Microsimulator *(version 4.0)*

Revision History
1/8/2010 Edited by AECOM Consult, Inc.
4/20/2010 Edited by RSG, Inc.

The **Microsimulator** program is used to:

1. Simulate the second-by-second movements of vehicles and persons through the network.
2. Generate performance statistics, track individual travelers, and summarize events.

Microsimulator is a console-based program that runs in a command window on either Windows or Linux. The command syntax is:

Microsimulator [-flag] [control_file]

The control_file is the file name of an ASCII file that contains the control strings expected by the program. The control_file is optional. If a file name is not provided, the program will prompt the user to enter a file name. The flag parameters are also optional. Any combination of the following flag parameters can be included on the command line:

-Q[uiet] = execute without screen messages
-H[elp] = show program syntax and control keys
-K[eyCheck] = list unrecognized control file keys
-P[ause] = pause before exiting
-N[oPause] = never pause before exiting
-B[atch] = execute in batch processing mode

The program automatically creates a printout file based on the control_file name. If the file name includes an extension, the extension is removed and ".prn" is added. The printout file will be created in the current working directory and will overwrite an existing file with the same name.

Control File Parameters

Control parameters are defined using a control key followed by a string or number. The control parameters can be specified in any order. If a given key is defined more than once, the last instance of the key is used. The default value for each key is 0 or "Null". Null parameters do not need to be included in the file. Note that comment lines or extraneous keys can be included in the file. They will be ignored by the program.

A typical **Microsimulator** control file is shown below:

```
PROJECT_DIRECTORY                d:\software\test\case3
NET_DIRECTORY                    d:\software\test\case3
NET_NODE_TABLE                   Node
```

```
NET_LINK_TABLE                  Link
NET_POCKET_LANE_TABLE           Pocket_Lane
NET_PARKING_TABLE               Parking
NET_LANE_CONNECTIVITY_TABLE     Lane_Connectivity
NET_ACTIVITY_LOCATION_TABLE     Activity_Location
NET_PROCESS_LINK_TABLE          Process_Link
NET_UNSIGNALIZED_NODE_TABLE     Unsignalized_Node
NET_SIGNALIZED_NODE_TABLE       Signalized_Node
NET_TIMING_PLAN_TABLE           Timing_Plan
NET_PHASING_PLAN_TABLE          Phasing_Plan
NET_DETECTOR_TABLE              Detector
NET_SIGNAL_COORDINATOR_TABLE    Signal_Coordinator

VEHICLE_FILE                    Vehicle.txt
VEHICLE_TYPE_FILE               VehType.txt

PLAN_FILE                       TimePlan.txt
NODE_LIST_PATHS                 Yes

CELL_SIZE                       7.5
TIME_STEPS_PER_SECOND           1
TIME_OF_DAY_FORMAT              24_HOUR_CLOCK
SIMULATION_START_TIME           0:00
SIMULATION_END_TIME             27:00

SPEED_CALCULATION_METHOD        CELL-BASED

PLAN_FOLLOWING_DISTANCE         525
LOOK_AHEAD_TIME_FACTOR          1.0
LOOK_AHEAD_LANE_FACTOR          4.0
LOOK_AHEAD_DISTANCE             260
ENFORCE_PARKING_LANES           Yes

DRIVER_REACTION_TIME            0.7
PERMISSION_PROBABILITY          50.0
RANDOM_NUMBER_SEED              1122687672

MINIMUM_WAITING_TIME            120
MAXIMUM_WAITING_TIME            3600
MAX_ARRIVAL_TIME_VARIANCE       60
MAX_DEPARTURE_TIME_VARIANCE     60

NEW_PROBLEM_FILE                MsimProblem.txt
NEW_PROBLEM_FORMAT              VERSION3
MAX_SIMULATION_PROBLEMS         100000

OUTPUT_TRAVELER_RANGE_1         158501
OUTPUT_TRAVELER_FILE_1          Traveler.txt
OUTPUT_TRAVELER_FORMAT_1        TAB_DELIMITED

OUTPUT_SNAPSHOT_FILE_1          Snapshot.txt
OUTPUT_SNAPSHOT_INCREMENT_1     900
OUTPUT_SNAPSHOT_TIME_RANGE_1    0..86400

OUTPUT_SUMMARY_FILE_1           LinkDelay.txt
OUTPUT_SUMMARY_FORMAT_1         TAB_DELIMITED
OUTPUT_SUMMARY_INCREMENT_1      900
OUTPUT_SUMMARY_TIME_RANGE_1     0..86400
OUTPUT_SUMMARY_TURN_FLAG_1      YES

OUTPUT_PROBLEM_TYPE_1           WAIT_TIME
```

```
OUTPUT_PROBLEM_FILE_1              Problem_Link.txt
OUTPUT_PROBLEM_FILTER_1            100
OUTPUT_PROBLEM_INCREMENT_1         3600
OUTPUT_PROBLEM_TIME_RANGE_1        0..86400

OUTPUT_SYSTEM_EVENT_TYPE_1         TIMING_CHANGE, PHASE_CHANGE
OUTPUT_SYSTEM_EVENT_FILE_1         Timing.txt
OUTPUT_SYSTEM_EVENT_TIME_RANGE_1   8:30..9:00
OUTPUT_SYSTEM_EVENT_NODE_RANGE_1   100201, 20033

OUTPUT_EVENT_TYPE_1                START_TIME, END_TIME
OUTPUT_EVENT_FILE_1                Event.txt
OUTPUT_EVENT_FILTER_1              60
```

This example simulates time sorted plans from the "TimePlan.txt" file. The program generates output data for traveler 158501; a snapshot of vehicle positions every 15 minutes; 15 minute link summaries in the "Link Delay.txt" file in tab delimited format with turning penalties; links with 100 or more Wait_Time problems in one hour; and Start_Time and End_Time event summaries.

The keys recognized by the **Microsimulator** program are listed below. These keys can be defined in a variety of different ways to perform different tasks.

TITLE

Any text string can be used on this line. This text is printed on the top of each output page.

REPORT_FILE

The report file name is optional. If a file name is not provided, the program automatically creates a report file name based on the input control file name. The report file will overwrite an existing file with the same name if the Report Flag key is False or not specified.

REPORT_FLAG

The report flag key is optional. If it is specified as Yes or True, the report file or default printout file will be opened in "Append" mode rather than "Create" mode. This permits the user to consolidate the output of several programs into a single report file.

MAX_WARNING_MESSAGES

When the program generates a warning message, a counter is incremented and the total number of warning messages is reported and a warning return coded (2) is set at the end of the execution. By default the program prints up to 100,000 warning messages to the print-out file. If more than 100,000 warning messages are sent, the program stops printing additional messages to the file or terminates the program with an error message based on the MAX_WARNING_EXIT_FLAG. This parameter enables the user to modify the default warning limit.

MAX_WARNING_EXIT_FLAG

If the maximum number of warning messages is exceeded, this flag directs the program in what to do. If the flag is TRUE (the default), the program is terminated with an error message about the warning messages. If the flag is FALSE, the program continues execution, but no additional

warning messages are sent to the screen or written to the printout file. The warning message counter continues to count the messages and reports the total at the end of the execution.

PROJECT_DIRECTORY

The project directory key is not required. If it is specified, it is added to all non-network file names required by the program. If it is not specified, all non-network file names should fully specify the file path.

NET_DIRECTORY

The network directory key is not required. If it is specified, it is added to all network table names. If it is not specified, the network table names should fully specify the file path.

NET_NODE_TABLE

The node table key is required. It specifies the name of the TRANSIMS node file within the network directory. The full path and file name for the node table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_LINK_TABLE

The link table key is required. It specifies the name of the TRANSIMS link file within the network directory. The full path and file name for the link table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_POCKET_LANE_TABLE

The pocket lane table key is required. It specifies the name of the TRANSIMS pocket lane file within the network directory. The full path and file name for the pocket lane table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_LANE_CONNECTIVITY_TABLE

The network lane connectivity table key is required. It specifies the name of the TRANSIMS lane connectivity file within the network directory. The full path and file name for the lane connectivity table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_PARKING_TABLE

The network parking table key is required. It specifies the name of the TRANSIMS parking table file within the network directory. The full path and file name for the parking table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_ACTIVITY_LOCATION_TABLE

The network activity location table key is required. It specifies the name of the TRANSIMS activity location file within the network directory. The full path and file name for the activity location table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_PROCESS_LINK_TABLE

The network process link table key is required. It specifies the name of the TRANSIMS process link file within the network directory. The full path and file name for the process link table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_LANE_USE_TABLE

The network lane use table key is optional. It specifies the name of the TRANSIMS lane-use file within the network directory. The full path and file name for the lane-use table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TURN_PROHIBITION_TABLE

The network turn prohibition table key is optional. It specifies the name of the TRANSIMS turn prohibition file within the network directory. The full path and file name for the turn prohibition table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_UNSIGNALIZED_NODE_TABLE

The network unsignalized node table key is optional. It specifies the name of the TRANSIMS unsignalized node file within the network directory. The full path and file name for the unsignalized node table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_SIGNALIZED_NODE_TABLE

The network signalized node table key is optional. It specifies the name of the TRANSIMS signalized node file within the network directory. The full path and file name for the signalized node table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TIMING_PLAN_TABLE

The network timing plan table key is required if the NET_SIGNALIZED_NODE_TABLE is specified. It specifies the name of the TRANSIMS timing plan file within the network directory. The full path and file name for the timing plan table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_PHASING_PLAN_TABLE

The network phasing plan table key is required if the NET_SIGNALIZED_NODE_TABLE is specified. It specifies the name of the TRANSIMS phasing plan file within the network directory. The full path and file name for the phasing plan table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_DETECTOR_TABLE

The network detector table key is required if the NET_SIGNALIZED_NODE_TABLE is specified and demand actuated signals are to be modeled. It specifies the name of the TRANSIMS detector file within the network directory. The full path and file name for the

detector table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_SIGNAL_COORDINATOR_TABLE

The network signal coordinator table key is optional. It specifies the name of the TRANSIMS signal coordinator file within the network directory. The full path and file name for the signal coordinator table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TRANSIT_STOP_TABLE

The transit stop table is optional. If the stop table is not provided, transit vehicles cannot be built. This key specifies the name of the TRANSIMS transit stop file within the network directory. The full path and file name for the transit stop table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TRANSIT_ROUTE_TABLE

The transit route table is required if the transit stop file is provided. This key specifies the name of the TRANSIMS transit route file within the network directory. The full path and file name for the transit route table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TRANSIT_SCHEDULE_TABLE

The transit schedule table is required if the transit stop file is provided. This key specifies the name of the TRANSIMS transit schedule file within the network directory. The full path and file name for the transit schedule table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

NET_TRANSIT_DRIVER_TABLE

The transit driver table is required if the transit stop file is provided. This key specifies the name of the TRANSIMS transit driver file within the network directory. The full path and file name for the transit driver table is constructed by appending the value of this key to the value of the NET_DIRECTORY key.

VEHCILE_FILE

The vehicle file is required. The vehicle file contains the parking location of each vehicle used for drive activities. The full path and file name is constructed by appending the value of this key to the value of the PROJECT_DIRECTORY key.

SORT_VEHICLES

The sort vehicles key is optional. If provided, it tells the program if the vehicle file needs to be sorted. The default value is TRUE. This implies that the vehicle file will be sorted by vehicle ID. If the key is FALSE and the vehicle file is not sorted, the program is likely to take considerably long to read the vehicle file. Setting this key to TRUE can significantly reduce the time required to read the vehicle file.

VEHICLE_TYPE_FILE

The vehicle type file is required. The vehicle type file contains the performance characteristics and length for each vehicle type used in the simulation. The full path and file name is constructed by appending the value of this key to the value of the PROJECT_DIRECTORY key.

PLAN_FILE

The plan file key is required. When appended to the PROJECT_DIRECTORY key, it specifies the file name for the travel plans to be simulated. The plan file must be sorted by time of day. If the data are split into multiple files, the key should end with ".t*" or ".*". The program will read each file in the AA, AB, AC… sequence. If the plan files have a companion *.def file, the PLAN_FORMAT and NODE_LIST_PATHS keys are not required.

PLAN_FORMAT

The plan format key is optional. If provided, it defines the file format of the plan file. The default plan file is in VERSION3 (unformatted text) format. This parameter enables the user to specify that the plan file is in BINARY format.

NODE_LIST_PATHS

The node list paths key is optional and when provided specifies the way the path is identified in the plan file. The key is "true" by default. This means that the plans include a list of the node ID numbers along the travel path. If the key is "false", the plans contain a list of link ID numbers. If the first character of the key is "0", "N", "n", "F", or "f", the key is interpreted as "false".

NEW_PROBLEM_FILE

The new problem file key is optional. When appended to the PROJECT_DIRECTORY key, it specifies the file name for the output problem file created by the program. The problem file is a tab-delimited ASCII file that lists the households and persons for whom complete plans could not be generated. A sample problem file generated by the **Microsimulator** is shown below.

TIME	LINK	HHOLD	PERSON	TRIP	MODE	PROBLEM	START	ORIGIN	ARRIVE	DESTINATION
3272	0	178	1	1	2	3	3272	5	3485	7
1089	0	343	1	1	2	3	1089	24	1302	22
3178	0	356	1	1	2	3	3178	11	3391	10
3264	0	359	1	1	2	3	3264	11	3477	10
997	0	503	1	1	2	3	997	10	1210	11

The PROBLEM field identifies the problem type for a given record. The message corresponding to each problem code is listed below. An interpretation of the messages generated by this program is provided in the algorithm section of this document.

