Proceedings of the American Control Conference
Arlington, VA June 25-27, 2001

Meso-Microscale Traffic Simulation of an AHS Control Architecture*

Gabriel Gomes†, Laura Muñoz‡, Jingang Yi§, Charmaine Toy¶, Sara Cinnamon‖, Roberto Horowitz**, Luis Alvarez††

†,‡,§,‖,** *Department of Mechanical Engineering, University of California, Berkeley, CA 94720-1740, USA*
¶ *DiCon Fiberoptics, Inc., 1331 Eighth Street, Berkeley, California 94710, USA*
†† *Instituto de Ingeniería Universidad Nacional Autónoma de México, 04510 Coyoacán DF, México*

Abstract

A recently developed meso-microscale traffic simulator allows a stationary region of microsimulation to be defined within a larger, mesosimulated Automated Highway System (AHS). This simulator permits analysis of traffic behavior in situations where both microscopic (vehicle-level) and mesoscopic (aggregate flow) effects are important, while avoiding the prohibitive computational cost of microsimulating a large-scale AHS. This paper describes the software structure of the meso-micro simulator, which implements the PATH AHS control architecture [4, 5]. Simulation results indicate that the meso-micro simulator generates consistent traffic flows across both meso and microsimulated regions.

1 Introduction

Simulation tools available for analyzing highway traffic behavior, and specifically AHS, generally fall into two categories, according to their level of modeling detail: *microscale* and *mesoscale* simulators.

Mesoscale simulators are usually based on fluid-like models of traffic flow, applied to average traffic characteristics such as density, flow, and velocity. They do not provide information about particular vehicles but are instead capable of simulating large highway networks. Microscale simulators are based on models of individual vehicle dynamics, where the behavior of each vehicle depends on its locally perceived environment.

Currently, the mesosimulation software used by the California PATH program for analyzing AHS is SmartCAP [1]. This software, based on the flow theory introduced in [2], uses a conservation model of traffic density, coupled with rules of traffic flow behavior given by a finite set of vehicle *activities* (e.g. join, cruise, change lane). SmartCAP allows monitoring of aggregate traffic quantities in an AHS subdivided into sections of approximately 0.5 miles.

The highway microsimulation software currently under development at PATH is SmartAHS. This software package is based on the hybrid systems language SHIFT [3]. SmartAHS was designed to demonstrate several AHS concepts, including the PATH hierarchical control architecture [4, 5] described in Sec. 2.

In order to simultaneously capture the occurrence of small scale events and their impact on the larger highway, multiple degrees of modeling abstractions are needed. Traffic simulation frameworks that enable multiple levels of resolution are not commonplace; however, there are many traffic scenarios that are characterized by both mesoscopic and microscopic effects. For example, in an AHS, it is important to study: 1) the impact of fault detection and management strategies at vehicle and roadside levels on the overall AHS performance, 2) onramp metering control schemes where rates of entrance are a function of the overall state of the highway, but are also influenced by the ability of each vehicle to complete an *entry* maneuver.

Although it is in principle possible to microsimulate the entire AHS, the computational cost is prohibitively high, if the interest is in the detailed behavior of AHS components: radars, vehicle-to-vehicle and roadside-to-vehicle communications, maneuver controllers and protocols, etc. This increased complexity severely compromises the number of vehicles that can be included in a simulated experiment. Recent trials with SmartAHS place its capabilities in the range of 30 vehicles with a 25/1 real-to-simulation time ratio on a 4 processor Sun Sparc Ultra-II computer. Furthermore, in many cases, such as the previous examples, it may only be necessary to examine selected portions of the AHS at the microscopic scale, while mesoscale simulation is sufficient for

*Research supported by UCB-ITS PATH grant MOU-383.
†Graduate student; Email: gomes@me.berkeley.edu.
‡Graduate student; Email: lmunoz@me.berkeley.edu.
§Graduate student; Email: jgyi@me.berkeley.edu.
¶Automation Engineer; Email: ctoy@diconfiber.com.
‖Undergraduate; Email: azulo@uclink4.berkeley.edu.
**Professor and corresponding author; Email: horowitz@me.berkeley.edu.
††Professor; Email: alvar@pumas.iingen.unam.mx.

the remainder of the highway.

