User Guide for Simulation Application

Project: AMS Testbed for the Evaluation of DMA and ATDM Applications/Strategies
Prepared by: Booz Allen Hamilton, University of Arizona

The purpose of this user-guide is to help users of this repository to utilize the MMITSS application, specifically, the simulation side of applications. A separate user-guide is provided for the controller side for Linux-based installation. Booz Allen team has acquired the MMITSS system from University of Arizona. The MMITSS system is a software in the loop system (SILS) and uses Vissim's Econolite/ASC3 control system to replace the innate signal control behavior using specifically designed Docker containers.

The system also uses two inputs: Loop-detector inputs and Connected Vehicle Data. Loop detector inputs are collected using data collection devices at intersection approaches. Driver behavior model in Vissim is used to read vehicle data and generate BSM (Basic Safety Messages) to provide connected vehicle inputs to the MMITSS system. The setup is shown in Figure 1.

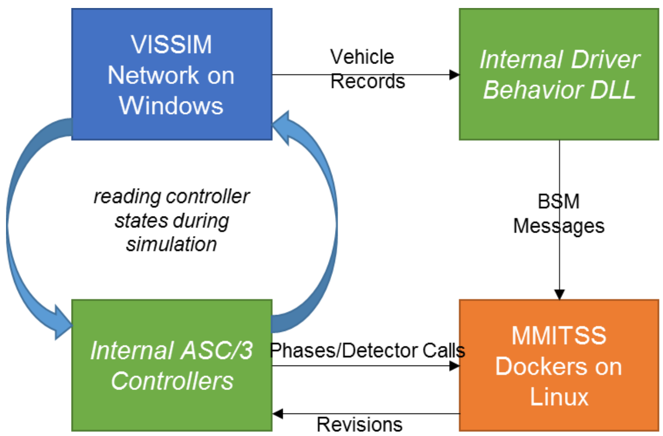

Figure 1. MMITSS Integration in AMS Testbeds [Source: Booz Allen]

As shown in Figure 1, the MMITSS is modeled as a complex software-in-the-loop system to mimic real-world controller operations. The MMITSS application is coded to Docker Containers which are issued specific sub-net IP addresses and require two forms of inputs.

1. The ACS/3 controllers in Vissim are coded to communicate with these containers using their IP addresses which will provide them with phasing and detector call inputs as NTCIP 1202 objects.
2. The other input is provided by the vehicles itself in the form of Basic Safety Messages. BSMs are generated by the CV class of vehicles using BSM-generating driver-behavior models. These models communicate the BSMs to a BSM emulator which classifies them to the Road-Side-Equipment which is in its range. Based on the RSE-mapping, each of these BSMs go to their specific MMITSS containers to form the second set of inputs.

Using these two inputs, the containers produce NTCIP commands to intelligently control the ASC/3 controllers which are provided as output to the specific controller files.

MMITSS Components:

MMITSS application suite consists of two major components – The On-Board Unit (OBU) and Road-Side Unit (RSU) and each of these components have multiple sub-components within it as shown in Figure 2.