1. Path Building
2. Time Schedule
3. Zero Node
4. Vehicle Type
5. Path Circuity
6. Travel Mode
7. Vehicle Access
8. Walk Distance
9. Wait Time
10. Walk Access
11. Path Size
12. Park-&-Ride Lot
13. Bike Distance
14. Departure Time
15. Arrival Time
16. Link Access
17. Lane Connectivity
18. Parking Access

19. Lane Merging
20. Lane Changing
21. Turning Speed
22. Pocket Merge
23. Vehicle Spacing
24. Traffic Control
25. Access Restriction
26. Transit Stop
27. Activity Location
28. Vehicle Passenger
29. Activity Duration
30. Kiss-&-Ride Lot
31. Vehicle ID
32. Data Sort
33. Walk Location
34. Bike Location
35. Transit Location

NEW_PROBLEM_FORMAT

The default output format is VERSION3 (tab delimited). Other options include BINARY, FIXED_COLUMN, COMMA_DELIMITED, SPACE_DELIMITED, TAB_DELIMITED, and DBASE.

MAX_SIMULATION_PROBLEMS

The maximum number of simulation problems key is optional. It defaults to 100,000 problems. This parameter defines the number of simulation problems that are permitted before the program terminates execution. A value of zero will disable this feature.

PRINT_PROBLEM_MESSAGES

The problem messages can optionally be sent to the print out file. This option should be limited to small test networks with relatively few problems.

CELL_SIZE

The default cell size is 7.5 meters. In general, cell size should be set based on the typical vehicle size. Vehicle sizes are defined in the Vehicle Type file. It is best to set the cell size to an integer multiple of the vehicle size. For example, if the typical vehicle size is 7.5 meters, the cell size should be 7.5, 3.75, or 2.5 (i.e., 1, 2 or 3 cell vehicles).

TIME_STEPS_PER_SECOND

The default time step is one second long. This key permits the user to simulate traffic based on sub-second time steps. A value of 2 will simulate movements every 0.5 seconds. If the Speed Calculation Method is set to CELL-BASED, it is desirable to synchronize the cell size and time steps to maintain a reasonable cells per time step ratio. The default values of 7.5 and 1 result in a maximum speed of five cells per second. The simulation works best with ratios of five or greater.

TIME_OF_DAY_FORMAT

The default time of day format will display values in 24 hour clock time. Other options include SECONDS, HOURS, 24_HOUR_CLOCK, and 12_HOUR_CLOCK.

SIMULATION_START_TIME

The default simulation start time is zero (midnight). The value can be coded as seconds, hours, or clock time.

SIMULATION_END_TIME

The default simulation end time is 24 hours or midnight. For full day simulations, most users code the end time greater than 24 hours to enable the trips that start just before midnight to complete their trip. A value of 27:00 or 97200 seconds is often used to continue the simulation until 3:00 AM the following day.

SPEED_CALCULATION_METHOD

The default speed calculation method is CELL-BASED. A cell-based simulation models speeds as whole cell movements during each time step. This is equivalent to the Cellular Automata method implemented by the original TRANSIMS **Microsimulator**. The other option is DISTANCE-BASED. This method attempts to simulate speeds using actual distance changes and stopping distance criteria. The method has not as yet been calibrated.

PLAN_FOLLOWING_DISTANCE

Plan following distance defines the distance ahead of each lane restriction when the vehicle will begin to make lane changes to get into the required lane. The plan following distance is divided into five stages or levels of urgency. As the vehicle approaches the lane restriction the urgency increases. This limits the vehicle's ability to make discretionary lane changes and increases the probability that a vehicle will force itself into a congested traffic stream. The default value is 525 meters. To avoid undesirable impacts, the plan following distance should be greater than the look-ahead distance.

LOOK_AHEAD_DISTANCE

Vehicles choose to make discretionary lane changes based on the look-ahead criteria. The look-ahead analysis compares the speeds and lane changes of remaining in the current lane to the speeds and lane changes of changing to the lane to the right or left. The option of a lane to the right or left is determined by odd and even time step values. If an adequate gap is available and the cumulative look-ahead score for the lane changes is better than the current lane, the vehicle changes lanes. The look-ahead score is defined by the Look Ahead Lane Factor and the Look Ahead Time Factor. This parameter defines the distance ahead that is scanned to cumulate the look-ahead score. The default value is 260 meters.

LOOK_AHEAD_LANE_FACTOR

The look-ahead lane factor defines the weight associated with lane changes required within the look-ahead distance on the travel path. Each required lane change results in a zero speed and a penalty of the vehicle acceleration rate times the lane factor. The default value is 4.0. The look-ahead weight for considering a change into the neighboring lane will have at least one lane factor at the start of the maneuver.

LOOK_AHEAD_TIME_FACTOR

The look-ahead time factor defines the weight associated with the speed of each vehicle encountered within the look-ahead distance on the travel path. The assumed travel speed is reset to match the speed of each vehicle encountered on the path. The speed is also set to zero for each lane change. After a vehicle is passed, the acceleration rate is used to define the speed for

subsequent unoccupied cells. The sum of the speeds times the time factor determines the travel time weight. The default value is 1.0.

MAXIMUM_SWAPPING_SPEED

To avoid deadlock situations where two vehicles need to exchange cells to continue their trip, a cooperative lane swapping concept was introduced to break the deadlock and permit the vehicles to continue their trips. This parameter defines the maximum speed at which lane swapping will be allowed. The default value is 37.5 meters per second.

MAXIMUM_SPEED_DIFFERENCE

In addition to the maximum swapping speed, the lane swapping concept is limited by the difference in speeds of the two vehicles. This parameter defines the maximum speed difference at which lane swapping will be permitted. The default value is 7.5 meters per second.

ENFORCE_PARKING_LANES

For parking lots on standard arterials (i.e., not boundary parking lots), the simulation will attempt to move the vehicle to the side of the street where the parking lot is coded prior to exiting the simulation. If the vehicle is unable to move to within two lanes of the parking location before needing to exit, a parking problem is encountered. This parameter can be used to define what happens under these conditions. If the value is YES or TRUE, the vehicle will stop and continue to attempt lane changes until it is in the appropriate exit lane. If the value is NO or FALSE, a problem message is generated and the vehicle is immediately moved to the parking lot. The default value is NO.

FIX_VEHICLE_LOCATIONS

In activity-based plan files, the location of the vehicle may be lost if the Router converted the vehicle to a magic move. Vehicle locations can also be in correct if the Router was executed with the IGNORE_VEHICLE_ID key is true. This key permits the **Microsimulator** to automatically move the vehicle to a parking lot attached to the origin activity location to avoid vehicle access problems. The default value is FALSE.

DRIVER_REACTION_TIME

The default reaction time is 1.0 second. This value is used to calculate the gap between vehicles at a given speed. Current experience suggests that a value of 0.7 generates logical results using the CELL-BASED method. Smaller values will increase the throughput of the simulation. The random number seed is used to implement the random rounding procedure used to convert the fractional cell gaps to integer cell gaps needed by the CELL-BASED algorithm.

If more than one value is provided for this parameter, each value corresponds to the facility type code assigned to the link. The facility type equivalence is: 1 = freeway, 2 = expressway, 3 = principal arterial, 4 = major arterial, 5 = minor arterial, 6 = collector, 7 = local, 8 = frontage road, 9 = ramp, 10 = bridge or other, 11 = walkway, 12 = bikeway, 14 = light rail, 15 = heavy rail, 16 = ferry, and 17 = external. If fewer than 17 values are provided, all facility types after the last key value are assigned the last value.

PERMISSION_PROBABILITY

The permission probability defines the likelihood that a vehicle will permit another vehicle to change lanes to the cell ahead when the traffic is stopped. The default value is 50 percent. A value of 50 indicates that the vehicles will permit other vehicles to cut in front of them 50 percent of the time.

If more than one value is provided for this parameter, each value corresponds to the facility type code assigned to the link. The facility type equivalence is: 1 = freeway, 2 = expressway, 3 = principal arterial, 4 = major arterial, 5 = minor arterial, 6 = collector, 7 = local, 8 = frontage road, 9 = ramp, 10 = bridge or other, 11 = walkway, 12 = bikeway, 14 = light rail, 15 = heavy rail, 16 = ferry, and 17 = external. If fewer than 17 values are provided, all facility types after the last key value are assigned the last value.

SLOW_DOWN_PROBABILITY

The slow down probability defines the likelihood that a vehicle will slow down for no apparent reason. The default value is zero (i.e., no random slow down). A value of 20 indicates that the vehicles will randomly slow down twenty percent of the time. The amount the vehicle slows down is defined by the Slow Down Percentage.

If more than one value is provided for this parameter, each value corresponds to the facility type code assigned to the link. The facility type equivalence is: 1 = freeway, 2 = expressway, 3 = principal arterial, 4 = major arterial, 5 = minor arterial, 6 = collector, 7 = local, 8 = frontage road, 9 = ramp, 10 = bridge or other, 11 = walkway, 12 = bikeway, 14 = light rail, 15 = heavy rail, 16 = ferry, and 17 = external. If fewer than 17 values are provided, all facility types after the last key value are assigned the last value.

SLOW_DOWN_PERCENTAGE

The slow down percentage defines the amount a vehicle will randomly slow down. The default value is zero which means that the slow down will be based on the deceleration rate. If the value is greater than zero is represents the percentage of the current speed that the vehicle will slow down. The random number seed is used to implement the random rounding procedure used to convert the fractional speed reductions to integer speed reductions needed by the CELL-BASED algorithm.

If more than one value is provided for this parameter, each value corresponds to the facility type code assigned to the link. The facility type equivalence is: 1 = freeway, 2 = expressway, 3 = principal arterial, 4 = major arterial, 5 = minor arterial, 6 = collector, 7 = local, 8 = frontage road, 9 = ramp, 10 = bridge or other, 11 = walkway, 12 = bikeway, 14 = light rail, 15 = heavy rail, 16 = ferry, and 17 = external. If fewer than 17 values are provided, all facility types after the last key value are assigned the last value.

RANDOM_NUMBER_SEED

The random number seed key is optional. This key specifies the random number seed to be used for the random impedance calculations. Any positive integer can be specified. If the value is zero or if no key is provided, the program uses the system clock to set the random number seed.

MINIMUM_WAITING_TIME

If a vehicle does not move from a given cell for a prolonged period of time, it is likely to be stuck in a deadlock situation. To break deadlocks, the simulation can give priority to vehicles that have not moved for some time. The vehicle is placed in a priority queue if it has not moved for more than the minimum waiting time. This parameter defaults to 180 seconds. The vehicle remains in the priority queue until it moves or the maximum waiting time is reached. Vehicles in the priority queue are given first opportunity to make lane changes or forward movements at the beginning of each time step.

MAXIMUM_WAITING_TIME

The maximum waiting time defines when a vehicle is removed from the simulation. If the vehicle has not moved for this amount of time, a Waiting Time problem message is generate, the vehicle is removed from the link, and moved to the destination parking lot. The default value is 3600 seconds.

MAX_ARRIVAL_TIME_VARIANCE

Each travel plan includes the expected arrival time at the destination activity location. If the vehicle is still traveling on the network at a time equal to the scheduled arrival time plus the maximum arrival time variance, the vehicle is removed from the simulation and moved to the destination parking lot, and an arrival time problem message is posted in the problem file. The default value for this parameter is 60 minutes.

MAX_DEPARTURE_TIME_VARIANCE

If the vehicle is unable to leave the parking lot at the beginning of the trip before the scheduled departure time plus the maximum departure time variance, the trip is abandoned and the vehicle is moved to the destination parking lot, and a departure time problem message is generated. The default value for this parameter is 60 minutes.

Microsimulator Output Files

The **Microsimulator** includes 10 different types of output data files. The generic name and purpose of each file type is listed in the following table. This is followed by a list of the keys that are available to control the output for each file type. The "#" at the end of each key represents the output key group. Any number of output groups (i.e., files) can be specified. All keys with the same group name and number operate as a set.

File Type	Description
Summary	Aggregates volumes and performance statistics (e.g., travel time) for each link direction and turning movement by time increment (e.g., 15 minutes). Turning movement volumes and delays are optional.
Snapshot	Lists the link direction, offset, lane, and speed of each vehicle at specified time points (e.g., every 5 minutes).
Occupancy	Lists the link direction, offset, lane, and cumulative occupancy of each cell by time increment (e.g., 15 minutes). May list the cells occupied at the maximum load point during the time increment or the total occupancy of each

	cell during the time increment.
Ridership	Summarizes the boardings and alightings at each stop on each route based on the scheduled and actual departure time for each run.
Event	Lists the scheduled and actual time and link direction and offset for each traveler and trip event (i.e., start time and end time).
System Event	Lists the time, node, and phasing information for each phase or timing plan change event at a traffic signal
Problem Link	Aggregates the number of problems by problem type (e.g., wait time) for each link direction by time increment (e.g., 15 minutes).
Traveler	Lists the link direction, offset, lane, and speed for each selected traveler by time step (e.g., second)
Turn	Aggregates the number of turning movements at a node by input and output link and time increment (e.g., 15 minute)
Speed Bin	Aggregates the number of vehicles of a specified vehicle type by speed bin traveling on link segments at specified time increments

Summary Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_SUMMARY_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_SUMMARY_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_SUMMARY_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_SUMMARY_INCREMENT_#	Time increment duration (default 15 minutes)	Opt	16 characters (3)
OUTPUT_SUMMARY_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_SUMMARY_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)
OUTPUT_SUMMARY_TURN_FLAG_#	True flag creates a nested file with turning movement volumes and delays (default false)	Opt	True/False (6)
OUTPUT_SUMMARY_PCE_FLAG_#	True flag applies the passenger car equivalence to volume data based on the vehicle type length	Opt	True/False (6)
OUTPUT_SUMMARY_PERSON_FLAG_#	True flag generates performance measures based on vehicle occupancy (i.e. persons) (default false)	Opt	True/False (6)

OUTPUT_SUMMARY_COORDINATES_#	Link selection coordinate range (default All)	Opt	x1, y1, x2, y2
OUTPUT_SUMMARY_VEH_TYPES_#	Vehicle selection range (default All)	Opt	ID range (5)