This paper describes development of a simulation package, in which SmartCAP and SmartAHS are combined to allow representation of microscopic and mesoscopic traffic and efficient simulation of all control layers of the AHS.

2 Overview

The hierarchical PATH AHS control architecture described in [4, 5] divides the task of controlling vehicles in an AHS among 5 layers. The upper levels of the architecture (network and link) are devoted to route planning and flow management across large expanses of highway. The lower levels (coordination, regulation, and physical) exist onboard each vehicle, and are responsible for vehicle-based communications, maneuver selection, motion control, and actuation.

The integrated environment consists of an AHS network, simulated by SmartCAP, with a microsimulated *μ-window*, maintained by SmartAHS, which may cover one or more SmartCAP sections (see Fig. 1a). The size and location of the μ-window are user-defined, and the μ-window is stationary with respect to the highway. Within the μ-window, SmartAHS-simulated vehicles interact using full coordination, regulation and physical layers, and communicate with roadside link layer components. The link layer is present in both mesoscopic and microscopic regions.

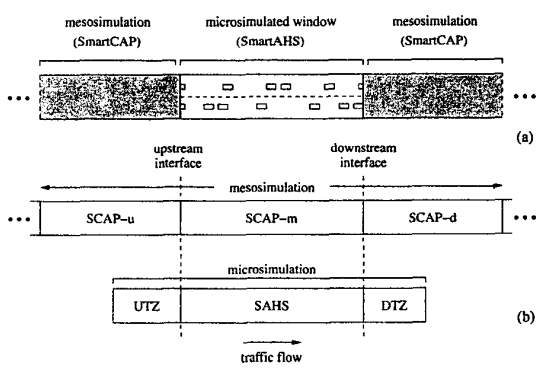

Figure 1: (a) Conceptual drawing and (b) main topological elements of meso-micro simulator

This article describes all aspects of the design and implementation of the integrated simulation software, which include:

- **Upstream interface**: The task at this interface is to produce a stream of microsimulated vehicles in a way that preserves the flow of each vehicle type (O/D pair) given by the mesosimulation. The methodology described in section 3 employs an additional section called the *upstream transition zone* (UTZ) (see Fig. 1b), where platoons of vehicles are generated in correspondence with upstream mesosimulation conditions.

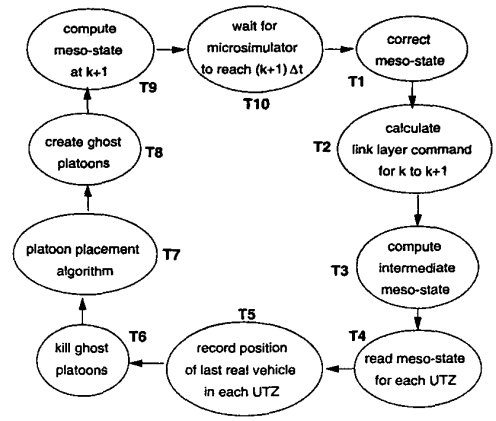

Figure 2: Scheduler

- **Downstream interface**: This interface converts microsimulated vehicles into mesoscopic traffic flow. It makes use of a *downstream transition zone* (DTZ) (see Fig. 1b) that allows backward propagating conditions from downstream mesosimulated sections to enter the μ-window. The average characteristics (e.g., velocity, flow rate) of the exiting microsimulated vehicles are used as inlet parameters for the flow in the downstream SmartCAP region.
- **Environment consistency**: Agreement of the highway state in both simulation environments is guaranteed by upstream and downstream *consistency checks* that adjust the meso-state in sections immediately before and after the μ-window (i.e. SCAP-u, SCAP-d in Fig. 1b), whenever discrepancies arise in the meso and microscale flows across the boundaries.
- **Scheduler**: The scheduler integrates and supervises the components of the meso-micro simulator. It is a finite state machine which coordinates the evolution of the traffic state in the micro and mesosimulated regions, and handles environment interfacing tasks such as the *platoon placement* algorithm and the consistency checks.

