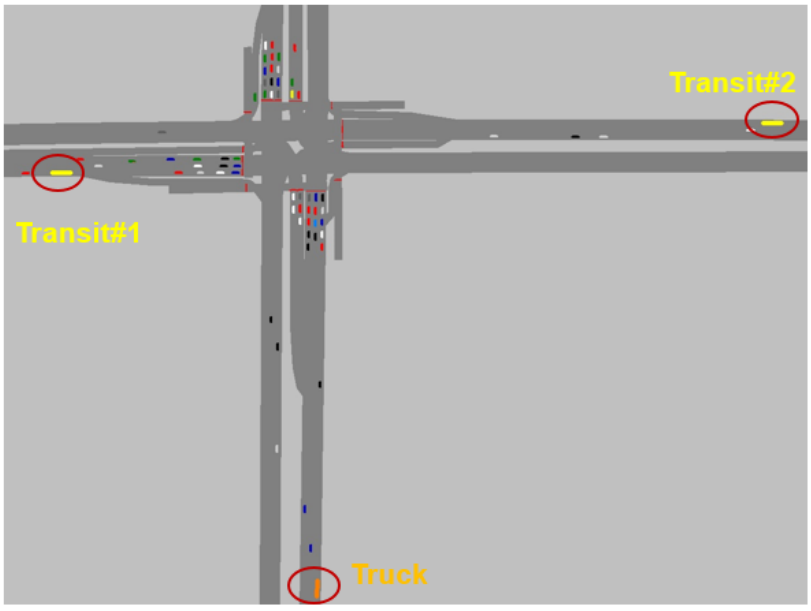


Figure 26 Multiple priority request example.

Transit#1 is close to intersection and will arrive at the stop bar in 10 to 15 seconds and will be served by Phase 2. Transit#2 will arrive at the stop bar in 42 to 47 seconds and

will be served by Phase 6. The Truck will arrive in 50 to 60 seconds and will be served by Phase 4. There are several factors that may affect arrival time of a vehicle such as driver behavior and congestion, so the MRP_Priority_Solver considers a range for the arrival time to provide some flexibility in estimating the arrival time.

Table 4 Multiple priority request example

	Vehicle ID	Vehicle Type	Range (seconds)	Requested Phase	Queue Clearance Time
1	Transit#1	Transit	[10,15]	2	5
2	Transit#2	Transit	[42, 47]	6	0
3	Truck#1	Truck	[50, 60]	4	10

Table 5 shows the intersection configuration data for an eight-phase intersection. Minimum green time, maximum green time, yellow time, and red time are described for each phase.

Table 5 Intersection configuration data

	Phase							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Gmin	8	8	8	8	8	8	8	8
Yellow	3.5	4.3	3	4.9	3.5	4.3	3	4.9
Red	1.5	1.5	1	2	1	1.8	1	2
Gmax	16	36	15	29	16	36	15	29
PedWk	0	10	0	10	0	10	0	10
PedClr	0	15	0	15	0	15	0	15

Table 6 shows the current signal status for the example illustrated in Figure 26. The controller is currently in Phases 3 and 7, which are denoted as the starting phases (SP1 and SP2 for ring 1 and ring 2, respectively) and the elapsed green time (Init1 and Init2) of these active phases is 1.5 seconds each. The controller has been in the Red clearance interval for 0.5 seconds (of a total of 2.0 seconds). Phases 3 and 7 will start in 1.5 second.

Table 6 Signal status data

SP1	3
SP2	7
Init1	1.5
Init2	1.5
Grn1	0
Grn2	0

Table 7 shows the modes that will be considered in this intersections. Each mode has its determined weights based on N-Level priority policy. This table also shows protected left-turn option and coordination weight. In this table, |1| means EV has highest absolute priority. If there is an EV in the ART, the EV has the absolute importance and other priority eligible vehicles should yield to it. Truck weight is 1.0 and Transit weight is 1.0. Phase skipping is not allowable and coordination weight is zero.

Table 7 Modes weights

EV	Transit	Truck	Coordination	Protected Left-Turn	Phase Skipping
|1|	1.0	1.0	0	Yes	No

Figure 27 shows the OPT schedule for the ART in Table 4. A precedence graph (Head et al. 2006) shows the dual phase sequence and timing. Transit #1 has 38.2 seconds delay while Transit#2 has no delay. By looking at Figure 27, it is revealed that phases 3, 4, and 1 timed their minimum green time such that the delay for Transit#1 becomes 38.2. On the other hand, this OPT schedule times phase 2 and 3 in the second cycle to their minimum green time such that Truck#1 gets minimum delay of 33.3 seconds.

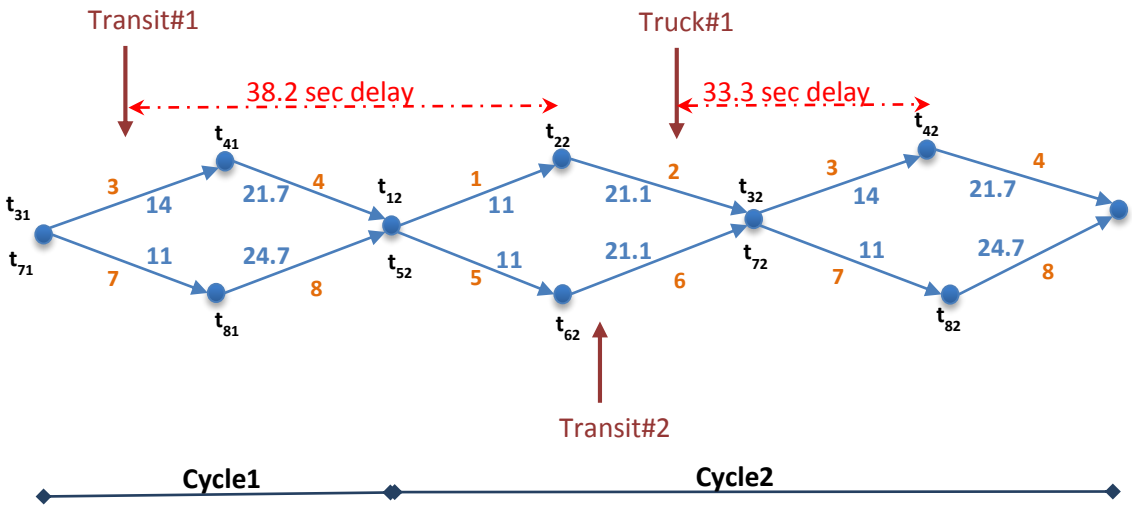


Figure 27 OPT Schedule for the given example

5.3.4.8 Sequence Diagrams

Figure 28 shows the sequence diagram of MRP_Priority_Solver and its interactions with other software components in MMITSS.

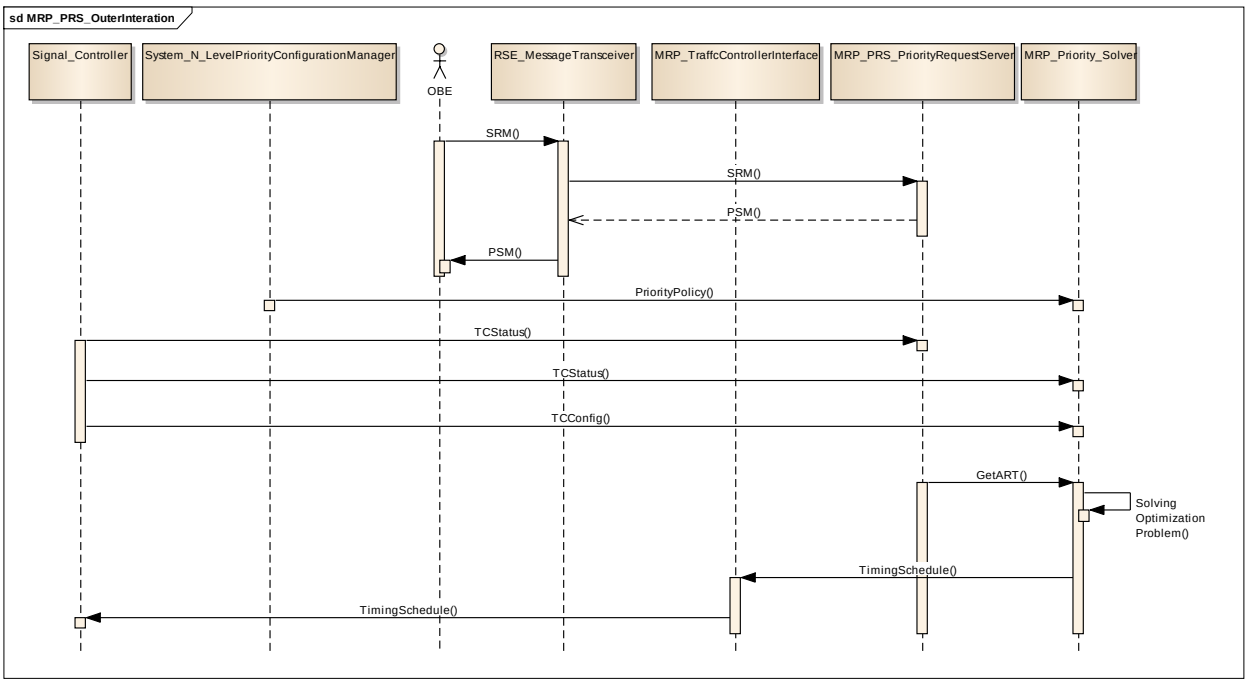


Figure 28 Sequence Diagram of interaction of MRP_PRS_PriorityRequestServer with other components

5.3.4.9 References

1. Head, L., Gettman, D. & Wei, Z., 2006. A Decision Model for Priority Control of Traffic Signals, *Transportation Research Record: Journal of the Transportation Research Board*, Washington, D.C, 1978, 169-177.

2. Zamanipour, M., K.L. Head, Feng, Y., Khoshmagham, S, "A Priority Optimization Control Framework for Multi-Modal Intelligent Traffic Signal Systems", Working Paper.
3. Ding, J., He. Q., Head, L., Saleem, F., Wu, W., 2013 Development and Testing of Priority Control System in Connected Vehicle Environment. *Transportation Research Board ,* Washington, D.C, 1925, pp. 227–234.
4. He, Q., K.L. Head, and J. Ding, "Multi-Modal Traffic Signal Control with Priority, Signal Actuation and Coordination, Transportation Research Part C: Emerging Technologies. 46, 2014, pp 65-82.
5. GNU project http://www.gnu.org/software/glpk/

5.3.5 MRP_TrafficControl

5.3.5.1 Overview of Functionality

The MRP_TrafficControl component is responsible for providing signal control for all non-priority vehicles at an intersection. MRP_TrafficControl integrates information from connected vehicles and traditional vehicle detectors. MRP_TrafficControl performs detection matching, phase calls, phase extensions, dilemma zone protection for each of the different modes (and dynamics) of vehicles.

5.3.5.2 Functional Requirements (reference)

Node: MRP	**Component Name:** MRP_TrafficControl
	Traceability: C2011.001, C2011.302, C2011.303, C2011.304, C2011.005, C2011.006, C2012.001, C2012.002, C2014.001, C2101.001, C2101.002, C2101.003, C2101.004, C2101.305, A2106, 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; §11.0, §11.0.2, §11.1.1, §11.1.1.2, §11.1.1.3, §11.1.4, §11.2.1, §11.2.2, §11.2.3, §11.3, §11.3.2, §11.3.3, §11.4.1, §11.5.1, §12.7.1, §5, §8

Description of Responsibility: The MRP_TrafficControl is responsible for integrating all equipped vehicles from the MRP_EquippedVehicleTrajectoryAware into the traffic control logic through detection matching, phase calls, phase extensions, dilemma zone protection for each of the different modes (and dynamics) of vehicles.
Supporting Text: *The MCDOT SMARTDrive network utilizes NTCIP compliant controllers that allow data acquisition, including detector calls, phase status, etc., and control using NTCIP objects.*

5.3.5.3 Derived Requirements

Timing
The traffic control component optimization must execute in less than minimum green time.

5.3.5.4 Interfaces

Required (input)
Receive Vehicle Trajectory from MRP_EquippedVehicleTrajectoryAware component

This function will request Vehicle Trajectories from the MRP_EquippedVehicleTrajectoryAware component and the MRP_EquippedVehicleTrajectoryAware component will return the Vehicle Trajectories through the UDP socket. The data structure of the Vehicle Trajectory message is defined in MRP_EquippedVehicleTrajectoryAware interface section (trajectory message).

Supporting Code:

```
#ifndef TRAJECTORY_MSG_SIZE
    #define TRAJECTORY_MSG_SIZE  (5000)
#endif
char Recvbuf[TRAJECTORY_MSG_SIZE];

int GetTRAJECTORY (int sockfd)
{
       int recv_data_len;
       recv_data_len=Socket_Receive(sockfd);
       return recv_data_len;
}
```

Traffic Control Configuration Parameters

The Traffic Control Configuration Parameters (TCConfig) include phase in use, ring structure, minimum green time, maximum green time, yellow change interval, red

clearance time, pedestrian walking time, and pedestrian clearance time. The Traffic Control parameters will be stored in a local file. The structure of the message is defined in MRP_PRS_PriorityRequestServer component.

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>
#include "NMAP.h"

class SignalTimingPlan
{
public:
	int No_phases;
	int Phase_Seq[8];   //Maximum allow 8 phases
	float Gmin[8];
	float Yellow[8];
	float RedClr[8];
	float Gmax[8];
	float PedWk[8];
	float PedClr[8];
};

int ReadSignalTimingPlan (char * Plan_File_Name)
{
	SignalTimingPlan Plan;
	// Parse the MAP file
	ReadSigPlan(Plan_File_Name, Plan);
	printf("Read the Signal Plan successfully At (%d).\n",time(NULL));
	return 1;
}
```

Signal Status Parameters

The signal status parameters (TCStatus) are the real-time signal status obtained from MRP_TrafficControllerInterface component. Those parameters include current green phase, and green elapsed time. A request will be sent to MRP_TrafficControllerInterface for those parameters. The structure of the signal status parameters is defined in MRP_PRS_PriorityRequestServer component.

Supporting Code:

```
#ifndef SignalStatus_MSG_SIZE
    #define SignalStatus_MSG_SIZE  (45)
#endif
char Recvbuf[SignalStatus_MSG_SIZE];

int GetSignalStatus(int sockfd)
{
       int recv_data_len;
       recv_data_len=Socket_Receive(sockfd);
       return recv_data_len;
}
```

Provided (output)

The Signal Timing Schedule messages contains the duration and sequence of each phase. The phase sequence consists of combination of two current phases. After phase allocation algorithm generating the message, it will be sent to MRP_TrafficControllerInterface for traffic control. The structure of the signal status parameters is shown as follows:

```
SigTimScheMessage :: = Sequence {
   Header      InternalMsgHeader          -- two bytes (0xFFFF)
   msgID       SigSchemsgID               -- one byte (0x01)
   timeStamp   CurEpochTime               -- four bytes (0.01s)
   datablob    SigTimScheblob                 -- SigTimSche message payload}
```

The SigScheblob is a compilation of objects in binary form that provides the duration and sequence of each phase for signal control. Table 8 describes each object identifier.

Table 8 Object Identifiers of SigTimScheblob Data

Object Identifier	**Object Type**
0x01	Phase Sequence
0x02	Phase Duration

Object Format – Phase Sequence

Field	Note
Object Identifier	0x01
Size	0x08
Phase Sequence	Unsigned Char [8]

Object Format – Phase Duration

Field	Note
Object Identifier	0x02
Size	0x10

Phase Duration	Unsigned Integer [8]

Supporting Code:

```
#include <string>
#include <stdio.h>
#include <stdlib.h>
char Sendingbuf[102400]; //at most 100k
int SendSigSche(int sockfd, mschedule SigTimSche)
{

    //Process the Signal Schedule to an octet string before sending
    ScheduleProcessing (Sendingbuf, SigTimSche)
    //Send the trajectory
    int numbytes = sendto(sockfd, Sendingbuf,sizeof(Sendingbuf)+1 , 0,(struct
sockaddr *)&recvaddr, addr_length);
    return numbytes;
}
```

5.3.5.5 Functional Specification/Description

EVLS Algorithm

Due to the low penetration rate of the connected vehicles, an algorithm that estimates the states of unequipped vehicle (EVLS) based on connected vehicle data is developed to construct a complete arrival table for the phase allocation algorithm. The road segment before the intersection is divided into three regions: queuing region, slow-down region and free-flow region as shown in Figure 29. Different traffic flow models are applied to each region to estimate unequipped vehicle states. Shockwave theory is applied in queuing region to estimate the queue length. In slow-down region, Wiedemann's car following model is applied to insert unequipped vehicles and calculate their location and speed. In free-flow region, vehicles are randomly inserted on the roadway based on market penetration rate.

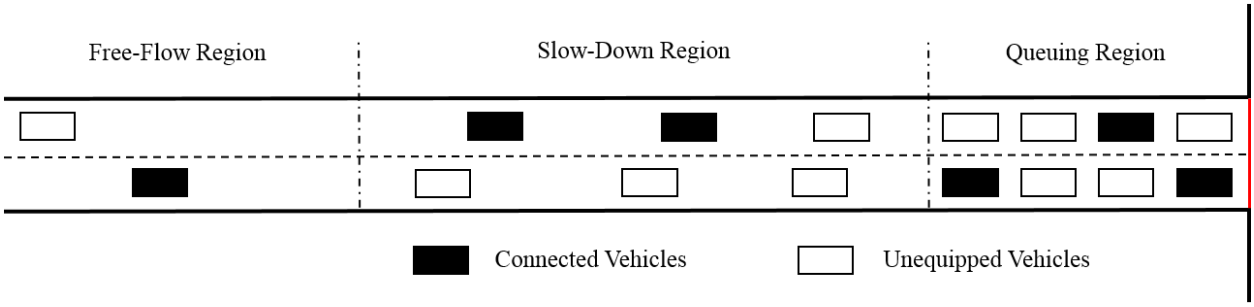


Figure 29 EVLS Algorithm Illustration

Constructing the arrival table

A trajectory-based arrival table is constructed based on ETAs and requested phases of all vehicles after executing EVLS algorithm. After MRP_TrafficControl computes the trajectory-based arrival table, it will combine the data with detector data that is acquired from the MRP_TrafficControllerInterface to create a modified arrival table that will be the input for phase allocation algorithm. Figure 30 shows the structure of the arrival table. The arrival table is a two dimensional matrix with time and phase respectively. The value of each cell is the number of vehicles that will arrive at the stop bar at time point *t* requesting phase *p*. An example arrival table can be seen in the application example section.

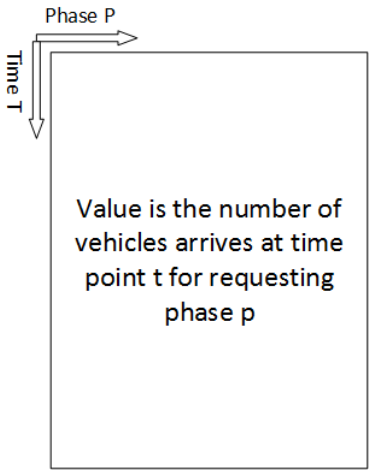


Figure 30 Arrival Table Structure

The Phase Allocation Algorithm

The principle part of MRP_TrafficControl component is the phase allocation algorithm. The phase allocation algorithm provides the adaptive control capabilities that the stakeholders desired. It is based on an improvement of controlled optimization of phases algorithm (COP) (Sen and Head, 1997) which is used in the RHODES adaptive traffic control system (Mirchandani and Head, 2001). The algorithm is based on dynamic programming that can result in optimal phase sequences and timing considering a variety of performance indices such as delay, stops and queue length. The algorithm applies a rolling-horizon approach where the optimization problem is solved at the end of each stage of implementation in order to include more recent arrival data.

The original COP algorithm was based on a sequence of stages, e.g. A, B, C, D, where a stage could represent phases 1 and 5 (denoted A), or phases 2 and 6 (denoted B),

but it didn't support flexible, or dual ring, phase sequences. The MMITSS phase allocation algorithm applies a two-level optimization scheme based on the dual ring controller as shown in Figure 31. At the upper level, a dynamic programming framework similar to COP is applied to each barrier group (e.g. phases 1 and 2 in ring 1 and phases 5 and 6 in ring 2) The sequence of barrier groups is assumed to be fixed, but the order of phases within each ring in each barrier group can vary. Initially, two stages of barriers with a total time horizon of 120 seconds are planned, but this could be modified to include three or more barrier groups depending on the time horizon and availability of data.

First, the minimum and maximum allowed barrier group lengths are calculated according to variety of signal parameters. Those parameters include the minimum green, maximum green, yellow interval and red clearance time of each phase. For the first barrier group, the elapsed time since the start of green of the current and previously served phases is taken into consideration. If one of the barrier group phases has no demand, then that phase can be skipped, otherwise, it will be served for at least the minimum green time. To determine the value function of a barrier group with a fixed length, the lower optimization level considers the phase duration and sequence within that barrier group. The four phases in one barrier are further divided into two rings. For each ring, the combinations of the two phase lengths and sequences are iterated to find the minimum delay combination. This combination is considered as the optimal phase split and sequence for that barrier length.

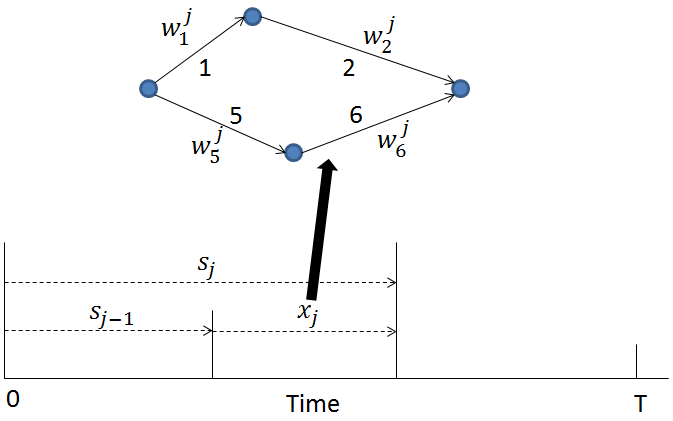


Figure 31 Two-level Optimization of Phase Allocation

The output of the phase allocation algorithm is a signal plan for three barriers groups with phase sequence and duration. The MRP_TrafficControl component then generates a schedule for controlling the signal. The schedule is a list of event which consists of control time, control command (e.g. call, hold, force-off, and omit) and which phase to

control. The control commands are sent to MRP_TrafficControllerInterface component and then to signal controller.

Dilemma Zone Protection

The objective of the MRP_TrafficControl component not only considers the efficiency part of the intersection operation such as delay or number of stops, but also the safety part. The number of vehicles in the dilemma zone (NVDZ) is used to represent the safety issue. An analytical model (Feng et.al, 2014) is developed to predict the NVDZ when yellow onset. In a connected vehicle environment, NVDZ can be directly obtained by vehicle trajectory data from MRP_EquippedVehicleTrajectoryAware component. The phase allocation algorithm is solved first and generate the optimal solution in terms of delay. Then green time can be adjusted around the optimal duration to extend longer or terminate earlier to reduce the NVDZ. An economic model can be applied to minimize the total combined delay of delay and NVDZ.

5.3.5.6 Example Applications

The sequence diagram of the component is shown in Figure 32.

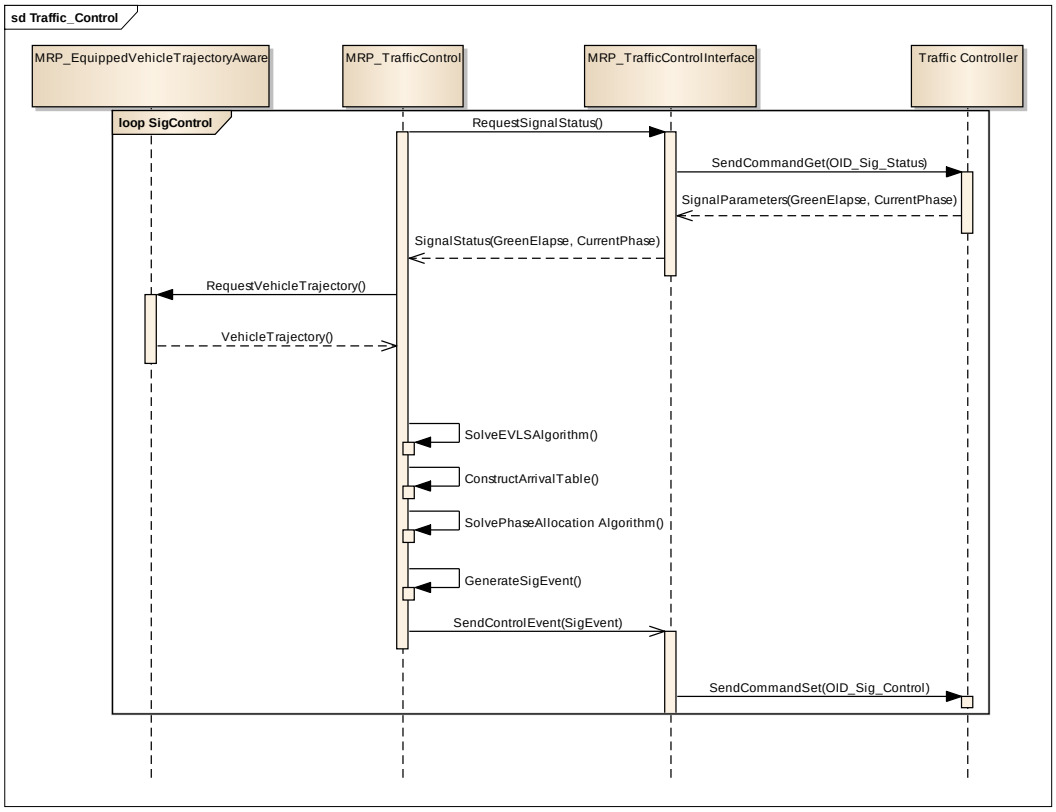


Figure 32 Sequence Diagram of MRP_TrafficControl Component

A sample phase allocation problem is described below. Each row represents one second of time and the columns represent the estimated number of vehicles to be served during that second of time.

A sample arrival table for 20 seconds of time:

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 7	Phase 8
0	0	4	0	2	0	7	0	4
1	0	1	0	0	1	0	2	0
2	0	0	0	0	1	0	0	1
3	1	0	0	0	0	1	0	0
4	0	1	0	0	0	2	0	0
5	0	0	0	0	0	0	0	1
6	0	1	0	0	1	0	0	0
7	0	0	0	0	0	1	0	0
8	0	0	0	1	0	0	0	1
9	1	0	0	0	0	0	0	0
10	0	1	0	1	0	2	0	0
11	0	0	0	0	0	1	0	1
12	0	1	0	0	0	0	0	0
13	0	0	0	0	0	1	0	0
14	0	1	0	1	0	0	1	1
15	0	0	0	0	0	0	0	0
16	1	1	0	0	0	0	1	0
17	0	0	1	0	1	0	0	1
18	0	0	0	1	0	0	0	0
19	0	0	0	0	0	1	0	0

The Signal Parameters for this example are assumed to be:

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6	Phase 7	Phase 8
MinGreen	8	20	7	10	8	20	7	10
MaxGreen	35	39	35	32	15	39	35	32
Yellow	3	3	3	3	3	3	3	3
RedClr	2	2	2	2	2	2	2	2

Assume that the current phases are 2 and 6 and that the green elapse time is 4 second for phase 2 and 4 seconds for phase 6.

The phase sequence (based on an initial lagging left turn configuration) and duration optimized by the algorithm:

Phase Seq R1	Phase 2	Phase 1	Phase 4	Phase 3	Phase 1	Phase 2
Phase Duration R1	16	8	11	15	8	20
Phase Seq R2	Phase 6	Phase 5	Phase 8	Phase 7	Phase 5	Phase 6
Phase Druation R2	16	8	19	7	20	8

The signal control event list based on the optimal schedule (1st barrier):

Time	Command	Phases
16.0	VEH_FORCE_OFF	2 and 6
16.0	VEH_PHASE_CALL	1 and 5
29.0	VEH_FORCE_OFF	1 and 5
29.0	VEH_PHASE_CALL	4 and 8

5.3.5.7 References

1. Sen S. and Head K.L., Controlled Optimization of Phase at an Intersection, Transportation Science. Vol. 31, No. 1, pp. 5 – 17, 1997.
2. Mirchandani P. and Head L., A Real-time Traffic Signal Control System: Architecture, Algorithms, and Analysis. Transportation Research Part C Vol. 9, pp. 415-432, 2001.
3. Feng, Y., Head, K.L., and Ma, W., Estimating Vehicles in the Dilemma Zone and Application to Fixed-time Coordinated Signal Optimization. Transportation Research Record, 2014 (in press).
4. M. Li, Y. Yin, W.B. Zhang, K. Zhou and H. Nakamura (2010). *Modeling and Implementation of Adaptive Transit Signal Priority on Actuated Control Systems*. Computer-Aided Civil and Infrastructure Engineering, 26(4): 270-284.

5.3.6 MRP_TrafficControllerInterface

5.3.6.1 Overview of Functionality

The *MRP_TrafficControllerInterface* component is responsible for providing the protocol specific interface to the traffic signal controller. The MCDOT SMARTDrive network uses Econolite ASC/3 NTCIP controllers with a fiber Ethernet backbone.

The functionalities[2] of this component include:

1. Collect controller's configuration data from traffic signal controller.

[2] Functions 1 and 2 are currently implemented in other components of MMITSS which requires controller information such as MRP_TrafficControl and MRP_PriorityRequestServer.

2. Collect vehicle detection data from traffic signal controller.
3. Command MMITSS traffic and priority control to traffic signal controller.

5.3.6.2 Requirements

Functional Requirements (reference)

Node: MRP	**Component Name:** MRP_TrafficControllerInterface
	Traceability: C2011.001, C2011.302, C2011.303, C2011.304, C2011.005, C2011.006, C2012.001, C2012.002, C2014.001, C2101.001, C2101.002, C2101.003, C2101.004, C2101.305, A2106, 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; §11.0, §11.0.2, §11.1.1, §11.1.1.2, §11.1.1.3, §11.1.4, §11.2.1, §11.2.2, §11.2.3, §11.3, §11.3.2, §11.3.3, §11.4.1, §11.5.1, §12.7.1, §5, §8
Description of Responsibility: This component is responsible for providing the protocol specific interface to the traffic signal controller.	
Supporting Text: *The MRP_TrafficControllerInterface component is a unique component for field test network implementation in terms the protocol supported for communications. The MCDOT SMARTDrive network uses NTCIP controllers over wired Ethernet. This component will provide the necessary protocol specific interfaces including messages and channel control/management capabilities.*	

Derived Requirements

5.3.6.2.1.1 Timing

The traffic controller (Econolite ASC/3) updates the active phase and active interval every 100 milliseconds, hence, the *MRP_TrafficControllerInterface* component should communicate with TSC at the same 100 milliseconds interval.

5.3.6.3 Interfaces

The interface of the MRP_TrafficControllerInterface component with other MMITSS components is described below.

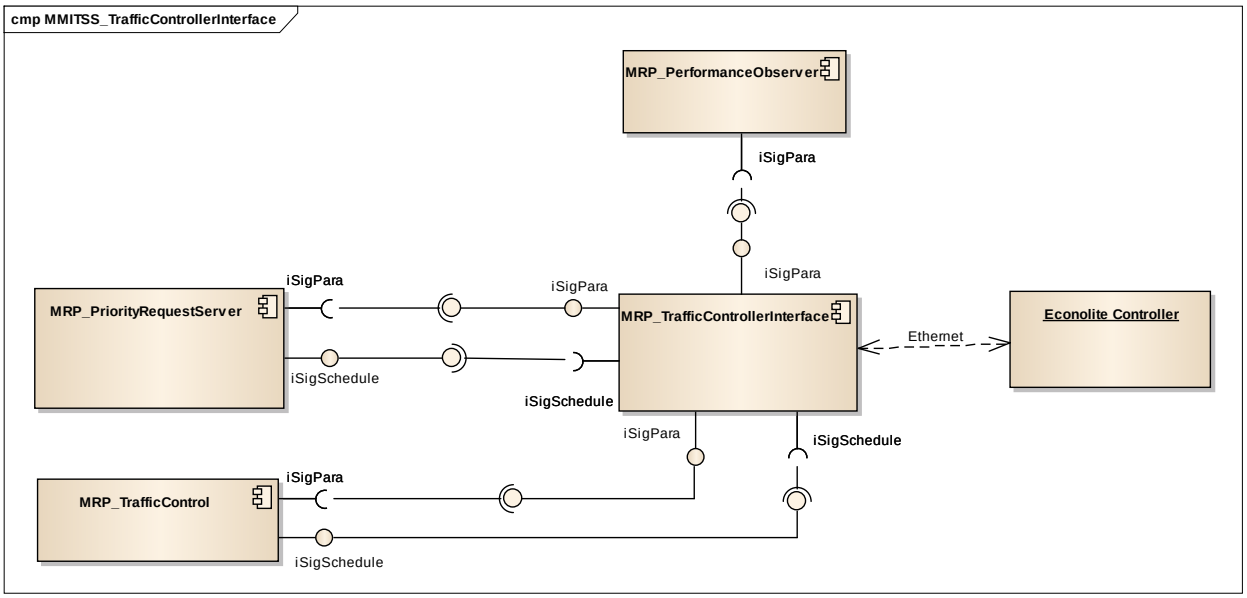


Figure 33 Message Exchange between MRP_TrafficControllerInterface and other MMITSS Components

Required (input):
Receive data objects of *PhaseTimingIntervals* and *DetectorParameters* (see Table 9 for definition of the data objects) from the Econolite ASC/3 NTCIP controller.

Table 9 Controller's Configuration Data

Data Object	Parameters	NTCIP 1202 Protocol Objects[0]	Usage
PhaseTimingIntervals	PhaseNumber PhaseWalk PhasePedestrianClear PhaseMinimumGreen PhasePassage PhaseMaximum PhaseYellowChange PhaseRedClear	Phase Parameters	Input to *MRP_TrafficControl* & *MRP_PRS_PriorityServer* for phase allocation
DetectorParameters	DetectorID DetectorType DetectorCallPhase NTCIPOccupancy NTCIPVolume	Detector Parameters	Input to *MRP_PerformanceObser* for performance estimation *MRP_TrafficControl* for phase allocation

Receive PhaseTimingIntervals

The *MRP_TrafficControllerInterface* component polls the controller's configuration data with NTCIP 1202 protocol over Ethernet from the Econolite ASC/3 controller with NTCIP 1202 protocol objects '*Phase Parameters*'. The controller's phase data is stored as local files as well as in the memory. Once it has been detected that the traffic signal controller

has changed the pattern, the *MRP_TrafficControllerInterface* component will poll the configuration data from traffic signal controller, compare the received data with previous saved data, and update the files and memory with any updated parameters. As traffic engineers may change the configuration data parameters in the field using controller's front panel, this data checking and verification will ensure that the MMITSS system always keep tracking the in-use configuration parameter.

Receive DetectorParameters

The *MRP_TrafficControllerInterface* component polls the system detector data (volume and occupancy) from Econolite ASC/3 controller with NTCIP 1202 protocol objects '*Vehicle Detector Status*' and '*Volume/Occupancy Report*'.

Receive SigTimScheMessage

SigTimScheMessage contains the desired signal control schedule parameters from *MRP_TrafficControl* and *MRP_PriorityRequestServer* for MMITSS traffic and priority control. The *MRP_TrafficControl* and *MRP_PriorityRequestServer* will send the Signal Timing Schedule messages through UDP socket after solving for optimal signal timing schedule. The data structure of Signal Timing Schedule message is defined in MRP_TrafficControl interface section (SigTimScheMessage).

Provided (output)
Send Signal Timing Plan Message

Signal Timing Plan messages contains the phase timing intervals and coordination control plan parameters. When requested by *MRP_TrafficController* and *MRP_PriorityRequestServer*, the *MRP_TrafficControllerInterface* will send *SigPlanMessage* to the requested component. The structure of *SigPlanMessage* is shown as follows:

```
SigPlanMessage :: = Sequence {
   Header       InternalMsgHeader          -- two bytes (0xFFFF)
   msgID        SigPlanmsgID               -- one byte (0x01)
   timeStamp    CurEpochTime               -- four bytes (0.01s)
   datablob     SigPlanblob                -- SigPlan message payload}
```

The SigPlanblob is a compilation of objects in binary form that provides the phase timing parameters and coordination plan parameters.

Table 10 Object Identifiers of SigPlanblob

Object Identifier	Object Type
0x01	Phase Timing Parameters
0x02	Coordination Plan Parameters

Object Format – Phase Timing Parameters

Field	Note
Object Identifier	0x01
Size	0x31
PermittedPhase	Byte, bit mapped with permitted phase
Gmin	Unsigned char[8], minimum green interval in seconds
Gmax	Unsigned char[8], maximum green interval in seconds
PedWk	Unsigned char[8], pedestrian walk interval in seconds
PedClr	Unsigned char[8], pedestrian flashing don't walk interval in seconds
Yellow	Unsigned char[8], yellow change interval in 10^{th} seconds
RedClr	Unsigned char[8], red clearance interval in 10^{th} seconds

5.3.6.3.1.1 System detector message

When requested, *MRP_TrafficControllerInterface* will send SysDectMessage to *MRP_PerformanceObserver*, which contains the volume and occupancy data obtained from TSC.

```
SysDectMessage :: = Sequence {
   Header       InternalMsgHeader          -- two bytes (0xFFFF)
   msgID        SysDectmsgID                 -- one byte (0x01)
   timeStamp    CurEpochTime               -- four bytes (0.01s)
   datablob     SysDectblob                 -- System detector message payload}
```

5.3.6.3.1.2 Control message

The control commands to the TSC from *MRP_TrafficControllerInterface* are defined in NTCIP 1202 standard. See [1].

5.3.6.4 Functional Specification/Description

Collect controller's configuration data from the traffic signal controller

The controller's configuration data includes a compilation of objects that specify the phase timing intervals. Table 9 lists the parameters for each of the objects, the

corresponding object in the NTCIP 1202 protocol that stores the object data, and the usage of the object data (as inputs to other MMITSS components).

Collect vehicle detection data from traffic signal controller
Vehicle detection data (count, occupancy, and detector fault/alarm) data provides inputs to *MRP_PerformanceObser* for performance estimation, which is then fed into *MRP_TrafficControl* and *MRP_PRS_PriorityServer* for MMITSS traffic and priority control, such as current queue length. The corresponding NTCIP 1202 Protocol objects are Vehicle Detector Status and Volume/Occupancy Report.

Control Signal Controller with NTCIP commands
When the MRP_TrafficControllerInterface receives the Signal Timing Schedule messages from the MRP_TrafficController or from the MRP_PriorityRequestServer, the MRP_TrafficControllerInterface component will communicate with the traffic signal controller (Econolite ASC/3 controller) to control the duration of signal phase with NTCIP 1202 protocol. NTCIP HOLD, FORCE-OFF, OMIT, and CALL commands are used to manipulate the controller phase timings.

5.3.6.5 References
[1] NTCIP 1202 v02.19. November 2005.

5.3.7 MRP_PedRequestServer (MMITSS Ped App)

5.3.7.1 Overview of Functionality
The MRP_PedRequestServer is responsible for reading the Ped SPaT status from MRP_MAP_SPAT_Broadcast and broadcasting it to the smartphone. It also receives the Ped calls from the smartphone and place those calls on the traffic controller.

5.3.7.2 Functional Specification
MRP_MAP_SPAT_Broadcast reads the SPaT information from the Signal Controller and writes it to a file. This information is in turn read by the MRP_PedRequestServer and broadcast to the network. The smartphone application has a class for decoding this information and find out current status of desire phase that pedestrian want to cross and shows it to pedestrian. If signal status is in “Flashing Don’t Walk” interval or in “Don’t Walk” interval application will send service request to MRP_PedRequestServer and it forward it to traffic signal controller.

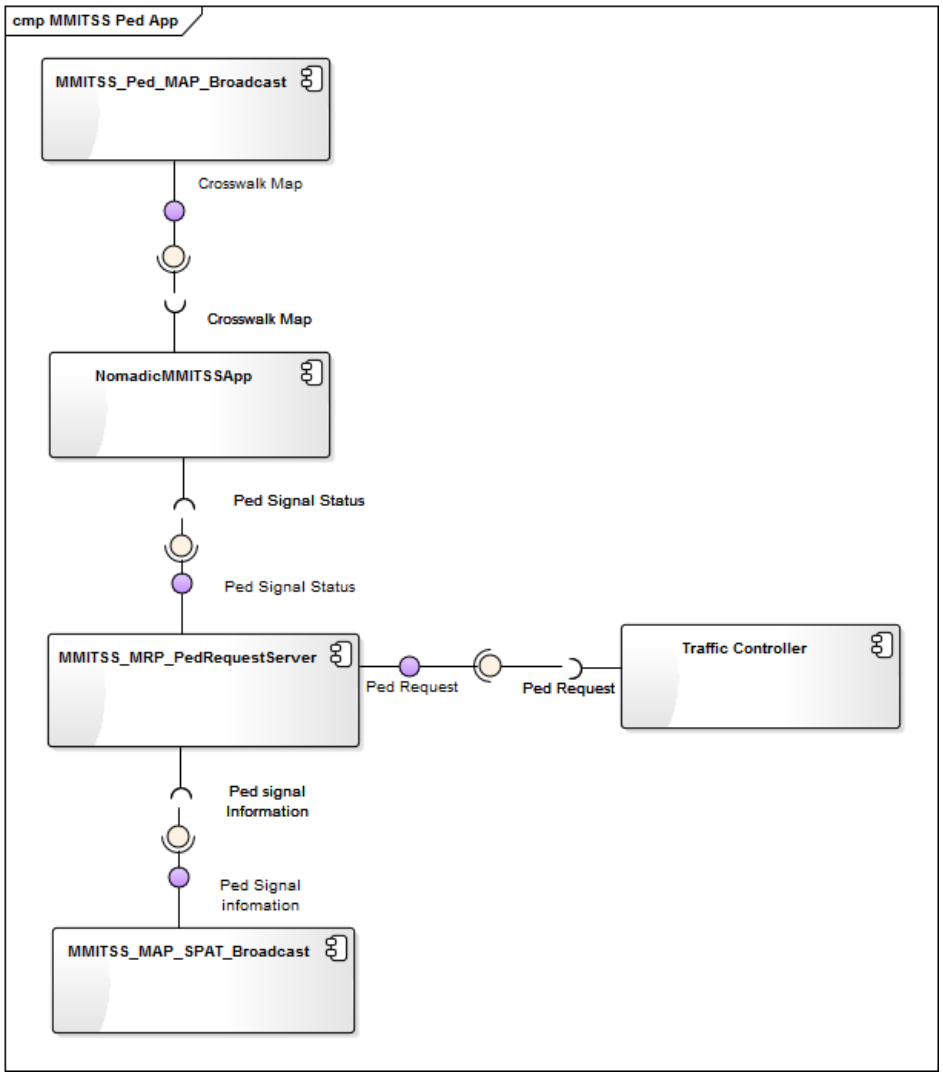


Figure 34 MMITSS Pedestrian Application Components

5.3.7.3 Unit Test Descriptions (debugging and testing)

This application has been tested at different intersections. It was tested several times at Daisy Mountain Drive and Gavilan Peak at the Anthem test bed. The RSE and smartphone are in the same local subnet at the intersection. The NomadicMMITSSAPP is installed on an Android smartphone. The RSE runs the applications, MRP_PedRequestServer and MRP_Ped_MAP_Broadcast.

The Controller will send real time information about the pedestrian signal indications to RSE and RSE will forward it to phone so the pedestrian can view this information. The application has been tested in all the eight possible crossing directions and verified its functionality. The same tests were successfully carried out at Speedway and Mountain Ave, Tucson.

5.3.8 MRP_Ped_MAP_Broadcast

5.3.8.1 Overview of Functionality

MRP_Ped_MAP_Broadcast broadcasts the crosswalk map that is received on the smartphone by the Nomadic_MMITSSApp. This map is then decoded on the smartphone by a class based on the pedestrian's GPS location can recognize if it is located near crosswalk.

5.3.8.2 Functional Specification

A MAP file that includes crosswalk information is created for every intersection. Each intersection map is defined as a series of Crosswalks. Each crosswalk is described as a rectangle by four GPS points taken from Google Earth. This information is stored in a MAP description file as shown below. This file is parsed and the information is serialized and broadcast to the network.

This Map Description file is for the Daisy Mountain Drive and Gavilan Peak intersection (Figure 35) at the Anthem Test Bed is shown below:

```
MAP_Name  Mountain_Speedway.nmap  /* 08082014 */
RSU_ID    Daisy_Gavlin
IntersectionID 301
Reference_point     32.235982  -110.952402 2449   /*lat, long,
evelation(decimeter)*/
No_Approach    4
Approach  1
Approach_type  2                    /*1: Approach, 2: Phase*/
No_nodes  4
2.1       33.842719 -112.135399
2.2       33.842683 -112.135378
2.3       33.842733 -112.134946
2.4       33.842777 -112.134953
end_approach
Approach  2
Approach_type  4                    /*1: approach, 4: phase*/
No_nodes  4
4.1       33.843127 -112.135404
4.2       33.843107 -112.135444
4.3       33.842701 -112.135405
4.4       33.842694 -112.135344
end_approach
Approach  3
Approach_type  6                    /*1: approach, 6: phase*/
No_nodes  4
6.1       33.843164 -112.135014
6.2       33.843204 -112.135024
```

```
6.3       33.843142 -112.135408
6.4       33.843096 -112.135432
end_approach
Approach  4
Approach_type  8                    /*1: approach, 8: phase*/
No_nodes  4
8.1       33.842742 -112.134988
8.2       33.842768 -112.134922
8.3       33.843193 -112.135038
8.4       33.843181 -112.134990
end_approach
end_map
```

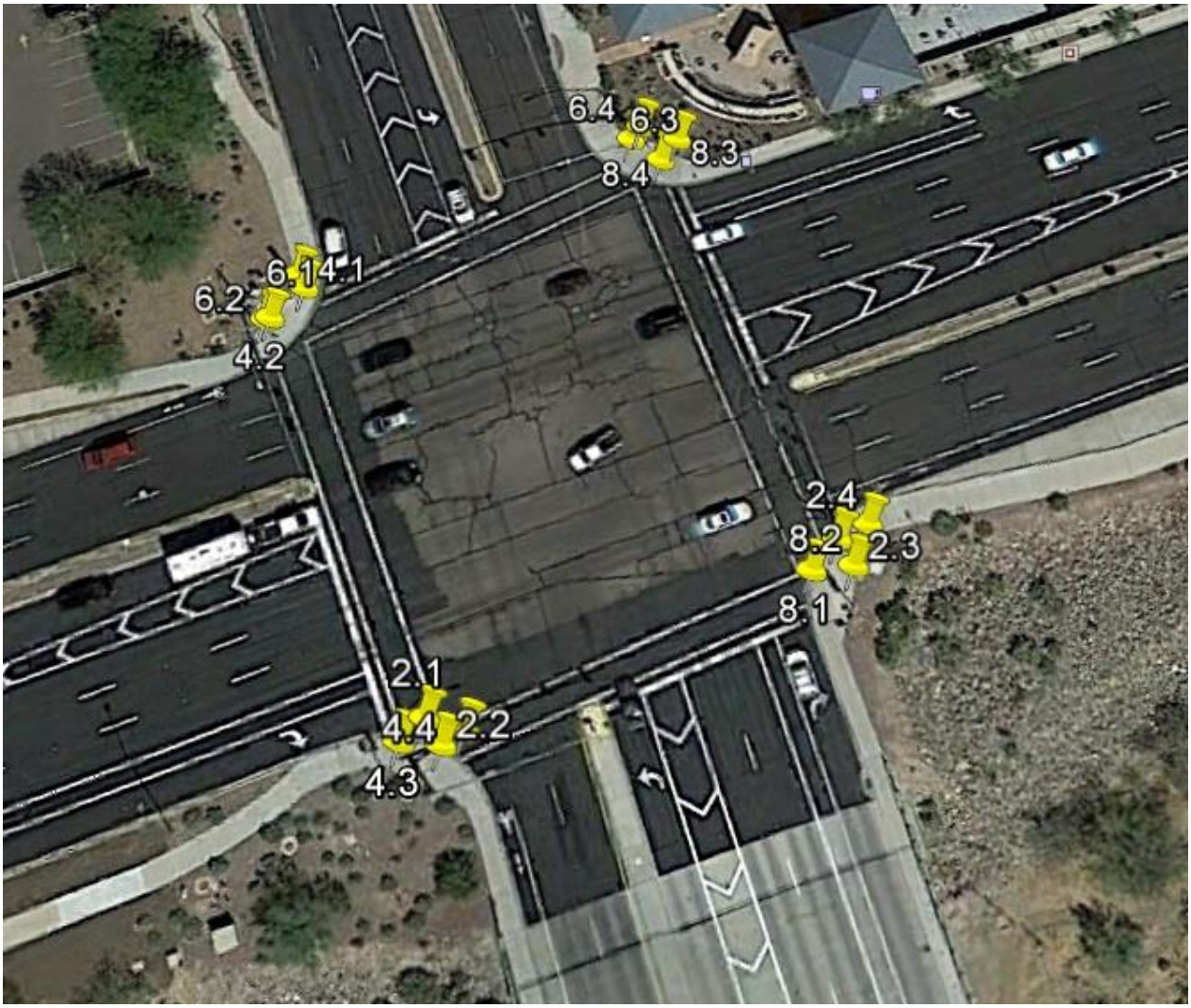


Figure 35 Waypoints for crosswalk at Daisy Mountain Dr and Gavilan Peak

5.3.8.3 Unit Test Description (debugging and testing)

The MMITSS_Ped_MAP_Broadcast testing dealt with:

- Verifying the serialization of the crosswalk points corresponding to the approach numbers on the RSE side.
- The accuracy of the GPS points, which is crucial to enable the smartphone to locate itself on the map and compare its heading with that of the crosswalk.
- Verifying that the smartphone's location and heading read from the sensors against the values calculated from the MAP.
- Using the above information to test for the accuracy and efficacy of geo-fence that the smartphone uses to locate itself and enable itself to request for Ped calls.

5.3.9 MRP_PerformanceObserver

5.3.9.1 Overview of Functionality

The main responsibility of the MRP_PerformanceObserver is to acquire data from other MRP components, including the MRP_EquippedVehicleTrajectoryAware and the MRP_TrafficControllerInterface components, to compute observed performance measures. The MRP_PerformanceObserver includes estimation, managing, and archiving the data. All the acquired performance observations and metrics are collected and reported in terms of peak period, all day, or specially defined time period.

5.3.9.2 Requirements

Node: MRP	**Component Name:** MRP_PerformanceObserver
	Traceability: C2013.001, C2013.002, C2013.003, C2013.004, C2013.005, C2013.006, C2013.007, C2013.008, C2015.001, C2015.002, C2015.003, C2015.004, C2015.005, C2015.006, C2015.007, C2015.008, C8101.101, 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; §5, §8, §11; §11.1.1, §11.1.3, §11.2.1, §11.2.2, §11.2.3, §11.4.1, §11.5.1
Description of Responsibility: This component is responsible for acquiring data and estimating intersection level performance measures. The collected measures may be requested by section or system level performance observers, or other MRP control processes such as MRP_TrafficControl and MRP_PriorityRequestServer.	
Supporting Text: *The MRP_PerformanceObserver is responsible for the estimation of performance measures from data available from the other MRP level components.*	

Functional Requirements

As initiated in the Stakeholder feedback and furthered in the MMITSS Concept of Operations Workshop, the collection, measurement, and estimation of performance observations serves a variety of users and needs. The requirements defined for this component are for acquiring, processing and evaluation of the data at the intersection level.

Performance observations shall be collected based on a defined period of collection, e.g. all day, peak period, 1 hours, and at a defined frequency. For example, the system might observe travel time based during the AM peak period from 7:00 AM to 9:00 AM and report travel times every 15 minutes.

The Basic Safety Message contains the vehicle temporary ID, latitude, longitude, elevation, speed, heading, and vehicle size. It doesn't have the vehicle type information, although it can be determined according to the Signal Request Message – if a vehicle sends a signal request message. So observing the performance of the travelers characterized by mode is currently achievable only after after receiving the SRMs. Table 11 depicts the available data and the corresponding modes. [Note: the issue of including vehicle classification is currently being considered by the SAE DSRC Technical Committee (May 22, 2015). Different classification schemes are being discussed that may include mode classification in the Part II BSM that would be transmitted at 1 Hz. Passenger vehicles would not report the mode classification and would be classified as passenger vehicles by default. Note that modal performance cannot be accurately achieved without including mode information in the BSM since vehicles may not request priority – such as a bus that is on schedule.]

Table 11 Data available from BSM and SRM

Mode	BSM Data	SRM Data
Vehicle	✓	✓
Transit		✓
Truck		✓
Emergency Vehicle		✓
Pedestrian	✓	✓

The Observation of performance measures and metrics in terms of structure and quantity at intersection level are presented as follows:

Intersection Delay: MMITSS performance shall be characterized based on delay of all vehicles and travelers traversing the intersection. Vehicle delay is the

broadest aggregate measure of intersection performance, and is relevant to all MMITSS use cases, including those that are not intended to improve delay and may make it worse (e.g., accommodation for pedestrians with disabilities). This parent requirement is decomposed into child requirements for any specified observation period that could include all day or peak period.

- **Average Vehicle Delay:** MMITSS performance shall be characterized based on the average delay of all vehicles and travelers during the specified observation period.

- **Average Transit Delay:** MMITSS performance shall be characterized based on the average transit vehicle delay during the specified observation period.

- **Average Truck Delay:** MMITSS performance at individual intersections shall be characterized based on the average truck delay during the specified time period.

- **Average Emergency Vehicle Delay**: MMITSS performance shall be characterized based on the average emergency vehicle delay during the specified observation period.

- **Average Pedestrian Delay:** MMITSS performance will be characterized based on the average delay during the specified observation period experienced by pedestrians while traversing the intersection.

Intersection Delay Variability: MMITSS performance will be characterized based on the variability of delays for all vehicles traversing the intersection. The variability of delay of vehicles traversing the intersection is the statistical dispersion, or variation of delay of those vehicles observed over the time period of interest. This parent requirement is decomposed into child requirements for any specified observation period that could include all day or peak period.

- **Vehicle Delay Variability:** MMITSS performance will be characterized based on the variability of delays during the specified observation period.

- **Transit Delay Variability:** MMITSS performance will be characterized based on the variability of transit delays during the specified observation period.

- **Emergency Vehicle Delay Variability:** MMITSS performance will be characterized based on the variability of emergency vehicle delays during the specified observation period.

Intersection Travel Time: MMITSS performance shall be characterized based on the total travel time associated with all vehicles traversing the intersection. Total intersection travel time is defined as the cumulative time that all vehicles require to travel when inside the boundaries of the intersection. This parent requirement is decomposed into child requirements for any specified observation period that could

include all day or peak period.

- **Vehicle Total Travel Time:** MMITSS performance shall be characterized based on the total travel time during the specified observation period for vehicles traversing the intersection.

- **Emergency Vehicle Average Travel Time:** MMITSS performance shall be characterized based on the emergency vehicle average travel time during the specified observation period while responding to emergencies, traversing the intersection. A meaningful measure of emergency vehicle operational performance is the response time for emergencies, which is best measured at the system level, but is impacted at the intersection and section levels by the travel time.

Intersection Travel Time Variability: MMITSS performance will be characterized based on the variability of the vehicles travel time. Travel time variability indicates the operational consistency of the intersection and a quality of service to users, who can plan their travel more efficiently. This parent requirement is decomposed into child requirements for any specified observation period that could include all day or peak period.

- **Vehicle Travel Time Variability:** MMITSS performance will be characterized based on the variability of the vehicles travel time during the specified observation period.

- **Freight Travel Time Variability:** MMITSS performance will be characterized based on the variability of freight travel times during the specified observation period. Although this measure is most relevant for complete trips spanning the system-level, information gathered at the intersection and section can provide a level of granularity for identifying localized problems and performance contributions. This may not be measurable directly from MMITSS data sources, but may require dispatch data from truck fleet operators.

Intersection Number of Stops: MMITSS performance shall be characterized based on the number of stops incurred by all vehicles at an intersection. This parent requirement is decomposed into child requirements for any specified observation period which could include all day or peak period.

- **Vehicle Number of Stops:** MMITSS performance shall be characterized based on the number of stops during the specified observation period.

- **Freight Vehicle Number of Stops:** MMITSS performance shall be characterized based on the freight vehicle number of stops during the specified observation period.

- **Emergency Vehicle Number of Stops:** MMITSS performance shall be characterized based on the emergency vehicle number of stops during the specified observation period. Whenever and where ever possible, MMITSS does not want to stop active emergency vehicles at an intersection.

Intersection Throughput: MMITSS performance shall be characterized based on total vehicle throughput traversing the intersection. The throughput measures include only vehicle counts in this measure to account for vehicular interactions among multiple travel modes (e.g., transit).

- **Vehicle Throughput:** MMITSS performance shall be characterized based on throughput during the specified observation period by vehicles traversing the intersection.

5.3.9.3 Interfaces

The MRP_PerformanceObserver component acquires trajectories and detectors data as input, and sends different performance measures, including queue length, as output. The structure of the interfaces are shown in Figure 36.

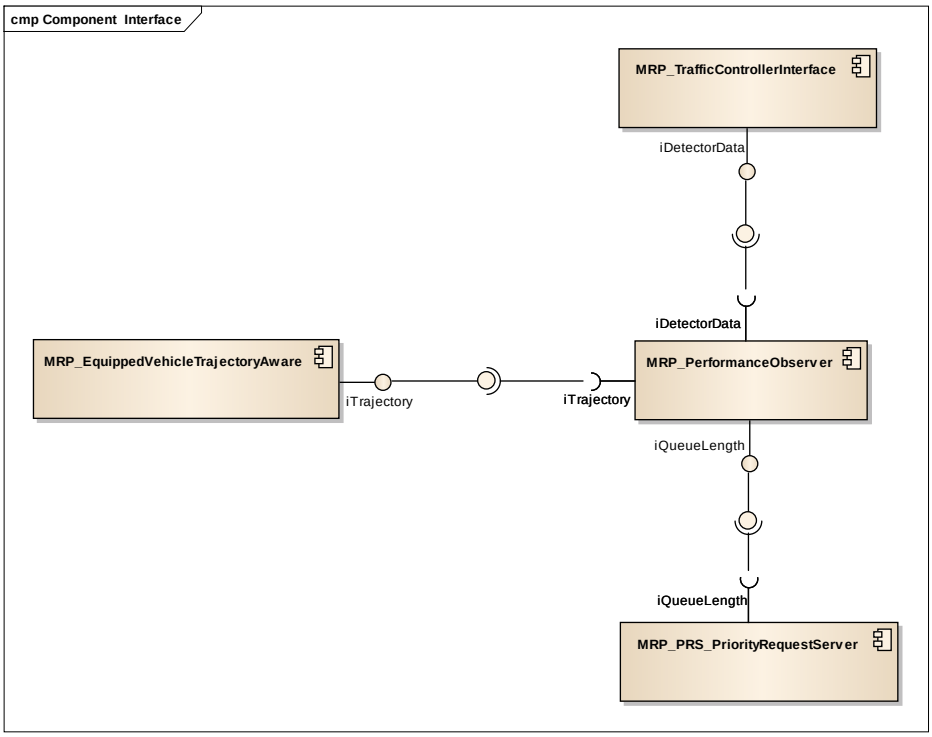


Figure 36 MRP_PerformanceObserver Component Interface.

Figure 37 below, shows the interaction between the MRP components. This component sends request to two other components to get the vehicle trajectory data and vehicle

detection data. After receiving this data it estimates the performance observations. It sends the queue length estimate to the MRP_PriorityRequestServer to be used in estimating a more accurate arrival time for priority vehicles.

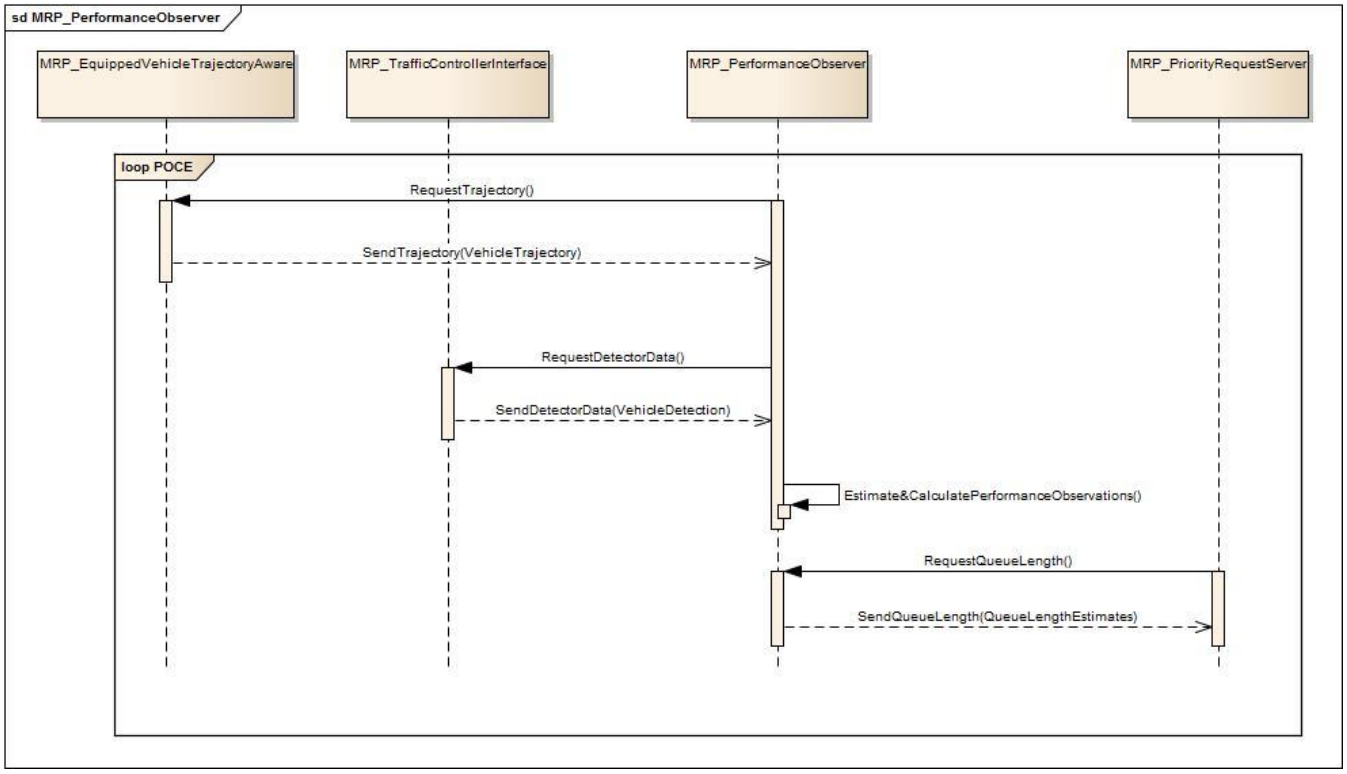


Figure 37 Sequence Diagram Showing the Interaction between Components

Figure 38 shows the system structure of Performance Observer component. The left side of the figure shows the data flows between the RSE and the OBE and Traffic Controller. The right side of the structure depicts the Linux or Windows server which acquires the performance metrics from the intersection application, stores them into a database, and provides tools for visualization in a dynamic web application (WAMP/LAMP: Windows/Linux Apache MySQL PHP).

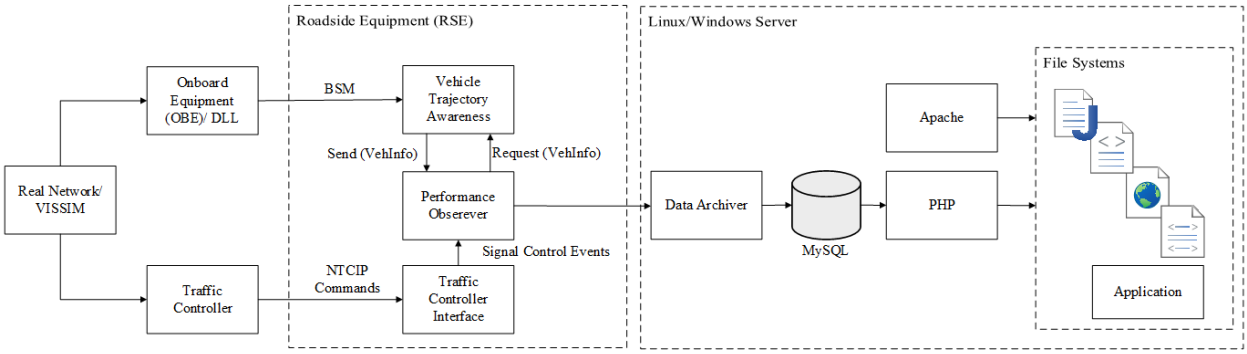


Figure 38 MRP_PerformanceObserver System Structure

5.3.8.3.1 Required (Input)

As Figure 36 depicts, the data required for this component is obtained from another MRP level component, MRP_EquippedVehicleTrajectoryAware, and includes the Vehicle Trajectory Database which is a processed collection of vehicle position, vehicle temporary ID, speed, heading, and associated time in the format of the Basic Safety Message. In addition, traditional loop detector data, including volume and occupancy, is acquired from the MRP_TrafficControllerInterface.

Receive Trajectory Data

This function reads Trajectory data from a UDP socket and returns the number of Bytes received. The trajectory message (octet stream) is defined in “Send Vehicle Trajectory” part of the MRP_EquippedVehicleTrajectoryAware component. The structure of trajectory message is shown below:

```
TrajMessage:: = Sequence {
Header      InternalMsgHeader      -- two bytes (0xFFFF)
MsgID       TRAJmsgID              -- one byte (0x01)
Datablob    TRAJblob               -- trajectory message payload
```

The TRAJ blob is a compilation of objects in binary form that provides the vehicle trajectory information while the vehicle was within the DSRC range. Table 12 describes each object identifier.

Table 12 Object Identifiers of Trajectory Blob Data

Object Identifier	**Object Type**
0x01	Vehicle ID
0x02	nFrame (Number of trajectory points)
0x03	Vehicle Trajctory

Object Format – Vehicle ID

Field	Note
Object Identifier	0x01
Size	0x02
Vehicle ID	Unsigned 16-bit Integer

Object Format – nFrame

Field	Note

Object Identifier	0x02
Size	0x02
nFrame	Unsigned 16-bit Integer

Object Format – Vehicle Trajectory

Field	Note
Object Identifier	0x03
Size	0x12
Latitude	Signed 32-bit Integer
Longitude	Signed 32-bit Integer
Elevation	Signed 16-bit Integer - optional
N_Offset	Signed 32-bit Integer
E_Offset	Signed 32-bit Integer

Note: N_Offset and E_Offset are local coordinates to the center of the intersection

The following code snippet shows how the MRP_PerformanceObserver can get data from the MRP_EquippedVehicleTrajectoryAware component using the socket based peer-to-peer architecture. After initializing the socket connection, the MRP_PerformanceObserver requests trajecotry data at 1 Hz, then listens for the data to be pushed from the MRP_EquippedTrajectoryAware component:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <getopt.h>
#include <unistd.h>
#include "ConnectedVehicle.h"
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include "LinkedList.h”
int ReceiveTrajectory(int sockfd, LinkedList <ConnectedVehicle> trackedveh)
{
//Establish socket connection to data provider
struct sockaddr_in recvaddr;
struct sockaddr_in sendaddr;
      //start socket communication
      if((sockfd = socket(PF_INET,SOCK_DGRAM,0)) == -1)
    {
        perror("sockfd");   exit(1);
    }
    if((setsockopt(sockfd,SOL_SOCKET,SO_BROADCAST,
        &broadcast,sizeof broadcast)) == -1)
```

```
    {
        perror("setsockopt - SO_SOCKET ");  exit(1);
    }
    //Network is unreachable
    sendaddr.sin_family = AF_INET;
    sendaddr.sin_port = htons(30000);
    sendaddr.sin_addr.s_addr = INADDR_ANY;
    memset(sendaddr.sin_zero,'\0',sizeof sendaddr.sin_zero);

    if(bind(sockfd, (struct sockaddr*) &sendaddr, sizeof sendaddr) == -1)
    {
        perror("bind");
        exit(1);
    }

    recvaddr.sin_family = AF_INET;
    recvaddr.sin_port = htons(30000);
    recvaddr.sin_addr.s_addr = inet_addr("127.0.0.1") ; //SEND TO LOCAL HOST,
//assuming MRP_PerformanceObserver resides in the same device
//change the IP address to the remote device if not on the same device
    memset(recvaddr.sin_zero,'\0',sizeof recvaddr.sin_zero);
    int addr_length = sizeof ( recvaddr );

    //Receive the trajectory message
    numbytes   =   recvfrom(sockfd,   Sendingbuf,   traj_data_size+1,0,(struct
sockaddr *)&recvaddr, addr_length);
    return numbytes;
}
```

Receive Detectors Data

This function reads the System Detectors data from MRP_TrafficControllerInterface. The data includes volume and occupancy. MRP_PerformanceObserver reads the data regarding the number of active system detectors and their associated lane at each intersection from Det_Numbers.txt.

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>
void get_Det_Numbers()
{
```

```
      fstream sh;
      sh.open(Det_Numbers_file);
      char temp[128];
      string temp_string;
    getline(sh,temp_string);  //Reading the first line which explains the
detectors assignment
    getline(sh,temp_string);  //Read the second line which indicate the
number of detectors
    char tmp[128];
      strcpy(tmp,temp_string.c_str());
      sscanf(tmp,"%d",&tot_num);
      Det_Number = new int[tot_num];
      getline(sh,temp_string);  //Reading the third line which has the
Detector Number for each lane
      if(temp_string.size()!=0)
      {
            strcpy(tmp,temp_string.c_str());
            char * pch;
            pch=strtok(tmp," ");
            for(int i=0; i<tot_num ; i++)
            {
                  sscanf(pch,"%d",&Det_Number[i]);
                  pch=strtok(NULL," ,.-");
            }
    }
    else
    {
        sprintf(temp,"Reading Det_Numbers_file problem!!");
        cout<<temp<<endl;
        outputlog(temp);
        exit(0);
    }
    getline(sh,temp_string);  //Reading the fourth line which explains the
Approach & Lane Combination for Distance from System Detectors to stop bar
      getline(sh,temp_string);  //Reading the fifth line which has the values
for the corresponding approach and lane
      if(temp_string.size()!=0)
      {
            strcpy(tmp,temp_string.c_str());
            char * pch;
            pch=strtok(tmp," ");
            for(int i=0; i<tot_num; i++) //going through approaches 1,3,5,7
and lanes 1,2,3,4,5
            {
                  sscanf(pch,"%f",&Detector_stopbar[i]);
                  pch=strtok(NULL," ,-");
            }
    }
    else
```

```
    {
        sprintf(temp,"Reading Stop bar distance of detectors problem!!");
        cout<<temp<<endl;
        outputlog(temp);
        exit(0);
    }
    getline(sh,temp_string);  //Reading the sixth line which explains the
Association between the Approach & Lane Combination and Phases
      getline(sh,temp_string);  //Reading the seventh line which has the
values for the corresponding phases
      if(temp_string.size()!=0)
      {
            strcpy(tmp,temp_string.c_str());
            char * pch;
            pch=strtok(tmp," ");

            for(int i=0; i<tot_num; i++) //going through approaches 1,3,5,7
and lanes 1,2,3,4,5
            {
                  sscanf(pch,"%d",&Lane_phase[i]);
                  pch=strtok(NULL," ,-");
            }
    }
    else
    {
        sprintf(temp,"Reading Phase Lane Mapping problem!!");
        cout<<temp<<endl;
        outputlog(temp);
        exit(0);
    }
    getline(sh,temp_string);  //Reading the eighth line which explains the
Turning Movement Proportions
    for(int i=0; i<tot_num ;i++) //Reading line by line the next 20 lines
which has the values for the corresponding TMPs
    {
            getline(sh,temp_string);
            if(temp_string.size()!=0)
            {
                  strcpy(tmp,temp_string.c_str());
                  char * pch;
                  pch=strtok(tmp," ");
                  for(int j=0;j<3;j++)
                  {
                        sscanf(pch,"%f",&Turning_proportion[i][j]);
                        pch=strtok(NULL," ,-");
                  }
            }
            else
            {
```