Snapshot Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_SNAPSHOT_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_SNAPSHOT_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_SNAPSHOT_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_SNAPSHOT_INCREMENT_#	Time increment duration (default 15 minutes)	Opt	16 characters (3)
OUTPUT_SNAPSHOT_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_SNAPSHOT_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)
OUTPUT_SNAPSHOT_COORDINATES_#	Link selection coordinate range (default All)	Opt	x1, y1, x2, y2
OUTPUT_SNAPSHOT_MAX_SIZE_#	Maximum size of the snapshot file in megabytes (default = 0 = unlimited)	Opt	Integer {0..2048}
OUTPUT_SNAPSHOT_LOCATION_FLAG_# (8)	Add X, Y, and Bearing fields to the output file (default = false)	Opt	True/False (6)

Occupancy Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_OCCUPANCY_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_OCCUPANCY_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_OCCUPANCY_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_OCCUPANCY_INCREMENT_#	Time increment duration (default 15 minutes)	Opt	16 characters (3)
OUTPUT_OCCUPANCY_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)

OUTPUT_OCCUPANCY_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)
OUTPUT_OCCUPANCY_MAX_FLAG_#	True creates a file with vehicle locations when the link contains the highest number of vehicles during the increment. False totals the number of seconds each cell on the link is occupied. (default false)	Opt	True/False (6)
OUTPUT_OCCUPANCY_COORDINATES_#	Link selection coordinate range (default All)	Opt	x1, y1, x2, y2

Ridership Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_RIDERSHIP_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_RIDERSHIP_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_RIDERSHIP_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_RIDERSHIP_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_RIDERSHIP_ROUTE_RANGE_#	Route number range (default ALL)	Opt	ID range (5)
OUTPUT_RIDERSHIP_ALL_STOPS_#	True creates a ridership file including all transit stops regardless of the number of boardings, alightings or riders at each transit stop. False restricts the output to only the transit-stops with at least one boarding, alighting or rider (default False)	Opt	True/False (6)

Event Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_EVENT_TYPE_#	Comma separated list of event type codes	Req	START_TIME, END_TIME, RUN_TIME
OUTPUT_EVENT_FILE_#	File name to be created	Req	255 characters

	within the project directory		
OUTPUT_EVENT_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_EVENT_FILTER_#	Number of seconds difference criteria (default 0)	Opt	Integer {0..2,147,483,647}
OUTPUT_EVENT_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_EVENT_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_EVENT_MODE_RANGE_#	Link number range (default ALL)	Opt	ID range (5)

System Event Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_SYSTEM_EVENT_TYPE_#	Comma separated list of event type codes	Req	PHASE_CHANGE, TIMING_CHANGE
OUTPUT_SYSTEM_EVENT_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_SYSTEM_EVENT_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_SYSTEM_EVENT_TIME_ FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_SYSTEM_EVENT_TIME_ RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_SYSTEM_EVENT_NODE_ RANGE_#	Node number range (default ALL)	Opt	ID range (5)

Problem Link Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_PROBLEM_TYPE_#	Comma separated list of problem type codes	Req	255 characters (7)
OUTPUT_PROBLEM_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_PROBLEM_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_PROBLEM_FILTER_#	Minimum problems per time increment (default 0)	Opt	Integer {0..2,147,483,647}
OUTPUT_PROBLEM_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)

OUTPUT_PROBLEM_INCREMENT_#	Time increment duration (default 24 hours)	Opt	16 characters (3)
OUTPUT_PROBLEM_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_PROBLEM_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)

Traveler Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_TRAVELER_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_TRAVELER_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_TRAVELER_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_TRAVELER_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_TRAVELER_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)

Turn Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_TURN_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_TURN_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_TURN_FILTER_#	Minimum turns per time increment (default 0)	Opt	Integer {0..2,147,483,647}
OUTPUT_TURN_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_TURN_INCREMENT_#	Time increment duration (default 24 hours)	Opt	16 characters (3)
OUTPUT_TURN_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_TURN_NODE_RANGE_#	Node number range (default ALL)	Opt	ID range (5)

Speed Bin Files

MICROSIMULATOR CONTROL KEY	Descriptions	Use	Values
OUTPUT_SPEED_FILE_#	File name to be created within the project directory	Req	255 characters
OUTPUT_SPEED_FORMAT_#	File format to be created (default Version3)	Opt	Format code (1)
OUTPUT_SPEED_VEHICLE_TYPE_#	A vehicle type code number (default 0 = ALL)	Opt	Integer {0..99}
OUTPUT_SPEED_FILTER_#	Minimum number of vehicles per time increment (default 1)	Opt	Integer {1..2,147,483,647}
OUTPUT_SPEED_TIME_FORMAT_#	Output time format (default seconds)	Opt	Time code (2)
OUTPUT_SPEED_INCREMENT_#	Time increment duration (default 24 hours)	Opt	16 characters (3)
OUTPUT_SPEED_TIME_RANGE_#	Time period range (default ALL)	Opt	Time range (4)
OUTPUT_SPEED_LINK_RANGE_#	Link number range (default ALL)	Opt	ID range (5)
OUTPUT_SPEED_SAMPLE_TIME_#	The time frequency in seconds at which the speed bins will be summarized (default 1 second)	Opt	Integer {>= 1}
OUTPUT_SPEED_BOX_LENGTH_#	The length in meters of the link segments for which speed bins are summarized (default = 0 = link length)	Opt	Floating point (1 decimal)
OUTPUT_SPEED_NUM_BINS_#	The number of speed bins that are summarized (default = 6)	Opt	Integer {>= 1}

Notes

1	VERSION3, TAB_DELIMITED, COMMA_DELIMITED, SPACE_DELIMITED, FIXED_COLUMN, DBASE, BINARY
2	HOURS, 24_HOUR_CLOCK, 12_HOUR_CLOCK, SECONDS, TIME_CODE
3	NOON, MIDNIGHT, d@hh:mm:ss.xAM/PM, d@hh:mm:ss.x, d@hh:mm, d@hh:mmAM/PM, d@hh.xxx, d@ssssss, hh:mm:ss, hh:mm:ss_AM/PM, hh:mm, hh:mm.x , hh.xxx, ssssss, wwwhh:mm where www = SUN, MON, TUE, WED, THU, FRI, SAT, WKE, WKD, ALL
4	One or more comma separated time ranges (start_time..end_time) of standard time codes (3) (e.g.,

	0:00..6:00, 18:00..23:00)
5	One or more combinations of comma separated ID numbers or ID ranges (ID..ID)
6	True/False values: T, TRUE, Y, YES, 1 or F, FALSE, N, NO, 0
7	TOTAL, PATH_BUILDING, TIME_SCHEDULE, ZERO_NODE, VEHICLE_TYPE, PATH_CIRCUITY, TRAVEL_MODE, VEHICLE_ACCESS, WALK_DISTANCE, WAIT_TIME, WALK_ACCESS, PATH_SIZE, PARK-&-RIDE_LOT, BIKE_DISTANCE, DEPARTURE_TIME, ARRIVAL_TIME, LINK_ACCESS, LANE_CONNECTIVITY, PARKING_ACCESS, LANE_MERGING, LANE_CHANGING, TURNING_SPEED, POCKET_MERGE, VEHICLE_SPACING, TRAFFIC_CONTROL, ACCESS_RESTRICTION, TRANSIT_STOP, ACTIVITY_LOCATION, VEHICLE_PASSENGER, VEHICLE_LOCATION, KISS_&_RIDE_LOT, VEHICLE_ID, DATA_SORT, WALK_LOCATION, BIKE_LOCATION, TRANSIT_LOCATION
8	Since the link shape file is not processed by the **Microsimulator**, the X, Y, and bearing data is based on the straight line centerline location of the vehicle. Typically these values are updated by ArcSnapshot to generated the correct X, Y, and bearing values based on link shapes and lane offsets.

Algorithm Notes

The high level processing within the **Microsimulator** can be outlined as follows:

1. Read the network and boundary speed files into memory
2. Read the vehicle type and vehicle files into memory
3. Convert the network into cell grids (defaults to 7.5 meters / cell)
4. Summarize link connections
5. Initialize the traffic controls
6. Construct a travel plan for each transit route and synthesize a vehicle for each run
7. Loop through each time step (defaults to 1 step per second)
    a. When the time step equals a whole second
        i. Process output increments for the second
        ii. Update the network use restrictions for the second
        iii. Update the signal timing plans for the second
        iv. Initiate transit runs scheduled to start during the second
    b. Process vehicles in the priority queue
    c. Process lane change requests
    d. Process the vehicle loading queue
    e. When the time step equals a whole second
        i. Read travel legs from the plan file that start at the second
        ii. Convert the travel leg to a link-lane travel plan
        iii. Load the vehicle to the network or the travel queue
    f. Calculate movements for each traveler and transit vehicle
    g. Move each vehicle to its new location and travel speed
8. Close the output files and print processing and problem summaries

Data Preparation

Several data processing steps are required to prepare the network, vehicle, and traffic control data for a cellular automata simulation where speeds and locations are measured as an integer number of cells per time step. This data processing is described in the sections immediately following.

Read the Network Files

Required files include Node, Link, Pocket Lane, Lane Connectivity, Parking, Activity Location, and Process Link. Optional files include Lane Use, Turn Prohibitions, Transit Stop, Transit Route, Transit Schedule, Transit Driver, Unsignalized Node, Signalized Node, Timing Plan, Phasing Plan, Detector, Signal Coordinator, and Boundary Speeds. All files are cross checked and converted to internal record numbers (e.g., link ID, node ID, etc.). Note that all lengths, offsets, speeds, and travel times are rounded to one decimal place and converted to integers (e.g., 100.25 meters = 1003 decimeters). Also note that the **Microsimulator** does not read the shape file. This means that the XY coordinates calculated for the snapshot file are based on the straight line location along the link. The ArcSnapshot program uses the link offset to recalculate the link and lane position based on shape points.

Read Vehicle Files

A Vehicle Type and Vehicle file are required. All files are cross checked and converted to internal record numbers. The vehicle type information is also converted to cells and cells per time step. All vehicles are initially located at a parking lot. Memory is allocated for each vehicle based on its cell size (i.e., single cell or multi-cell vehicles). Multi-cell vehicles are processed as a cell chain. Each cell of the vehicle can occupy a different directional link, lane, and cell location. All cells of the vehicle follow the path set by the lead cell. Since all vehicles are read into memory and stored in a single memory block, the program is currently limited to approximately 20 million vehicles.

Construct Cell Grids

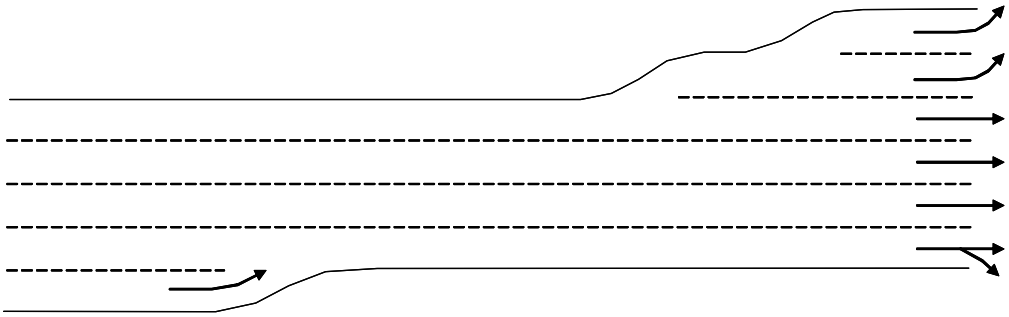
All movements within the network are processed as cells per time step (by default 7.5 meters per second). This typically means that a vehicle can move between zero and 5 cells per time step. In other words, at any given point in time, a vehicle will be traveling at one of six possible speeds (0.0, 16.8, 33.6, 50.3, 67.1, or 83.9 mph). Vehicles accelerate based on the vehicle type's acceleration rate or a minimum of one cell per time step. The vehicle's deceleration rate, however, is not used in calculating cell-based travel speeds. A vehicle can stop from any speed in one time step.

The network is physically converted to a cell grid for simulation. Each direction of each link that permits vehicle movements (the use code includes car, truck, bus, and/or rail) and includes at least one travel lane is converted to a cell grid. The number of cells in the grid is based on the link length.

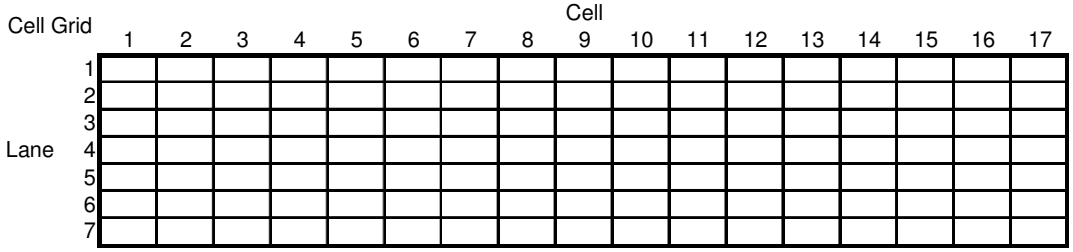
Cells = max ((length + half_cell) / cell_size, 1)

Remember that length and cell size are rounded to the nearest one decimal place. A cell size of 7.5 meters is processed as 75 decimeters. Half cell is therefore (75 + 1) / 2 = 38 decimeters.