Results from simulation runs are presented and demonstrate the functionality of the integrated software.

3 Software Design

The time intervals used by SmartCAP and SmartAHS are very different. While on-board vehicle control laws and communication components require intervals of approximately 0.05 sec, the mesosimulation typically uses intervals of 15 to 20 sec. Thus, the majority of meso-micro simulation effort takes place in SmartAHS, with a

periodic update and correction of mesoscopic values and corresponding boundary conditions. At the end of each meso-time interval the following tasks are performed:

- The mesoscopic flows across upstream interfaces, which are *real* numbers, are converted to integer numbers of vehicles under a residual bounding scheme.
- The microscopic flows across upstream and downstream boundaries, registered during the previous time step, are used to correct the present meso-state.
- The meso-state is updated (i.e. projected into the next time interval).

3.1 Scheduler

The simulation is coordinated by a SHIFT component called the *scheduler* (Fig. 2). The scheduler tasks are:

T1: At the end of the meso-time interval (Δt), values for the actual flows of microsimulated vehicles across each of the interfaces are available. These may differ from the values used by SmartCAP to compute the present meso-state, and are used to correct the error (Sec. 3.5).

T2: Using the corrected highway state, the link layer controller generates velocity and activity plans for the upcoming interval.

T3: The activity plan determines the intermediate meso-state Eq. (1).

T4: Intermediate densities and velocities are copied to the SmartAHS environment.

T5: The position of the real vehicle farthest upstream is recorded, for safe spacing purposes.

T6: Ghost platoons remaining in the UTZ are removed (sec. 3.3.2).

T7: The intermediate meso-state is used to generate an arrangement of vehicles in the UTZ (sec. 3.3.2)

T8: Ghost platoons are created with initial condition given by the previous step.

T9: The meso-state is updated for the next time step (Eq. (2)).

T10: SmartAHS continues for one meso-time step.

3.2 SmartCAP dynamics

The mesoscopic state of the highway is composed of the following variables:

$n^c(i,j,k)$ ... Number of vehicles of type j, in section i, at time $k\Delta t$
$n^c_f(i,j,k)$ ... Number of followers
$n^c_l(i,j,k)$ ... Number of leaders
$v^c(i,k)$ ... Average traffic speed in section i, during time interval $[k\Delta t, (k+1)\Delta t]$
$\pi^c_m(i,j,k)$... Proportion of $n^c(i,j,k)$ performing maneuver m during $[k\Delta t, (k+1)\Delta t]$
$APS(i,k)$.. Average size of platoons in section i at time $k\Delta t$

where
$$n^c(i,j,k) = n^c_f(i,j,k) + n^c_l(i,j,k)$$
$$APS(i,k) = \frac{\sum_j n^c(i,j,k)}{\sum_j n^c_l(i,j,k)}$$

The superscript ^{c} is used to distinguish these from the analogous variables in SmartAHS. Beginning each meso-time interval, the link layer controller issues activity and velocity *plans* $(\pi^c_m(i,j,k), v^c(i,k))$, which are both feasible and can be safely completed by the end of the interval, and within each section.

The assumption that vehicles can complete their maneuvers within the section is key to the design of the upstream interface because it allows us to disregard the possibility of vehicles entering the μ-window in mid-maneuver.

SmartCAP variables are updated in 2 steps:
Step 1: Intermediate meso-state. All maneuvers are completed with zero velocity (**T3**). We denote the intermediate variable with a bar and assume, for simplicity of notation, that *join* and *split* are the only maneuvers (lane change maneuvers are treated by coupling these equations among adjacent lanes).

$$\begin{aligned}\bar{n}^c_l(i,j,k) &= n^c_l(i,j,k)[1 + \pi^c_{split}(i,j,k) - \pi^c_{join}(i,j,k)]\\ \bar{n}^c_f(i,j,k) &= n^c_f(i,j,k)[1 - \pi^c_{split}(i,j,k) + \pi^c_{join}(i,j,k)]\end{aligned} \quad (1)$$