```
                    sprintf(temp,"Reading Turning Movement Proportions
problem!!");
                    cout<<temp<<endl;
                    outputlog(temp);
                    exit(0);
                }
            }
        sh.close();
}
```

5.3.8.3.2 Provided (Output)

The input data is used to compute the defined performance measures (observations) and can be used to draw the trajectory of the vehicles for visualization.

The trajectories acquired by processing the BSMs are deployed to decide on the vehicle counts, market penetration rate, and the traffic states (whether the vehicle is in the queue or not) at the intersection. Queue length is provided to the MRP_PRS_PriorityRequestServer component to improve the accuracy of the estimated arrival time of the priority requesting vehicle.

5.3.9.4 Functional Specification/Description

Based on the data originating from the MRP_EquippedVehicleTrajectoryAware component and detector data from the MRP_TrafficControlInterface component, the performance measures related to number of vehicles, travel time, delay, number of stops, and queue length are calculated. The trajectory data consists of the x-y offset coordinates (or GPS data) of the vehicle location, time, vehicle ID, and the phase to which the vehicle is approaching. When a vehicle enters the range of the DSRC radio, the information becomes available.

The component is configured to observe performance measures over the defined observation period at a minimum frequency of once per minute or a maximum of once per observation period. If no connected vehicle data is observed during the reporting interval a warning message is logged as “No Connected Vehicle Available!”

Table 13 shows all the performance observations being calculated and estimated at MRP_PerformanceObserver component and the data resource regarding each measure.

Table 13 Performance Observations and Associated Data Resources.

Performance Metric	Abbreviation	Unit	Data Source
Travel Time	TT	Second	MRP_EquippedVehicleTrajectoryAware

Delay	D	Second	MRP_EquippedVehicleTrajectoryAware
Travel Time Variability	TTV	Second	MRP_EquippedVehicleTrajectoryAware
Delay Variability	DV	Second	MRP_EquippedVehicleTrajectoryAware
Queue Length	QL	Meter/number of vehicles	MRP_EquippedVehicleTrajectoryAware
Number of Stops	NS		MRP_EquippedVehicleTrajectoryAware
Volume	V		MRP_TrafficControllerInterface
Occupancy	O	%	MRP_TrafficControllerInterface
Market Penetration Rate	MPR	%	MRP_EquippedVehicleTrajectoryAware & MRP_TrafficControllerInterface

Equation (1) defines how travel time for each individual vehicle for a specific phase is calculated by subtracting the time that the vehicle leaves the radio range from the time that the vehicle entered the radio range. This range can be measured separately for each intersection based on the received BSMs on RSE side, and doesn't have to be exactly the same as the DSRC radio range.

$$Travel\ Time\ =\ Leaving\ Time - Entrance\ Time \quad (1)$$

One of the principle concerns confronting the development of connected vehicle applications is ensuring privacy of the public travelers. The connected vehicle system is being developed with this privacy issue as a requirement. According to "Annex E: Traffic Probe Message Use and Operation" in the SAE J2735 (2009) Standard *(1)*, the data collection of a connected vehicle does not begin until 500m or 120 seconds (whichever occurs first) after the vehicle start up to aid anonymity. If a vehicle changes its temporary ID number while it is being observed, it is not difficult to associate its new ID number with its old number based on the position data that is received every 0.1 seconds. However, making this association would violate the privacy requirement. Hence, trajectory fragments can be utilized to estimate the travel time over a certain road segment (i.e. within the boundaries of DSRC range). When a vehicle randomly generates a new ID, it's considered as a new entity in the system in order to avoid matching the two trajectory fragments. Keeping the trajectories for a short amount of time and assuming the trajectories fragments are independent, are two ways to help

ensure no vehicle is tracked. In other words, privacy ensured connected vehicle environment is characterized as both non-tracing and non-matching in this methodology.

Figure 39 shows how the Travel Time (TT) and Distance (D) of a single connected vehicle are divided into two separate fragments. At time t_1, the vehicle enters the data collection range (or the DSRC radio range). From the time that the ID is changed (t_2) till the vehicle leaves the data collection range (t_3), it's treated as a different connected vehicle in the system. The two fragments of one trajectory are designated as Type 1 and Type 2 trajectory fragments designating the period before changing ID's (Type 1) and after changing ID's (Type 2). The number of trajectory fragments stored in the database is twice the number of vehicles that changed their IDs.

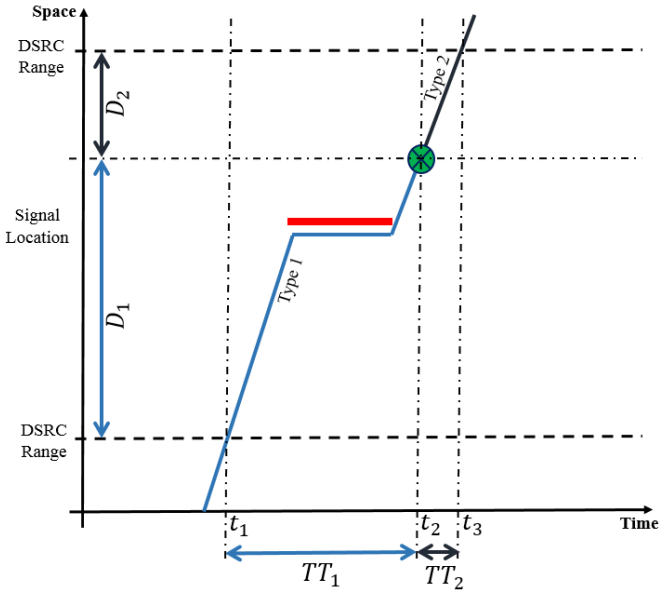


Figure 39 Trajectory Fragments of a single Connected Vehicle with a changed ID

Considering the fact that IDs of the vehicles are randomly changed every 300 seconds, there is a definite probability that it occurs while the car is passing the intersection and is in the range of the DSRC radio. The probability that a vehicle will change its ID can be estimated by dividing the vehicle's travel time by the 300 seconds. The probability of this happening for a moving object in the range is shown in equation (2). Meanwhile the chance of not changing ID in the same data set would be "1-p" accordingly.

$$Probability\ of\ changing\ ID = p = \frac{Travel\ Time\ of\ vehicle_j}{300} \quad (2)$$

Travel Time Estimation Using the Extended Tardity Function

Equation (3) shows how the average Extended Tardity is calculated based upon the trajectory fragments in the time-space diagram to obtain an estimate of the average link travel time of the vehicles in the system. In this Equation *n* is the number of the vehicles that didn't change their IDs; m_1 is the number of Type 1 vehicles that changed their IDs, and accordingly m_2 is the Type 2 number of cars with changing ID.

$$ETT = \left[\sum_{i=1}^{n} \frac{TT^i}{D^i} + \sum_{j=1}^{m_1} \frac{TT_1^j}{D_1^j} + \sum_{j=1}^{m_2} \frac{TT_2^j}{D_2^j}\right] \times \left[\sum_{i=1}^{n} D^i + \sum_{j=1}^{m_1} D_1^j + \sum_{j=1}^{m_2} D_2^j\right] / [n + m_1 + m_2] \quad (3)$$

where,

TT^i = Travel Time of Vehicle *i* that doesn't change its ID,
D^i = Distance Traveled by Vehicle *i* that doesn't change its ID,
TT_1^j = Travel Time of Vehicle *j* of Type 1,
TT_2^j = Travel Time of Vehicle *j* of Type 2,
D_1^j = Distance Traveled by Vehicle *j* of Type 1,
D_2^j = Distance Traveled by Vehicle *j* of Type 2.

The estimation of delay depends on two factors. One is the vehicle total travel time and the other is the free flow travel time. The travel time is estimated above, and as shown in equation (4) the free flow travel time is obtained based on the free flow speed and is specified as a pre-determined value in this design. However there are different definitions of free flow speed recognized by different agencies (e.g. Speed Limit), and they can be implemented to the logic as a configurable variable.

$$Free\ Flow\ Travel\ Time = \frac{Distance\ Range}{Free\ Flow\ Speed} \quad (4)$$

By knowing the values of travel time of each vehicle and free flow travel time at the intersection, delay is calculated as:

$$Delay = Travel\ Time - Free\ Flow\ Travel\ Time \quad (5)$$

Vehicle stopped delay is also measured based on the amount of time that the vehicle's speed is less than a threshold (designed to be 2 miles per hour) before reaching the stop bar.

When the number of vehicles traversing the intersection in a specific movement is known, total travel time and delay for movement "*i*" are calculated based on equations (6) and (7).

$$(Total\ Travel\ Time)_i = Number\ of\ Vehicles \times (Average\ Travel\ Time)_i \quad (6)$$

$$(Total\ Delay)_i = Number\ of\ Vehicles\ \times (Average\ Delay)_i \quad (7)$$

Based on the simple equations presented above, travel time, delay, and the total values of these measures are calculated for each specific movement. For obtaining the intersection travel time or delay the equations below are used, where *"i"* is an indicator of the available movements at the intersection:

$$\sum_{i=1}^{12}(Total\ Travel\ Time)_i \quad (8)$$

$$\sum_{i=1}^{12}(Total\ Delay)_i \quad (9)$$

The variability of travel time can be related to the Expected Value and Variance through the following relations (TT: Travel Time; FTT: Free Flow Travel Time; D: Delay).

$$\begin{gathered} TT = FTT + D \\ E(TT) = FTT + E(D) \\ Var(TT) = E[(FTT + D) - E(FTT + D)]^2 = E[D - E(D)]^2 = Var(D) \\ \sigma(TT) = \sigma(D) \end{gathered} \quad (10)$$

Travel time is free flow travel time plus delay. The expected value of travel time (average) is the free flow travel time (which is assumed to be deterministic) plus the expected value of delay (average delay). By definition of the variance, it can be shown that the variance of travel time is equal to the variance of delay.

An important performance metric being analyzed at the intersection level is queue length. The functional description of this measure is divided into two parts: Observing and estimating the queue length in real time in terms of the distance of the last stopped vehicle from the stop bar for each lane, and/or measuring the queue in terms of the number of stopped vehicles at each lane. These measures can be easily transformed into each other. For example, according to the data available from BSMs coming from the OBEs (vehicle trajectory database), the length of each individual vehicle is known,

and by knowing the number of vehicles in the queue, the overall queue length in terms of the distance from the stop bar can be calculated.
The main goal is to have an accurate and acceptable estimation of this measure when the market penetration rate is low. The algorithm developed addresses the market penetration problem by adding a parameter called Queue Increase Rate. Basically, by having information about the last stopped vehicle a lower bound for the queue length is provided. Then by adding the increase rate factor that is calculated based on the combination of historical and real-time data coming from the loop detectors to the current known queue status, an accurate estimate of the queue length is achieved.

Figure 40 shows the flow chart for how the algorithm works and estimates the queue length by lane either with or without system detectors. Numerical results show that the accuracy of the algorithm using system detectors data is higher than using only connected vehicle data.

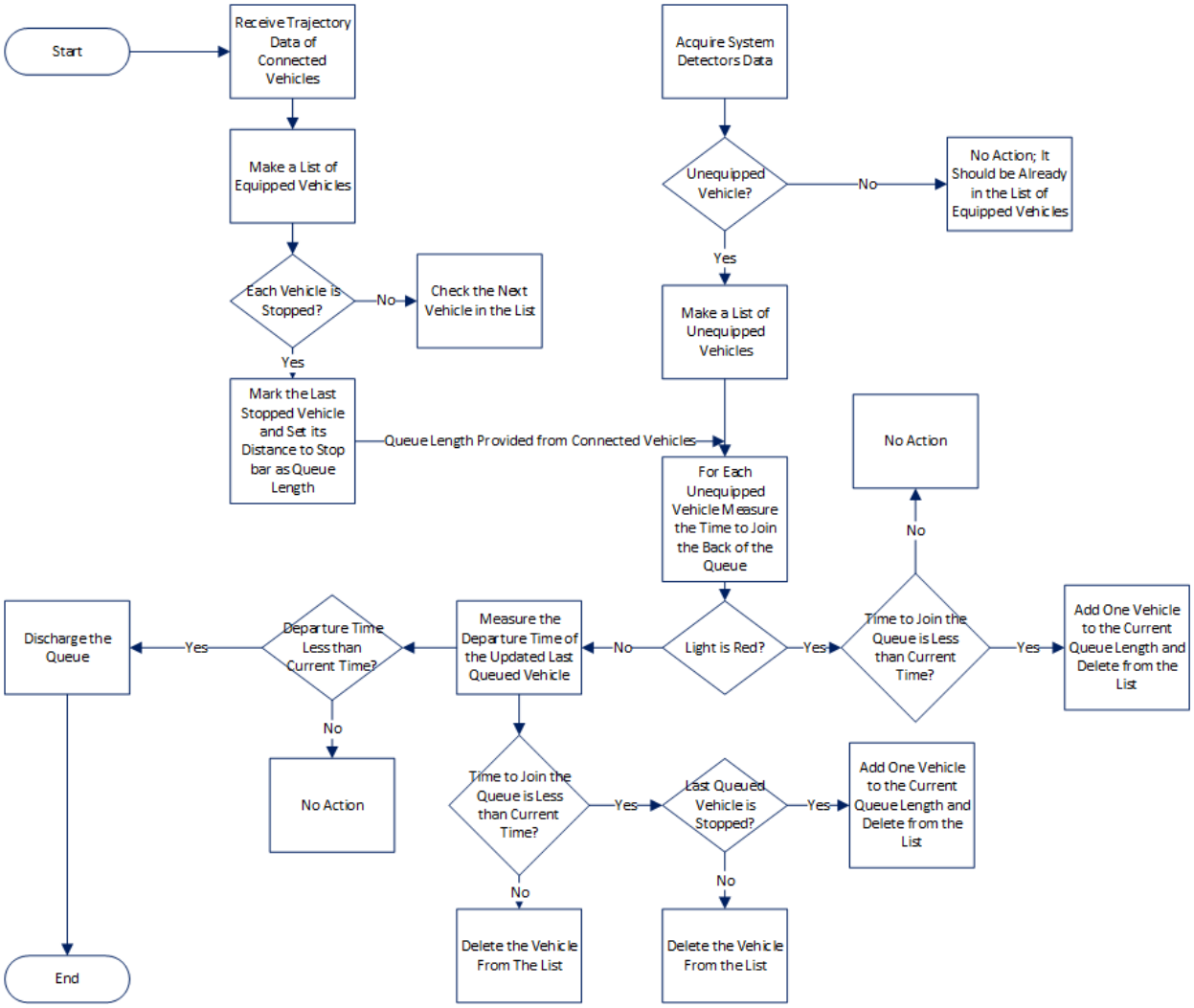


Figure 40 Queue Length Estimation Flow Chart using Detectors and BSMs Data

Performance observation categorizes performance by a combination of multiple modes and under various scenarios (Peak Period, All Day, Volume Level, etc.). It captures performance of vehicles by traffic movement (e.g. phases) at the intersection. Visualization of the complex performance characteristics can be challenging. Radar diagrams are utilized to visualize performance measures in an intuitive way. Radar diagrams are widely used in the field of organizational development to demonstrate multivariate data in the form of a two-dimensional diagram. In the context of traffic control, this is a tool that can help monitoring the performance measures at the intersections level, section level and system level in the format of a dashboard. This tool is used concurrently with other techniques for analyzing the strengths and weaknesses of the existing system. Not only is this tool useful for presenting the results to the stakeholders, but it is also useful for comparing before and after studies. Several examples of these diagrams are shown in the next section and detailed explanation of radar diagrams can be found in reference *(2)*.

The additional performance metrics that represent the connected vehicle operations include the number of equipped vehicles traversing a section and Market Penetration Rate (MPR). It is measured based on the data from System Detectors and received BSMs from OBEs.

By having an accurate vehicle counts from system detectors and information about the number of connected vehicles at the intersection during a predefined time period, the market penetration rate is computed as follows:

$$Market\ Penetration\ Rate\ (\%) = \frac{Number\ of\ Connected\ Vehicles}{Total\ number\ of\ vehicles} \times 100 \quad (11)$$

This metric is calculated for each approach, and then by computing a volume weighted sum, the penetration rate for each intersection is estimated. Given the market penetration rate of each individual approach, the average MPR value is calculated based on equation (12) where “n” is the total number of approaches available at intersection:

(12)

$$Average\ Intersection\ MPR\ (\%) = \sum_{k=1}^{n} \frac{(MPR)_k}{n}$$

5.3.9.5 Example Application

As an illustration, according to the data originates from the MRP_EquippedVehicleTrajectoryAware component, trajectories of the vehicles is plotted in Figure 41 and shown how to derive the metrics for an individual vehicle from a time-space diagram.
The performance measures related to travel time and delay are calculated and estimated based on the equations 1-8 as discussed above.

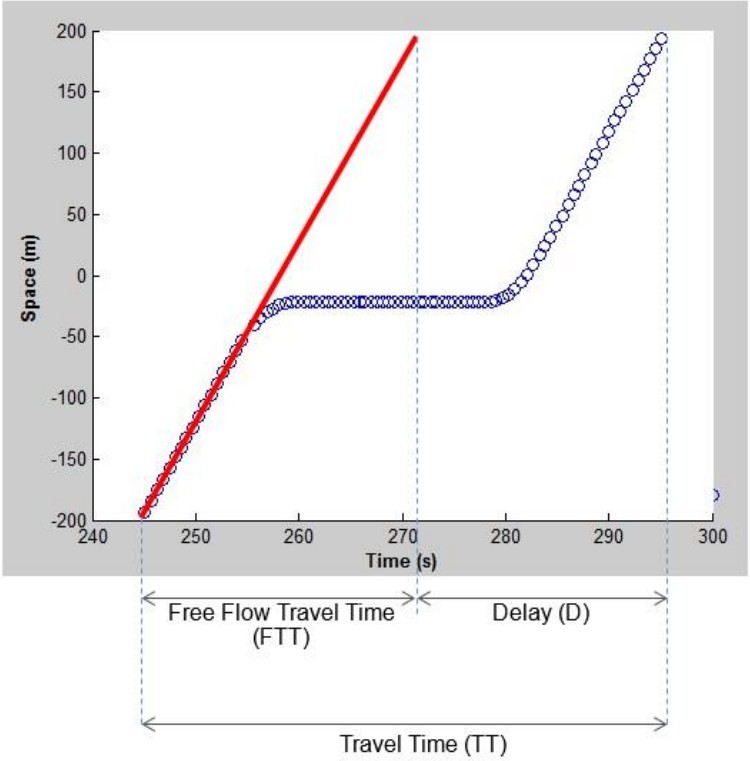


Figure 41 Travel Time and Delay Calculation on a Trajectory Sample

According to the functional description section, the current queue length during each cycle could be observed using the algorithm developed. In this methodology, the last stopped vehicle is critical in the definition of the back of the queue, and in the case of 100% penetration rate by focusing on this specific car during each cycle, the queue length distance is accurately observed. Figure 42 shows how the queue length increased and discharged during each cycle in a lane for phase 2.

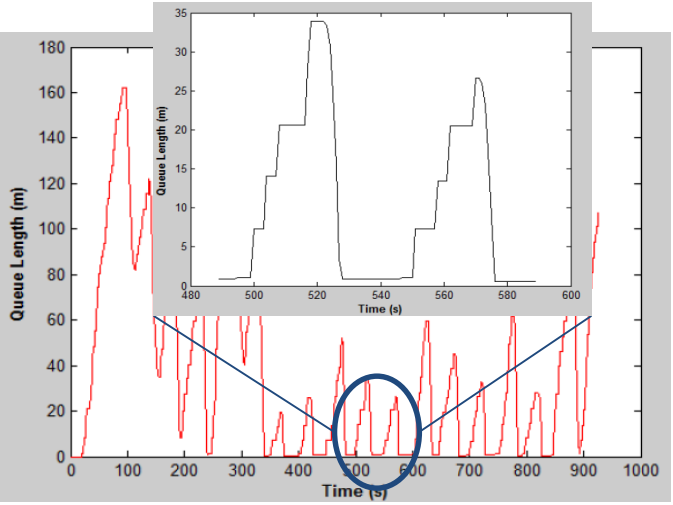
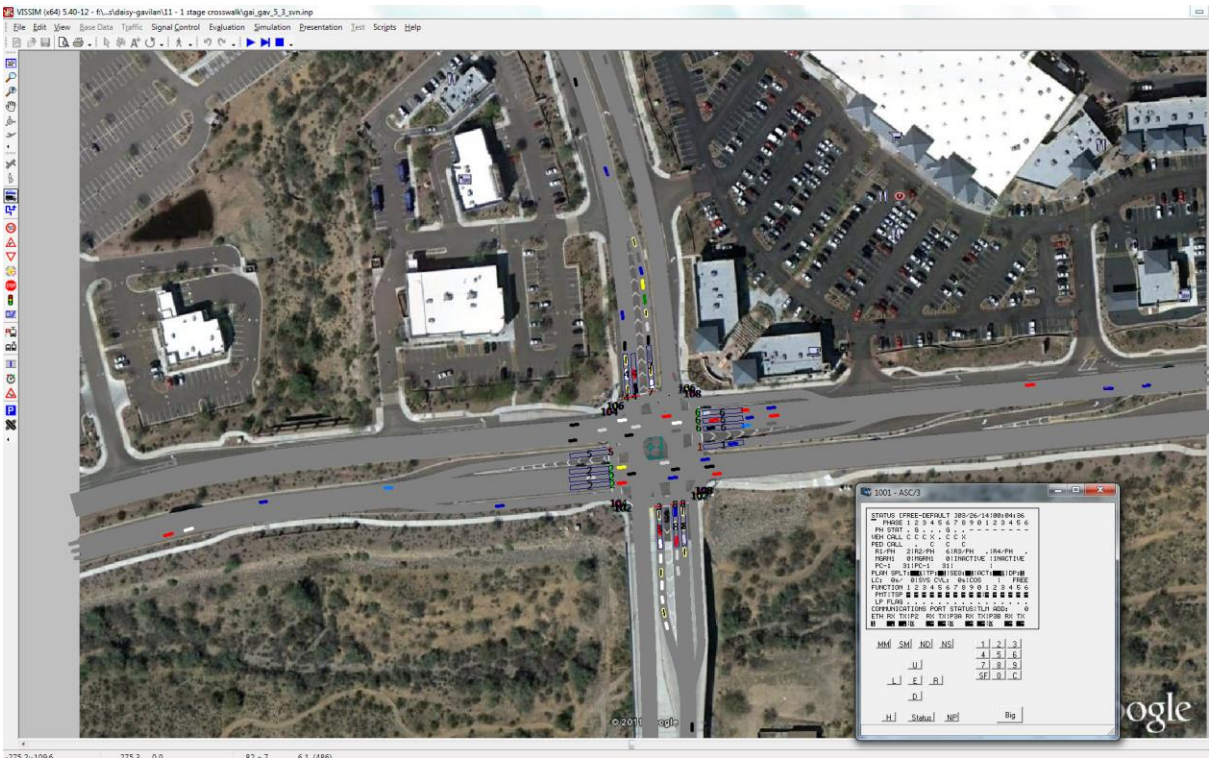


Figure 42 Queue Profile of a lane for Phase 2

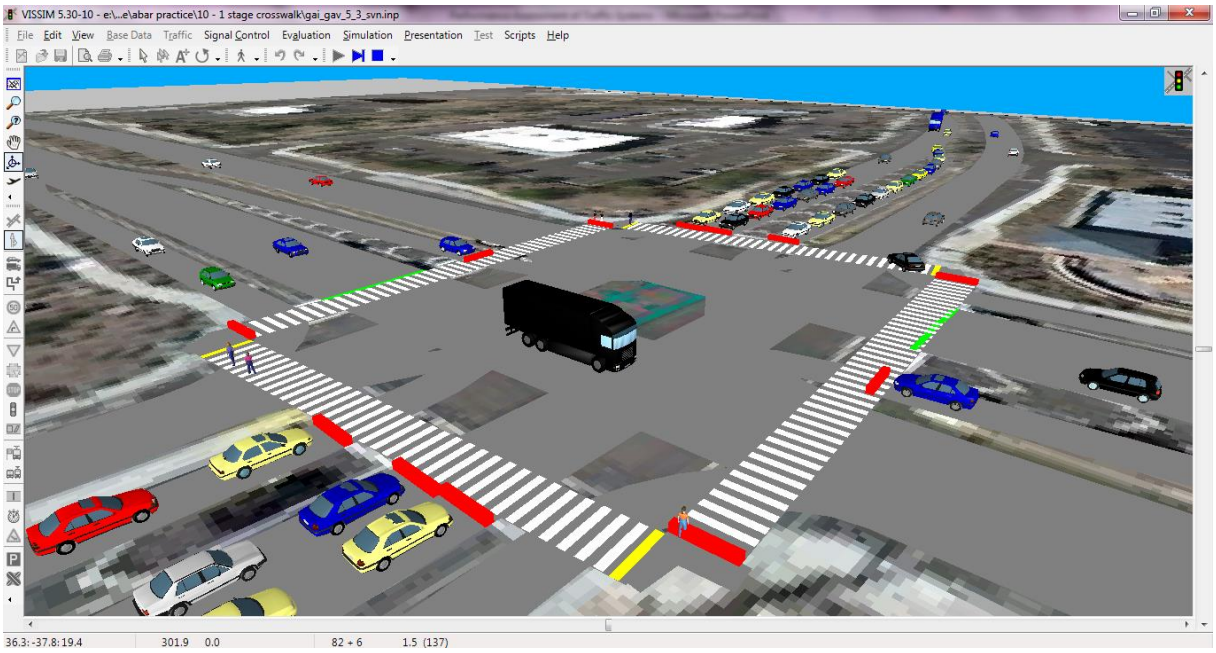
5.3.8.6 Unit Test Description (debugging and testing)

All the individual units of the source code are written in C++ for this component were tested separately to ensure that they perform as designed. Even the smallest testable part of a class was verified through the debugging processes, and the output of each function was compared with the simulation results of a known data set to validate the accuracy of the code.

As an example, travel time, delay, and queue length calculations and estimations were made and compared with the VISSIM output files estimating the same performance measures. The results showed less than a 5% difference in the final values. Figure 43 shows a screenshots of the simulation model which ran for one hour with an extra 15 minutes warm up period for Daisy Mountain and Gavilan Peak. 10 different random seeds were used for replications in order to make allowances for the stochastic variations.



(a)



(b)

Figure 43 Screenshots of Simulation Models ran in VISSIM 5.3 in 2D (a) and 3D (b) with multiple modes

Figure 44 is a time-space diagram based on a VISSIM simulation showing the movements of 96 vehicles approaching Daisy Mountain and Gavilan Peak during phase 2 over a 1000 second time horizon.

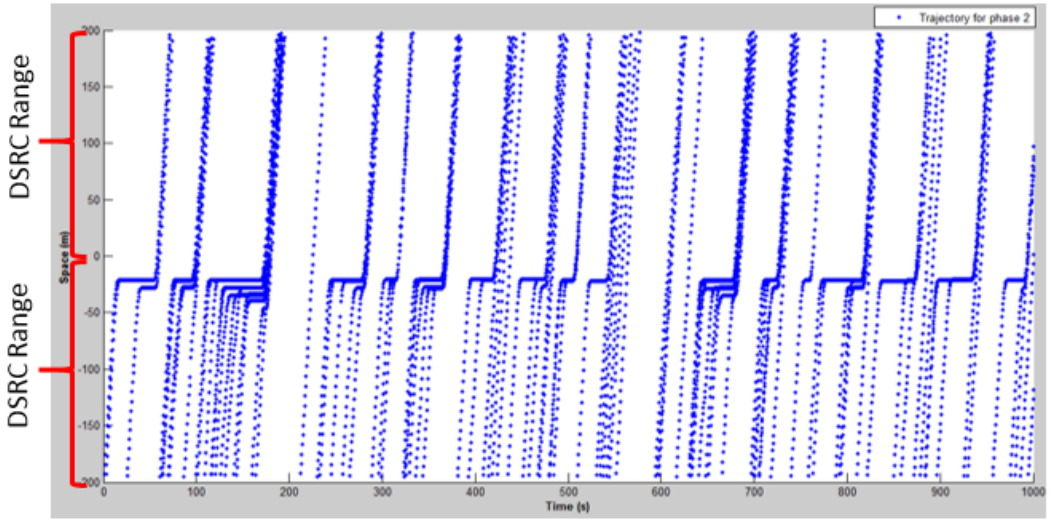


Figure 44 Trajectories of All the vehicles in the range of DSRC radio channel.

In the real world, 100% market penetration rate is unrealistic, so by lowering the rate, a random sample of the above population could be a representative of the connected vehicles out in the street. Figure 45 shows the trajectories of a 20% random sample of the vehicles demonstrated in a time-space diagram over the same period of time for the same phase.

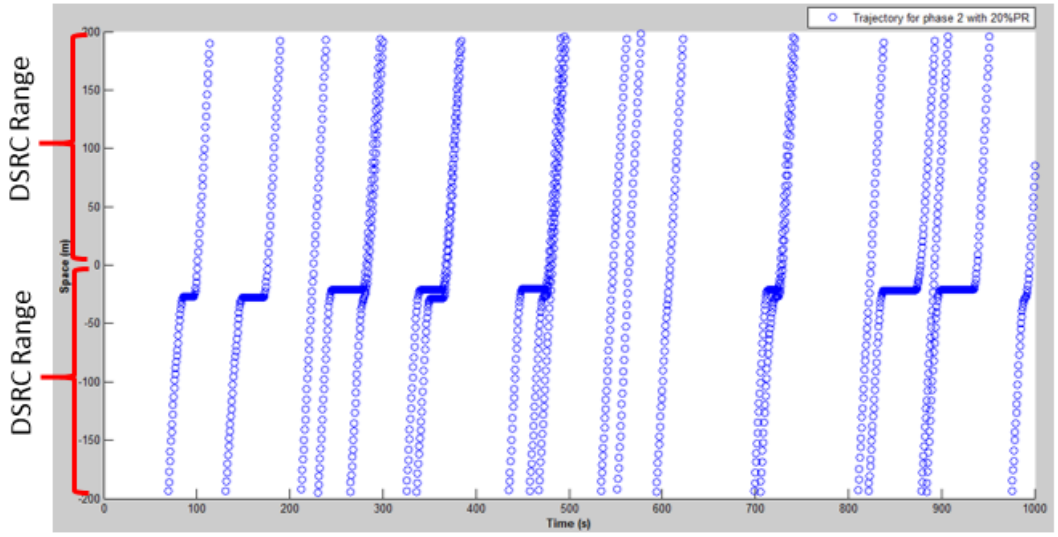
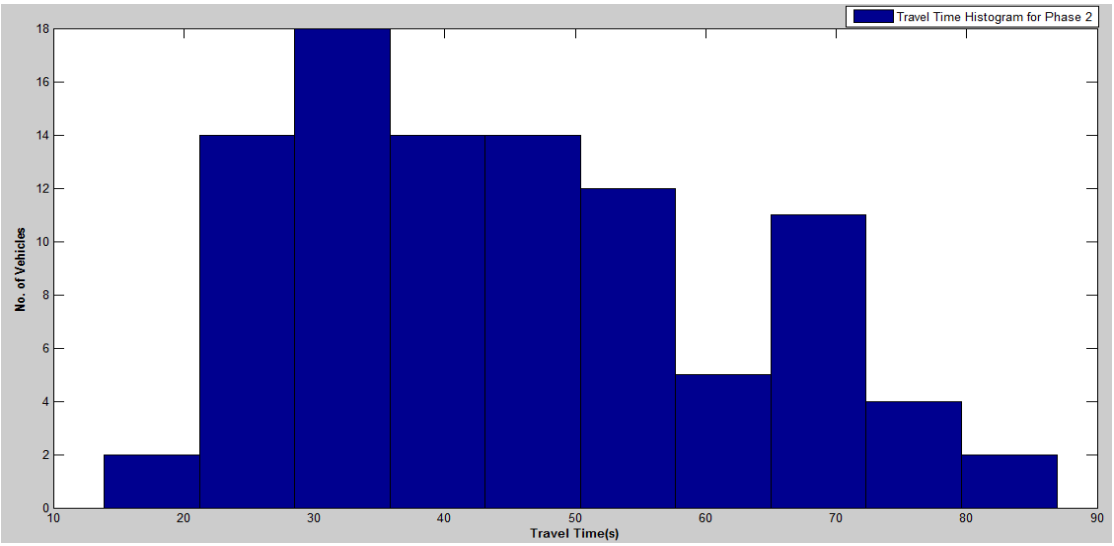


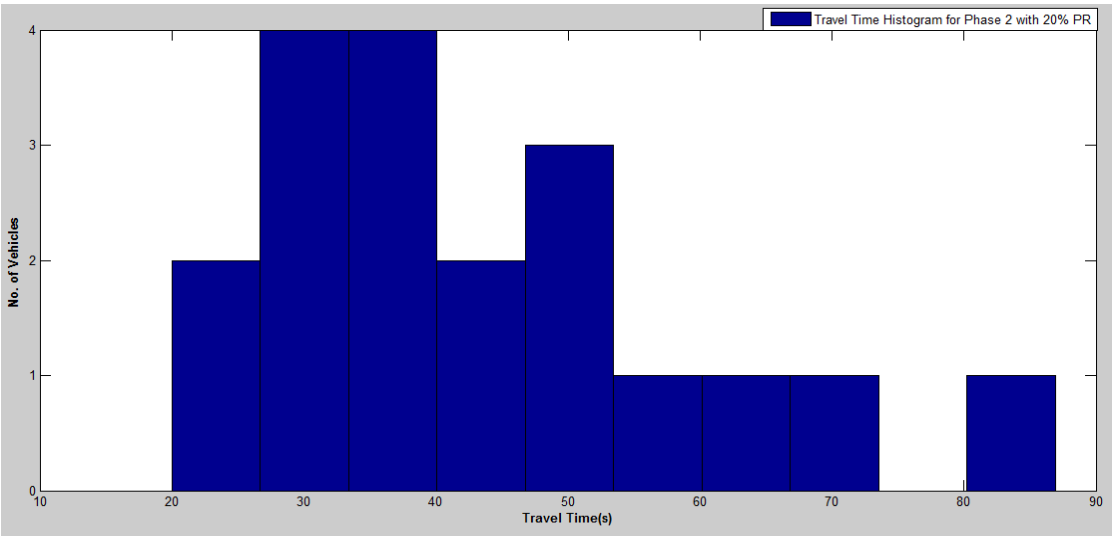
Figure 45 Trajectories of 20% of the vehicles in the range of DSRC radio channel

Travel time and delay for each individual vehicle are calculated and the average, total, and the variability of these measures are computed. A histogram to show how the travel times of the vehicles traversing the intersection are distributed over the defined time period is shown in Figure 46. This could be done for both the population and sample

based on the information from the trajectories. Figure 46 (a) and (b) are the travel time histograms of the whole population and the randomly selected sample.



(a)



(b)

Figure 46 Travel Time Histogram of the Population (a) and Travel Time Histogram of the Sample (b)

The performance measures resulting from the figures above, are all shown in Table 14. It's worth noting that the total number of vehicles in the simulation run of Daisy Mountain and Gavilan Peak intersection during a 1000 second time horizon was 96, and the randomly selected sample contained only 19 vehicles.

Table 14 Travel Time and Delay Performance Metrics

	Population	Sample (20% Penetration Rate)
Average Travel Time (s)	45.39	43.10
Average Delay (s)	19.62	17.33
Standard Deviation (s)	16.6	17.23
Total Travel Time (s)	4357.44	818.9
Total Delay (s)	1883.52	329.27

5.3.9.6 References

1. J2735: Dedicated Short Range Communications (DSRC) Message Set Dictionary - SAE International. http://standards.sae.org/j2735_200911/. Accessed Jul. 29, 2014.
2. Khoshmagham, S., L. Head, and F. Saleem. Performance Assessment of Multi-Modal Traffic System Using Micro-Simulation Methods. Presented at Transportation Research Board Annual Meeting, Washington D.C. 2014.

5.4 MMITSS Central System Components

5.4.1 MMITSSUserInterface

5.4.1.1 Overview of Functionality

The MMITSSUserInterface component is responsible for providing a user interface for the system level MMITSS component.

5.4.1.2 Requirements

Node: MMITSS System	**Component Name:** MMITSSUserInterface
	Traceability: (Overall MMITSS need)
Description of Responsibility: The MMITSSUserInterface component is responsible for providing a user interface on the system level MMITSS component.	
Supporting Text: *The user interface is the point where users/operators can access the PerformanceObserver Visualizations, System_ConfigurationManager, and System_N_LevelPrioriityConfigurationManager. The user interface can provide information about the status of the MMITSS system as well as general system information.*	

5.4.1.3 Interfaces

Required (input)

Desired Performance Metrics, Modes Weight, Coordination Configuration, N-Level Priority Hierarchy.

Provided (output)
Performance Measure Reports, Modes Weight, Coordination Configuration, N-Level Priority Hierarchy.

5.4.1.4 Functional Specification/Description

The MMITSSUserInteface is the point where user/operator can view system status (e.g. on-line, off-line), access the System_ConfigurationManager, System_N_LevelPriorityConfigurationManager and visualization of performance measures at the intersection, and section levels. The goal of this user interface is to make the traffic manager's interaction with the MMITSS system as simple and efficient as possible in terms of getting and accomplishing operator traffic preferences (such as establishing the modes that will receive priority at the intersection level or section level, allowable phase sequences, dilemma zone protection, etc.). The user interface can provide information about the status of the MMITSS system as well as general system information.

The MMITSSUserInteface is a web-based application. The web server restricts access to website content to only those traffic operators who have permission to access control information. The status and performance web pages get data from these components as well as from a database.

Figure 47 illustrates the architecture of the MMITSS Central user interface. The MMITSSUserInterface supports both the Arizona test bed and the California test bed. In addition, there are pages providing MMITSS project information including the MMITSS teams members who are working on the project.

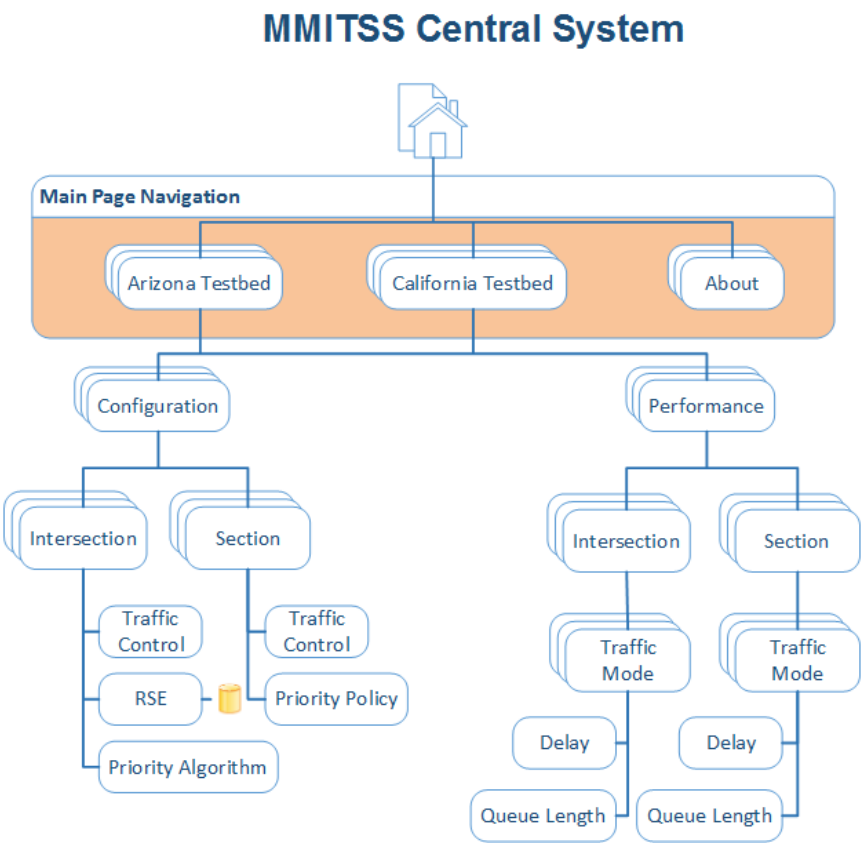


Figure 47 Web Application Map of MMITSS Central System

Figure 48 shows the system structure of Performance Observer component. The right side of the structure depicts the Linux or Windows server which gets the performance metrics from the application, stores them into database, and visualize them in a dynamic web application (WAMP/LAMP: Windows/Linux Apache MySQL PHP).

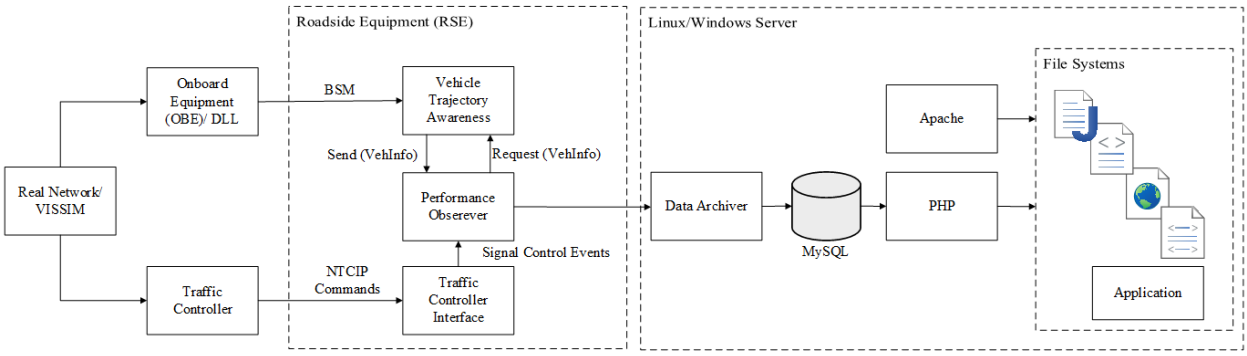


Figure 48 Performance Observer System Structure

The user can also configure the N-Level Priority Hierarchy for different modes and class of vehicles. Emergency Vehicles, Trucks, Transits and non-motorized vehicles are considered as different modes. Within each mode, there are different classes, such as BRT, express, and local transit classes, that the user can rank according to the desired level of priority. For example, within emergency vehicles, fire trucks may have the highest rank, ambulances are ranked second, and police are in ranked third.

5.4.1.5 Example Applications

Figure 49 shows the sequence diagram of interactions between the MMITSSUserInterface and the System_ConfigurationManager, System_N_LevelPriorityConfigurationManager, and Section_PerformanceObserver.

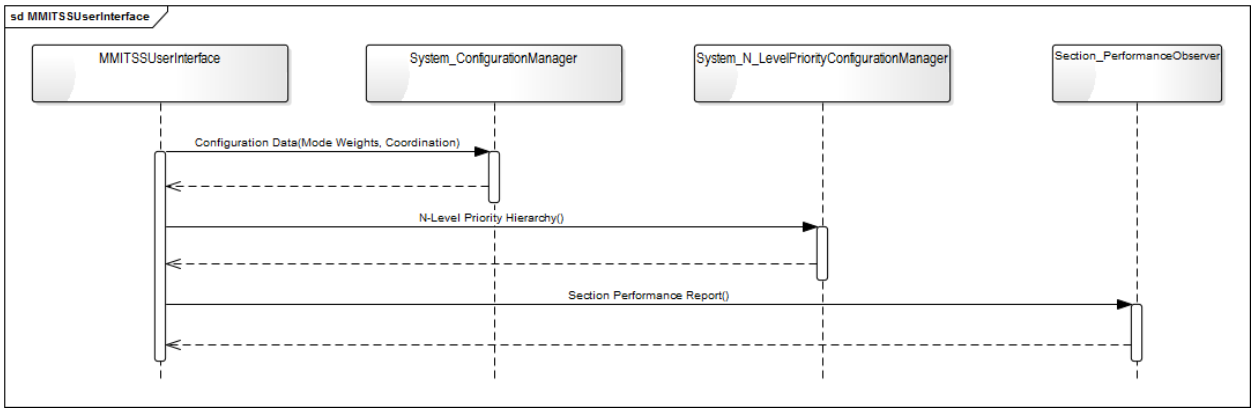


Figure 49 Sequence Diagram of Interactions between MMITSS User Interface, System_Configuration Manager, System_N_LevelPriorityConfigurationManager, and Section_PerformacneObserver

To visualize all the performance measures observed at intersection level radar diagrams are deployed. Figure 50 is a sample radar diagram. On each axis of this diagram one of the twelve possible movements at an intersection is shown and the selected performance measure is average travel time in seconds for passenger vehicles on each approach. For example, passenger vehicles on the Northbound Right Turn approach spend the least amount of time (17 seconds), and cars on Eastbound Left Turn movement have the most travel time (78 seconds).

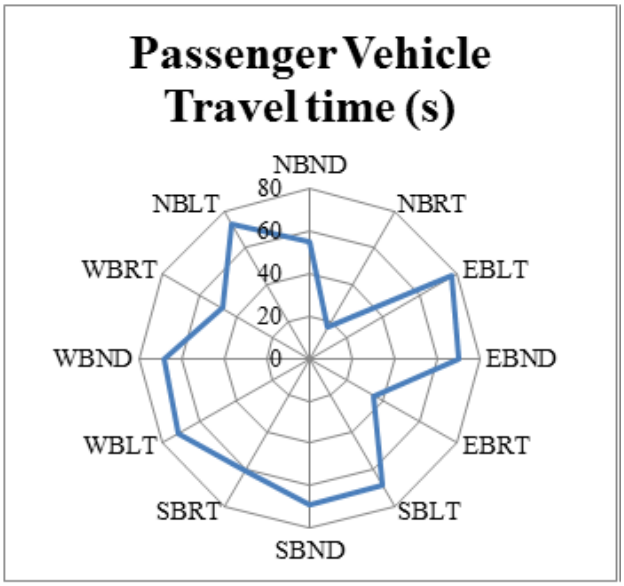


Figure 50 Sample key to radar diagrams used to describe traffic performance

When considering the complex interactions of the different transportation modes, there is a need for a tool showing all the modes at the same time. The dashboard provided here is able to show different measures of various modes under specific scenarios. Figure 51 below is an example of the dashboard resulting from the simulation model of the mentioned intersection at two different operational hours (peak vs. off-peak). Vehicle and pedestrian demand are assumed to be 825 vehicles per hour per lane and 350 pedestrian per hour per crosswalk during the peak period and to be 325 vehicles per hour per lane and 180 pedestrian per hour per crosswalk during the off-peak period. This dashboard consists of 4 radar diagrams, one for each mode concentrating on the following measures: travel time for passenger vehicles, delay for transit, number of stops for trucks, and delay for pedestrians. The pie chart in the middle depicts the distribution of modes in the system.

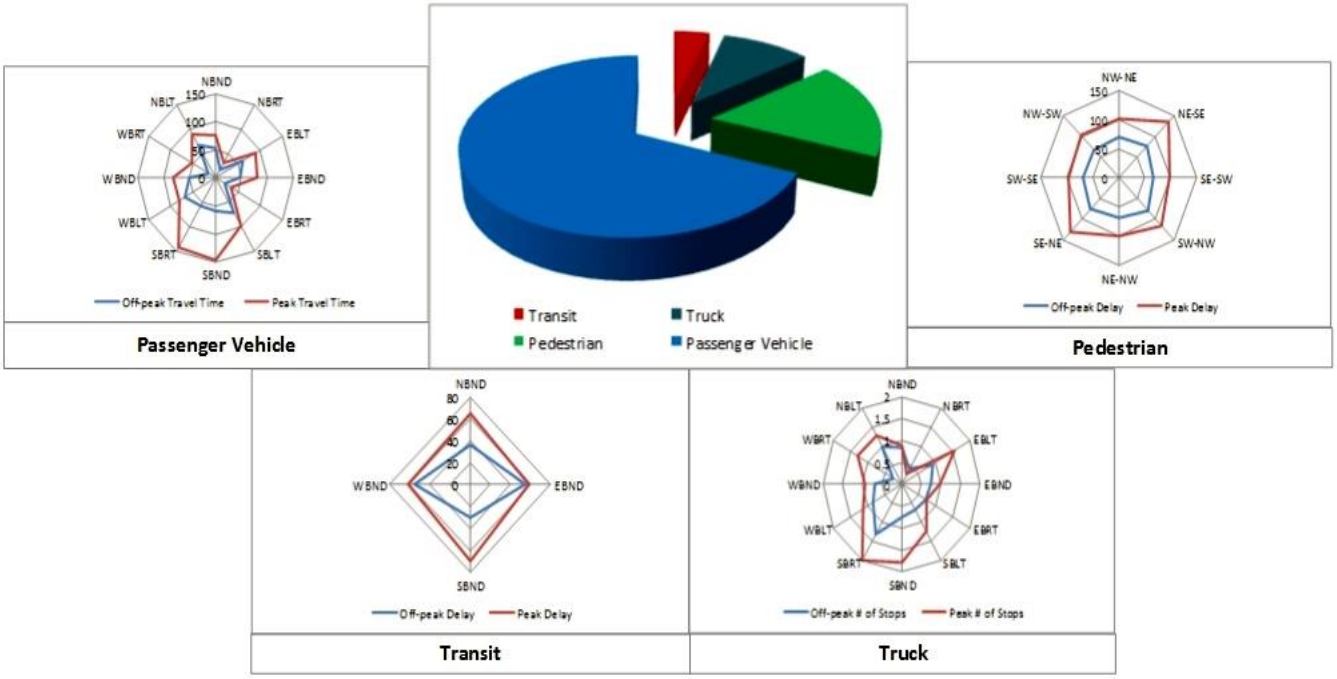
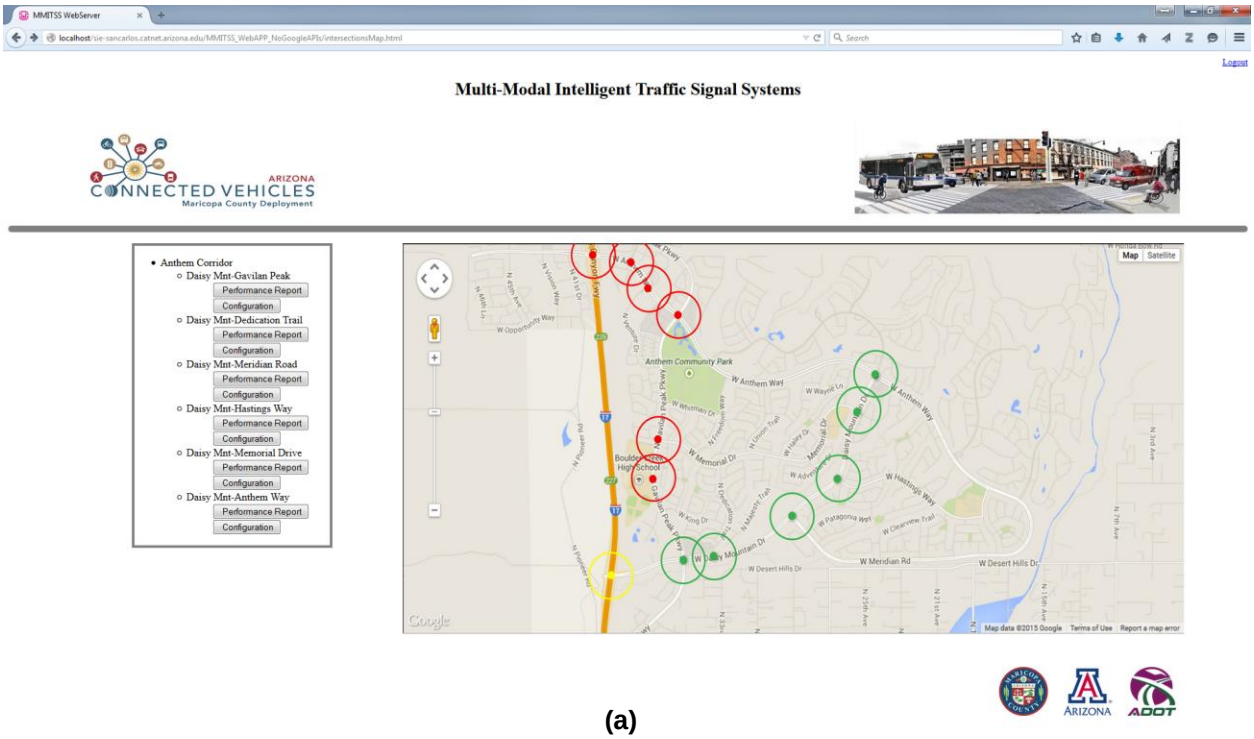
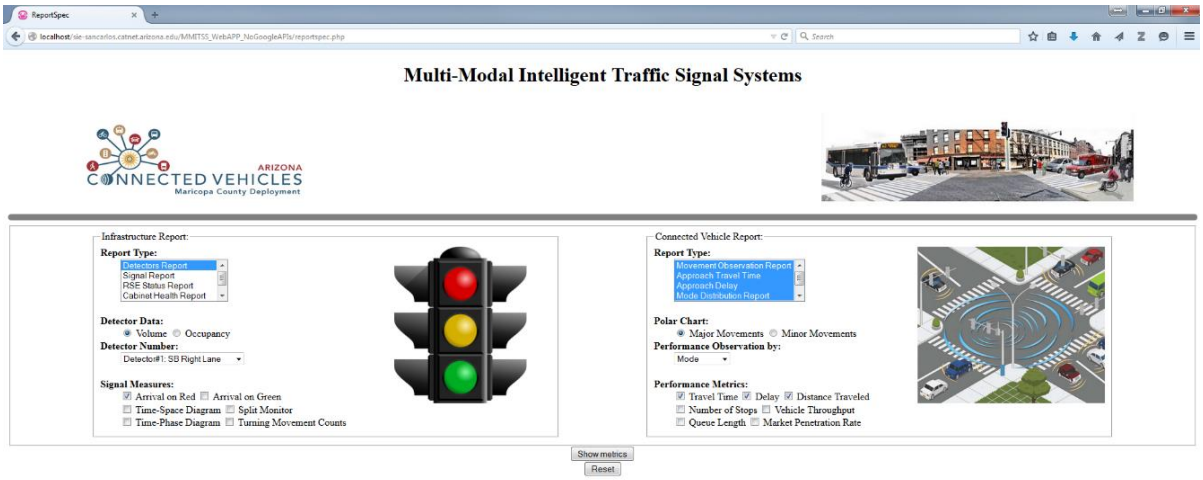


Figure 51 Dashboard consisting of four radar diagrams and a pie chart for comparing Peak and Off-peak periods

Figure 52 shows three screenshots of the web application that is used to visualize the outputs of MRP_PerformanceObserver component. As stated previously in the system structure, this web app runs on a Windows or Linux server in the same local network. On the first page (a), the end user can choose between the configuration page or the performance report page for active intersections. By selecting the report page (b), the type of metrics and reports to view can be selected. The report page (c) shows the sample of the types of graphs and visualizations available.


MMITSS WebServer
Logout
Multi-Modal Intelligent Traffic Signal Systems
ARIZONA
CONNECTED VEHICLES
Maricopa County Deployment
• Anthem Corridor
Daisy Mnt-Gavilan Peak
Performance Report
Configuration
Daisy Mnt-Dedication Trail
Performance Report
Configuration
Daisy Mnt-Meridian Road
Performance Report
Configuration
Daisy Mnt-Hastings Way
Performance Report
Configuration
Daisy Mnt-Memorial Drive
Performance Report
Configuration
Daisy Mnt-Anthem Way
Performance Report
Configuration
Map
Satellite
Anthem Community Park
W Anthem Way
W Memorial Dr
W Hastings Way
W Meridian Rd
W Desert Hills Dr
Boulder Creek High School
Google
Map data ©2015 Google
Terms of Use
Report a map error
ARIZONA
ADOT

(a)


ReportSpec
Multi-Modal Intelligent Traffic Signal Systems
ARIZONA
CONNECTED VEHICLES
Maricopa County Deployment
Infrastructure Report:
Report Type:
Detectors Report
Signal Report
RSE Status Report
Cabinet Health Report
Detector Data:
Volume
Occupancy
Detector Number:
Detector#1: SB Right Lane
Signal Measures:
Arrival on Red
Arrival on Green
Time-Space Diagram
Split Monitor
Time-Phase Diagram
Turning Movement Counts
Connected Vehicle Report:
Report Type:
Movement Observation Report
Approach Travel Time
Approach Delay
Mode Distribution Report
Polar Chart:
Major Movements
Minor Movements
Performance Observation by:
Mode
Performance Metrics:
Travel Time
Delay
Distance Traveled
Number of Stops
Vehicle Throughput
Queue Length
Market Penetration Rate
Show metrics
Reset

(b)

(c)

Figure 52 User Interface_PerformanceObserver Web Application

5.4.2 System_ConfigurationManager

Node: MMITSS System	**Component Name:** System_ConfigurationManager
	Traceability: A4101, A4102, 13.3.1; §11.1.2, §11.1.3
Description of Responsibility: The System_ConfigurationManager is responsible for allowing operators to add intersections, create and name sections, assign intersections to sections, and access other critical configuration components including the System_N_LevelPriorityConfigurationManager.	
Supporting Text: *The configuration manager allows operators the ability to create, change, and delete sections and intersections in the system. This functionality can be accomplished using configuration files, but a simple user interface to view the configuration will support the demonstration of the system.*	

In the Configuration page, the MMITSSUserInterface provides options for both section and intersection level traffic control. At section level, the user can specify whether the section is to be operated as a coordinated section of signals. If so, information related to coordination, such as the coordinated intersection names, the desired cycle length, the

coordinated phase, the offset for each intersection, and phase splits and coordination weight should be provided. Figure 53 shows the configuration page of Dedication – Daisy Mountain Dr. intersection.

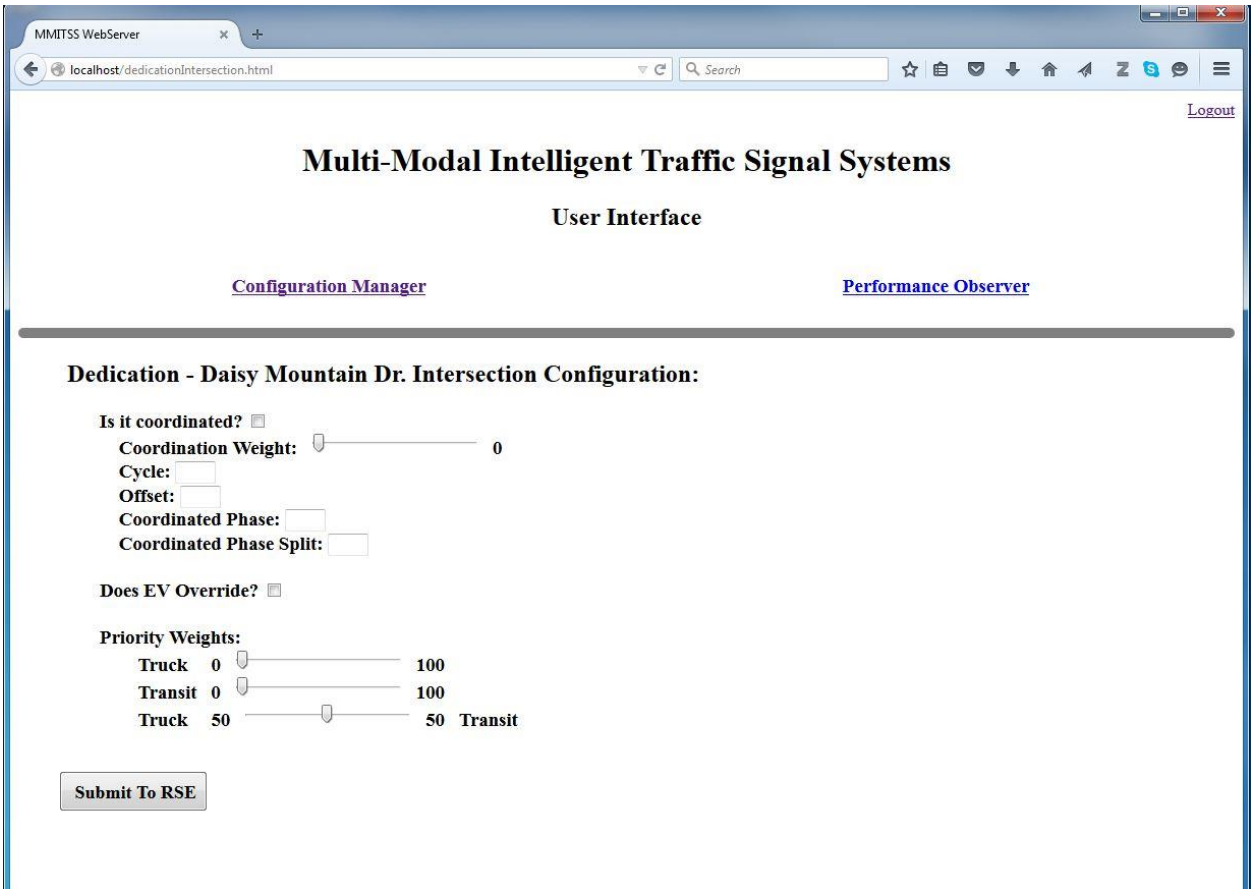


Figure 53 MMITSS Configuration Manager Web Page for Dedication and Daisy Mountain.

5.4.3 System_N_LevelPriorityConfigurationManager

5.4.3.1 Overview of Functionality

In each traffic control section, the decision makers define a preference for granting priority to different classes of vehicles. For example BRT transit is more important that Express transit and all transit is more important than trucks. In another section trucks are more important than transit. The System_N_LevelPriorityConfigurationManager is the mechanism where an operator can configure the policy. Given a policy setting, the

configuration manager will set the associated parameters on the MRP and within section level priority request servers.

5.4.3.2 Requirements

5.4.3.2.1

Node: MMITSS System	**Component** **Name:** System_N_LevelPriorityConfigurationManager
	Traceability: A8001, A8002, C8002.402, C8002.503, 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; §11.0, §11.1.1, §11.2, §11.3, §11.4, §11.5
Description of Responsibility: This component is responsible for allowing an operator to set the policy preferences for the N-Level Priority policy.	
Supporting Text: *In each traffic control section, the decision makers define a preference for granting priority to different classes of vehicles. For example BRT transit is more important that Express transit and all transit is more important than trucks. In another section trucks are more important than transit. The System_N_LevelPriorityConfigurationManager is the mechanism where an operator can configure the policy. Given a policy setting, the configuration manager will set the associated parameters on the MRP and within section level priority request servers.*	

5.4.3.3 Interfaces

Figure 54 shows the interface of the System_N_LevelPriorityConfigurationManager component.

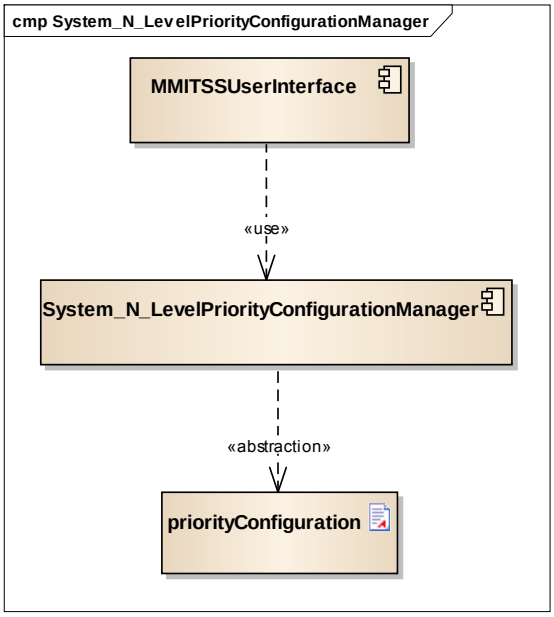


Figure 54 Interfaces of the System_N_LevePrioirityConfigurationManager

5.4.3.3.1 Provided (output)

5.4.3.3.2 Send Priority Policy

The System_N_LevelPriorityConfiguratuionManager is responsible for providing a data structure for the MRP_Priority_Solver and the MRP_PRS_PriorityRequestServer. The data structure contains weights for transit and truck. If the coordination is considered as a form of priority, the weight of coordination as well as the cycle, split, offset, and coordinated phases are determined in the data structure.

5.4.3.4 Functional Specification/Description

The System_N_LevelPriorityConfiguratuionManager uses MMITSSUserInterface to receive the priority configuration. The user may define the configuration for each intersection. Error! Reference source not found. shows the priority configuration webpage for Dedication and Daisy Mountain Dr. intersection. The user can set the priority policy by changing the weights. There are slider bars for setting the transit, truck, and coordination weight. By clicking the “Submit To RSE” button, the priority policy data will be transferred to the corresponding RSE (in this example Dedication-Daisy Mountain Dr.) and will be stored in a file. The MRP_Priority_Solver and MRP_PRS_PriorityRequestServer will read the file to implement the priority policy.

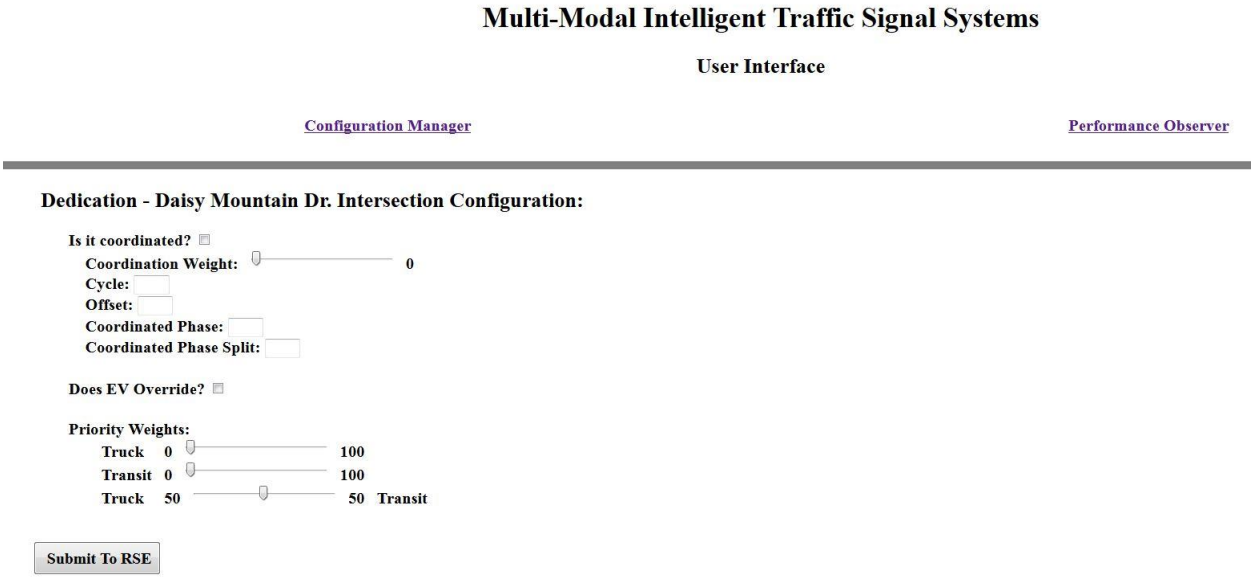


Figure 55 Example of Setting the Priority Policy System_N_LevePrioirityConfigurationManager

In “priorityConfiguration.txt” file, coordination plan such as cycle split and offset is also considered. (Note: The coordination plan should have obtained from Traffic Signal Control through TCConfig. But in the current system, it is read from a file)

An example of priorityConfiguration.txt is shown as follow:

```
coordination 1
coordinated_phase 2 6
cycle 100
offset 0
split 30
transit_weight 1
truck_weight 1
```

5.4.4 Section_PriorityRequestServer

5.4.4.1 Overview of Functionality

The Section_PrioirityRequestServer is responsible for the coordination of priority requests at a section level. [Implementation Note: This section was not implemented].

5.4.4.2 Requirements

Node: MMITSS System	**Component Name:** Section_PriorityRequestServer
	Traceability: C3001.202, C3001.403, C3001.504, C4003.201, C4003.402, A4004, C4004.201, 13.3.2; 13.3.4; 13.3.5; §11.0, §11.0.1, §11.0.2, §11.2, §11.2.3, §11.4, §11.4.2, §11.5
Description of Responsibility: This component is responsible for implementing section level priority control based on requests for priority that are acquired from the intersections.	
Supporting Text: *Section level priority strategy may include coordination of signals to provide priority for emergency vehicles, or transit/trucks. Typically these strategies include coordination changes (e.g. offsets), queue clearance, or creation of green bands. The N-level priority policy is used to determine which modes/classes of vehicles are given preference over other mode/classes of vehicles.*	

5.4.5 Section_Coordinator

5.4.5.1 Overview of Functionality

The Section_Coordinator provides priority requests for coordination. If the intersection is configured as part of a section of coordinated intersections, the phase splits, cycle length, and offset would be mapped to a Coordination Request Message (CoordRM). The Section_Coordinator receives trajectory of equipped vehicles from all of the coordinated intersections within a section. It uses the trajectories to determine of the offset needs to be adjusted to improve progression. [Implementation Note: The Section_Coordinator was not implemented. Coordination can use the controller coordination parameters to generate coordination priority requests.]

5.4.5.2 Requirements

Node: MMITSS System	**Component Name:** Section_Coordinator
	Traceability: C3003.001, C3003.002, C3003.003, C3003.004, C3003.005, C3003.006, A3004, C3005.001, C3005.002, A3006, A3101, C3101.001, C3102.001, C3102.002, A3103, 11.1.2; 13.3.1, 13.3.2, 13.3.3, 13.3.4, 13.3.5; §4.1.1, §4.2, §5, §11.0, §11.1.2, §11.4.2, §11.5.2

Description of Responsibility: This component is responsible for the identification of platoons, estimation of the platoon arrival time at intersections in the section, setting offsets and coordinated phase splits.
Supporting Text: *The Section_Coordinator provides adjustments to coordination timing that considers the movement/progression of platoons within the priority framework. Trade-offs between priority for different modal vehicles and coordination is part of the N-level priority policy.*

5.4.5.3 Interfaces

Figure 56 shows the interface of the Section_Coordinator.

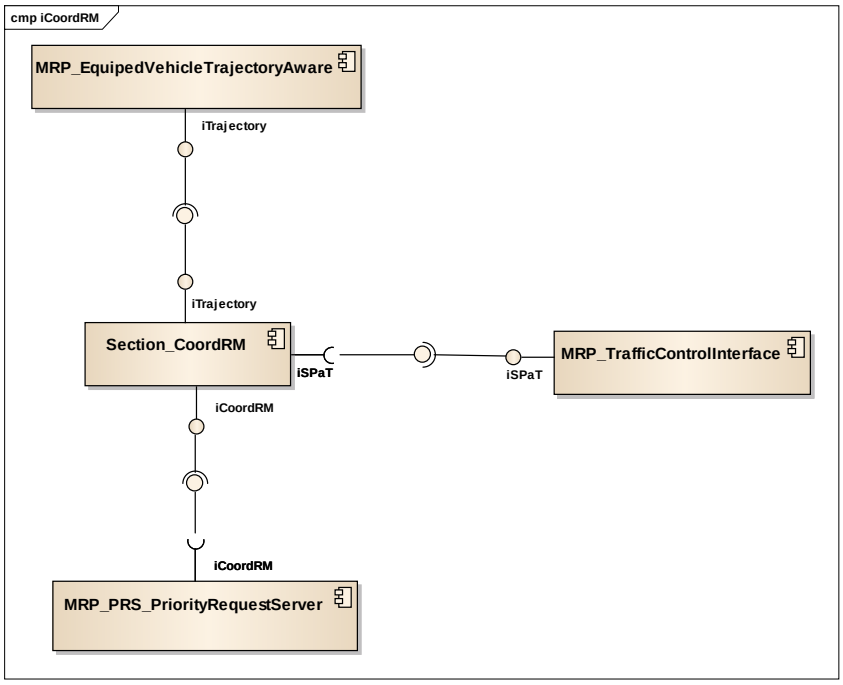


Figure 56 Interfaces of the Section_Coordinator

Required (input): Vehicle Trajectory, SPaT

5.4.5.3.1.1 Receive SPaT

The Section_Coordinator gets the Signal Timing Plan messages form the MRP_TrafficControlInterface for all coordinated intersections.

5.4.5.3.1.2 Receive Vehicle Trajectory

This function receives the tracked vehicle list data from MRP_EquipedVehicleTrajectoryAware component of all coordinated intersections. (See MRP_EquipedVehicleTrajectoryAware for definition)

Provided (output)

The iCoordRM interface generates coordination priority request (CoordRM) messages that contain the desired start and end times of the coordinated phase based on the coordination plan (cycle, offset, and splits).

5.4.5.3.1.3 Send CoordRM

This function send a Coordination Request Message (CoordRM) data structure to MRP_PRS_PriorityRequestServer. CoordRM has the same data structure as SRM.

Supporting Code:

```
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <time.h>

int ReadCoordRM (char * CoordRM_File_Name)
{
	SRM CoordRM;
	CoordRM.ReadfromFile(CoordRM_File_Name);
	sprintf(temp_log,"Read the CoordRM successfully At (%d).\n",time(NULL));
	return 1;
}
```

5.4.5.4 Functional Specification/Description

Coordination requests are priority requests intent on creating a “green wave” for passenger vehicles. Given arrival table of the vehicles and a pre-calculated optimized signal coordination plan, the “arrival times” of coordination requests are generated to request coordination phases for each cycle. If coordination is considered in PriorityPolicy, CoordRM and coordination weight are included every time the formulation is solved.

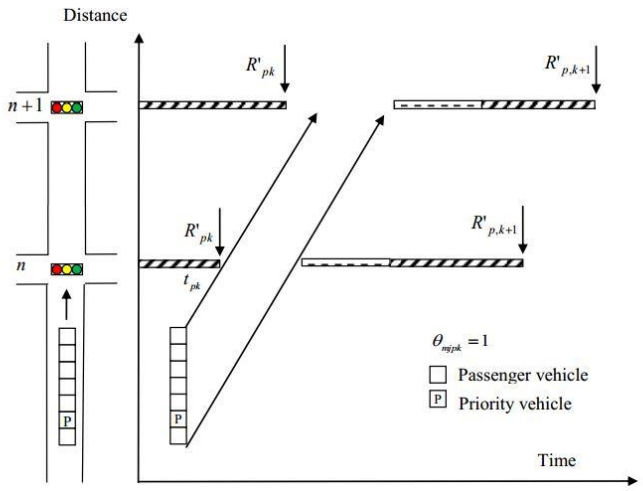


Figure 57 Illustration of CoordRM

5.4.6 Section_PerformanceObserver

5.4.6.1 Overview of Functionality

The Section_PerformanceObserver is responsible for acquiring data from MRP_PerformanceObserver component of each intersection and estimating and monitoring of the selected performance measures at section level.

5.4.6.2 Requirements

Node: MMITSS System	**Component Name:** Section_PerformanceObserver
	Traceability: C3002.001, C3002.002, C3002.003, C3002.004, C3002.005, C3002.006, C3002.007, C3002.008, C8101.102, §11, 11.1.2 (13.3.1), §12.7"
Description of Responsibility: This component is responsible for acquiring intersection performance estimates and combining them with section level data to estimate section level performance measures.	
Supporting Text: *Section level performance measures utilize intersection level measures together with section leve information (e.g. platoon size, head, tail, stops, travel time, etc.) to provide estimates of section level performance.*	

Functional Requirements

The functional requirements are concerned with the acquisition of data by the section from the intersection and infrequently from equipped and unequipped vehicles and travelers within the boundaries or neighborhood of the section. This includes the

geometric intersection description (GID) of the intersections comprising the section. The section data acquisition requirements are concerned with the function and specificity of acquiring data (what, where, when, from whom, and sometimes how).

The acquisition of performance measures and metrics is as follows:

Acquire Intersection Performance Measure Data: A MMITSS Section shall acquire performance measures from intersections. Performance measures collected and estimated at each intersection can be collected/acquired by the corresponding section. Performance measures at intersections will be used to determine the performance in sections.

- **Queue Length and Queue Length Variability:** A MMITSS Section shall acquire the performance measure of estimated queue length and its variability for each intersection. Estimated queue length will be used to determine the congestion level at intersections in a section.
- **Delay and Delay Variability:** A MMITSS Section shall acquire the performance measure of delay and its variability from each intersection. Delay at intersections will be used in determining signal priority timing that balances impacts on traffic in a section.

- **Throughput and Throughput Variability:** A MMITSS Section shall acquire the performance measure of throughput and its variability for each intersection. Throughput will be used to determine demands for the section or system, which serves as input for adjusting section/system signal timing plan parameters in response to demand fluctuations.

- **Traffic Counts and Traffic Count Variability:** A MMITSS Section shall acquire the estimate of traffic counts and their variability from each intersection. Traffic count will be used to determine the volume demands for the section or system, which serves as input for adjusting section/system signal timing plan parameters in response to demand fluctuations.

Determination/Monitoring Phase Failure Status: The MMITSS shall determine the appropriate strategy for phase failure management using alternative strategy evaluations for intersections within a section. The command and implementation parameters are determined by the corresponding section and the strategy is implemented at the intersection level by setting corresponding intersection parameters (e.g., offsets). Appropriate strategies include free operation, split adjustment, cycle length modifications, and queue management are evaluated prior to selection.

Simulation is used as the verification method by creating models that have high demand on some movements so that phase failure occurs.

Also, The MMITSS shall monitor phase failure status to determine if the intersection phase is in failure mode or if the intersection phase is no longer in failure mode. This requirement allows MMITSS to determine when to start and stop the use of alternative strategies to improve traffic flow at a congested intersection.

The evaluation of performance measures and metrics in terms of structure and quantity at section level could be presented as follows:

Section Delay: MMITSS performance shall be judged on the delay of all vehicles traversing the section. Vehicle delay is the broadest aggregate measure of section performance, and is relevant to all MMITSS use cases, including those that are not intended to improve it and may make it worse (e.g., accommodation for pedestrians with disabilities). Since this is a performance evaluation requirement, it does not rely on a MMITSS performance measurement and can employ field study measurements. This parent requirement is decomposed into child requirements for all day vehicle delay and peak period vehicle delay.

- **Average Vehicle Delay (All day/Peak Period):** MMITSS performance shall be judged on average delay throughout the day or during the peak period of vehicles traversing the section.

- **Average Transit Delay (All day/Peak Period):** MMITSS performance shall be judged on the average delay throughout the day or during the peak period for transit vehicles traversing the section. The average transit vehicle delay reduction is the best aggregate measure of effects on transit service, with implications not only for passenger service but also for operating efficiency of the transit fleet.
- **Average Truck Delay (All day/Peak Period):** MMITSS performance at individual sections shall be judged on average delay throughout the day or during the peak period of trucks traversing the section. Average truck delay is the broadest aggregate measure of intersection performance on trucks, and is relevant to all MMITSS use cases, including those that are not intended to improve it and may make it worse.

- **Average Emergency Vehicle Delay (All day/Peak Period):** MMITSS performance shall be judged on the average delay throughout the day or during the peak period experienced by emergency vehicles responding to emergencies while traversing the section. Emergency vehicle delays can lead to deaths or exacerbated injuries of victims needing emergency assistance, making this an

important performance measure. These delays can only be measured for equipped emergency vehicles, so this performance measure should be focused on the EV priority use cases.

Section Delay Variability: MMITSS performance will be judged on the variability of delays for all vehicles traversing the section. The variability of delay is the statistical variability, dispersion, or variation of delay of those vehicles over the time period of interest. Since this is a performance evaluation requirement, it does not rely on a MMITSS performance measurement and can employ field study measurements. This parent requirement is decomposed into child requirements for all day vehicle delay variability and peak period vehicle delay variability.

- **Vehicle Delay Variability (All day/Peak Period):** MMITSS performance will be judged on the variability of delays throughout the day or during the peak period for vehicles traversing the section. The delay variability for vehicles traversing the section is the statistical variability, dispersion, or variation of delay of those vehicles measured over an entire 24-hour period or the time period defined by local agencies as the peak period.

- **Transit Delay Variability (All day/Peak Period):** MMITSS performance will be judged on the variability of delays throughout the day or during the peak period for transit vehicles traversing the section. The variability of delay is an important measure of reliability of transit service, with implications for both passenger service and operating efficiency of the transit fleet. It is relevant not only to the transit signal priority use cases, but also to the other use cases that may impose delays on transit. The variability is most crucial for peak period operations, when transit bus service is under maximum stress.

- **Emergency Vehicle Delay Variability (All day/Peak Period):** MMITSS performance will be judged on the variability of delays throughout the day or during the peak period for emergency vehicles responding to emergencies while traversing the section. The delay variability for emergency vehicles responding to emergencies while traversing the section is the statistical variability, dispersion, or variation of delay of those vehicles measured over an entire 24-hour period or the time period defined by local agencies as the peak period. A large variability indicates that the system is not adequately responding to the importance (high level of priority).

Section Travel Time: MMITSS performance shall be judged on the travel time associated with all vehicles traversing the section. Section total travel time is defined as

the cumulative time that all vehicles require to travel when inside the boundaries of the section. Since this is a performance evaluation requirement, it does not rely on a MMITSS performance measurement and can employ field study measurements. This parent requirement is decomposed into child requirements for all day vehicle travel time and peak period vehicle travel time.

- **Vehicle Total Travel Time (All day/Peak Period):** MMITSS performance shall be judged on the total travel time throughout the day or during the peak period for vehicles traversing the section. Total travel time evaluated on an all-day or a peak-period basis is defined as the cumulative time that all vehicles require to travel when inside the boundaries of the section.

- **Emergency Vehicle Response Time (All day/Peak Period):** MMITSS performance shall be judged on the emergency vehicle response time throughout the day or during the peak period while responding to emergencies while traversing the section. A meaningful measure of emergency vehicle operational performance is the response time for emergencies, which is best measured at the system level, but is impacted at the section and section levels by the travel time.

Section Travel Time Variability: MMITSS performance will be judged on the variability of the travel time by vehicles traversing the section. Travel time variability indicates the operational consistency of the intersection and a quality of service to users, who can plan their travel more efficiently. This parent requirement is decomposed into child requirements for all day vehicle travel time variability and peak period vehicle travel time variability. Since this is a performance evaluation requirement, it does not rely on a MMITSS performance measurement and can employ field study measurements.

- **Vehicle Travel Time Variability (All day/Peak Period):** MMITSS performance will be judged on the variability of the travel time throughout the day or during the peak period by vehicles traversing the section.
- **Freight Travel Time Variability (All day/Peak Period):** MMITSS performance will be judged on the variability of travel times throughout the day or during the peak period of freight vehicles traversing the section. Reliability of travel time is an important measure of operational efficiency for trucking, so that pickups and deliveries can be scheduled more tightly. Although this measure is most relevant for complete trips spanning the system-level, information gathered at the intersection and section can provide a level of granularity for identifying localized problems and performance contributions.

Section Number of Stops: MMITSS performance shall be judged on the number of stops incurred by all vehicles at a section. The number of stops is an important measure

of section level inefficiency, with a particularly strong relationship to energy consumption and pollutant emissions and goods movement efficiency and pavement damage. This parent requirement is decomposed into child requirements for all day vehicle number of stops and peak period vehicle number of stops for two modes: Passenger vehicles and Freight.

Section Throughput: MMITSS performance shall be judged on total vehicle throughput traversing the section. Section throughput is an important measure of the productivity and capacity of the section when its performance is being pushed to its limits. This parent requirement is broken into two section measures: all-day throughput and peak period throughput. The throughput measures include only vehicle counts in this measure to account for vehicular interactions among multiple travel modes (e.g., transit).

5.4.6.3 Interfaces

Required (Input)
The required data for this component is obtained from MRP_PerformanceObserver in which the main focus is on collecting intersection performance measures.

Provided (Output)
The input data will be utilized to calculate, estimate, monitor, and archive the section performance measures. Also some coordination parameters are driven by section level performance observation, such as Offset (based on number of arrivals during red) or Split Adjustment (based on Phase Utilization/Phase Failures)

5.4.6.4 Functional Specification/Description

Based on the data originating from the MRP_PerformanceObserver, the performance measures related to travel time, delay, number of stops, etc. are calculated.
Some other performance metrics including the number of equipped vehicles traversing the section, and Market Penetration Rate (MPR) are also observed based on the data coming from System Detectors, and OBEs.

By having an accurate vehicle counts from system detectors installed on the roadways, and the information of the number of connected vehicles in the system during a predefined time period, the market penetration rate is achieved as follows:

$$Market\ Penetration\ Rate\ (\%) = \frac{Number\ of\ Connected\ Vehicles}{Total\ number\ of\ vehicles} \times 100 \quad (1)$$

This metric could be calculated for each approach, and then by adding the data from all the other approaches at an intersection, the penetration rate for the intersection is attained. Finally, knowing the market penetration rate at each individual intersection, its average value is calculated for the section based on the simple following equation where “n” is the total number of intersections assigned to the section:

$$Average\ Section\ MPR\ (\%) = \sum_{k=1}^{n} \frac{(MPR)_k}{n} \quad (2)$$

5.4.6.5 Example Application

Figure 58 shows the Section_PerformanceObserver dashboard that is utilized to visualize the performance measures at different levels. For this component, all 6 intersections are considered and different performance metrics and modes are studied.

The observation at this level is divided into two parts: Performance by Mode, and Performance by Metric. Figure 58 and Figure 59 below are the two dashboards resulting from the simulation runs of the whole Arizona Test Bed corridor conducted in VISSIM. The duration of the simulations is determined to be 60 minutes (with an extra 10-minute warm up) and the simulation resolution chosen is 1 second. 10 different random seeds are considered for replications in order to make allowances for the stochastic variations.

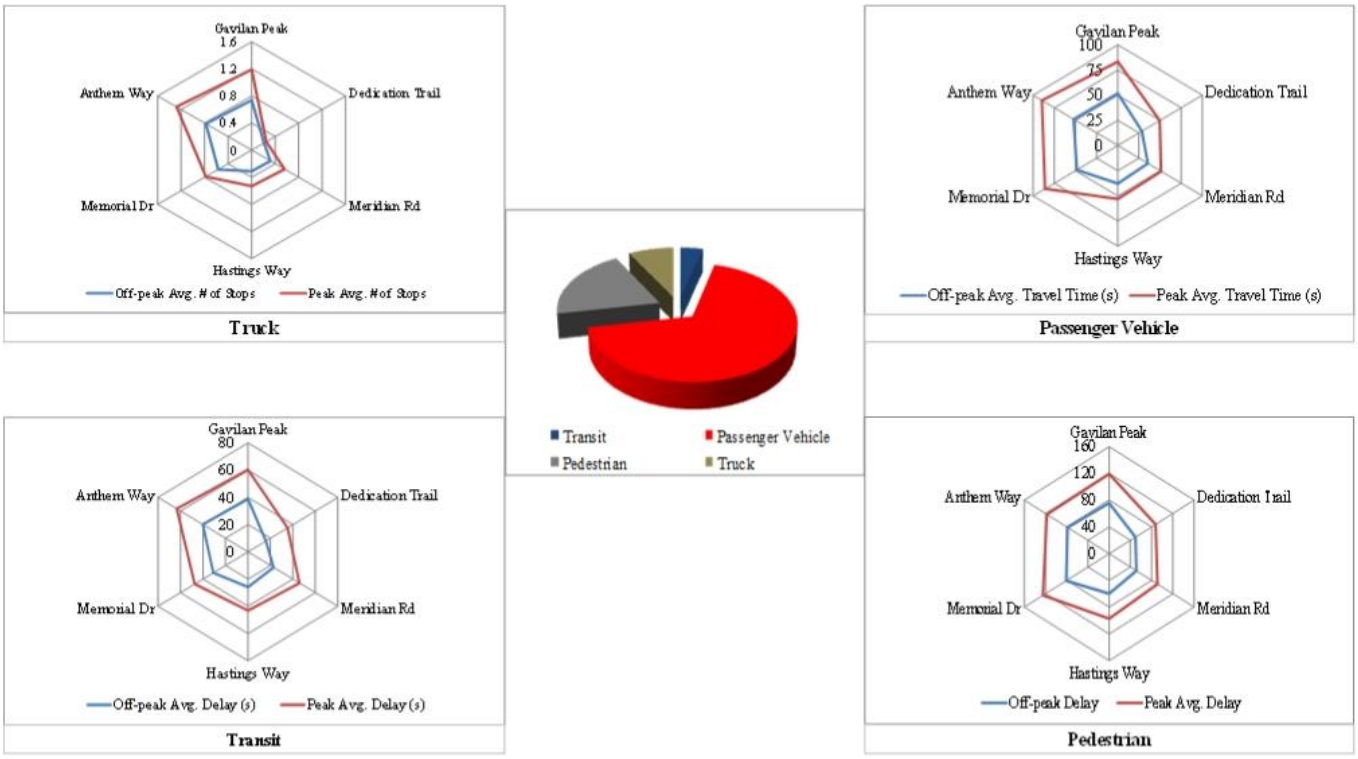


Figure 58 Section Level Dashboard by Mode

Figure 58 compares different modes and their performance measures at each of the intersections in the selected section during the peak period and off-peak period. The pie chart in the middle shows the distribution of modes in the section.

Figure 59 displays the average number of stops for a truck at each intersection of the corridor. Some intersections have all the 12 possible movements (e.g. Gavilan Peak Pkwy, Memorial Dr., and Anthem Way), while some others only have 6 movements because of the geometry of the intersection (e.g. Dedication Trail, Meridian Rd., and Hastings Way).

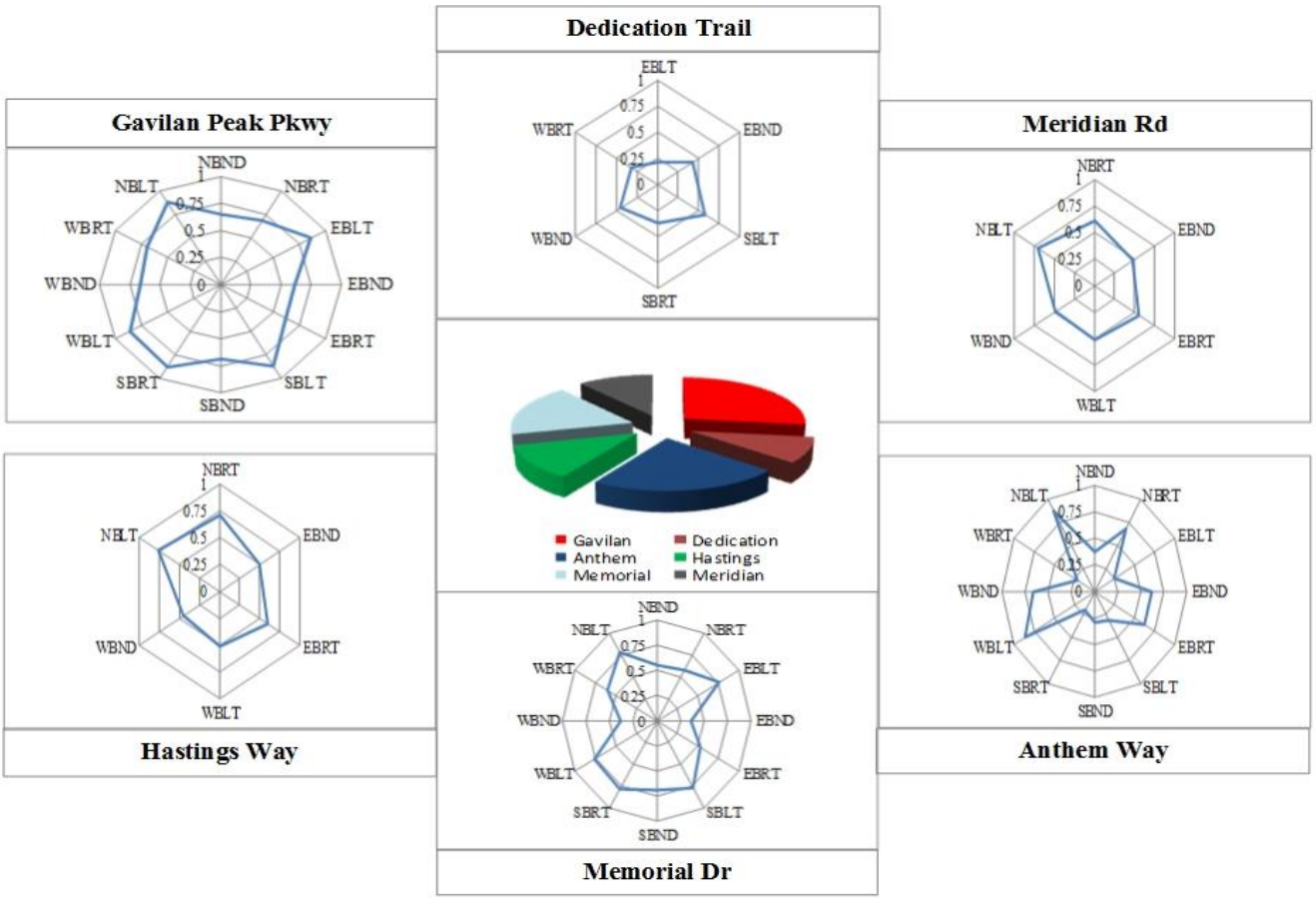


Figure 59 Section Level Dashboard by metric: Average Number of Stops for Truck

5.4.6.6 References

1. *MMITSS System Requirements Document, CDRL 130, Version 3.0; January 2012.*

5.5 Nomadic Device Components

The Nomadic device components are designed based on two efforts: the Savari SmartCross SBIR project and the MMITSS Ped App design. In the Phase I Concept of Operations and High Level Design Tasks, it was determined that the Savari SmartCross project would be able to provide a much higher quality application that would meet the needs of pedestrians and generalize non-motorized travelers. Hence, it was decided to adopt the SmartCross application as a solution to the MMITSS Pedestrian needs. The Savari SmartCross application uses standard SPaT and MAP messages that are sent using cellular communications from the RSE to a smartphone based on the smartphone

location and a cloud based server. The SmartCross application can send an SRM message to the MMITSS MRP_PriorityRequestServer that would request pedestrian service (e.g. place a Ped Call for the appropriate signal phase).

During the MMITSS development, it was decided to also develop a simple MMITSS Ped App that would use only wifi communications and would simply send a call for pedestrian service based on a simple text based map and simple text based signal status data. The purpose of this app was to fill the need to have an app for demonstrations and testing throughout the MMITSS development cycle. The Savari SmartCross app was being developed under a separate project and had different schedule and deliverable milestones.

The high level design of the Savari SmartCross app is presented as well as the detailed design of the MMITSS Ped app components, where appropriate.

5.5.1 Nomadic Priority Request Generator

The MMITSS Ped App does not support this component. The logic for sending a request for pedestrian service is a function in the MMITSS PedApp component.

5.5.1.1 Overview of Functionality (Savari SmartCross)

This component is responsible for tracking the orientation of the phone and determining the signal phase status for the relevant phase. The MMITSS application on the smartphone receives SPaT and MAP messages from the backend infrastructure (Cellular) and determines the position of the user with respect to the crosswalk. It uses data from other on-board sensors on the phone to determine the user's intended direction of travel (heading). Based on this data and, if the user requests a WALK phase for the crosswalk, this component sends a Signal Request Message (SRM). The SRM is a standard SAE J2735 message that is transmitted to the backend infrastructure which in turn transmits it to the RSE at the intersection.

5.5.1.2 Requirements

Node: Nomadic Device	**Component Name:** Nomadic_PriorityRequestGenerator
	Traceability: C1303.302
Description of Responsibility: This component on the nomadic device is responsible for formulating and sending a request for service/priority from the nomadic device.	
Supporting Text: *The Nomadic_PriorityRequestGenerator is analogous to the OBE based priority request generator, with the additional capability of using the nomadic devices location services for determining the location and orientation of the device.*	

5.5.1.3 Interfaces

Required (input/output) Internal to the Nomadic MMITSS Application.

5.5.1.3.1 Required (input):

This component uses the internal Android APIs to acquire the following on-board sensor information:

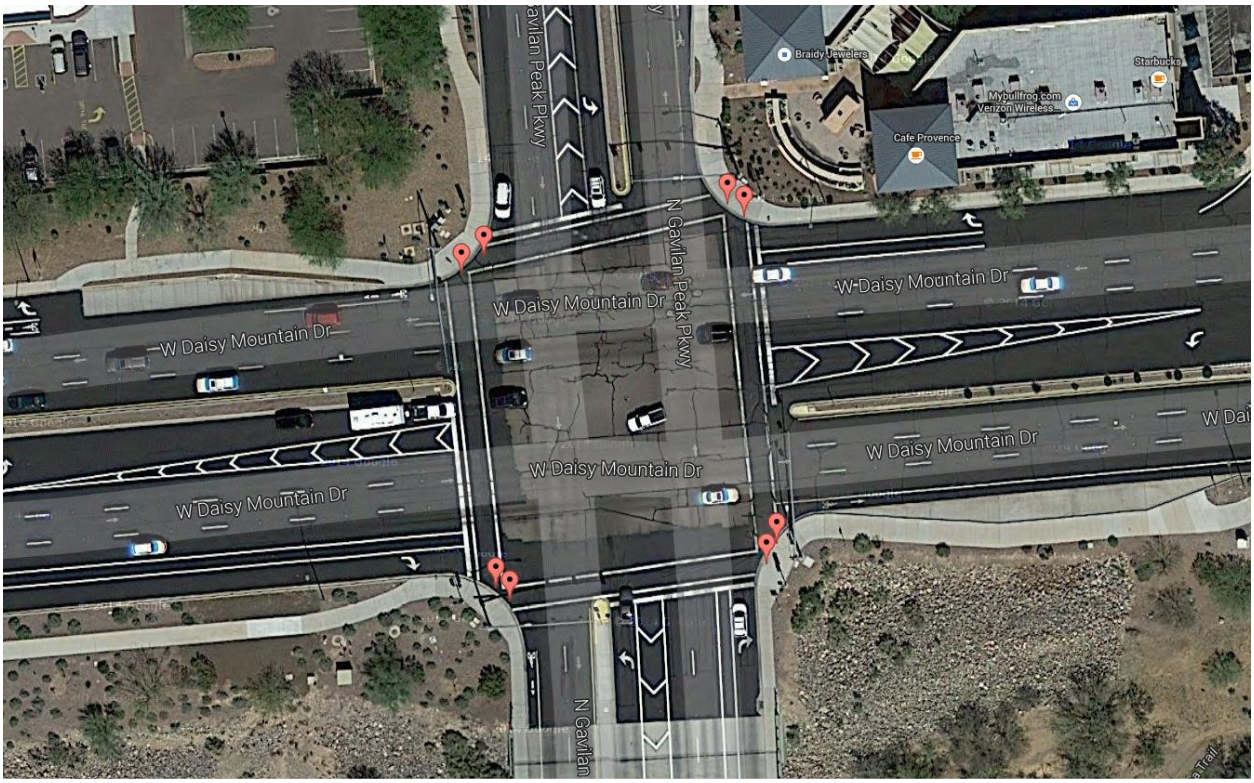
1. GPS location of the user
2. GPS speed
3. Magnetometer reading

The component also uses the decoded MAP structure that is populated when the SAE J2735 MAP messages is received from the RSE. Required (output):

This component sends out an ASN.1 encoded SAE J2735 Signal Request Message that contains the lane number of the crosswalk that the user intends to cross. The SRM is sent to the Nomadic Traveler Service.

5.5.1.4 Functional Specification/Description (Savari SmartCross)

The Nomadic Priority Request Generator is used to formulate and send a signal request message. This component shall take into account, user position, phone orientation (compass heading), speed as well as SPaT and MAP data to determine the user's position with respect to the crosswalk. Based on this data, and specific required input from the user indicating an intent to cross the street, the component shall transmit a Signal Request Message to the Nomadic Traveler Service containing all relevant information. The Signal Request cannot be sent unless the application user is at the curb and facing one of the two possible crosswalks as shown in the figure below.



The thresholds to determine how far the user has to be from the edge of the crosswalk as well as tolerance for magnetometer errors is configurable. It is a component of the MMITSS Nomadic Application for the smartphone.

5.5.1.5 Unit Test Descriptions (debugging and testing) (Savari SmartCross)

This component shall be tested in conjunction with all other components comprising the Nomadic Traveler Service node. The most important tests to be carried out when debugging this component are as follows:

1. Testing the magnetometer errors and user notifications when the sensor readings are not reliable.
2. GPS mode calibration and adjustment.

5.5.2 Nomadic Signal Status Receiver

The MMITSS Ped App does not support this component. The logic for receiving the signal status is a function in the MMITSS PedApp component.

5.5.2.1 Overview of Functionality (Savari SmartCross)

This component is responsible for tracking the orientation of the phone and determining the signal phase status for the relevant phase.

The Signal Status Message (Note: This is assumed to be the modified version of the J2735 2009 SSM and is discussed at the Priority Status Message (PSM) in the MRP_PriorityRequestServer design) is a message sent by an RSE at a signalized intersection. It is used to relate the current status of the signal and any collection of pending or active preemption or priority events acknowledged by the controller. The data contained in this message allow other users to determine their "ranking" for any request they have made as well as see the currently active events. When there have been no recently received requests for service messages, this message may not be sent. The outcome of all pending requests to a signal can be found in the Signal Status Message, and the current event may also be reflected in the SPaT message contents if successful.

Based on the received Signal Status Message and the user's location, speed and past history of whether or not a Signal Request Message was sent from the user, the status for the relevant request is relayed to the user.

5.5.2.2 Requirements

Node: Nomadic Device	**Component Name:** Nomadic_SignalStatusReceiver
	Traceability: A1301, A1302, 13.3.3; §5, §8, §11.1, §11.3"
Description of Responsibility: This component is responsible for receiving signal status data from an intersection and making the data available to the NomadicMMITSSApp.	
Supporting Text: *This is the analogous component to the OBE_MAP_SPaT_Receiver for the nomadic device. It is responsible for receiving status data and making it available to the app.*	

5.5.2.3 Interfaces

5.5.2.3.1 Required (input):

ASN.1 encoded Signal Status Message for the intersection lanes from the backend infrastructure.

5.5.2.3.2 Required (output):

This component is internal to the nomadic device and will provide information to the application logic to determine the phase status for relevant crosswalk based on the Signal Request Message. The output from this component is sent to the user interface (UI) display component that in turn alerts the user that his/her message has been acknowledged by the system.

5.5.2.4 Functional Specification/Description

This component of the SmartCross application will receive the Signal Status Messages from the Nomadic Traveler Service. It then decodes the message and determines if the signal status is applicable based on the user's location, orientation, speed and phone ID. It also checks if the user has requested a pedestrian service phase. If all the constraints are met, the component notifies the user facing components to generate an alert confirms that the SRM has been received by the system.

Unit Test Descriptions (debugging and testing)

This component shall be tested in conjunction with all other components comprising the Nomadic Traveler Service node. The primary tests for this component are:

1. Test component output with different user movements
    a. User requests a ped phase for a crosswalk and moves away from the curb.
    b. User requests a ped phase for a crosswalk and faces the other crosswalk.
    c. User is at the expected location but GPS errors cause component to fail.

5.5.3 Nomadic MMITSS Application (Savari SmartCross)

5.5.3.1 Overview of Functionality (Savari SmartCross)

This is the end-user application that comprises components defined in Sections 5.5.1 and 5.5.2 in addition to:

1. User Interface components
2. SAE J2735 encode/decode modules.
3. Cloud connectivity modules.

The smartphone application serves as the user interface component of the SmartCross system. It enables the pedestrians to interact with the system to:

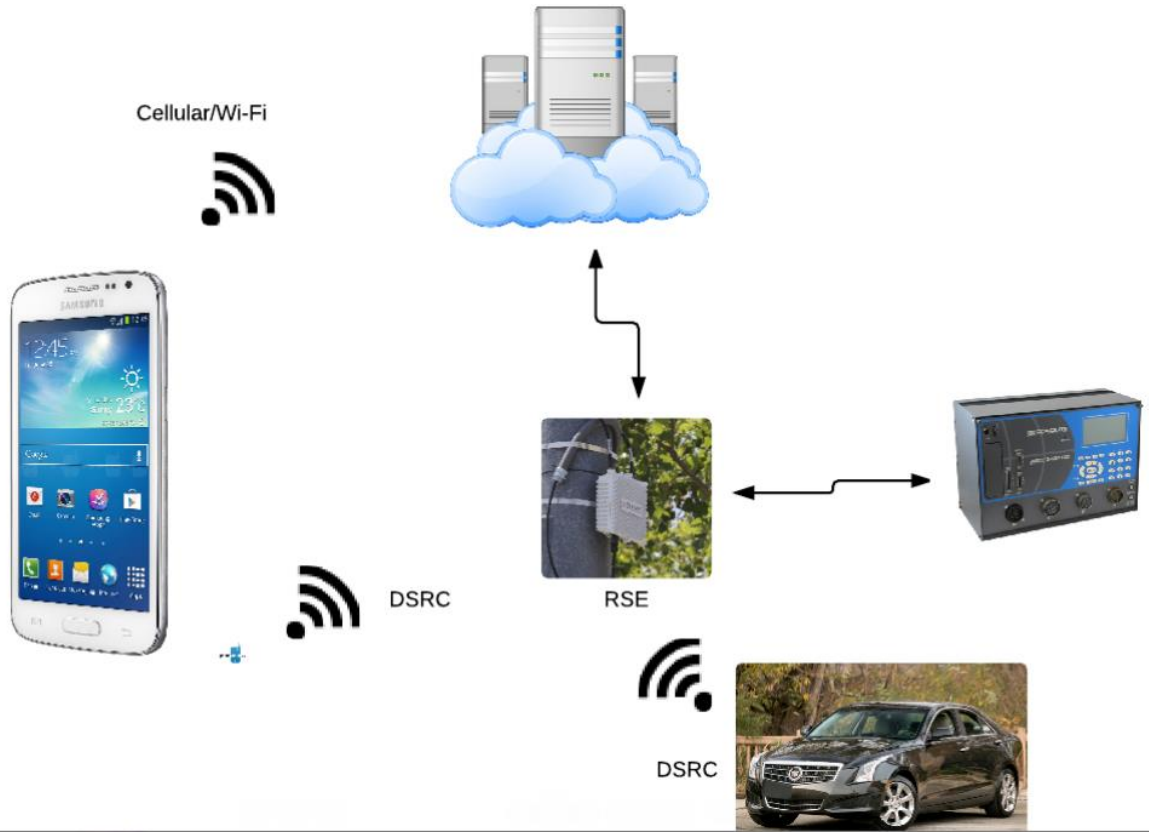
1. Get near real time notifications of crosswalk phases and along with the interval time.
2. Get audio,visual and haptic alerts when the signal state changes or if the user enters a crosswalk when unsafe to do so.
3. Request ped phase by pointing the smartphone at the crosswalk the user wants to use.
4. Get acknowledgement that the system has received the users request.

5.5.3.2 Requirements

Node: Nomadic Device	Component Name: NomadicMMITSSApp
	Traceability: A1303, 13.3.3; §4, §4.1.5, §5, §9.3.4, §11.0, §11.0.1, §11.0.2, §11.3
Description of Responsibility: The NomadicMMITSS App is a downloadable app that individuals can install on their nomadic device. It is the base application for the nomadic device.	
Supporting Text: It is assumed that the Nomadic Device is a smartphone (ios based or android based) and that users will be able to download the app to their device. The app provides the functionality available to the nomadic device including knowing how to connect to the Nomadic_PriorityDataServer to get status data and send requests for service. The Nomdic Device app can be configured to indicate that a user (pedestrian) is disabled, authorized, and provided additional crossing time if needed.	

5.5.3.3 Functional Specification/Description

The application's main purpose is to provide signal timing information in the form of audio, visual and haptic alerts to the user based on his/her location and orientation. The application received the SPaT and MAP information from a cloud based system that is in turn connected to the RSE's at the intersection via Internet. The architecture for the same is as shown below:



When the user launches the application, the UI (shown aside) is presented. If the phone is used in accessibility mode, then each of the buttons shown on the UI will have a Talkback mode that provides audio inputs based on where the user is touching the screen.

Using this screen, the user selects the mode in which the application needs to operate. When the first mode is selected, the UI shown on the below(left) is presented. The user interfaces for the other modes(bicycle, visually impaired and wheelchair) are as shown below:

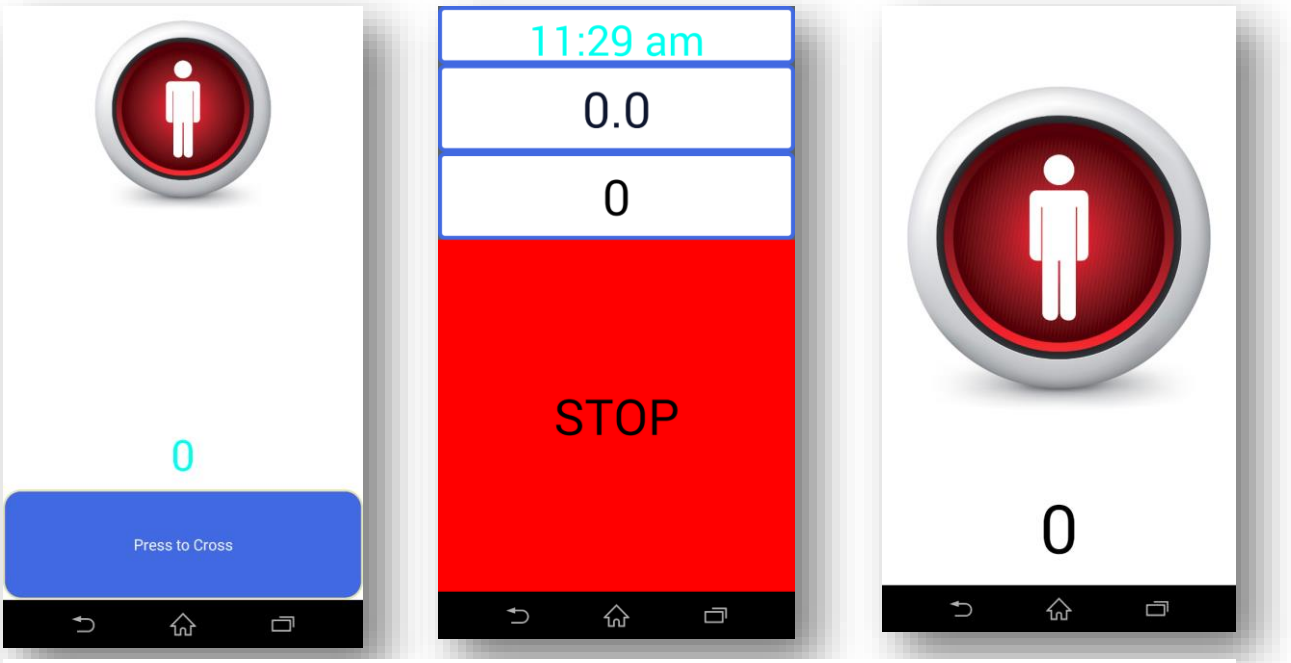


Figure 60 Ped Mode, Bicycle Mode, and Disabled Traveler Mode.

For general pedestrians, the left mode is used. This interface provides the time remaining in seconds if the pedestrian phase interval is active. The snapshot above shows a screen capture in a scenario where the user is not facing a crosswalk or the pedestrian interval is inactive. When facing the crosswalk, the user can request a WALK phase by pressing the blue “Press to Cross” button. When pressed, the application sends out a Signal Request Message as detailed in 5.5.1. On receipt of this message, the SmartCross backend server sends a Signal Status Message back to the pedestrian application. The application will receive this message and notify the user that the request has been acknowledged.

In the bicyclist mode, there is no mechanism to request a phase. The interface merely displays the time remaining for the phase and along with the signal light state. It also displays the bicyclist’s speed.