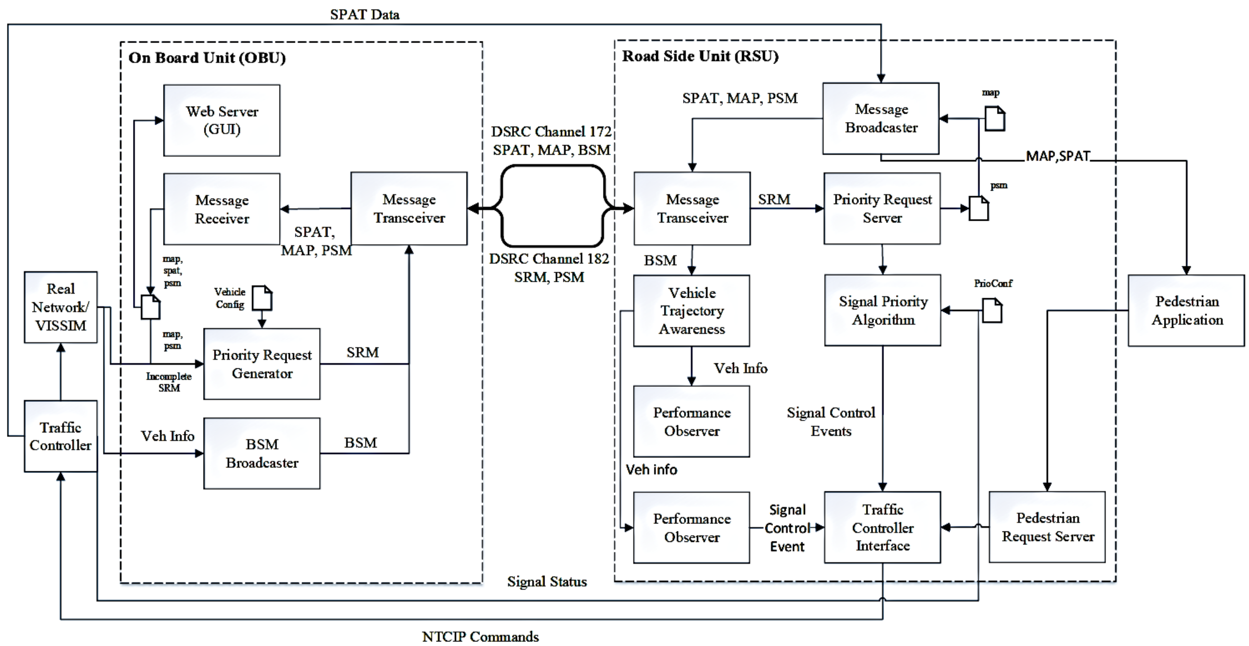

Figure 2. MMITSS Component Diagram [Source: University of Arizona]

For the SILS set up of the application, there are two interfaces between the Vissim simulation and the MMITSS-enabled Docker-based applications – (a) interface to communicate BSM messages to the MMITSS for Intelligent Signal Control and (b) interface to communicate "revised" signal timing back to the simulation.

These are explained below:

a. The BSM Distributor works by detecting vehicles within the vicinity of intersections and sending this data to that controller's Listener port as shown in Figure 4.
b. The Traffic Controller Interface interfaces the MMITSS application output to the traffic controller in the simulation. At a high-level, this component is responsible for receiving signal timing schedule from the controller (`MinGreen`, `MaxGreen`, `Yellow` and `Red` for all 8 phases) and send control commands back (`NTCIP: FORCE_OFF`, `VEH_CALL`, `PHASE_OMIT`, `PHASE_HOLD`).

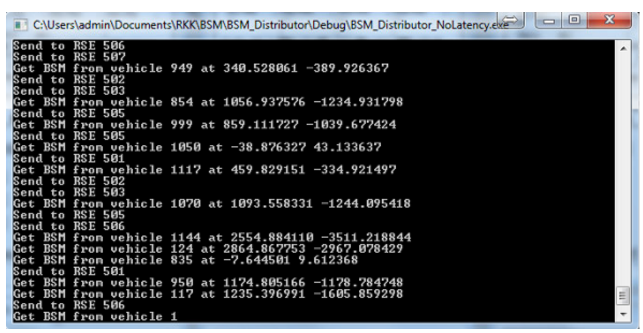

Figure 3. BSM Distributor distributing vehicle data to specific controller unit's RSE port.

Vissim Implementation Pseudocode:

Below is a step-by-step simplified pseudocode of how MMITSS is currently implemented in Vissim:

1. CV class of vehicles will use a "driver-behavior model" which basically communicates vehicle speed, position to a `PORT` as shown in Figure 4.
2. BSM Distributor will "listen" to this `PORT` and distributes the vehicles to multiple controller PORTS which are connected to specific MMITSS containers as shown in Figure 5.
3. Signal Controllers within Vissim are also connected to specific MMITSS container `PORTS` to read NTCIP commands and write the signal plans to them as shown in Figure 5.
4. Once simulation is initiated, steps 2 and 3 will contribute enough data to MMITSS containers for them to generate commands for changing the signal phases in a desired way as shown in Figure 6.