Step 2: Vehicles flow downstream (**T9**).

$$\begin{aligned}n^c_l(i,j,k+1) &= \bar{n}^c_l(i,j,k)(1 - \tau(i,k)) + \bar{n}^c_l(i-1,j,k)\tau(i-1,k)\\ n^c_f(i,j,k+1) &= \bar{n}^c_f(i,j,k)(1 - \tau(i,k)) + \bar{n}^c_f(i-1,j,k)\tau(i-1,k)\end{aligned} \quad (2)$$

where
$$\tau(i,k) \triangleq \frac{v^c(i,k)\Delta t}{L_i} \qquad L_i \ldots \text{section lengths}$$

3.3 Upstream Interface

Vehicles crossing the upstream boundary are influenced by traffic both inside and outside the μ-window. Their nominal speed is dictated by the upstream section, but may be diminished if traffic is congested inside the μ-window. Furthermore, the backward propagation speed of congestion is determined by the chosen AHS regulation layer control laws. The *upstream transition zone* (UTZ) is designed to make the meso-micro simulator independent of the particular car-following model being used. Vehicles entering the μ-window during the next time step are placed inside the UTZ with initial densities and velocity given by the intermediate meso-state

and an arrangement given by the platoon placement algorithm. While in the UTZ, vehicles are not allowed to perform any maneuvers, since these are assumed to be completed in the intermediate meso-state. If entry into the μ-window is not impeded by congestion, the flow measured across the upstream boundary in both meso and micro environments should be approximately equal, with differences due to rounding and randomness in the initial vehicle distribution. These differences are cancelled by the meso-state correction task (**T1**). Vehicles in the UTZ are of a simple type called the *ghost platoon*, which lacks many of the AHS vehicle components. A description of this SHIFT type is given below.

3.3.1 Ghost Platoons: The primary reason for using a specialized vehicle type in the UTZ, instead of the fully-equipped SmartAHS vehicle type, is computational efficiency. The total number of vehicles simulated is reduced by replacing each platoon with a single rigid ghost platoon. Another reason is that there is no easy way in the SHIFT language to delete the subset of vehicles remaining in the UTZ at the end of the meso-time interval (**T6**).

A single ghost-platoon component contains all the information needed to create a platoon of real vehicles (i.e. number, type and order of vehicles in the platoon), but unlike a standard SmartAHS vehicle, a ghost platoon lacks detailed regulation and coordination layers. It travels along the UTZ, controlled by the standard regulation layer *leader* law, and materializes when it enters the μ-window. Because it is important to avoid maneuvers taking place inside the UTZ, leaders are not allowed to request maneuvers until their entire platoon has crossed the boundary. Congestion inside the μ-window is manifested as a decrease in the expected flow of vehicles across the upstream boundary, and is propagated to the mesosimulation by the consistency check.

3.3.2 Platoon Placement Algorithm: This algorithm translates the upstream mesosimulated densities into a set of discrete SmartAHS vehicles, in a way that preserves the number of leaders and followers and the average platoon sizes. The algorithm also produces initial positions for each platoon within the UTZ, and a randomized arrangement of vehicles within each platoon.

The main steps of the algorithm are:

1. Get SmartCAP values and check for space: Using the number of leaders and followers from the intermediate SmartCAP state $(\bar{n}^c_l(i,j,k), \bar{n}^c_f(i,j,k))$ and the velocity $(v^c(i,k))$, integer numbers of leaders and followers $(n^a_l(i,j,k), n^a_f(i,j,k))$, and initial positions for each ghost platoon are generated. The algorithm also checks that sufficient space is available in the UTZ to accommodate the calculated number of leaders and followers.
2. Platoon sizes: The average platoon size for the intermediate state is:

$$APS(i,k) = \frac{\sum_j \left(\bar{n}^c_f(i,j,k) + \bar{n}^c_l(i,j,k)\right)}{\sum_j \bar{n}^c_l(i,j,k)}$$