For the visually impaired and mobility impaired modes, the underlying operation is the same as Mode 1 (general pedestrians). The interface is modified to eliminate the cross button. Instead, if the user wants to request a crosswalk phase, he/she can do so by pressing and holding any part of the screen for 3 seconds or until an audio notification is heard.

In addition to the above operation, multiple scenarios are addressed by the application to enhance safety.

1. If the user enters a crosswalk when the crosswalk is not active.
    a. An audio alert (“Do not walk”) and a strong haptic alert are triggered.
2. When the user is in an active crosswalk
    a. The time remaining along with the ped signal status and audio beeps accompanied by haptic feedback. The audio beep frequency and haptic feedback strength are different for WALK and FDW.
3. When the system is unreliable.
    a. A notification communicating that the system is unreliable and should not be used.
4. The user does not complete crossing the road when the FDW times out
    a. The green could be delayed. This depends on how the local agency wants to address such a scenario.

The flowchart in Figure 61 illustrates user interaction with the SmartCross system.

Data Flow: The application sends out Pedestrian Location Messages (PLMs) to the backend server at an adaptive rate based on the proximity to a connected intersection. Once at the intersection, the application sends out PLMs once a second.

When the backend server receives the PLM message, it updates the list of users at each intersection and begins transmission of SPaT and MAP data to the appropriate pedestrian users.

The application then parses the SPaT and MAP data and uses it to display alerts/ notification to the user based on the Nomadic_SignalStatusReceive and the Nonadic_PriorityRequestServer components.

If the user requests a walk phase, the application encodes the crosswalk lane into a SAE J2735 SRM message and transmits it to the backend server which is forward to the appropriate RSE. The application then receives a Signal Status Message confirming receipt of the SRM.

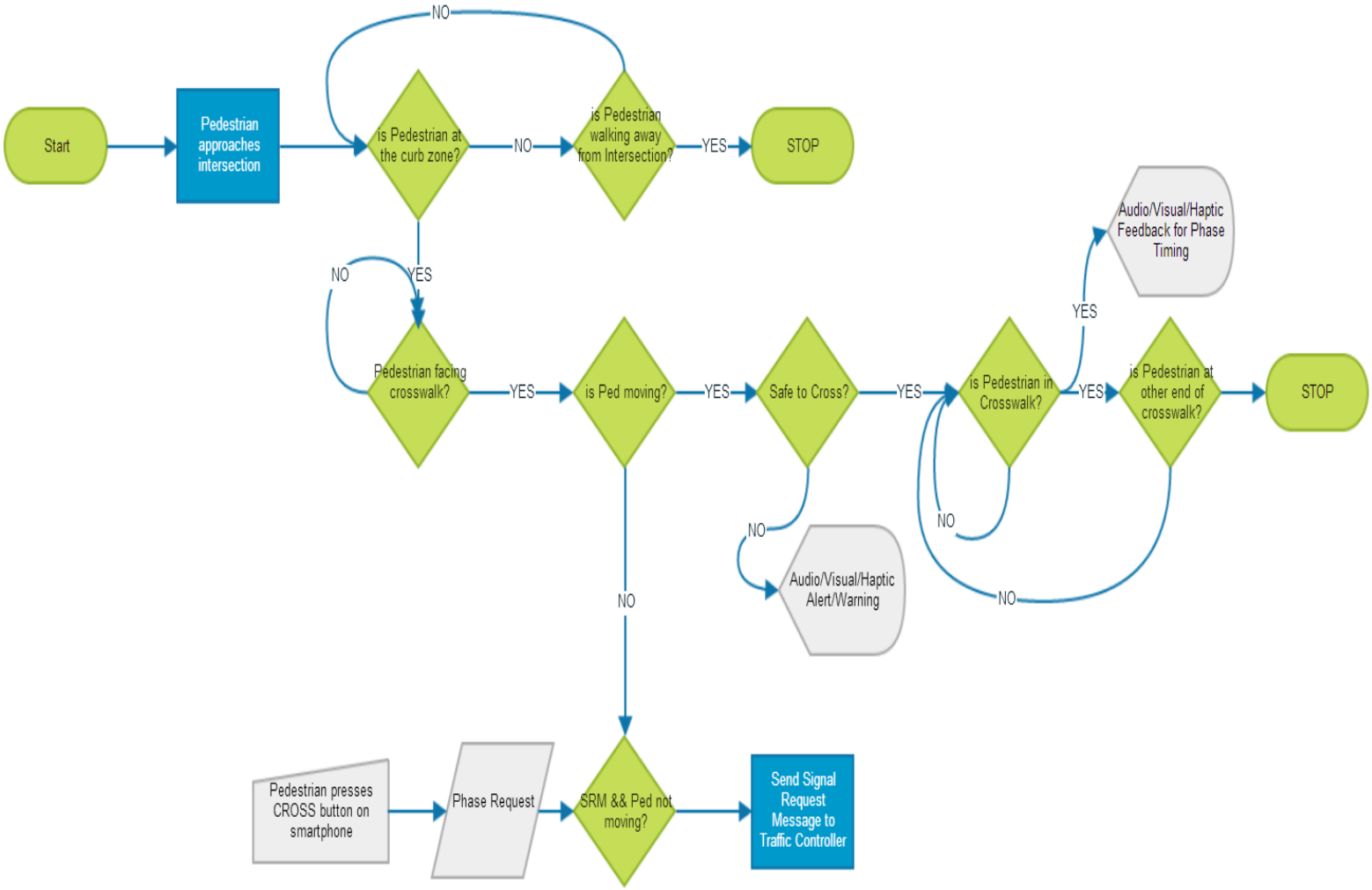


Figure 61 SmartCross Flow Chart.

5.5.3.4 Unit Test Descriptions (debugging and testing)

This component shall be tested in conjunction with all other components comprising the Nomadic Traveler Service node. The application needs real world testing with the following variables:

1. Different types of phones
2. Different intersection geometries
3. Different environments (urban jungles vs. open to sky intersections)
4. Different cellular networks.

5.5.4 Nomadic MMITSS Application (MMITSS PedApp)

5.5.4.1 Overview of Functionality (MMITSS PedApp)

The MMITSS Ped application developed for Android. This application can be used as an aid for pedestrians to request for service and receiving real-time information about the signal indication. It is capable of providing haptic and audio feedback to help pedestrians to cross safely. It has the features and their descriptions as listed in Table 15 that have been developed and tested. **Error! Reference source not found.** shows the MMITSS Ped App user interface.

Table 15 Summary of the MMITSS Ped App Features

Features	Completed and Tested	Description
Mobile device can receive MAP	√	RSE broadcasts MAP and the application will connect to RSE through Wi-Fi and receive it.
Mobile device can locate itself on the MAP	√	The application uses smartphone GPS location and compare it with MAP to find out it is near any crosswalk or not
Mobile device knows when it is located at the entrance to a crosswalk	√	When GPS location of the smartphone matches to MAP information the application will recognize it is near crosswalk
Mobile device knows when it is properly aligned when located at the entrance to a crosswalk	√	The application uses smartphone compass heading to know it is aligned with crosswalk or not.
Mobile device can send request for pedestrian service	√	Pedestrian can send service request through the application and phone send request through Wi-Fi to RSE and RSE forwards it to traffic signal controller.
Mobile device can receive SPaT(Ped) information from intersection	√	The application is connected to RSE and RSE connected to traffic signal controller and forward SPaT data to the application.
Mobile device can display current status (from the Ped point view)	√	The application has a real-time connection to RSE and RSE is connected to traffic signal controller and pedestrian can have real time status of pedestrian signal.

Mobile device will show DON'T WALK when pedestrian should not enter the crosswalk	√	The application doesn't let pedestrian to start walking if pedestrian signal indication is in DON'T WALK interval.
Mobile device will show WALK when pedestrian is properly aligned at the entrance and the current signal control interval is WALK	√	The application shows Walk interval to pedestrian to start walking.
Mobile device will show countdown when the pedestrian is in the crosswalk and the signal is in countdown interval	√	The application provides audio countdown for pedestrian if pedestrian signal indication is in Flashing Don't walk (countdown).
Mobile device will show DON'T WALK when the pedestrian is in the crosswalk and the signal interval is DON'T WALK	√	The application notifies pedestrian if WALK and FLASHING DON'T WALK intervals finish.
Mobile device has haptic and audio alarm	√	

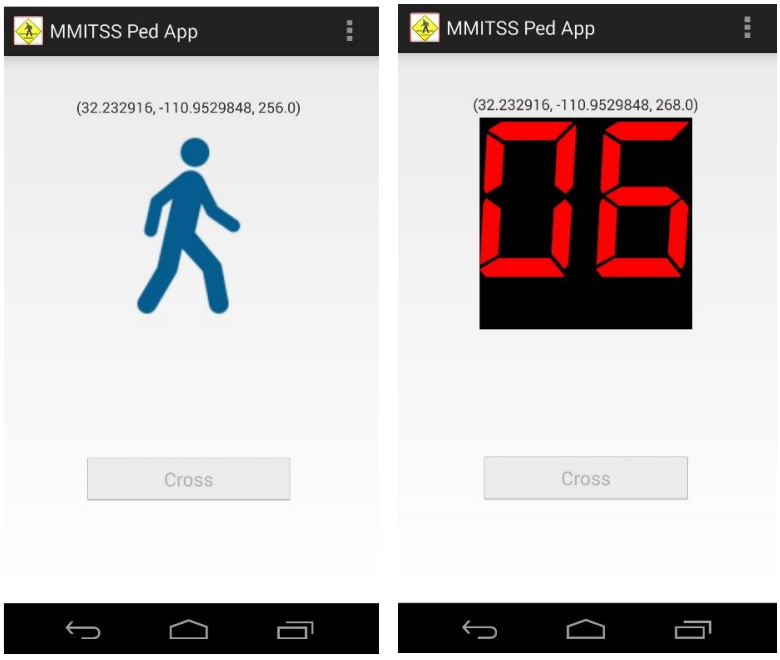


Figure 62 MMITSS Ped App User Interface

5.5.4.2 Requirements

Node: Nomadic Device	Component Name: NomadicMMITSSApp
	Traceability: A1303, 13.3.3; §4, §4.1.5, §5, §9.3.4, §11.0, §11.0.1, §11.0.2, §11.3

Supporting Text: The Nomadic Device is a smartphone (Android based) and that developer team can install application on users' devices. The app provides the functionality available to the nomadic device including knowing how to connect to the RSE to get status data and send requests for service.

Node: Nomadic Device	Component Name: Nomadic_SignalStatusReceiver
	Traceability: A1303, 13.3.3; §4, §4.1.5, §5, §9.3.4, §11.0, §11.0.1, §11.0.2, §11.3
Supporting Text: The Nomadic_SignalStatusReceiver component is a part of the NomadicMMITSSApp Pedestrian Application. It is used to receive Phase information from the RSE whenever there is a request sent by a Pedestrian.	

Node: Nomadic Device	Component Name: Nomadic_PriorityRequestGenerator
	Traceability: A1303, 13.3.3; §4, §4.1.5, §5, §9.3.4, §11.0, §11.0.1, §11.0.2, §11.3
Supporting Text: The Nomadic_PriorityRequestGenerator is a part of the NomadicMMITSSApp Pedestrian Application. It is used to send the signal request from the smartphone to the RSE whenever there is a request sent by a Pedestrian.	

5.5.4.3 Functional Specification/Description

The MMITSS Ped application uses the smartphone's GPS location and compass services to determine which crosswalk the pedestrian is facing. The orientation of the smartphone is very important and if it is not oriented towards a crosswalk, it will not allow the pedestrian to request service. With the phone oriented towards the crosswalk, the pedestrian presses the Cross button and a request for pedestrian service is sent to the RSE and forwarded to the controller.

The communication between the smartphone, RSE and the controller happens inside a 10.254.56.* subnet[3]. Applications on RSE will send and receive packets to the static IP of the phone which is connected to the subnet over WiFi via a wireless router. First, the

[3] The current implementation uses fixed IP addresses for the smartphone and the RSE

MAP that is composed of GPS Waypoints that describe the crosswalk locations and associate each crosswalk with signal data from the controller is broadcast over Wi-Fi and is received by smartphone. Smartphone will receive SpaT data that is broadcast. Figure 63 shows the communication between smartphone and RSE and traffic signal controller. Figure 64 shows a sequence diagram of the interactions.

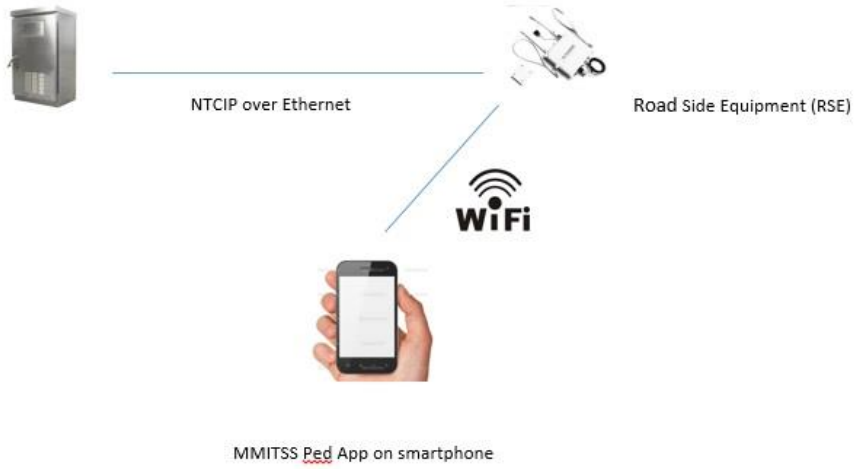


Figure 63 Communication between system components

The application will display the real-time status of the pedestrian indications. It uses the standard pedestrian signal head graphics and a combination of auditory and haptic signals. During the “WALK” interval a standard WALK graphic is displayed. During the “Flashing Don’t Walk” interval, the standard pedestrian graphic and a countdown timer will be shown to the pedestrian. An auditory countdown (e.g. 10, 9, 8,) is played. During “Don’t walk” it shows a standard “Don’t Walk” graphic.

5.5.4.4 Unit Test Description (Debugging and Testing)

If the “Cross” button is pressed when it is not near any crosswalk, it will play an audible message, “you are not near any crosswalk”. When the smartphone in near the curb of crosswalk and the user presses the “Cross” button, if phone is not oriented toward any crosswalk application plays an audible message, “you’re not facing any crosswalk”. When the phone is oriented to the desired crosswalk it will send pedestrian service request to RSE and RSE forwards it to controller. An audible message will confirm the receipt of the request.

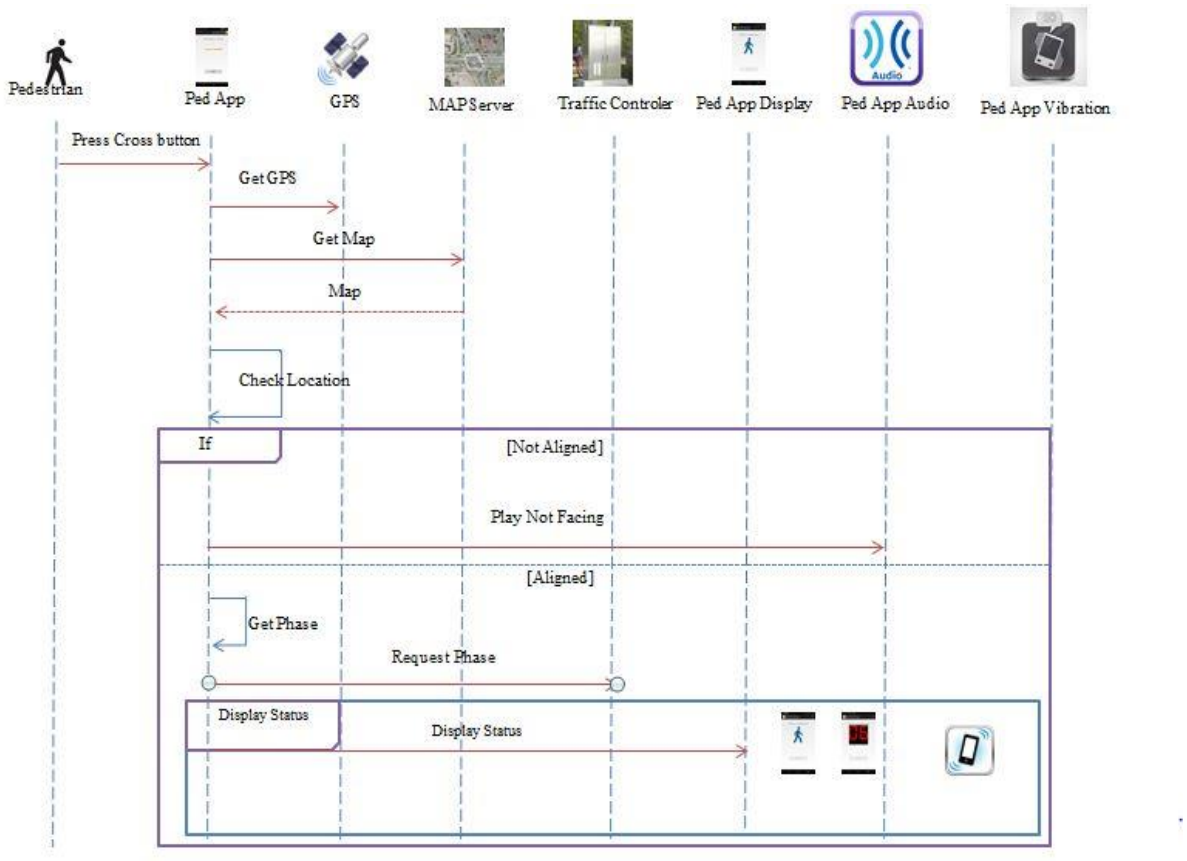


Figure 64 Sequence Diagram showing Ped App interaction.

5.5.5 Nomadic_PriorityDataServer (Savari SmartCross)

5.5.5.1 Overview of Functionality:

This component is part of the Nomadic Traveler Service and is responsible for receiving PLMs from the nomadic users. This component is also responsible for notifying MRP_MAP_SPaT_Broadcast component about the user locations to enable routing of the SPaT data from the appropriate RSE.

5.5.5.2 Requirements

Node: Nomadic Server	Component Name: Nomadic_PriorityDataServer
	Traceability: C1303.302

Description of Responsibility: This component is responsible for acquiring infrastructure data and providing it to nomadic devices using cellular, wifi, and/or DSRC enabled smartphones.
Supporting Text: Infrastructure data, including the MAP, SPaT, Signal Status Messages, etc. needs to be relayed from the MPR to the nomadic device through a cloud based (or server based) capability. The Nomadic_PriorityDataServer is responsible for providing this service. This component will host data for all (many) intersections and the nomadic device will request data (using the NomadicMMITSS App) based on the devices location.

5.5.5.3 Functional Requirements (reference)

Based on the incoming location data from a user, this component shall identify the intersection in which the user is present. Once identified, its forwards the information to the MRP_MAP_SPaT_Broadcast component that will forward MAP and SPaT messages to the server. It shall also have the capability to forward all PLMs received to a port if required.

5.5.5.3.1 Interfaces

5.5.5.3.1.1 Required (input)

Cloud server/TMC based server with internet connectivity to communicate with Google Cloud Messaging (GCM) servers.

Connection with RSEs at intersections via UDP.

Each RSE shall forward SPaT and MAP information to this component.

This component receives SRMs from the smartphone application.

5.5.5.3.1.2 Provided (output)

This component packages SPaT and MAP data and transmits to specific users based on the intersection.

This component relays the incoming SRM to the appropriate RSE based on the location of the user.

This components also relays SSM from the RSE to the smartphone user.

5.5.5.4 Functional Specification/Description

This component receives PLMs from all application users at all connected intersections. Based on their location and predetermined geo-fences, the application puts each user into different bins based on the location- intersection id match.

Once a user has been assigned to an intersection ID, the application begins transmission of SPaT and MAP data to the user at 1Hz.

When the user moves out of the intersection and is no longer within the geo-fence, the application removes the user from the list and stops all outgoing message transmission.

If a user sends an SRM, based on the users location and stored RSE information, the SRM is relayed to the appropriate RSE. The SSM from the RSE is also relayed back to the user.

5.5.5.5 Unit Test Descriptions (debugging and testing)

Since this information is transmitted over the internet using a cellular connection, end to end latency needs to be tested and the time values should be appropriately compensated using the timemark data element in the J2735 SPaT message.

The system should also be tested for reliability over extended periods of time ranging from few hours to weeks.

An assessment of system when any component of the backend infrastructure fails needs to be completed before deployment.

5.5.6 Authorized Special User Service (Savari SmartCross)

5.5.6.1 Overview of Functionality:

This component is part of the Nomadic Traveler Service and is responsible for authorizing services for users with special needs. This includes but is not limited to pedestrians with on wheelchairs, visually impaired and hearing impaired pedestrians. Depending on the type of pedestrians, this component may allow for an extended pedestrian phase that particular user. The functionality of this component is currently limited only to sending a request for an extended phase to the RSE.

5.5.6.2 Requirements

Node: Nomadic Server	**Component Name:** AuthorizedSpecialUserService
	Traceability: C1303.301, C1303.302

Description of Responsibility: This component is responsible for ensuring that nomadic devices are authorized participants in the MMITSS system and to authorize special service for pedestrians with a disability.
Supporting Text: *This service is required to ensure unauthorized devices do not request service or have any other impact on the MMITSS system. Special users, e.g. pedestrians with disabilities, are required to have special authorization before being allowed to request extra crossing time.*

5.5.6.2.1 Functional Requirements (reference)

Based on the incoming Pedestrian Location Message/ABSMs, this component shall verify the identity of the user and authorize special services.

5.5.6.3 Interfaces

This is internal to the Nomadic Traveler Service and does not have any external interfaces

5.5.6.3.1 Required (input):

Pedestrian Location Message/ABSMs

5.5.6.3.2 Required (output):

Verification of the identity of the user and authorization special services

5.5.6.4 Functional Specification/Description

This component receives the incoming ABSM's from nomadic devices and verifies the identity of the user requesting special services from a known list of users. Upon verification, it authorizes the use of special services. These special services are dependent on support from the traffic signal controller as well as traffic agency policies.

5.5.6.5 Unit Test Descriptions (debugging and testing)

This component will be tested with a known group of users requiring special services.

6 Appendices

6.1 Acronyms

ABSM	Alternate Basic Safety Message
AC	Alternating Current
ADA	Americans with Disabilities Act (1990)
AQ	Air Quality
APS	Accessible Pedestrian Signals
ASC	Actuated Signal Controller
ATIS	Advanced Traveler Information Systems
ATDM	Active Traffic and Demand Management
ATV	All-Terrain Vehicle
BRT	Bus Rapid Transit
BSM	Basic Safety Messages
CA	California
CDRL	Contract Deliverables Requirements List
CMMI	Capability Maturity Model Integration
CONOPS	Concept of Operations
CoordRM	Coordination Request Message
CTS	Cooperative Transportation System
CV	Connected Vehicle
DC	Direct Current
DMA	Dynamic Mobility Applications
DOT	Department of Transportation
DSRC	Dedicated Short Range Communication
EMS	Emergency Medical/Management Services
ESD	Electro-static Discharge
ETA	Estimated Time of Arrival
EV	Emergency Vehicle
EVP	Emergency Vehicle Preemption
FHWA	Federal Highway Administration
FOM	Figure of Merit
FPS	Feet Per Second
FTA	Federal Transit Administration
FYA	Flashing Yellow Arrow
GID	Geometric Intersection Description
GPS	Global Positioning Systems
IC	Information Center
ID	Identification
IM	Incident Management
INCOSE	International Council on Systems Engineering
ISIG	Intelligent Traffic Signal System
ITS	Intelligent Transportation System

LOS	Level of Service
MD	Maryland
MHz	Megahertz (10^6 Hertz)
MMITSS	Multi-Modal Intelligent Traffic Signal System
MOE	Measures of Effectiveness
MPH	Miles Per Hour
MRP	MMITSS Roadside Processor
MTBF	Mean Time Between Failure
MTTF	Mean Time to Failure
NHTSA	National Highway Traffic Safety Administration
NTCIP	National Transportation Communications for ITS Protocol
OBE	On-Board Equipment
OD	Origin-Destination
OEM	Original Equipment Manufacturer
PATH	Partners for Advanced Transportation Technology
PFP	Pooled Fund Project
PFS	Pooled Fund Study
PI	Principal Investigator
PII	Personally Identifiable Information
PMPP	Point to Multi-Point Protocol
POV	Privately Owned Vehicle
R&D	Research and Development
RSE	Roadside Equipment
RV	Recreational Vehicle
SCMS	Security Certificate Message Server
SE	Systems Engineering
SPaT	Signal Phase and Timing
SRM	Signal Request Message
SSM	Signal Status Message
STMP	Simple Transportation Management Protocol
SVN	Subversion (PFP Repository with Version Control)
SyRS	System Requirements
TMDD	Traffic Management Data Dictionary
TSC	Traffic Signal Controller
TSP	Transit Signal Priority
UA	University of Arizona
UC	University of California
UML	Unified Modeling Language
USDOT	United States Department of Transportation
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VDOT	Virginia Department of Transportation
VMT	Vehicle Miles Traveled
WAVE	Wireless Access in Vehicle Environment
WSA	WAVE Service Advertisement

MMITSS Detail Design

6.2 Appendix A: Savari RSE Product Specification Sheets

TM

StreetWAVE™ Roadside Unit is a ruggedized outdoor DSRC pla
Vehicle Deployments. StreetWAVE™ is

	–
	45) ports with Auto Uplink™

TM

Specifications

DATA SHEET | LocoMate™ RSU

WAVE Protocols
- 802.11p (WAVE)
- EEE 1609.2
- IEEE 1609.3
- IEEE 1609.4
- SAE J2735

Frequency
- 5.85 - 5.925 GHz
- 5.7 - 5.8 GHz (Europe)

DSRC Radio
- High power miniPCI optimized for 5.9 GHz
- 5.9 GHz: +23dBm at 64QAM from -40°C- +85°C

GPS Device
- GPS with internal RF antenna
- Accuracy <1m

Power Supply
- 802.3af PoE compliant
- IEC60950 compliant

Multi-channel operation
- Consistent 3 mS channel switch time

Supplementary 802.11 MAC features
- Control Channel (CCH) and Service Channels coordination
- 50 mS channel dwell time
- CCH for broadcast, high-priority and single-use safety messages and SCH for IP data

Channel Access
- Alternative, continuous

Channel Switching
- Consistent 3 mS switch time at every 50 mS

Software Queuing
- Transmit queues per channel
- Prioritized channel access queues, with configurable channel access parameters

Database Configuration
- CLI
- Database file backup, restore

Platform
- Linux/Unix compatible
- SDK with C libraries

Interactive Communication
- ssh/telnet

IP Protocols
- ipv4 / ipv6

Network Configuration
- Wired and DSRC
- ipv4 configuration
- ipv6 configuration
- SIT Tunnel Support

US DOT RSE spec
- QPL vendor

GPS Applications
- Approx. 1m accuracy
- Path history implementation
- Path prediction implementation

Local Time Synchronization
- GPS along with PPS

Security
- Signing and verification of messages, encryption and decryption of messages
- Signing and verification of WSAs

Message Logging
- DSRC Transmit packets, DSRC Receive Packets, Ethernet packets
- System events
- Heartbeat messages with configuration (ipv4 or ipv6)
- Log offload configuration (ipv4 or ipv6)
- Wave Service Announcement configuration

LEDs
- DSRC packet transmission
- Firmware upgrade

Software Development Kit
- Linux based tool chain
- Application library
- Sample applications
- Programmer guide
- User guide
- SAE J2735 ASN library
- Sample applications include the following J2735 message formats: BSM, SPAT, MAP, TIM
- Sample applications include GPS data extraction

Data and Management Planes
- UDP/TCP and WAVE Short Messaging Protocol (WSMP) support
- Manages WAVE Basic Service Set (WBSS)
- Application management

Channel Bandwidth
- WAVE mode (802.11p) at 5.9 GHz: reduced to 10 MHz, supports 20 MHz channels

DSRC Message Set - SAE J2735
- BSM Part I, BSM Part II
- SPAT, MAP, TIM

Flash/RAM
- 16 MB Flash
- 64 MB SDRAM (512 Mbits)

Shared Library
Applications Shared Library with Windows/Linux support for application development

Applications Support
- Menu-driven tool
- IP based applications
- WSM-based applications
- Periodic transmit of GPS data
- Remote and logging applications

Certificate Management
- 1609 certificate update
- Support for time limited 1609 certificate

DSRC Channel Support	
10 MHz Channels	Frequency (MHz)
172	5860
174	5870
176	5880
178	5890
180	5900
182	5910
184	5920
20 MHz Channels	Frequency (MHz)
173	5865
175	5875
177	5885
179	5895
181	5905
183	5915

Throughput Traffic Test Results Half-Rates on Channel 172 (Mbps) Without Channel Switch								
Rates	3M	4.5M	6M	9M	12M	18M	24M	27M
TCP	2.36	3.37	4.34	6.32	7.97	11.23	13.54	14.75
UDP	2.38	3.50	4.37	6.99	9.00	12.96	15.81	17.32

Throughput Traffic Test Results Full-Rates on Channel 175 (Mbps) Without Channel Switch		
20 MHz Data Rates	TCP	UDP
6M	4.7	5.0
9M	6.7	7.2
12M	9.8	10.5
18M	12.9	14.52
24M	16.6	18.661
36M	22.630	26.022
48M	27.782	32.231

6.3 Appendix A: Savari OBE Product Specification Sheets

Savari
www.savari.net

MobiWAVE On Board Equipment

Overview

The MobiWAVE™ On Board Equipment (OBE) family of products includes the Vehicle Awareness Device (VAD), the Automotive Safety Device (ASD), the Modular Communications Platform (MCP), and the Software Development Kit (SDK). The MobiWAVE™ OBE is designed to address the needs of the connected vehicle market. The MobiWAVE™ OBE supports a variety of automotive safety and commercial applications.

MobiWAVE™ VAD and ASD are compact, ruggedized safety devices. They are capable of transmitting signed "Here I Am" basic safety messages (BSM) to other vehicles and devices over a dedicated short range communications (DSRC) 5.9 Gigahertz (GHz) wireless network using the protocol stack and other standards associated with DSRC for vehicular communications. These include: IEEE 802.11p, IEEE 1609.1 through 1609.4, and J2735 and a performance standard under development by the automobile industry. Both feature a highly accurate GPS receiver and a 5.9GHz DSRC radio. The DSRC radio is used to communicate to other vehicles. The GPS receiver is used to determine accurate location of the vehicle. These devices have a provisioning/test interface that can receive and load new versions of software, new configurations and credentials, and instructions to perform logging functions and download log messages to external storage. The VAD and ASD can be mounted on different classes of vehicles like light passenger cars, trucks, and public transit buses.

MobiWAVE™ MCP is a module, which is easily integrated into a variety of embedded vehicle devices and platforms. The MCP features a high power DSRC/Wi-Fi radio transceiver, a highly accurate GPS receiver, and a powerful application processor. It comes pre-loaded with Savari firmware. It is ready to be deployed to support a variety of connected vehicle applications.

MobiWAVE™ SDK is integrated with VAD, ASD, MCP and can be used to develop new features and applications.

Technical Specifications

Device	Power	Wireless	GPS	Port	Antenna	Storage
VAD	12VDC USCAR connector	1 25dbmDSRC/ Wi-Fi 5.15-5.9GHz, 10, 20 MHz channels, 802.11a	+/- 2M Position Accuracy, 50% CEP	1 Ethernet 1 RS-232 2 USB 2 FAKRA	Multiband Wi-Fi/DSRC/GPS	Up to 512MB internal, USB external
ASD	12VDC USCAR connector	2 concurrent 25dbmDSRC/ Wi-Fi 5.15-5.9GHz, 10, 20 MHz channels, 802.11a	+/- 2M Position Accuracy, 50% CEP	1 Ethernet 1 RS-323 2 USB 3 FAKRA	Multiband Wi-Fi/DSRC/GPS	Up to 4GB internal, USB external
MCP	3.3VDC	1 25dbmDSRC/ Wi-Fi 2.4-2.4835GHz 5.15-5.9GHz, 10, 20 MHz channels, 802.11a/b/g/n	+/- 2M Position Accuracy, 50% CEP	1 SDIO 3 MMCX	N/A	N/A

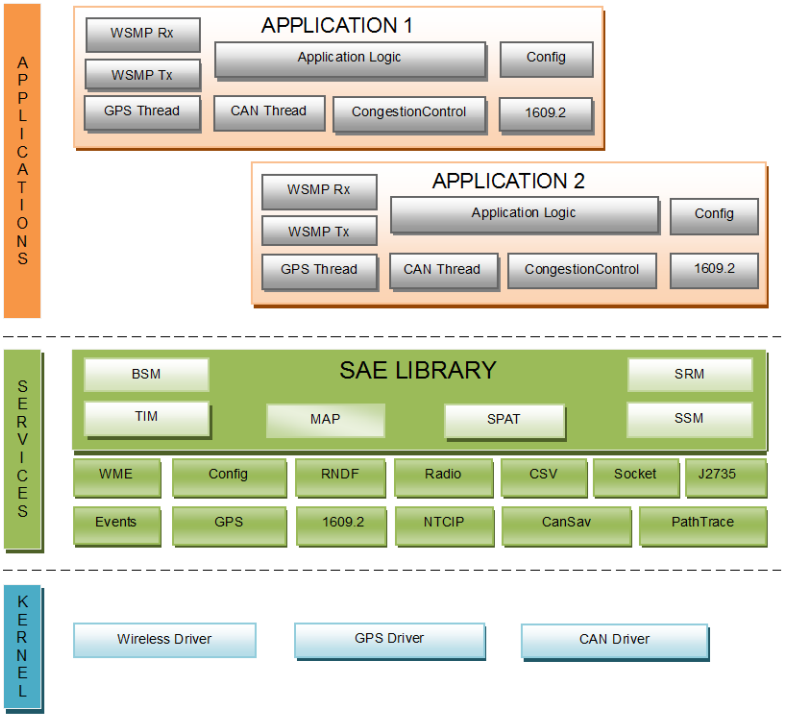
SDK

MobiWAVE™ Software Development Kit (SDK) provides the tools for implementing a variety of safety applications for the ASD and MCP platforms. The SDK includes the Savari On-Board Operating System (SOBOS), which is based on Linux, the APIs, the example safety applications as well as testing and integration tools. All of the MobiWAVE™ devices use the SOBOS as the software framework.

The APIs support GPS, CAN, SAE 2735, and WME. Additional APIs can be used to configure radio interfaces for channel specific parameters along with retrieving debug information. All application messages are signed using the IEEE 1609.2 standard.

All program development is done using the Linux environment. SDK can support the following applications:

- Emergency Brake Light Warning
- Forward Collision Warning
- Intersection Movement Assist
- Blind Spot and Lane Change Warning
- Do not pass Warning
- Control Loss Warning



Savari On Board Operating System

Savari
www.savari.net
Savari - 2005 De La Cruz Blvd #128, Santa Clara, CA 95050
Tel: 408 833 6369 - www.savarinetworks.com - Fax: 408 583 4061

6.4 Appendix B: MMITSS User Guide

Multi-Modal Intelligent Traffic Signal System (MMITSS)

Field Applications User Guide

University of Arizona

Version 1.0

6/7 2015

RECORD OF CHANGES

A – Added, M- Modified, D - Deleted

Version Number	Date	Identification of Figure, Table, or Paragraph	Title or Brief Description	Change Request Number
1.0	6/7/2015	N/A	Initial Draft	

Table of Contents

1. Purpose of the Document

This document is a user guide for the Multi-Modal Intelligent Traffic Signal Systems (MMITSS) applications. This document contains the detailed usage instructions for each of the MMITSS applications including a description of each configuration file that is required.

2. MMITSS Introduction

MMITSS is a USDOT Cooperative Transportation Systems Pooled Fund project that focuses on connected vehicle Dynamic Mobility Applications (DMA). The project investigates the next generation of traffic signal system in a connected vehicle environment that serve multi modal including general vehicles, transit, emergency vehicles, fright vehicles, pedestrians and bicyclists. The project mainly consists of four parts: Intelligent Traffic Signal System (ISIG), Signal Priority (SP), Mobile Accessible Pedestrian Signal System (PED-SIG) and real-time performance observer (PERF-OBS). I-SIG provides real-time adaptive signal control to general vehicles which is the major topic of this dissertation. Signal Priority provides priority to different modes of vehicles including transit, trucks and emergency vehicles. PED-SIG includes a smartphone application that can send pedestrian signal request and provide assistance to disabled pedestrians. PERF-OBS monitors all the users on the road and collect real-time performance data by movement by mode such as travel time, delay, and queue length etc. The component diagram of MMITSS is shown in Figure 1. More details of the introduction of the project can be found in (Head et al., 2015) and (University of Arizona, 2015). WAVE message transceiver (MSGTRANS) provides a physical-layer agnostic network interface, mainly used for DSRC communication using the WAVE protocol. The message transceivers are required on both RSE and OBE and are used by ISIG, SP and PERF-OBS. All the applications will be introduced in following sections.

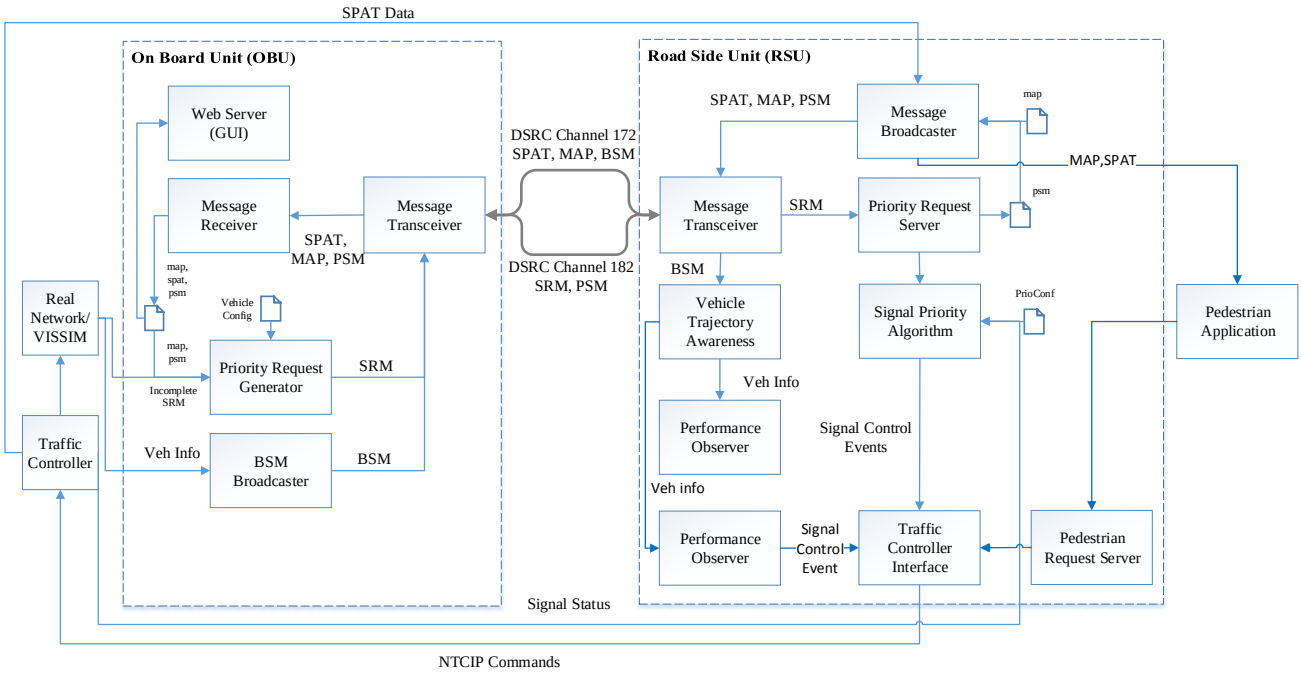


Figure 1 MMITSS Component Diagram

MMITSS is composed of a collection of applications that each performs critical system functions. Table 1 summarizes 16 MMITSS applications with its hardware location (e.g. processor) and the MMITSS functions that it supports.

Table 1 MMITSS Applications

Application Name	Location	MMITSS Components
MMITSS_MRP_EquippedVehicleTrajectoryAware	RSE	ISIG, PERF-OBS
MMITSS_MRP_MAP_SPAT_Broadcast	RSE	SP
MMITSS_MRP_Priority_Solver	RSE	SP
MMITSS_MRP_PriorityRequestServer	RSE	SP
MMITSS_MRP_Signal_control	RSE	I-SIG
MMITSS_MRP_TrafficControllerInterface	RSE	I-SIG, SP
MMITSS_Ped_MAP_Broadcast	RSE	PED-SIG

MMITSS_MRP_PerformanceObserver	RSE	PERF-OBS
MMITSS_MRP_Priority_Webserver	RSE	PERF-OBS, SP
MMITSS_MRP_PedRequestServer	RSE	PED-SIG
MMITSS_RSE_Message_Transceiver	RSE	ISIG, SP, PERF-OBS
MMITSS_OBE_BSMData_Transmitter	OBE	ISIG, PERF-OBS
MMITSS_OBE_MAP_SPAT_Receiver	OBE	SP
MMITSS_OBE_PriorityRequestGenerator	OBE	SP
MMITSS_OBE_Webserver	OBE	SP
MMITSS_OBE_Message_Transceiver	OBE	ISIG, SP, PERF-OBS

The current implementation of MMITSS requires that all the field applications need to be run on Savari StreetWAVE RSE and Savari MobiWAVE OBE. To compile the source code, the Savari SDK is needed. There is no need to install the applications. After compiling, the binary file of each application should be copied to /usr/local/bin folder of RSE or OBE.

3. WAVE Message Transceiver

The WAVE Message transceiver works in the same way on both OBE and the RSE. The application is invoked as below:

OBE: MMITSS_OBE_Message_Tranceiver –f CONF [-d]

RSE: MMITSS_RSE_Message_Tranceiver –f CONF [-d]

Argument: -f specifies which file to use to read configuration information. (e.g. /etc/config/wmefwd/bsm_wme.conf)

Extra Argument: -d debug mode to print logs to screen rather than writing to a file.

Sample Configuration file is as follows. Inline comments explain the role of each option:

```
########################################
###   Sample config file for WME app   ###
########################################

### Configuration enabled: 0 -> Disabled; 1-> Enabled; Default -> 1
Enabled = 1

### Name of the Message: Default -> WME_MSG
MsgName = BSM

### Direction: 0 -> Outgoing; 1 -> Incoming; Default -> 1
Direction = 1

### Channel number to use for WME: Default -> 172
ChannelNumber = 172

### Interface to use for WME: Default -> ath0
InterfaceName = ath1

### PSID of the message which uses this config file: Default -> 0x10
PSID = 0x20

### Channel Mode: 0 -> Alternating; 1 -> Continuous; Default -> 1
ChannelMode = 1

### Remote IP to which socket is opened: Default -> 127.0.0.1
RemoteIP = 127.0.0.1

### Remote port to which socket is opened: Default -> 9999
RemotePort = 3333

### Protocol to use for socket: Default -> UDP
RemoteProtocol = UDP

### Provider Service Context: Default -> "PSC"
PSC = MMITSS

### Message Length: Default -> 128 bytes
MsgLength = 75
```

This configuration file shows an example of BSM transceiver in RSE. For different message types on different devices, users need to modify: Name, Direction, Channel, Interface, PSID, Remote Port, and Message Length as well as the configuration file name (e.g. srm_wme.conf). All the WAVE message transceiver configuration files should be copied to /etc/config/wmefwd

4. Intelligent Signal Control

The Intelligent Signal Control applications requires the following applications in RSE and OBE need to be run:

RSE:

MMITSS_MRP_EquippedVehicleTrajectoryAware PORT SPEED_CO LOG

Argument PORT: port that receives BSM, should be 3333

Argument SPEED_CO: speed coefficient, should be 1.0

Argument LOG: 1- log incoming BSM data; 0- don't log incoming BSM data

MMITSS_MRP_Signal_control PORT PR OBJ

Argument PORT: Port that request vehicle trajectory from VehicleTrajectoryAware, should be 3333

Argument PR: Market penetration rate (change when needed)

Arguemnt OBJ: 0 - objective is to minimize total vehicle delay; 1- objective is to minimize total queue length

MMITSS_MRP_TrafficControllerInterface

There is no argument for this application

MMITSS_RSE_Message_Tranceiver -f /etc/config/wmefwd/bsm_wme.conf

Argument 1: location of the configuration file (e.g. /etc/config/wmefwd/bsm_wme.conf)

OBE:

MMITSS_OBE_BSMData_Transmitter

There are no arguments with this application

MMITSS_OBE_Message_Tranceiver -f /etc/config/wmefwd/bsm_wme.conf

Argument 1: location of the configuration file

Several configuration files need to be constructed before running the applications.

RSE: All the RSE configuration files are located under /nojournal/bin

1. Configinfo.txt

The content of the file should be: /nojournal/bin/ConfigInof_XXXXX.txt where XXXXX is the name of the intersection.

2. Configinfo_XXXXX.txt (XXXXX is the name of the intersection)
   The content of the file will be read from the controller when the applications started.
3. Signal_Config_COP.txt

   The content of the file is the signal parameters of the signal controller including number of phases in use, phase sequence, minimum green time, maximum green time, yellow time, red clearance time, pedestrian walking time and pedestrian clearance time.

4. XXXXX.nmap (XXXXX is the name of the intersection)

   The content of the file is the intersection map description file. Details of how to construct the map and an example map file can be found in (University of Arizona, 2015).

5. DSRC_Range.txt
   The content of the file is the geo-fencing area of the intersection which specifies the length of the extension of map nodes from the reference point for each leg of the intersection.
6. IPInfo.txt
   The content of the file is the IP address and port information for the (virtual) signal controller
7. ntcipIP.txt
   The content of the file is the IP address and port information for the (virtual) signal controller
8. Lane_Phase_Mapping.txt
   The content of the file is the phase mapping to lane in order to calculate requested phase. The value is the phase number. The sequence of the number in line 2 is phase of approach 1 through, approach 1 left, approach 3 through, approach 3 left, approach 5 through, approach 5 left, approach 7 through, approach 7 left. The approach number should match your map. If a phase doesn't exist, put zero, if left turn and through share the same phase, put the same value.
9. nmap_name.txt
   The content of the file is the name of the map description file: /nojournal/bin/XXXXX.nmap

5. Signal Priority

In the field version, the following applications in RSE and OBE need to be run:

RSE:

MMITSS_MRP_MAP_SPAT_Broadcast MAP_IP SPAT_IP LOG

Arguments MAP_IP: Local host IP for sending MAP to

Argument SPAT_IP: Local host IP for sending SPaT to

Argument LOG: 1: log incoming SPaT data; 0: don't log SPaT data

MMITSS_MRP_Priority_Solver –s NETWORK –c INTEGRATION

Argument NETWORK: -s 1: for congested network; –s 0: for regular network

Argument INTEGRATION: -c 1: for using signal priority control with actuated control; -c 2: for using signal priority control with I-SIG

MMITSS_MRP_TrafficControllerInterface

MMITSS_MRP_PriorityRequestServer_field –o COOR –c INTEGRATION

Argument COOR: -o 1: for considering coordination as a form of priority; –o 0: without considering coordination

Argument INTEGRATION: -c 1: for using signal priority control with actuated control; -c 2: for using signal priority control with I-SIG

MMITSS_RSE_Message_Tranceiver -f /etc/config/wmefwd/spat_wme.conf

Argument 1: location of the configuration file

MMITSS_RSE_Message_Tranceiver -f /etc/config/wmefwd/art_wme.conf

Argument 1: location of the configuration file

MMITSS_RSE_Message_Tranceiver -f /etc/config/wmefwd/srm_wme.conf

Argument 1: location of the configuration file

OBE:

MMITSS_OBE_MAP_SPAT_Receiver

MMITSS_OBE_PriorityRequestGenerator_field

MMITSS_OBE_Message_Tranceiver -f /etc/config/wmefwd/spat_wme.conf

Argument 1: location of the configuration file

MMITSS_OBE_Message_Tranceiver -f /etc/config/wmefwd/art_wme.conf

Argument 1: location of the configuration file

MMITSS_OBE_Message_Tranceiver -f /etc/config/wmefwd/srm_wme.conf

Argument 1: location of the configuration file

Several configuration files need to be constructed before running the applications.

RSE: All the RSE configuration files are located under /nojournal/bin

1. Configinfo.txt

   The content of the file should be: /nojournal/bin/ConfigInof_XXXXX.txt where XXXXX is the name of the intersection.

2. Configinfo_XXXXX.txt (XXXXX is the name of the intersection)
   The content of the file will be read from the controller when the applications started.
3. Configinfo_EV.txt
   The is generated by MMITSS_MRP_Priority_Solver. The file is used whenever there is an EV in the active request table.
4. XXXXX.nmap (XXXXX is the name of the intersection)

   The content of the file is the intersection map description file. Details of how to construct the map and an example map file can be found in (University of Arizona, 2015).

5. IPInfo.txt
   The content of the file is the IP address and port information for the (virtual) signal controller
6. ntcipIP.txt

The content of the file is the IP address and port information for the (virtual) signal controller

7. rsuid.txt

   This file contains the intersection name.

8. InLane_OutLane_Phase_Mapping.txt

   The content of the file is the inlane / outlane mapping to phases in order to calculate requested phase. This file is generated by MMITSS_MRP_MAP_SPAT_Broadcast

9. nmap_name.txt

   The content of the file is the name of the map description file: /nojournal/bin/XXXXX.nmap

10. requests.txt

    The content of the file is the active requests table.

11. requests_combined.txt

    The content of the file is the active requests table.

12. Results.txt

    The content includes the solution of the optimizer whenever the optimizer is called.

13. signal_status.txt

    The content includes 8 numbers that show the status of each phase.

14. NewModel.mod

    The file includes the mathematical model that will be used by optimizer (GLPK).

15. NewModel_EV.mod

    The file includes the mathematical model that will be used by optimizer (GLPK) whenever there is an EV at the intersection.

16. NewModelData.dat

    The file contains the input for the mathematical model.

OBE: All the OBE configuration files are located under /nojournal/bin

1. vehicleid.txt

   The content of the file shows the vehicle ID, vehicle type and vehicle name

2. signal_status.txt

The content includes 8 numbers that show the status of each phase.

3. Intersection_maps.txt

   The content of the file is the list of received maps

4. Intersection_MAP_XXX.txt (XXX is the MAP ID)

   The content of the file is the map description file with MAP ID XXX.

5. psm.txt

   The content of the file indicates all of the active priority requests at the intersection. The first line shows the number of requests. The rest lines are the information of each request such as vehicle id, type, ETA, inlane, outlane, start time of service, end time of service, and vehicle status.

6. ActiveMAP.txt

   The content of the file indicated which intersection the vehicle is approaching and which one it is leaving.

7. busStopsRange.txt

   This file is used only for transit vehicles. It indicated the distance between each bus stop in the route of the bus to the next intersection. The bus stop distance to intersection is important only when it is less than 300 meters.

6. Performance Measurement

In the field version, the following applications in RSE and OBE need to be run:

RSE:

MMITSS_MRP_EquippedVehicleTrajectoryAware PORT SPEED_CO LOG

Arguments PORT: port that receives BSM, should be 3333

Argument SPEED_CO: speed coefficient, should be 1.0

Argument LOG: 1: log incoming BSM data; 0: don’t log incoming BSM data

MMITSS_MRP_PerformanceObserver_Field PORT PR SERVER_IP

Argument PORT: Port that request vehicle trajectory from VehicleTrajectoryAware

Argument PR: Market penetration rate (change when needed)

Arguemnt SERVER_IP: IP Address of the Database Server Machin (change when needed)

MMITSS_RSE_Message_Tranceiver -f /etc/config/wmefwd/bsm_wme.conf

Argument 1: location of the configuration file (e.g. /etc/config/wmefwd/bsm_wme.conf)

OBE:

MMITSS_OBE_BSMData_Transmitter

MMITSS_OBE_Message_Tranceiver -f /etc/config/wmefwd/bsm_wme.conf

Argument 1: location of the configuration file

Several configuration files need to be constructed before running the applications.

RSE: All the RSE configuration files are located under /nojournal/bin

1. Configinfo.txt

The content of the file should be: /nojournal/bin/ConfigInof_XXXXX.txt where XXXXX is the name of the intersection.

2. Configinfo_XXXXX.txt (XXXXX is the name of the intersection)
   The content of the file will be read from the controller when the applications started.
3. XXXXX.nmap (XXXXX is the name of the intersection)

   The content of the file is the intersection map description file. Details of how to construct the map and an example map file can be found in (University of Arizona, 2015).

4. GeoFence_Range.txt
   The content of the file is the geo-fencing area of the intersection which specifies the length of the extension of map nodes from the reference point for each leg of the intersection.
5. IPInfo.txt
   The content of the file is the IP address and port information for the (virtual) signal controller
6. ntcipIP.txt

The content of the file is the IP address and port information for the (virtual) signal controller

7. Lane_Phase_Mapping.txt

   The content of the file is the phase mapping to lane in order to calculate requested phase. The value is the phase number. The sequence of the number in line 2 is phase of approach 1 through, approach 1 left, approach 3 through, approach 3 left, approach 5 through, approach 5 left, approach 7 through, approach 7 left. The approach number should match your map. If a phase doesn't exist, put zero, if left turn and through share the same phase, put the same value.

8. Lane_Movement_Mapping.txt

   The content of the file is the movement mapping to lane in order to do the performance observation by movement. The value is the lane number. The sequence of the number in line 2 is lane number of approach 1 right, approach 1 left, approach 3 right, approach 3 left, approach 5 right, approach 5 left, approach 7 right, approach 7 left. The approach number should match your map. If a designated left-turn or right-turn lane doesn't exist, put zero.

   The sequence of the numbers on line 4 is the arbitrary numbers assigned to each travel time section at intersection. This could be used for comparison and data validation in simulation to be matched with VISSIM travel time sections.

9. nmap_name.txt

   The content of the file is the name of the map description file: /nojournal/bin/XXXXX.nmap

7. Pedestrian Application

In the field version, the following applications in RSE need to be run:

MMITSS_MRP_MAP_SPAT_Broadcast MAP_IP SPAT_IP LOG

Arguments MAP_IP: Local host IP for sending MAP to

Argument SPAT_IP: Local host IP for sending SPaT to

Argument LOG: 1: log incoming SPaT data; 0: don’t log SPaT data

MMITSS_Ped_Map_Broadcast PHONE_IP

Argument PHONE_IP: IP address of the smartphone

MMITSS_MRP_PedRequestServer PHONE_IP

Argument PHONE_IP: IP address of the smartphone

The following applications in smartphone need to be run:

MMITSS Ped Application

The communication between RSE and the smartphone is through WIFI. A router should be configured and connected to the RSE in the same subnetwork as shown in Figure 2.

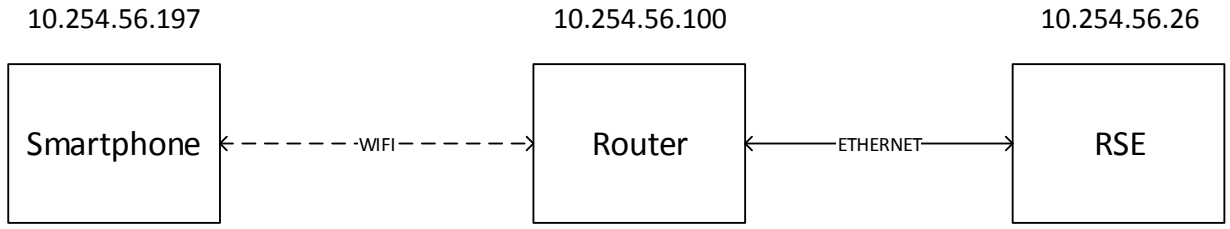


Figure 2 Communication Diagram of MMITSS Pedestrian Application

8. Central System

MMITSS Central System is intended to run on a server in the same network as the roadside equipment. It consists of different file formats including: .sql, .php, .html, .css, and .txt.

The login page is the first page that gets the username and password from the system operator. Currently the credentials are not verified. By clicking the submit button, the operator will be directed to a page with the map of the network on right and list of active intersections on left. For each intersection there is an option for selecting either the configuration page for the signal priority application (N Level Priority Policy) or the performance report page.

The performance Report button directs the operator to a page where they can specify the measurement reports for infrastructure and/or connected vehicles. Some of the fields are not fully functional in this version, for example the split monitoring measures under infrastructure report is not currenly implemented since this is typically a traffic signal management system capability. The “Show Metrics” button leads to the next page that shows the visualizations of the selected measures.

9. References

Head, L., Feng, Y., Zamanipour, M., Khoshmagham, S., Khosravi, S., Mucheli, S., 2015. A Multi-Modal Intelligent Traffic Signal System: Architecture, Components, and Implementations. Work. Paper.

University of Arizona, 2015. Multi-Modal Intelligent Traffic Signal System - Detailed System Design. University of Arizona.