The other dimension of the grid is the total number of lanes available at any time on the link direction. This is the total travel lanes plus the total number of pocket lanes on the left and right side of the link. Lanes are numbered from the left-most pocket lane. In other words, a roadway configuration that looks like:



is converted to a cell grid that looks like:



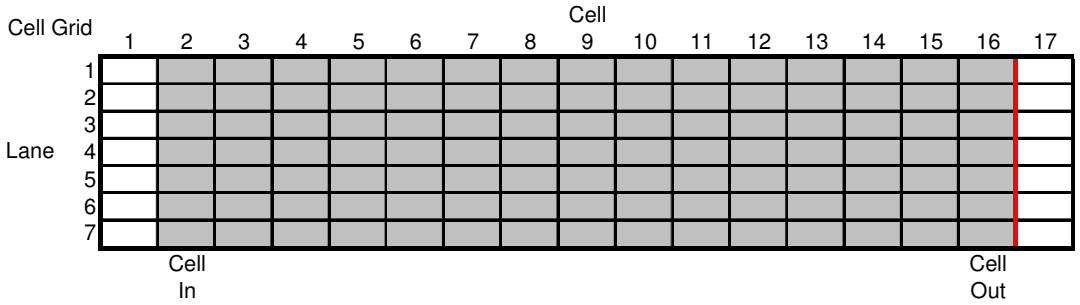
This configuration is coded as four travel lanes with one right side pocket lane and two left side pocket lanes for a total of seven lanes. The overall length is 130 meters = 17 cells.

In fact, two cell grids are created for each link direction. One cell grid records vehicle IDs at the beginning of each time step and the second grid records vehicle IDs at the end of the time step (i.e., the beginning of the next time step). Each vehicle record keeps track of the vehicle state (location and speed) at the beginning and end of the time step.

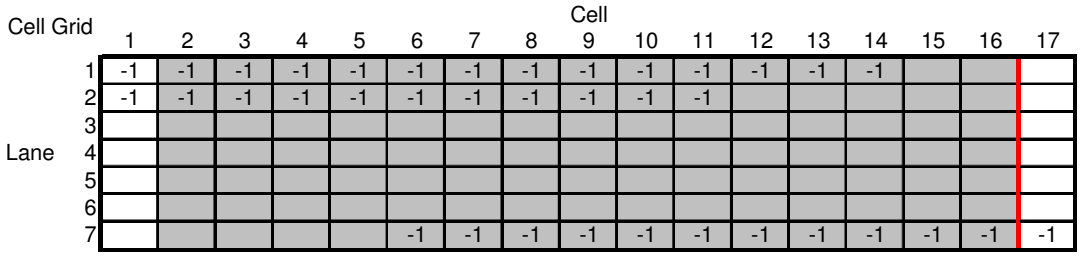
The program then identifies the first and last "travel" cells based on the coded setback distances at each end of the link. Travel cells are where lane use restrictions and traffic controls take affect. Setback cells are used for intersection movements or transition areas between lane use restrictions. The number of setback cells is calculated as:

Setback cells = max ((setback + half_cell) / cell_size, 1)

If our example has setback distances of 11.2 meters or less (e.g., 0), the travel cells will be the shaded cells in the following graphic.



The program then sets the pocket lane cells that are not included in a pocket lane to -1 to mark them as unavailable to traffic at all times. For this example, the initial grid cells look like this:



The maximum speed on the link is then calculated by rounding up the speed limit coded on the link:

Maximum speed = max ((speed_limit + cell_round) / cell_size, 1)

where cell_round is:

cell_round = (3 * cell_size) / 4

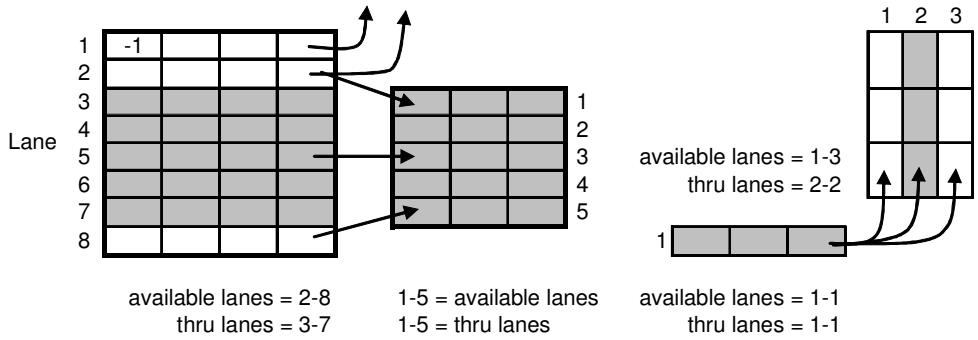
This is 56 decimeters for the default cell size. The result of this calculation is cells per second. This results in the following speed limit ranges having the same maximum speed:

Max Speed	**Speed Limit**
cells/second	meters/second
1	0.1 to 9.3
2	9.4 to 16.8
3	16.9 to 24.3
4	24.4 to 31.8
5	31.9 to 39.3

Summarize Link Connections

In most cases, the lane connectivity file defines a connection between a specific lane on an approach link to a specific lane on a departure link. The **Microsimulator** summarizes this data into a single link connection record with the directional approach and departure links as the record keys. The program includes in this record 12 additional data items. These are the first and last input and output lane numbers, the first and last input and output thru lane numbers, the movement type (i.e., left, thru, right, u-turn), and the movement penalty, speed, and capacity if

provided. The input and output lane numbers are derived directly from the lane connectivity file. The input and output thru lanes are a subset of these lanes that would not require lane merging or diverging to complete the movement. In other words, if the number of input and output lanes is different, the thru lanes identify the lane numbers that are directly aligned with each other. Movements between links that use these lanes are given priority over movements that use the other available lane connections. These concepts are shown in the following graphic.



The algorithm for setting thru connections starts by comparing the number of approach lanes to the number of departure lanes. If there are more approach lanes than departure lanes, the program reduces the approach lane range by removing pocket lanes from the approach link. Left pocket lanes are removed first followed by right pocket lanes. If there are still too many approach lanes, one lane is removed from the left side and then the right side until the lanes align. If there are more departure lanes than approach lanes, the process is similar. Left pocket lanes are removed first followed by right pocket lanes. If there are still too many departure lanes, one lane is removed from the left side and then the right side until the lanes align.

Initialize Traffic Controls

Converting the traffic signal and sign files into a coordinated sequence of events linked to specific movements within the network is a complex process. The process begins by verifying the signal timing plan for each signalized intersection. This includes confirming the signal type (actuated or timed), identifying the first phase, counting the number of rings, and checking the ring boundaries.

The program then initializes the signal controller for each phase. This confirms that a timing plan is provided for each phase and builds a list of the intersection movements that are controlled by the phase. For movements that represent a left turn, the program identifies up to two conflicting movements. These are thru movements in the opposite direction (i.e., oncoming traffic) that a left turning vehicle would need to cross over (or yield) to make the turn.

The procedure then sorts the phase lists for each ring and timing plan. This includes identifying lead phases within each ring and the ring barriers where phase changes need to be coordinated. The phase lists are then linked to detectors and link connection records. If a given connection does not include a phase controller, a warning message is issued.

It is at this point that the sign file is processed. Signs are added to the control lists in much the same way as traffic signals. Stop signs enter a link on a given approach and generate a list of connections for departing the intersection. The orientation of each of these movements is calculated and conflicting movements are identified. The conflicts for a left turn movement are links approaching from the left side or thru in the opposite direction. (Conflicts from vehicles approaching from the right side are captured by the cell merge logic.) The conflicts for a thru movement are the links approaching from the right and left sides (i.e., cross-street traffic).

The program than checks all of the uncontrolled intersections to determine if any of the movements at these locations include left turns that cross thru movements in the opposite direction. These conflicts are treated as a permitted green phase with a 100 percent green time.

Finally the program loops through all of the controls and for each control that has conflicting movements, identifies the number of lanes included in the conflicting movement. The lane number range is stored with the conflict so that the crossing vehicle will be able to check each lane before making the movement.

Construct Transit Plans

In the Version 3 **Microsimulator** transit routes were treated like any other vehicle. A driver travel plan was created for each run of each route. These plans list the origin and destination parking lot, the trip start and end times, and the travel itinerary (node list) for the trip. Transit vehicles were added to the vehicle file and initially located at the origin parking lot. From the **Microsimulator**'s point of view, this integrated transit into the network processing logic in a relatively simple and straightforward way. Unfortunately, it made transit network development and editing a very difficult and complex process.

The Version 4 software was designed to make the development and management of transit networks much simpler and more intuitive at the expense of more complex simulation pre-processing. Rather than code a driver travel plan for each run, the Version 4 network includes a driver path that lists the sequence of links the transit route uses. The same set of links is used for each run of the route. Also, the Version 4 approach does not include pre-defined vehicles. A transit vehicle is synthetically generated for each run of each route by the **Microsimulator**. These vehicles enter the network at the first stop on the transit route and exit the network at the last stop on the route. In this way there is no need to assign transit vehicles to parking lots and estimate the travel time between the parking lot and the first transit stop to coincide with the transit schedule.

The first step in the transit pre-processor is to calculate the transit vehicle IDs. The program retrieves the highest ID from the highway vehicles included in the simulation and rounds this up to create a unique transit vehicle code. This code is listed in the printout file to help the modeler know how to interpret transit vehicle IDs. The transit vehicle ID first identified the highest order digit in the vehicle file. For example, if the highest vehicle number is 2,134,531, the highest order digit is 2. The program adds one to this value and multiples the result by the order of magnitude of the highest vehicle number (i.e., 3,000,000). This value becomes the root of the transit vehicle ID.

The rest of the transit vehicle ID is a composite number based on the highest transit route number and the highest number of runs included in any of the transit routes. These values are concatenated as a character string image and added to the root ID. If the root ID is not large enough to accommodate the number of digits in the line and run string, additional characters are added to the root ID to satisfy the required number of digits. If, for example, the highest route number is 1234 and the maximum number of runs on any route is 120, the resulting transit vehicle ID would have the following template:

3x,xxx,###

where 30,000,000 is the root ID that identifies the vehicle number as a transit vehicle and the xxxx digits identify the route number and the ### digits identify the run number within that route. In other words, the vehicle ID for the first run of transit route 25 is 30,025,001.

The program adds a transit vehicle to the vehicle inventory for each run of each route in the transit network. These vehicles are initially loaded at the first transit stop on the route.

The pre-processor then converts each transit driver record into a **Microsimulator** travel plan. These plans are similar to standard highway vehicle plans with the exception that they start and end at transit stops rather than parking lots. In addition to lane control points at intersections, a transit plan also includes mid-link lane control points for each stop on the route. The lane control points force the transit vehicles to move into the right most lanes to serve transit stops. If this lane is blocked by a vehicle use restriction by time of day (e.g., offpeak parking), the **Microsimulator** will override this restriction to enable the transit vehicle to move in to and out of the stop location.

Note that the pre-processor creates only one travel plan for each transit route. This becomes a plan template that is indexed by the transit route ID and stored in a separate array. When the transit schedule indicates that a new run of a transit route should start, the software makes a copy of the corresponding template, adds the run specific vehicle information, and initiates the loading process.

Time Step Processing

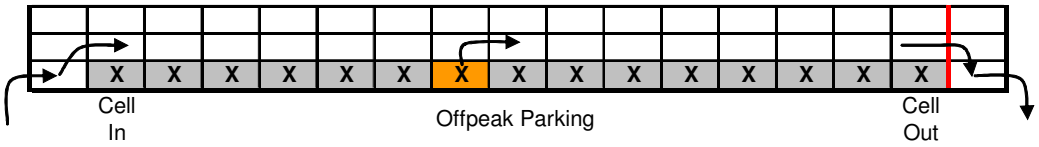
After the network, vehicles, and traffic controls are prepared, the program steps through each time step of the simulation. At the beginning of each time step, the program first checks if any of the time increment boundaries for the output options have been reached. If so, the output files are written and the summary statistics are cleared in preparation for the next time increment. This affects the link summary, problem link, turning movement, occupancy data, and speed bin output options. The program then processes the time-of-day changes in lane use restrictions and traffic controls and starts any scheduled transit runs.

Update Use Restrictions

For each time step at a whole second (i.e., normally every time step), the program checks the lane use restriction records to determine if any of the restrictions start or end at the time step. If

a time-of-day restriction is found, each lane is reset to the default link restriction and then each time-of-day restriction modifies the default values based on its time, lane, and cells ranges. In this process, lane-use records provided in the lane-use file cannot make a lane less restrictive than the default link use restriction.

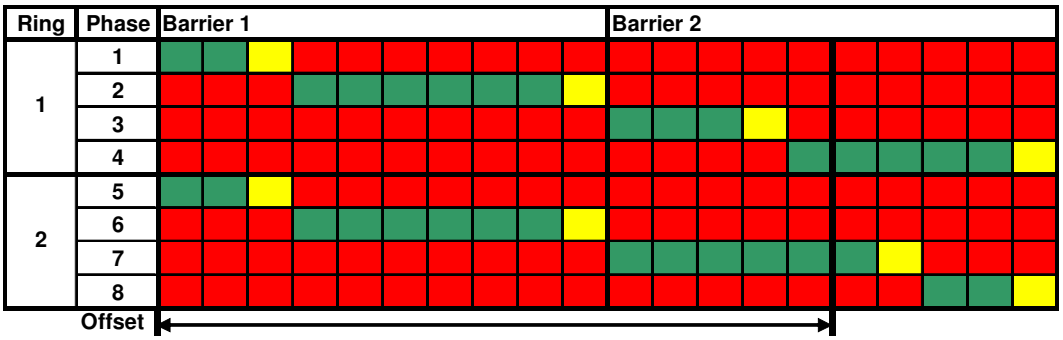
Since restrictions do not apply to setback cells, the lane connectivity is not directly impacted by closing or restricting travel cells. Vehicles may need to immediately change lanes to move out of the setback cells to continue their trip, but the travel paths are not "broken" by adding or changing lane-use restrictions by time-of-day. If all of the lanes are blocked or a vehicle is on the link when the restriction takes effect, the vehicle will attempt to move out of the restricted lane or wait until the restriction is removed. These concepts are depicted in the following graphic.