A total of $P = \sum_j n^a_l(i,j,k) + n^a_f(i,j,k)$ vehicles must be arranged in $\sum_j n^a_l(i,j,k)$ platoons, with an average size approximating $APS(i,k)$. Defining $\bar{APS}(i,k) = \text{trunc}(APS(i,k))$, the following rule is used to choose a platoon size PS_n for the nth platoon in the set:

$$PS_n = \begin{cases} \bar{APS}(i,k) & P - \sum_{j=1}^{n-1} PS_j \geq \bar{APS}(i,k) \\ P - \sum_{j=1}^{n-1} PS_j & 0 \leq P - \sum_{j=1}^{n-1} PS_j < \bar{APS}(i,k) \\ 0 & P - \sum_{j=1}^{n-1} PS_j = 0 \end{cases}$$

3. Vehicle arrangement in platoons: Once a size has been determined for each platoon, vehicle types are selected to create the platoons. The first vehicle in the platoon is randomly chosen from the leader types, and the rest from the pool of followers.
4. Ghost platoon spacing: once platoon size and vehicle arrangements have been determined, initial positions for each platoon within the UTZ are calculated. The position of the trailing real vehicle inside the UTZ determines the amount of space available for placing ghost platoons. The space required by ghost platoons is the sum of the physical length of vehicles and the headways required by leaders and followers to ensure safety. This space is subtracted from the total available space to calculate the free space. This amount is then distributed randomly among platoons.

3.4 Downstream Interface

At the downstream interface, vehicles are phased out of the microsimulation and transformed into an equivalent mesoscopic flow. The transformation is carried out in a smooth, computationally efficient manner by replacing microscale vehicles with *ghost vehicles*. Ghost vehicles are used in order to:

1. increase simulation speed, by reducing the number of SmartAHS components in simulation, as soon as vehicles exit the μ-window.
2. provide continuity between micro and mesosimulated regions. The ghost vehicles track the average velocity of the downstream mesosimulated flow. In addition, the positions and velocities of ghost vehicles are detected by the radar of vehicles in the μ-window. Since the ghost vehicles travel at the same speed as the downstream SmartCAP flow, and the

vehicles in the μ-window maintain safe spacing behind the ghost vehicles, downstream flow conditions (e.g. traffic jams) can be propagated backward into the μ-window.

The process of removing microsimulated vehicles from the simulation environment is illustrated by the example of Fig. 3. In the top drawing, vehicle $v1$ has exited the μ-window and has just crossed into the DTZ. The stages that follow are:

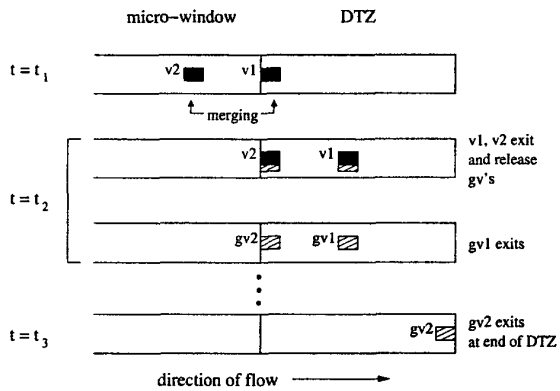

Figure 3: Example: downstream interface

1. $v1$ travels downstream through the DTZ until all its platoon members and/or upstream maneuver partners are inside the DTZ. An upstream maneuver partner of $v1$ is any vehicle upstream of $v1$ that is currently engaged in a pairwise maneuver with $v1$. In Fig. 3, $v2$ is merging with $v1$, thus $v2$ is the upstream maneuver partner of $v1$. If $v1$ were to disappear, with $v2$ still inside the μ-window, this would cause referencing errors for $v2$. Thus, the maneuver pair is kept intact until it crosses into the DTZ, where both partners can be removed safely and simultaneously from the microsimulation.
2. When both $v1$ and $v2$ are inside the DTZ, they spawn *ghost vehicles* ($gv1$ and $gv2$) before deleting themselves from the simulation environment. $gv1$ disappears, and $gv2$ remains, to avoid disrupting the car-following behavior of vehicles in the μ-window. These events take place within timestep t_2.
3. $gv2$ travels along the DTZ until (a) it reaches the downstream end, as in Fig. 3, or (b) another ghost appears upstream in the DTZ. In either case, $gv2$ is removed from the simulation.