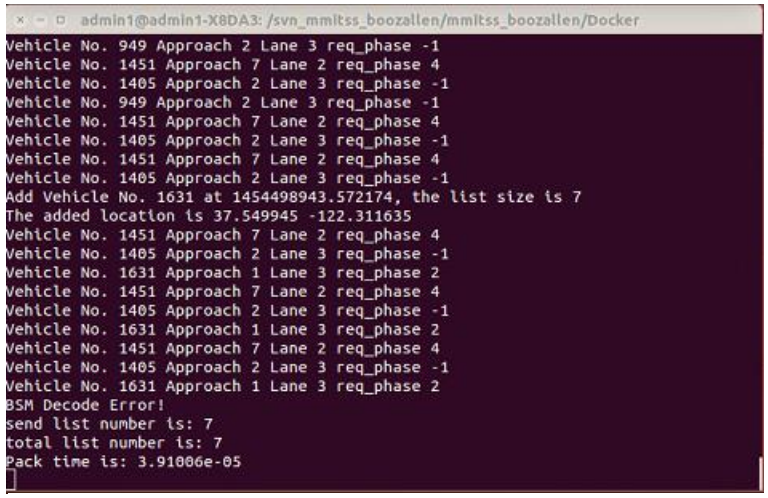

Figure 4. Docker Container Receiving Vehicle Data

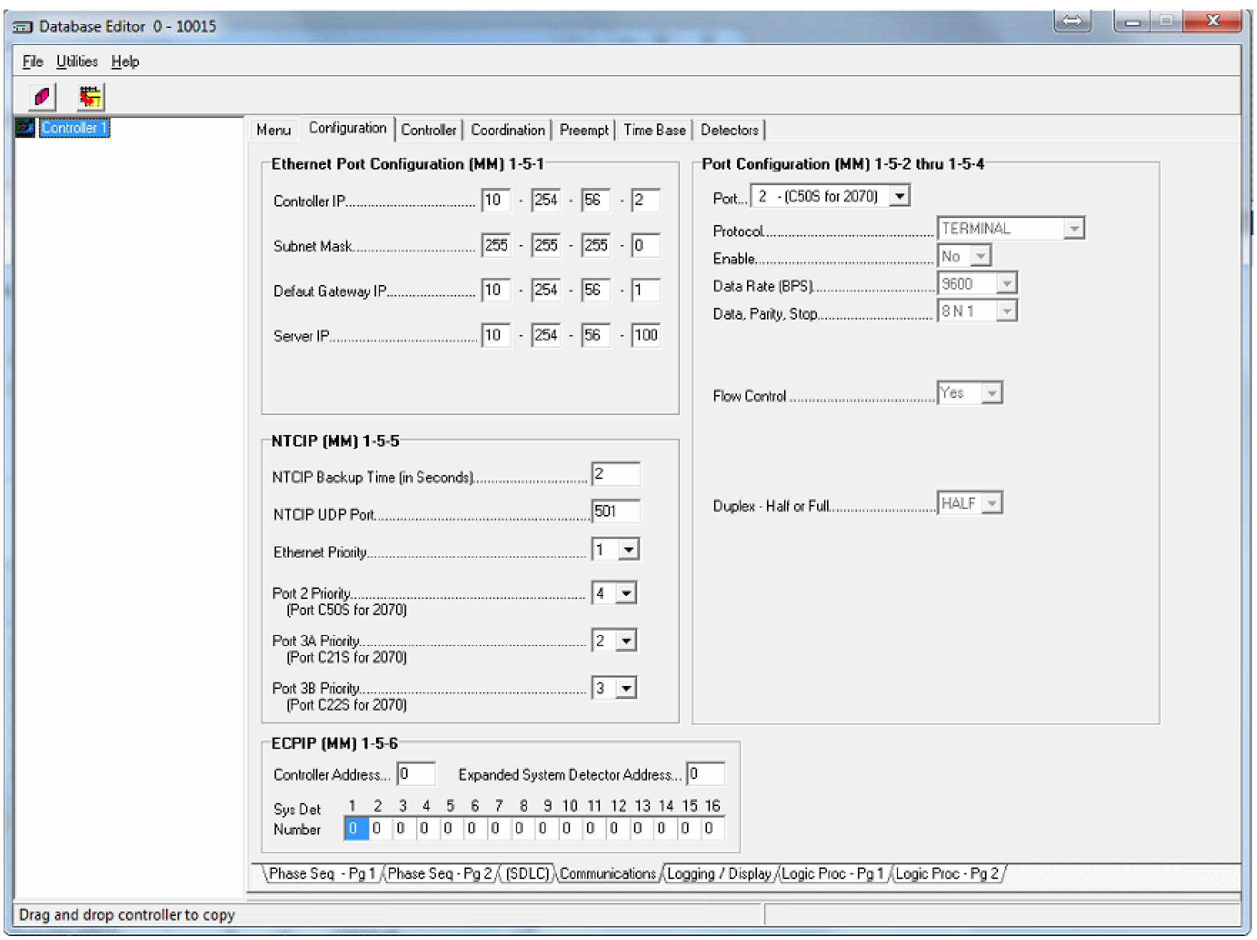

Figure 5. Setting up Controllers to Read/Write to a Port

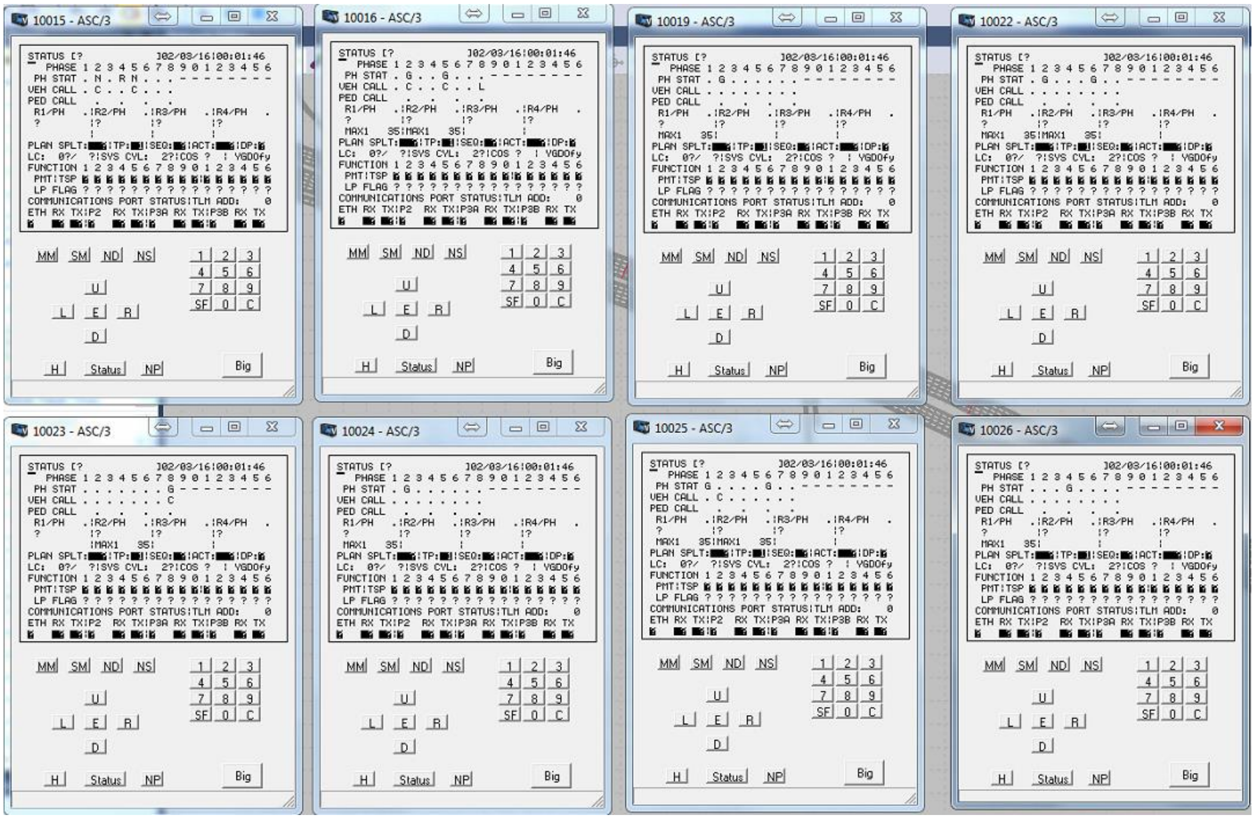

Figure 6. Virtual Controllers Initiated for each Intersection

MMITSS DATA REQUIREMENTS

This section is synthesized based on an internal memo with MMITSS Application's Principal Investigator Larry Head (University of Arizona). MMITSS Application requires 2 types of data from the simulation – (a) Vehicle Data and (b) Infrastructure Data. These are listed in detail below:

Vehicle Data:

1. **Each connected vehicle must produce BSM data at a frequency of 10 Hz**
   a. **BSM data based on DSRC SAE J2735 standards.**
   b. **BSM Position data must be accurate to 20 cm using the WGS-84 coordinate system**
   c. **BSM messages shall be constructed per SAE J2735 2009 message specification. [We should also consider moving to the new 2015 standard when it is approved].**
2. **BSM Data should be forwarded to a computer IP and port (e.g. 10.10.10.1 port 7754)**

Infrastructure Data