Update Traffic Controls

Traffic signals are updated in two ways. If this is the first time step or a time step when a signal is scheduled to change its timing plan, the signal must be initialized. Once it is initialized, the signal changes phases or extends phases at defined time intervals.

The initialization process begins by retrieving the signal type, number of rings and barriers, the timing plan ID, and the timing offset from the signal database. All movements under signal control are initialized to red. If the signal has multiple rings and barriers, the minimum phase time for each barrier is calculated based on the highest minimum phase time assigned to each phase in the barrier. The procedure then calculates the cycle length based on the sum of the minimum phase times for each barrier. If the signal has a timing offset, the program converts this offset into an offset within a specific cycle barrier. The program then resets the phase timers based on the barrier-offset time. Each phase will have a red, green, or yellow status with an appropriate event time defined for the next phase check. These concepts are depicted in the following graphic.



If the signal does not have barriers (also called groups in the network file), the process calculates the cycle length for each ring based on the sum of the minimum phase times for the ring. If the rings have different cycle lengths, the rings are padded with red time to correspond to the ring with the highest minimum cycle length. Given these adjustments, the offset position within each ring is identified and the phase status and timing events are set.

Now that the signal is initialized it proceeds through the standard update cycles based on the timing events. Each ring is checked to determine if the current time step is equal to the timing event for that ring. If it is, the status of each phase movement is updated. If the current status is red, the next phase flag is set. If the signal is currently yellow, the status is changed to red. If the current status is green and the signal is demand actuated, the program checks the current green duration against the maximum green time. If it is still less than the maximum, all of the detectors assigned to the movement are checked. If there are vehicles on the detectors, the green time is extended by one green extension increment. If vehicles are not detected or the signal is fixed timed, the status is changed to yellow (or red if yellow is not included in the timing plan).

If the next phase flag is set, the procedure checks if the next phase starts a new barrier group. If it does, all phases in the other rings must clear before the next barrier can be entered. If the next phase can be entered and the signal is demand actuated, all of the detectors for the next phase are checked to determine if there are any vehicles waiting for the phase. If not, the next phase in the sequence is called. If it cycles through all of the phases without detecting any vehicles, the phase that follows the last active phase is activated. When a new phase is activated, all of the movements for the previous phase are set to red and all of the movements for the new phase are set to green. The timing event for the controller is set to the minimum green time defined by the timing plan.

If output system events are requested, each timing event associated with each phase of the selected signals is reported. The output file includes the time and location for each instance where a signal phase changes status (i.e., green, yellow, red) and each time a demand actuated signal extends the green time. Changes from green to yellow and red to green also trigger cycle failure processing for Version 4 link summary output.

Start Transit Runs

For each simulation second, the program scans the transit routes to determine if any of the runs of a given route are scheduled to start at that second. If a run is scheduled to start, a copy of the transit travel plan template for that route is made and the vehicle ID and start and end times are updated. The procedure then attempts to load the vehicle to the network at the first stop on the route. If this fails, the vehicle is added to the travel queue for consideration in the next time step. When the vehicle is loaded to the network, all of the passengers waiting to board the vehicle are loaded to the vehicle to start the transit leg of their trip. Since the transit schedule file records the stop departure time, no dwell time is added for loading passengers at the first stop on the route. The program assumes the transit vehicle waits at an off-network location while it loads passengers and is ready to enter the network at its scheduled departure time.

Queue Processing

After all of the preliminary network adjustments are made, the program loops through various vehicle queues. The primary queue is the network traffic queue that includes all vehicles currently on the network. This queue is processed in the order in which vehicles enter the network. In the course of simulating these vehicles needs arise that make it desirable to change the processing order to give some vehicles priority over other vehicles. These vehicles are processed first and then given a second chance to move as part of the normal network traffic queue. These queues deal with vehicles that have not moved for a long time, vehicles that need to make a lane change to continue their trip, and vehicles that are waiting in parking lots or at transit stops for a chance to enter the network.

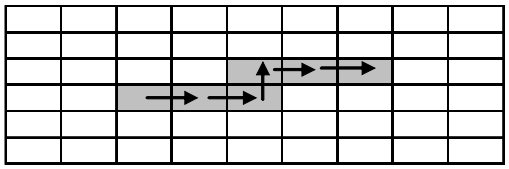
Priority Queue

The first set of vehicles that are given an opportunity to move are those vehicles included in the priority queue. Initially, the priority queue is empty. As transit vehicles are loaded they are added to the priority queue. All other vehicles move in to and out of the priority queue based on their urgency to move. This urgency is defined in a number of ways, but it primarily results from a vehicle stopped in one place for MINIMUM_WAITING_TIME seconds. This key defaults to 180 seconds or 3 minutes. Once a vehicle joins the priority queue it remains in the queue until it moves or is removed from the network. A vehicle can be removed from the network if it fails to move for MAXIMUM_WAITING_TIME seconds (defaults to 3600 seconds or one hour) or the MAX_ARRIVAL_TIME_VARIANCE or MAX_DEPARTURE_TIME_VARIANCE keys are violated. Both variance keys default to 60 minutes.

A priority vehicle will be processed as a lane change or as a forward movement depending on its needs. These procedures are discussed below. The only advantage is that the vehicles that have been in the priority queue longest are given an opportunity to attempt their maneuver before any other vehicles move. If they fail to complete their movement, they remain in the queue and are considered a second time as part of the network traffic queue. The order of processing in the network traffic queue is based on the time of day the vehicle entered the network.

Lane Change Requests

The **Microsimulator** maintains five lane changing queues. Each queue represents a different level of lane changing urgency. The vehicles in the queue with the highest urgency are given the first opportunity to attempt a lane change. If a lane change is feasible, the vehicle is immediately moved over into that lane in the current cell grid, the lane change flag is set to zero, and the vehicle is removed from its original cell in the current cell grid. If the vehicle is a multi-cell vehicle, the second cell of the vehicle will be moved up to the original cell in the original lane. This continues for all other vehicle cells. In other words, a lane change for a multi-cell vehicle is processed as a cell chain that follows the path of the lead cell.



The lane changing queues are cleared immediately after they are processed. This means that a vehicle does not automatically stay in a lane changing queue for more than one time step. Regardless of what happened during the lane changing step, the vehicle remains in its normal position in the network traffic queue and is processed a second time as part of the normal traffic processing loop. At this point the vehicle can move forward or request another lane change.

Vehicles are added to one of the lane changing queues if they need to make a lane change to continue their trip. This can be a lane change required at the end of a pocket lane or due to a lane-use restriction or a lane change required to continue the travel plan. A lane change required by network or lane-use restrictions is always added to the priority one lane changing queue. The priority level for plan following lane changes is a function of the number of required lane changes and the distance to the end of the link.

As part of the travel plan construction process described below, each link on the travel path has four lane number ranges. There are two ranges for entering the link and two ranges for exiting the link. One range specifies all possible lanes and the other range specifies the best possible lanes. Given this information, the lane change priority is defined as:

```
num_cells = last_cell - cell

if (num_cells > 0 and num_cells < plan_following and
        (lane < best_lane_low or lane > best_lane_high)) then

        if (lane < best_lane_low) then
                change = best_lane_low - lane
                if (change > 1 and lowest_lane < best_lane_low) change = change - 1
        else
                change = lane - best_lane_high
                if (change > 1 and highest_lane > best_lane_high) change = change - 1

        change_level = (5 * num_cells / change + plan_following / 2) / plan_following

        if (change_level < 1) change_level = 1
        if (change_level = 1 and speed > 1) speed = speed - 1
```

In these calculations “plan_following” refers to the PLAN_FOLLOWING_DISTANCE key rounded to cells (the default value is 525 meters or 70 (7.5m) cells). The equation basically divides the plan following distance into five sections and assigns a higher priority level or urgency as the vehicle approaches the end of the link. If more than one lane change is required, the priority level increases accordingly. Notice also that priority one vehicles slow down if they are traveling more than one cell per second.

Of course, adding a vehicle to a priority lane changing queue does not guarantee that a lane change will be made in the next time step. Vehicles are likely to attempt a lane change for many

time steps before the right conditions are found to make the change. These conditions are a function of the vehicle's current speed and the speed and location of vehicles behind and ahead in the neighboring lane.

Lane Changing

When a vehicle attempts a lane change, it first determines if a lane change is possible. If the cell in the neighboring lane is not available or permissible, the lane change is aborted immediately. The cell may not be available because it is a pocket lane and the lane does not exist at the current location along the link (i.e., it is marked as -1). It may also be unavailable if the lane has a vehicle use restriction that prohibits the current vehicle type (e.g., lane is used for parking or limited to HOV vehicles). If the lane is occupied by another vehicle, the program retrieves the neighboring vehicle and checks its processing status. If the vehicle has not yet been processed or it has been processed, but failed to move, the algorithm checks to see if a cooperative lane change can be made.

Cooperative Lane Change

The first step in a cooperative lane change is to check if the neighboring vehicle has a reservation placed on its movements. A reservation is a conditional processing rule based on a movement made by another vehicle. If the vehicle with a reservation moves, it immediately reevaluates the movements of the reserved vehicle. In other words, the reserved vehicle has first call on the cell vacated by the vehicle with a reservation. So in this case, if the vehicle does not have a reservation, the current lane change vehicle places a reservation on the neighboring vehicle.

The second step is to check the travel plans of the neighboring vehicle. If the neighboring vehicle is not also attempting a lane change or is a multi-cell vehicle and the neighboring cell is not the front of the vehicle, or the current vehicle is a multi-cell vehicle and it is not stopped, the lane change fails. If it passes these tests, it then checks the relative speeds of the two vehicles against the lane swapping parameters. MAXIMUM_SWAPPING_SPEED (defaults to 37.5 mps) and MAXIMUM_SPEED_DIFFERENCE (defaults to 7.5 mps) are the keys that control this process. In order for a swap to take place, both vehicles must be traveling at a speed less than or equal to the maximum swapping speed and the absolute difference in the vehicles' speeds must be less than or equal to the maximum speed difference.

If both vehicles are single cell vehicles, the program confirms that the neighboring vehicle is attempting to move into the lane occupied by the current vehicle. If this is the case, both vehicles exchange positions in the current cell grid and may consider additional movements from their new positions as part of the traffic processing loop.

If one of the vehicles is a multi-cell vehicle or the neighboring vehicle is not attempting a lane change into the current lane, the algorithm attempts a diagonal movement. This logic temporarily places the vehicle in the neighboring lane (i.e., on top of the neighboring vehicle) than calls the check-ahead routine (described in more detail below) to determine if the vehicle could move if it were located in the neighboring lane. If it cannot move (e.g., vehicles are stopped in a queue, the traffic signal is red, etc.), the lane change fails and the vehicle is placed

back in its original lane. If it can move, the lane change logic continues by considering a lane change to the cell in front of the neighboring vehicle.

Lane Change to an Unoccupied Cell

At this point in the process we have a vehicle that can make a diagonal lane change or a vehicle with a neighboring cell that is unoccupied. In other words, there is a cell the vehicle could change into. The vehicle is temporarily placed into this cell in the neighboring lane and the check-behind and check-ahead routines are called. These routines are described in more detail below, but in this case they basically check if a sufficient gap exists for the vehicle to merge into the new lane. If a sufficient gap does not exist, the lane change fails and the vehicle is place back in its original position.

If the merge is feasible, a permissive probability check is then made. If the cell immediately behind the merge cell in the new lane is occupied by a vehicle, a random number is generated and compared to the permissive probability to determine if this vehicle will permit the current vehicle to move in front of it. The permissive probability is controlled by the PERMISSIVE_PROBABILITY key and defaults to 50 percent. If the merge is permitted, the current vehicle adds a reservation for the vehicle that permitted the merge. When the current vehicle moves from its new position, the vehicle that permitted the merge will have first call on the merged cell. This helps to simulate alternating merge behavior.

If the merge is not permitted, the lane change fails and the current vehicle is placed back in its original position. Then the program checks the status of the vehicle that did not permit the merge. If this vehicle had been previously processed and failed to move, its status is reset and it is reprocessed next. This permits the vehicle to reconsider the available cell ahead of it that might not have been available when it was originally processed.

If the vehicle satisfies all of the conditions, the lane change is implemented by moving the vehicle into the new lane and cell in the current cell grid. If the vehicle is a multi-cell vehicle, each cell in the vehicle is moved up one cell. If the vehicle is a single-cell vehicle, the current grid cell is made available to other movements. After the move, the vehicle can decide to request another lane change or move forward during the normal processing loop.

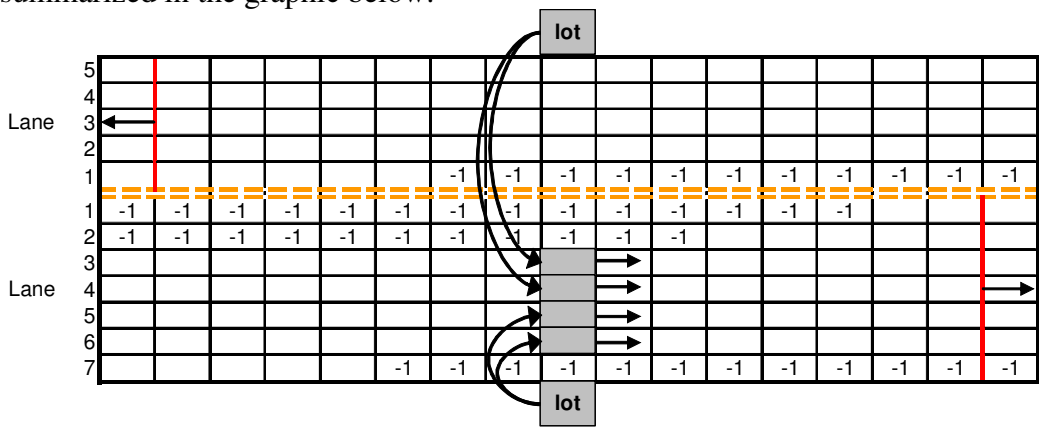
Loading Queue

Each time a travel plan involving a vehicle is created, the program attempts to immediately load that vehicle to the network. The vehicle is inserted into one of the lanes next to the parking lot and the travel plan is added to the network traffic queue. If none of the permissible loading lanes are available or the vehicle is currently in use on another trip, the travel plan is added to the travel queue and reconsidered at this point in the process. Vehicles are considered for loading in the time of day order they enter the queue. They wait in the queue until a loading lane is available or the MAX_DEPARTURE_TIME_VARIANCE key value is reached.