Flow characteristics are preserved across the downstream micro-meso boundary. The average velocity and flow rate of vehicles exiting the μ-window are used as inlet parameters for the downstream mesosimulated region.

3.5 Enforcing Consistency

We introduce here a set of corrections to the SmartCAP state, based on differences in flows across the interfaces, registered during the meso-time interval. There are three sources of discrepancy:

- The conversion of real densities to integer numbers of vehicles.
- The SmartCAP assumption that density is distributed uniformly within each section.
- The SmartCAP speed in the UTZ is calculated with the average density inside the μ-window. However, localized fluctuations in density may cause the actual flow of vehicles across the boundary to be different.

One of the important phenomena that we attempt to reproduce is the backward propagation of congestion, over the upstream and downstream boundaries. In order to guarantee that the speed of propagation is preserved, flows across the boundaries must agree in both environments, for each vehicle type. The actual numbers of vehicles that cross the boundaries is only known at the end of the time interval. These are denoted:

$\phi^{cu}(u,j,k)$ # veh. crossed upst. boundary (SmartCAP)
$\phi^{cd}(d,j,k)$ # veh. crossed downst. boundary (SmartCAP)
$\phi^{au}(u,j,k)$ # veh. crossed upst. boundary (SmartAHS)
$\phi^{ad}(d,j,k)$ # veh. crossed downst. boundary (SmartAHS)

3.5.1 Upstream correction: The upstream correction adjusts the velocities and densities in sections prior to the μ-window, for the time interval $[(k-1)\Delta t, k\Delta t]$. Velocity is modified so that the total flow entering the μ-window is equal in both environments, while the density of each vehicle type is adjusted to match individual flows.

The following density correction is applied for every vehicle type j:

$$n^c(u,j,k) \mathrel{+}= \bar{n}^c(u,j,k-1)\tau(u,k-1) - \phi^{au}(u,j,k-1)$$
$$n_l^c(u,j,k) \mathrel{+}= \bar{n}_l^c(u,j,k-1)\tau(u,k-1) - \phi_l^{au}(u,j,k-1)$$
$$n_f^c(u,j,k) \mathrel{+}= \underbrace{\bar{n}_f^c(u,j,k-1)\tau(u,k-1)}_{\phi^{cu}\text{'s}} - \phi_f^{au}(u,j,k-1)$$

Velocity for the previous time step is recalculated as follows:

$$v^c(u,k-1) = \frac{\sum_j \phi^{au}(u,j,k)}{\sum_j \bar{n}^c(u,j,k-1)} \frac{L(u)}{\Delta t}$$

A consequence of this correction method is that time variations in ϕ^{au} (or $\bar{n}^c$) may greatly affect v^c, the velocity upstream of the μ-window (see § 4). Development of a more robust correction strategy is a subject of ongoing work.

3.5.2 Downstream correction: Densities and velocities inside the μ-window are simply copied from the micro to the meso environments. This creates a problem for canceling the flow error out of the μ-window: the definition of flow out of a section in SmartCAP depends only on density and velocity within that section. Thus, the state of the highway in the μ-window fixes both outgoing flows. We resolve the dilemma by violating the assumption of uniform distribution of densities in SmartCAP, and adjust downstream densities as shown below. Notice that in this case, no velocity correction is needed for downstream sections.