1. **MMITSS must be able to acquire, using NTCIP get requests, the following signal controller data:**
   a. **Signal timing parameters**
      i. **Ring structure (we assume dual ring, 8-phase)**
      ii. **Phase min**
      iii. **Phase max**
      iv. **Walk Interval**
      v. **Ped Clear Interval**
      vi. **Yellow Change Interval**
      vii. **Red Clearance Interval**
      viii. **Coordination plan parameters:**
         1. **Cycle Length**
         2. **Phase Splits**
         3. **Coordination Offset**

3. For priority/preemption purposes, special classes of vehicles (transit, truck, EV) shall produce a Signal Request Message when they are within the DSRC range of the intersection to be controlled according to the SAE Standards.	**b. Signal timing flags** **c. Current phase state** **d. Current interval state** **e. Current detector state** **i. Detector Enabled** **ii. Detector Data** **1. Volume** **2. Occupancy**

The MMITSS application will use this information to optimize signal phases. The output will be in the form of following NTCIP Objects:

a. Holds (Phase)
b. Force-Off (Phase)
c. Omit (Phase)
d. Phase call

NTCIP Objects Communicated by MMITSS

SIGNAL CONTROLLER DATA	TYPE	NTCIP OID
RING STRUCTURE	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.7.3.1.3
PHASE MIN GREEN	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.4.p
PHASE MAX GREEN	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.6.p
WALK INTERVAL	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.2.p
PED CLEAR INTERVAL	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.3.p
YELLOW CHANGE INTERVAL	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.8.p
RED CLEARANCE INTERVAL	Signal Parameter	1.3.6.1.4.1.1206.4.2.1.1.2.1.9.p
CYCLE LENGTH	Coordination	1.3.6.1.4.1.1206.4.2.1.4.7.1.2.1
SPLIT (PHASE)	Coordination	1.3.6.1.4.1.1206.4.2.1.4.9.1.3.1.p
OFFSET	Coordination	1.3.6.1.4.1.1206.4.2.1.4.7.1.3.1
CURRENT TIMING PLAN	Status	1.3.6.1.4.1.1206.3.5.2.1.22.0
CURRENT DETECTOR VOLUME	Status	1.3.6.1.4.1.1206.4.2.1.2.5.4.1.1.p
CURRENT DETECTOR OCCUPANCY	Status	1.3.6.1.4.1.1206.4.2.1.2.5.4.1.2.p
HOLD (PHASE)	Control	1.3.6.1.4.1.1206.4.2.1.1.5.1.4.1
FORCE-OFF (PHASE)	Control	1.3.6.1.4.1.1206.4.2.1.1.5.1.5.1
OMIT (PHASE)	Control	1.3.6.1.4.1.1206.4.2.1.1.5.1.2.1
PHASE CALL	Control	1.3.6.1.4.1.1206.4.2.1.1.5.1.6.1