A permissible loading lane is defined by the loading type. For standard parking lots or transit stops or stations, the two lanes closest to the lot or stop are used. This includes pocket lanes that are available at the cell offset of the parking lot or transit stop. It does not, however, consider

any lane-use restrictions that make the lane unavailable at the time of the trip. This is one of the primary reasons why two lanes are included in the loading process. If one lane is unavailable due to a lane-use restriction, the other lane may still be available to load the vehicle. If both lanes are unavailable, the vehicle will wait in the parking lot until the restriction is removed.

For parking lots, the direction of travel is also considered. If the parking lot is on the opposite side of the street, the left two lanes are used for loading vehicles. If the parking lot is on the same side of the street, the right two lanes are used. Regardless of which side of the street the parking lot is on or which lane the vehicle loads into, there is no consideration of the traffic in the lanes the vehicle moves through to reach the vehicle loading lane. These concepts are summarized in the graphic below:



The lane rules apply to vehicles exiting the network as well. The vehicle will make lane changes to the appropriate side of the street before being pulled off into the parking lot. The modeler can override this behavior by setting the ENFORCE_PARKING_LANES key to false. In this case the vehicle will attempt to move into an exit lane, but if it fails to do so before reaching the parking lot, it can be removed from any lane.

The exception to the two lane rule is boundary parking lots and external transit stops. In these cases, all lanes are available for loading and unloading vehicles. The external flag also affect the speed at which the vehicle is loaded onto the network.

For an internal lot or stop, the vehicle enters the network at a speed equal to the acceleration rate of the vehicle type. This assumes the vehicle has a speed of zero in the parking lot and has accelerated for one second when it enters the network. Similarly, a vehicle must reduce its speed to a value less than or equal to the vehicle's deceleration rate before it can exit the network. Of course, all of these values must also be less than or equal to the maximum speed of the link.

For external entry and exit points, the speed of the vehicle defaults to the maximum speed of the link (or the maximum speed of the vehicle type). If a boundary speed file is provided, the speed associated with the time of day when the vehicle enters or exits the network is used. This speed may be adjusted by the random slow down procedure or constrained by the maximum speed of the link or the maximum speed of the vehicle type.

Given the permissible lanes and the entry speed, the algorithm then needs to select a lane for the vehicle. This process begins by identifying all of the "best" lanes with an unoccupied cell at the cell offset of the parking lot or transit stop. Best lanes are defined by the plan following procedure to help vehicles prepare for downstream turning movements. If, for example, the travel plan requires the vehicle to immediately exit a freeway soon after it enters from an external station, the algorithm would prefer to load the vehicle into the right-most lanes even though all lanes are permissible.

For all of the best lanes with an unoccupied cell, the procedure temporarily places the vehicle in the cell at its entry speed and then applies the check-behind and check-ahead routines to determine if the merge is acceptable. The output of the check-ahead routine is the speed of the vehicle. The algorithm selects the lane with the highest speed and loads the vehicle into that lane in the current cell grid.

If none of the best lanes have unoccupied cells, the search is expanded to include all permissible lanes. Once again, the vehicle is temporarily placed into unoccupied cells and the lane with the highest speed is selected. The cell grid and vehicle record are updated, and the travel plan is deleted from the travel queue and added to the end of the network traffic queue. The vehicle can then decide to move forward or change lanes as part of the normal traffic processing loop.

Notice that in this procedure, transit vehicles are loaded and unloaded from the network at the first and last stops on the route. This is different from the Version 3 procedure that required a transit vehicle to start at a standard parking lot and then travel to each transit stop and end at another parking lot. This may be more realistic, but it created a number of coding problems for transit routes that start or end on dead-end links. The only way to address many of these situations was to add a U-turn at the end of the link.

The Version 4 approach simplifies the transit route coding and minimizes the traffic impacts at the end of transit routes. The transit vehicle automatically materializes at the first transit stop at its scheduled departure time and is loaded onto the network with the same rules as highway vehicles. The procedure also assumes the transit vehicle waits at an off-network location while it loads passengers and is ready to enter the network at its scheduled departure time. The dwell time at all other transit stops is a function of the passenger boarding and alighting rates. However, if a transit vehicle is unable to leave a transit stop after the maximum dwell time is reached, it will close its doors to prevent additional passengers from boarding.

Read Travel Plans

The plan processing loop reads all of the records in the input plan file in time sorted order. If a travel plan has a start time that is earlier than the simulation start time, the traveler and any associated vehicles are repositioned to the destination location and parking lot of the trip. This is necessary to ensure that the vehicle is in the correct location for subsequent trips the traveler may make. If the start time is later than the simulation end time, further input is terminated, but all travelers that are currently active continue to be processed.

Records from the plan file continue to be read until the start time of the next plan is greater than the current time step. For each plan that starts between the simulation start and end times, plan processing is initiated. If the mode of the plan is an activity, school bus, or magic move, the plan is ignored. If the mode is transit and the driver flag is set, the plan is also ignored. These plans would be transit driver plans left over from a Version 3 application. If the plan has an auto mode and the driver flag is not set, the mode is reset to carpool. This plan represents an auto passenger or taxi trip. If the mode is walk or transit and the transit network files are not included in the **Microsimulator** control file, the plan is discarded. In other words, the **Microsimulator** will ignore transit trips if the transit network is not provided.

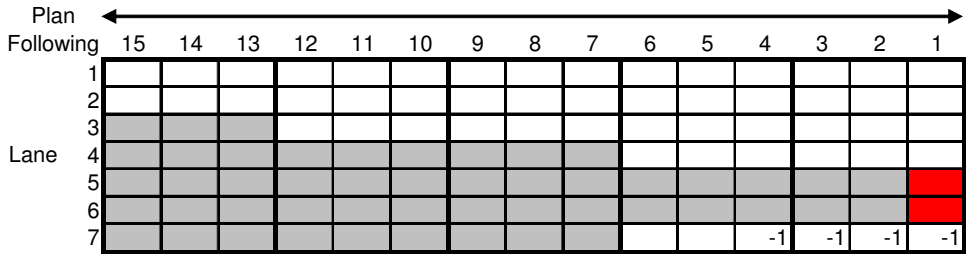
Plan processing starts by checking if the traveler is currently active. If the traveler is active, the current plan will not be loaded to the network until the previous plan is completed. A **Microsimulator** travel plan is then built for the traveler. This includes the planned start and end times, the mode, and the start and end locations converted to internal record indices. For a transit trip, the traveler is added to the end of the boarding queue at the initial transit stop of the specified route.

For an auto or truck trip, the specified vehicle is retrieved and it's location checked. Then the travel path (node-based or link-based) is converted to a link-lane list. A drive path starts at a parking lot offset, travelers along one or more links, and ends at another parking lot offset. At both parking lots and at every intermediate intersection, several sets of lane ranges are constructed. These lane ranges define the permissible and best lanes for entering and editing each link. The lane ranges are initialized based on the parking lot type and the link connectivity for the required movement. These values are then constrained by the plan following criteria.

Plan Following

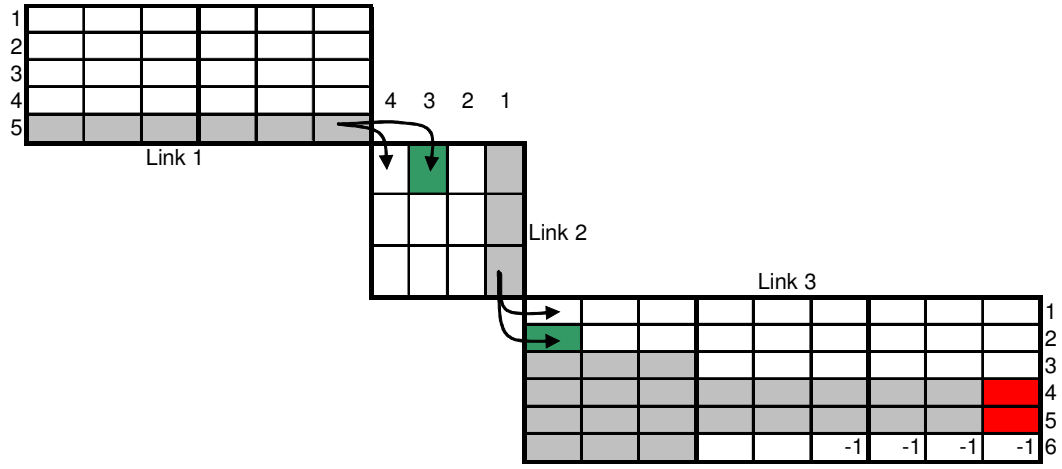
The best lane logic starts at the end of the trip and works back to the beginning of the trip, tracking the distance and lane restrictions as it proceeds. The initial lane restrictions are defined by the destination parking lot type. If the parking lot is an external boundary parking lot, all lanes will be permissible. For a right-side parking lot, the right two lanes will be used as the lane restriction. For a left-side parking lot, the left two lanes will be used. Given these restrictions and the distance to the prior intersection, the plan following algorithm may limit the best lanes for the movement at the prior intersection to a subset of the permissible lanes.

The plan following algorithm divides the plan following distance into five equal segments. No lane changes are desirable in the last two segments. The third and fourth segments are limited to one lane change, and the last segment is limited to two lane changes. This concept is shown in the following graphic.



In this example, the vehicle needs to move into lanes five or six to exit at the destination parking lot. The plan following distance is equal to 15 cells so each of the 5 segments is three cells long. For distances up to six cells upstream, the vehicle would be encouraged to use lanes five or six. For distances between 7 and 12 cells upstream, movements involving lanes four through seven would be considered best. Distances between 13 and 15 cells could use lanes three through seven and beyond 15 cells would have no restrictions.

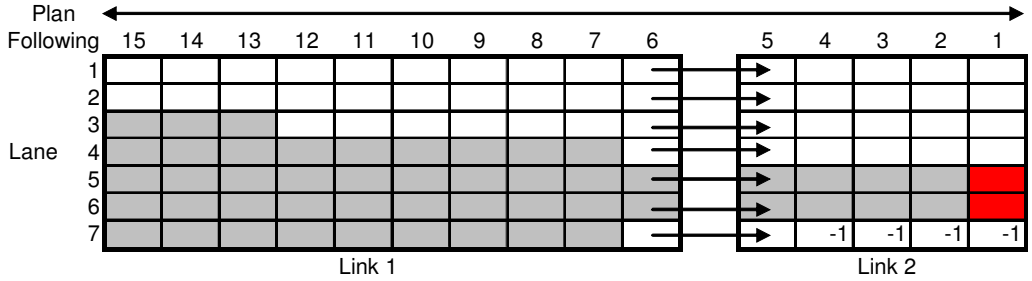
The plan following concepts can become complex when turning movements are involved. In the example below, the travel path includes three links and two turns in a distance equal to the plan following parameter. The vehicle needs to exit from lanes four or five on link 3. It enters link 3 by way of a left turn from link 2. The lane connectivity from link 2 to link 3 is from lane 1 on link 2 to lanes 1 or 2 on link 3. The plan following algorithm would flag this as a problem and select the closest turning movement as the best lane for entering link 3. This would favor the movement from lane 1 to lane 2 over the movement from lane 1 to lane 1.



Once the best lane for entering link 3 is set, the plan following distance is reset and the next upstream intersection is evaluated. In this case, the best lane from a plan following perspective would be to enter link 2 in lane 1. Since the path comes from link 1 and link 1 does not have a connection to lane 1, the plan following algorithm will select the best connection between link 1 and link 2 as lane 5 to lane 3. This would immediately become a priority one lane change request as the **Microsimulator** processes the vehicle.

Since the best lane entering link 2 is restricted, the plan following distance will be reset again in order to select lane 4 as the best lane to enter link 1. This restriction would affect the best lane alignment for any additional upstream links.

For situations were the distance to the prior intersection is less than the plan following distance and the best lane range entering the link is less than the permissible lanes for the movement, the algorithm adjusts the best exit lanes for the previous link to be less than or equal to the number of best lanes for entering the link. This helps to preserve the lane alignment across intersections.



In the example above the lane connectivity between links 1 and 2 permits the use of all lanes. The plan following algorithm would set the best lanes for entering link 2 to lanes 5 and 6. The procedure would then determine that the best lanes for exiting link 2 would also be lanes 5 and 6. Based on the plan following distance and permissible lane changes, the best lanes for entering Link 1 would be set to lanes 3 through 7.

Calculate Vehicle Movements

The core of the simulation program is the vehicle processing loop for a given time step. This loop reads through the network traffic queue in the order the vehicles were added to the network and attempts to move the vehicles from one cell to another cell along their travel path. The first step in this process it to check that the arrival time variance and maximum waiting time parameters have not been exceeded. If they have, a problem message is recorded and the vehicle is parked at the destination and removed from the simulation.

Discretionary Lane Changes

The vehicle is then given the opportunity to consider a discretionary lane change. Discretionary lane changes are lane changes that are not required by the travel plan, but may help the vehicle avoid congestion or slower vehicles ahead of it in the current lane (i.e., pass a slower vehicle). Like any other lane change, a lane must be available to change into before a lane change is even considered. These lane changes are considered on alternating sides of the current lane. On odd time steps the vehicle considers the current lane plus one. On even time steps the vehicle considers the current lane minus one. If the resulting lane number is out of range for the link, the process is aborted.