For every vehicle type j:

$$n^c(d,j,k) \mathrel{+}= -\bar{n}^c(d,j,k\text{–}1)\mathcal{T}(u,k\text{–}1) + \phi^{ad}(d,j,k\text{–}1)$$
$$n_l^c(d,j,k) \mathrel{+}= -\bar{n}_l^c(d,j,k\text{–}1)\mathcal{T}(u,k\text{–}1) + \phi_l^{ad}(d,j,k\text{–}1)$$
$$n_f^c(d,j,k) \mathrel{+}= \underbrace{-\bar{n}_f^c(d,j,k\text{–}1)\mathcal{T}(u,k\text{–}1)}_{\phi^{cd}\text{'s}} + \phi_f^{ad}(d,j,k\text{–}1)$$

4 Simulation Results

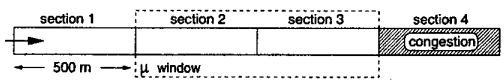

Figure 4: Highway geometry for simulation test

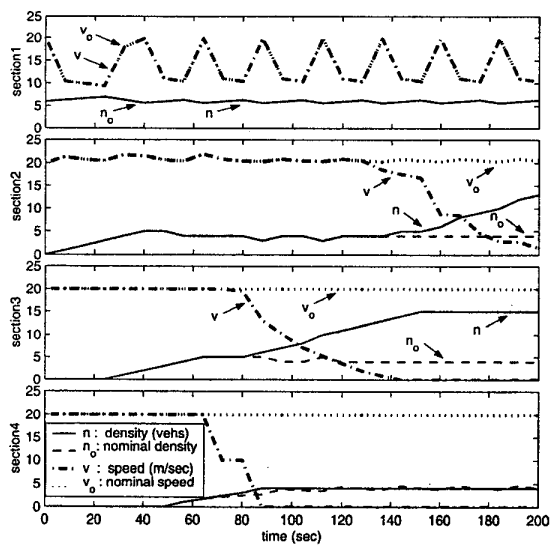

Figure 5: Propagation of congestion into micro-window

Fig. 5 shows the results of a test run designed to illustrate the propagation of a backward moving wave of congestion into the μ-window. The experiment was performed on the one-lane, four section stretch of highway shown in Fig. 4, with split, join and lane change maneuvers disabled. Congestion was mimicked in section 4 with a velocity profile that drops from 20 m/s to 0 m/s starting at t=62 seconds. When compared to the nominal uncongested case, the downstream drop in speed clearly triggered a backward moving wave which reached μ-window sections 2 and 3 at t=80 and t=130 seconds. After the congestion wave passed, the μ-window densities were seen to increase while the velocities decreased. As shown in the topmost plot of Fig. 5, there are fluctuations in the velocity of the section immediately upstream of the μ-window; this is expected due to large variations in ϕ^{au}. In this case, the average density of 6 veh/section and expected velocity of 20 m/s yield a predicted flow (ϕ^{cu}) of 1.92 vehicles per 8 sec. time interval into the μ-window. The alternating low and high speeds correspond to actual inlet flows (ϕ^{au}) of 1 and 2 vehicles per 8 sec., respectively. Future versions of the code will include a modified upstream velocity correction that produces more realistic upstream flows.

5 Conclusions

The integrated meso-microscale simulator incorporates a stationary "window" of microsimulation into a larger mesosimulated AHS, allowing traffic to be represented simultaneously at both the mesoscopic and microscopic levels. The main components of the software architecture have been described in this paper. This simulator is appropriate for testing AHS policies, such as entrance control and fault-handling, that are distinguished by both meso and microscopic phenomena. Future work will involve development and more extensive testing of the meso-micro simulator for a variety of highway topologies and traffic conditions.

References

[1] M. Broucke, P. Varaiya, M. Kourjankski, and D. Khorramabadi. Smartcap User's Guide. Technical report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, May 1996.

[2] Mireille Broucke and Pravin Varaiya. A theory of traffic flow in automated highway system. *Transportation Research, Part C: Emerging Technologies*, 4(4):181–210, 1996.

[3] Akash R. Deshpande, Aleks Göllü, and Luigi Semenzato. SHIFT *Reference Manual.* California PATH, Berkeley, CA, 1997.

[4] P. Varaiya. Smart Cars on Smart Roads: Problems of Control. *IEEE Transactions on Automatic Control*, 38(2):195–207, 1993.

[5] P. Varaiya and Steven E. Shladover. Sketch of an IVHS systems architecture. Technical Report UCB-ITS-PRR-91-3, Institute of Transportation Studies, University of California, Berkeley, 1991.