The discretionary lane changing process can be aborted for many other reasons as well. The new lane may be occupied by another vehicle or the lane use restrictions would not permit the vehicle. The vehicle may also be limited by the plan following criteria. If the distance to the end

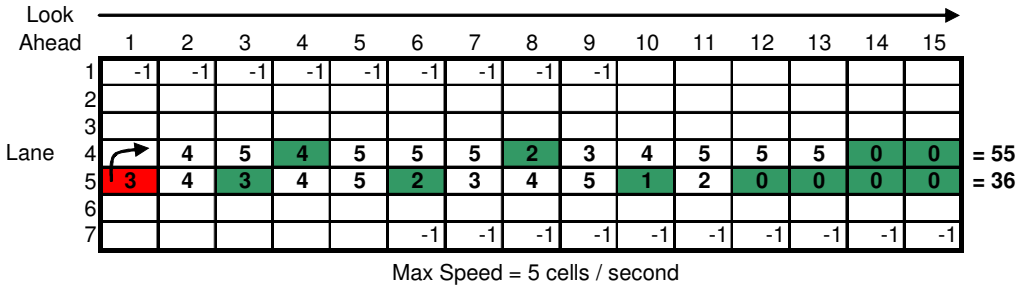
of the link is less than two fifths of the plan following distance, the discretionary lane change must move the vehicle closer to the required lane range at the end of the link rather than further away from it.

If the lane is available, the vehicle is temporarily placed in the neighboring cell of the new lane and the check-behind and check-ahead routines are applied. These routines check if a sufficient gap exists for the vehicle to merge into the new lane. If a sufficient gap does not exist, the lane change fails and the vehicle is placed back in its original position. If a gap exists, the look-ahead selection criteria are applied.

The look-ahead algorithm depends on three control keys: LOOK_AHEAD_DISTANCE, LOOK_AHEAD_LANE_FACTOR and LOOK_AHEAD_TIME_FACTOR. The distance key defaults to 260 meters (or 35 cells) and defines the distance the traveler will look ahead to determine if a lane change is desirable. The lane factor (defaults to 4.0) and the time factor (defaults to 1.0) define the relative weights of lane changes to travel time in selecting a lane.

The algorithm calculates the sum of the travel speeds minus the number of lane changes for a vehicle that starts in the neighboring lane versus a vehicle that remains in the current lane. If the difference is greater than the lane factor, the lane change is preferred. The vehicle is moved to the new lane and cell in the current cell grid and continues forward movements from this position.

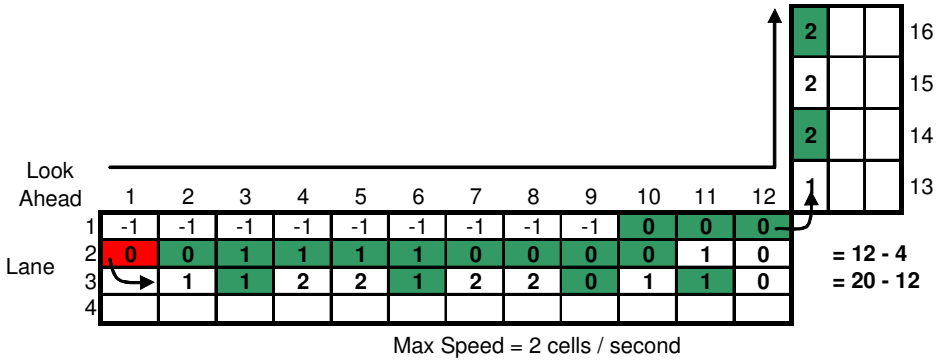
The lane preference calculation starts with the current speed of the vehicle and adds one second of acceleration per cell until it finds a vehicle in its path. It slows down to the speed of that vehicle and then adds one second of acceleration per cell until it finds another vehicle. This process continues until the number of look-ahead cells (distance) is reached. The maximum speed of the vehicle is limited by the maximum speed of the vehicle type. The resulting travel time weight is the sum of the speeds in each cell multiplied by the time factor. This calculation is depicted in the following graphic.



In this example, the vehicle is currently located in lane 5 and is considering a discretionary lane change to lane 4. The look-ahead distance is 115 meters or 15 (7.5m) cells. If the vehicle stays in lane 5, there are seven vehicles traveling at speeds between zero and 3 cells per second in the 15 cell area. The speed for each cell is based on the current speed of any vehicle in the cell or a speed equal to one second of acceleration (i.e., 1 cell per second) from the speed in the previous cell up to the link maximum speed of 5 cells per second. For the current lane this sums to a time

weight of 36 and in the neighboring lane the time weight is 55. The time weight of 55 is then reduced by 4 to account for the lane change weight. Since 51 is greater than 36, the vehicle will make the lane change.

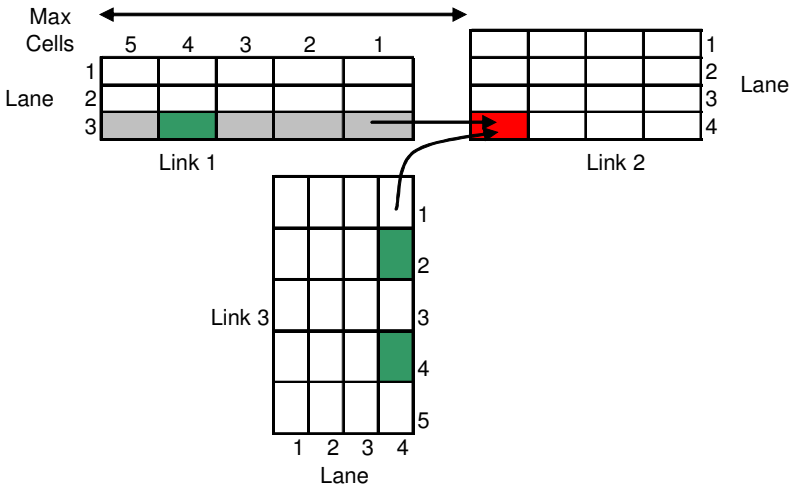
The previous example is a very simple situation that does not include any downstream lane changes. When downstream lane changes or turning movements are required the calculations can be much more complex. Consider the example shown below.



In this example, the vehicle is currently stopped in lane 2 in a queue of mostly stopped vehicles waiting to make a left turn at the next intersection. The discretionary lane change would pull the vehicle out of the queue and attempt to reenter the queue at the end of the link in order to complete the left turn required by the travel plan. Notice that the left turn needs to be made from the left turn pocket lane, but the look-ahead calculations continue in the current or neighboring lane until the end of the link. Since a lane change is required to continue the trip, the speed of the last cell on the link is set to zero and a lane change penalty is subtracted from the travel time weight. In the case of the option that starts by changing into lane 3, there is one lane change to move from lane 2 to lane 3 and then two lane changes to move from lane 3 to lane 1 to complete the left turn. Both paths then share the same lane on the second link. Since the resulting weights have the same net value, the vehicle would not choose to change lanes.

Check-Behind

The check-behind routine is used in lane changing calculations and forward movements to determine if an adequate gap exists for a vehicle to merge into a given cell. The routine starts with a vehicle in a specific location, traveling at a specific speed, and the number of time steps ahead that the vehicle is expected to arrive at this location. It first confirms that the cell is not currently assigned to some other vehicle at the end of the time step. It then estimates the number of cells a vehicle could travel at the maximum speed of the link for the specified number of time steps. The routine scans the cells upstream from the current cell to locate the first vehicle or pocket lane it encounters. If this search reaches the beginning of the link, it continues on the lane of the upstream link that is designated as the priority thru movement. This concept is shown in the following example:



In this example, the vehicle is attempting to merge into the first cell in lane 4 on link 2. This could be the consequence of a lane change from lane 3 on link 2 or the turning movement from link 3. In either case the program will check upstream a total of 5 cells to identify a vehicle that may have a priority claim to the first cell in lane 4 on link 2. In this case, the priority thru movement to lane 4 on link 2 is from lane 3 on link 1. The program will find the vehicle in the second cell in lane 3 on link 1.

If the identified vehicle has previously been processed within the time step, the speed and location of the vehicle at the end of the time step is available as part of the "to" data for the vehicle. If it has not been processed, the routine estimates what it will do in the current time step based on its current speed and travel plan. If the vehicle's movement is currently controlled by a red traffic signal or a stop sign, it would not interfere with the merging vehicle. Otherwise the speed of the vehicle times the number of time steps plus the length of the current vehicle defines the minimum cell gap required to permit the merge.

minimum gap = veh_speed * time_steps + length

If in the example above, the vehicle is one cell long and it arrives during the current time step, the speed of the vehicle in lane 3 of link 1 would need to be 3 cells per second or less to permit the vehicle to merge into lane 4 on link 2 (i.e., 3 * 1 + 1 <= 4). In other words, the vehicle should not need to slow down for the merge to take place.

Check-Ahead

All forward movements for a vehicle are initially set by the check-ahead routine. This routine can be called in test-mode or save-mode. The routine is called in test-mode for lane changing tests. The primary difference between test-mode and save-mode is how the passengers on transit vehicles are processed. When a transit vehicle arrives at a bus stop, all alighting and boarding passengers are processed in order to accumulate the individual processing rates and check the minimum and maximum dwell times. The actual time when the traveler exits or enters the vehicle is set by this accumulation and the processing order defined by the passenger queues. The travel plans for the passengers and the transit vehicle are put on "hold" until the

corresponding future time step is reached. While the transit vehicle is on hold, no forward movements are allowed.

The output of the check-ahead routine is the number of cells the vehicle can move during the time step (i.e., maximum speed) and flags for lane changing and simulation problems. The first problem check relates to lane use restrictions. If the vehicle is currently located on the travel section of the link and the current lane does not permit the vehicle type, the routine sets the speed to zero and checks if travel in one of the neighboring lanes is permitted. If the lane is available, the access restriction problem message is generated and a lane change request is set. If the lane is not available, a link access problem messages is generated and the vehicle waits in the current cell until the use restriction is removed or the maximum waiting time is exceeded.

The program then checks if the vehicle is at the link-offset of the destination parking lot. If it is and the ENFORCE_PARKING_LANES key is true, the current lane is checked against the exit lane range. If the vehicle is not in the appropriate lane, the speed is set to zero and a lane change request is issued. If parking is not enforced and the vehicle is not in an appropriate lane, a parking access problem message is generated and the vehicle exits the network.

If the vehicle is located in the last travel cell on the link and the link has traffic controls, the control for the appropriate link connection is retrieved and its status checked. If the status is red, stop sign, or right-on-red, the speed is set to zero and the routine exits. If the control is green or the intersection is uncontrolled, the current lane is checked against the exit lane range for the travel plan. If the lane is out of range, the speed is set to zero and a lane change request is issued.

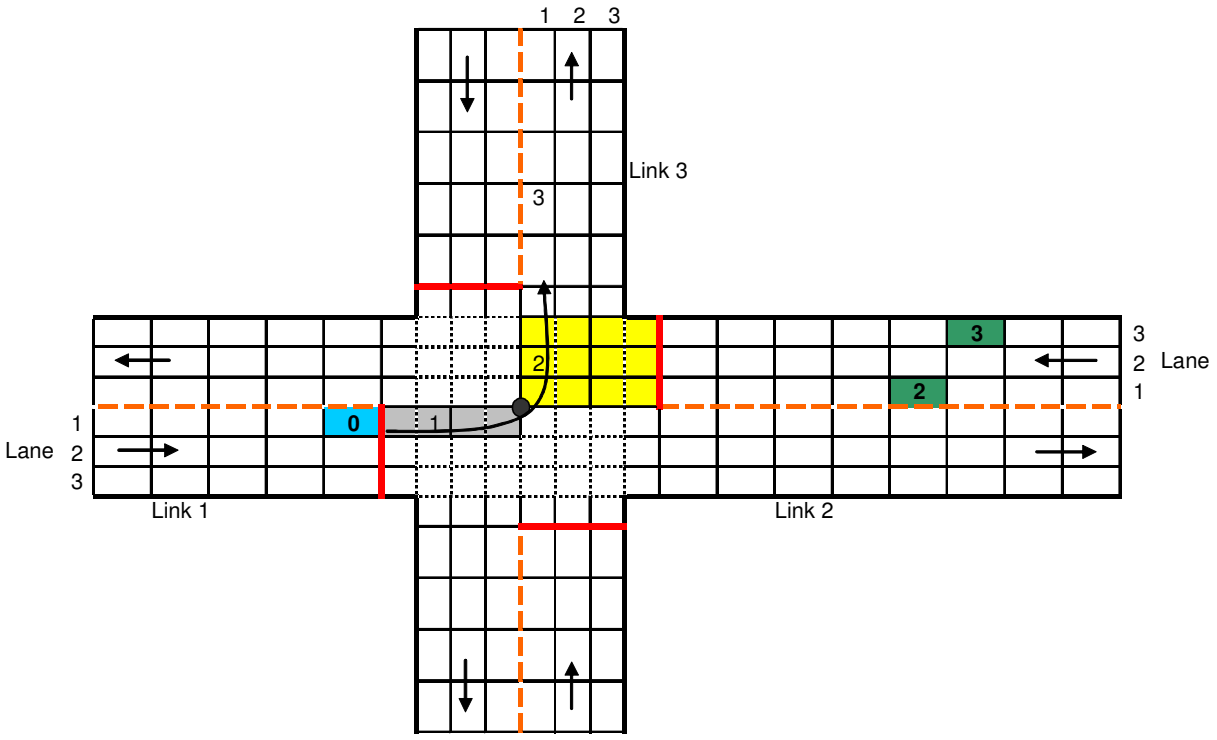
The routine then attempts to add one second of acceleration to the current speed of the vehicle. The current location and the new speed are checked against the plan restrictions at the end of the link. If the distance to the end of the link is less than the plan following distance and the vehicle is not currently in an acceptable lane for exiting the link, a priority level lane change request is issued and the vehicle may slow down. The resulting maximum speed defines the maximum number of cells the vehicle could consider moving during the time step.

The program then checks each cell on the path ahead. If the link has a traffic control and the forward check reaches the last travel cell on the link, the status of the controller that relates to the travel plan is checked. If the control is red, stop sign, or right-on-red, the maximum speed is set to the number of cells required to reach the last travel cell. If this value is greater than one, the speed is reduced by one to simulate a vehicle slowing down.

If the signal is yellow, the vehicle will continue through the intersection if the current speed of the vehicle is greater than the distance to the last travel cell. If the current speed is less than the gap, the vehicle will slow down and stop at the intersection.

If the signal status is a permitted green or the control is a stop sign or a right-on-red and the vehicle had previously stopped, the traffic on conflicting movements is checked. This procedure requires an estimate of the number of seconds it will take the vehicle to reach each conflict point.

This is a function of the number of cells in the intersection, the number of conflicting lanes, the current speed of the vehicle, and the vehicle acceleration rate. This concept will be described using the following graphic.



The example above shows a standard intersection with three approach and departure lanes in each direction. The setback distances are set at two cells for all legs. This means that the last travel cell for the link would be where the stop bars are shown. The black dot in the middle of the intersection is where the network node is located and each link starts or ends. (Note that intersection cells for cross streets overlap inside the intersection, but are processed independently by the **Microsimulator**.)

In this example, the blue vehicle is stopped at the intersection and receives a permitted green signal phase for the left turn from link 1 to link 3. The program first calculates the number of seconds it will take the vehicle to cross the intersection. In the first second the vehicle accelerates from a speed of zero to a speed of one cell per second. This places the vehicle in the first grey cell of the intersection at the end of one second. From here the vehicle accelerates once more to a speed of two cells per second. This places the vehicle on the first cell of the link 3 at the end of two seconds. At the end of three seconds the vehicle is on the fourth cell of link 3 and past the intersection. The conflicting lane calculations will therefore be based on a two second clearance time.

The conflicting movement for a permitted left from link 1 to link 3 is the approach from link 2. This approach has three thru lanes each with two cells within the intersection. These are the cells shown in yellow in the graphic. The program checks for conflicting movements by temporarily

placing a copy of the current vehicle into each of the intersection cells (i.e., the yellow cells) of the conflicting link and calling the check-behind routine with a two second clearance time. What this does is check each of the cells back from the intersection until a vehicle is found. The number of cells that are checked is based on the maximum speed of the link multiplied by the clearance time. In this case, the routine will find a vehicle traveling two cells per second in lane 1 and a vehicle traveling three cells per second in lane 3. The vehicle in lane 1 could therefore travel four cells during the two second clearance time and the vehicle in lane 3 could travel six cells. This means that the vehicle in lane 1 does not represent a conflict, but the vehicle in lane 3 is a conflict.

If the movement can be made without conflict, it is still likely to take several time steps to complete. In the example above the vehicle would not clear the intersection until the third time step. This means that the conflicting movement check would be made at least one more time. Each check accounts for the current location and speed of the vehicle, the status of the traffic controls, and changes in the conflicting lanes.

When the vehicle reaches the end of the link and is ready to enter the next link, lane alignment calculations are made to determine which lane the vehicle will enter based on the lane it was in when it exited the previous link. The range of exit and entry lanes is defined in the travel plan. These ranges include all permissible lanes and a subset of "best" lanes from a plan following perspective. If the exit lane is not part of the best lane range, but the entry alignment would permit the vehicle to move into one of the best entry lanes, the vehicle will select that lane.

The routine then checks if the current movement is the primary thru movement for the entry lane. The thru movement has priority access to the lane. Vehicles that enter a lane from a merging movement must yield to vehicles entering the lane from the thru movement. This test is made by placing a copy of the current vehicle into the entry lane and calling the check-behind routine to determine is an adequate gap exists to enter the lane. If the thru movement has a conflicting vehicle, the current vehicle will wait at the end of the previous link until the merge can be made.

In addition to traffic controls and link transitions, the forward path of a vehicle can be blocked in one of four ways: by a lane-use restriction, by the end of a pocket lane, by another vehicle, or by a reserved cell. If the cell ahead includes a lane-use restriction does not permit the current vehicle type, the vehicle will stop prior to the restricted cell and issue a lane change request. Similarly, if the cell ahead is coded as -1, it represents the end of a pocket lane. If the end is more than one cell ahead, the vehicle will slow down and issue a lane change request.

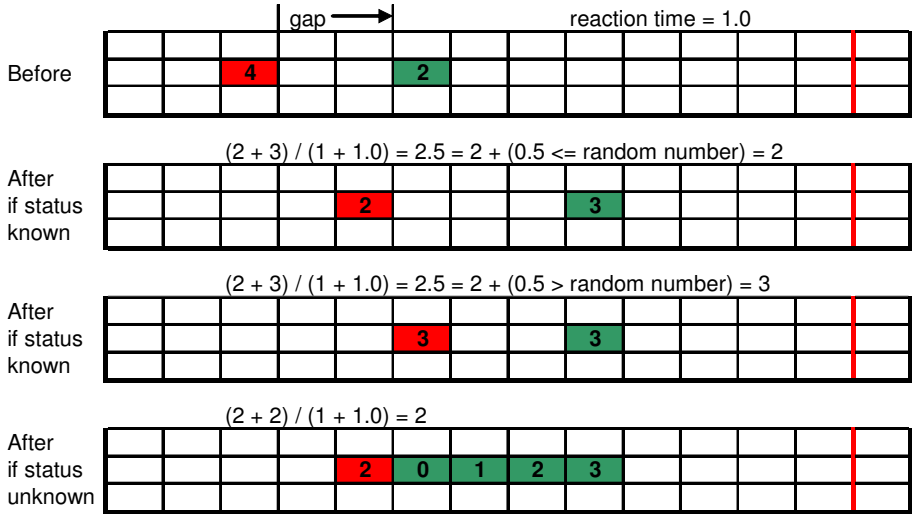
The third condition relates to a vehicle occupying the cell ahead. If the vehicle ahead had previously been processed during this time step, the result of its movement is considered by the vehicle behind. If it has not been processed, its current speed is considered. The speed of the vehicle ahead plus the gap between the current vehicle and the vehicle ahead is used in the reaction time calculation.

Reaction times can be specified by facility type. The default reaction time for all facility types is 1.0 second. The speed of the current vehicle is set by the following equation:

speed = (veh_speed + gap) / (1.0 + reaction_time)

This value is truncated to an integer and the remainder is used in a random probability calculation. If the remainder is greater than the random probability, an additional increment of speed is added to the truncated value. In other words, the speed value is rounded to an integer based on a random rounding probability. If the vehicle ahead has not yet been processed, the speed value cannot be greater than the gap between the vehicles. This speed must also be less than the maximum speed of the link and the maximum speed of the vehicle type.

Reaction time concepts are depicted in the following graphic. In this example, a vehicle traveling at 4 cells per second is following 2 cells behind a vehicle traveling at 2 cells per second. The second grid show the position at the end of the time step if the vehicle ahead is known to be accelerating to a speed of 3 cells per second during this time step. Based on the reaction time calculation, the current vehicle will have a target speed of 2.5 cells per second for this time step. A random number is generated and compared to the 0.5 value to determine if the speed is truncated to 2 cells per second as shown in the second grid or rounded up to 3 cells per second as shown in the third grid. The fourth grid shows the result if the status of the vehicle ahead is not known. Since the vehicle ahead could have speeds between zero and 3 cells per second, the current vehicle is limited to 2 cells per second in the current time step.



The final forward movement check relates to cell occupancy at the end of the time step. If the destination cell is reserved by another vehicle, the current vehicle will stop prior to this cell. This can occur as a result of a priority movement or multiple lane merges into the same cell. This check helps to avoid accidents caused by multiple vehicles being assigned to the same cell.

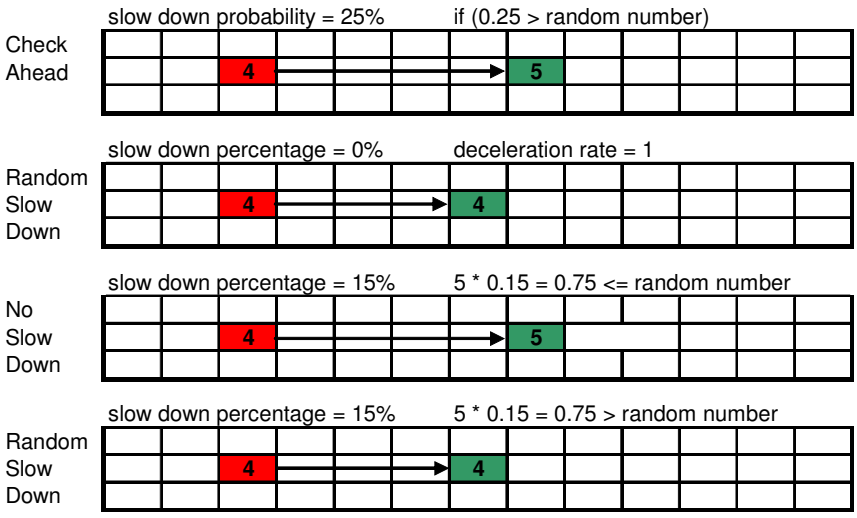
Check-Move

The check-ahead routine calculates the maximum speed of a vehicle during a given time step, but there are several conditions for which this speed is reduced. The first relates to the random slow down concept. Random slow down is controlled by the SLOW_DOWN_PROBABILITY and

SLOW_DOWN_PERCENTAGE keys. The default slow down probability is zero which disables random speed reductions for all facility types. If one or more percent values are included in the control file, the corresponding percentage of vehicles will slow down for no apparent reason in each time step of the simulation.

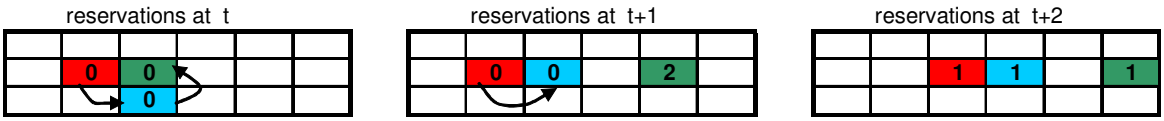
The magnitude of the speed reduction is controlled by the slow down percentage key. The default value is zero which means that vehicles slow down according to the maximum deceleration rate of their vehicle type (or a minimum of 1 cell per second). If percentage values are provided by facility type, the magnitude of the slow down is a percentage of the current speed of the vehicle. Since this value is likely to be a fraction, a random number is used to determine if the speed reduction is rounded up or down to an integer number of cells per second.

The following graphic shows the random slow down effects. In the first cell grid the check-ahead routine determined that the vehicle could accelerate from 4 to 5 cells per second during the current time step. With a slow down probability of 25 percent, the random number generator flagged this vehicle for slow down. If the slow down percentage is zero, the vehicle will automatically slow down by one deceleration unit or 1 cell per second. If the slow down percentage is 15%, the speed reduction from 5 cells per second is 0.75 cells per second. Since this is a fraction, a second random number is generated to determine if the slow down is rounded up or down. In the third grid the random number is higher than 0.75 so the slow down is ignored. The slow down is applied in the fourth grid.



If the resulting speed is zero and lane changes are not requested, the routine checks the cell ahead to determine if a vehicle is blocking the path. If this vehicle is also stopped, the current vehicle attempts to attach itself to the vehicle in front of it through a cell reservation. A cell reservation gives the current vehicle first call on the cell ahead when the vehicle ahead moves. If the vehicle ahead already has a reservation, the routine looks to see if the reserved vehicle is also stopped, but does not have a reservation. In this case the current vehicle is added as a reservation to the reserved vehicle.

This logic basically establishes a processing order for vehicles stopped in a queue. The following graphic shows the sequence of events that take place when the reservations are processed.



If the speed is greater than zero, the routine re-traced the vehicle path to ensure there are no conflicts in the destination cell grid at each cell along the path. If a conflict is found, the vehicle is stopped prior to the conflict. If the vehicle moves and has a reservation, the reserved vehicle is checked to ensure it is still waiting for the original cell. If it is, the reserved vehicle is processed next.

Move Vehicles

After all of the vehicles in the network traffic queue are processed and their desired movements identified, the network traffic queue is processed a second time to physically move the vehicles, resolve any conflicts that might result, and generate output summarizes. At the end of this process the current position of each vehicle is updated in the vehicle records and the cell grids are switched and cleared for the next time step.

The first check is to ensure that the vehicle is still on the network. If the vehicle is to be pulled off the network and parked, the program completes the traveler output, finishes the link summary calculations, and generates end-time and trip duration events. It then places the vehicle in the parking lot and activates any travelers that are waiting for the vehicle to arrive. If the vehicle is a transit bus or train, any remaining passengers are offloaded. The vehicle is removed from the network travel queue and the memory allocated for the travel plan is released.

Otherwise the traveler output, snapshots, summary statistics, and speed bin data are updated based on the before and after states of the vehicle. If the vehicle has a lane change request, it is added to the corresponding lane changing queue. If the vehicle moved, the after state is copied to the before state to prepare the vehicle for the next time step. If the vehicle did not move, the wait time counter is incremented and compared to the minimum waiting time parameter. When the values are equal, the vehicle is added to the priority queue.

After the network traffic queue is processed, the program checks if any vehicles are still active. If the network includes active vehicles, the cell grids are switched and the after grid is cleared. If there are no non-transit vehicles on the network and no travel plans waiting to be processed, the simulation terminates.

Exit Reporting

After all travelers have been processed or the simulation end time is reached, the output closeout routines for snapshots, summaries, occupancy data, problem links, turning movements, and speed bins are executed. The transit ridership file is then written, the processing summary statistics are printed, and the problem report is generated.

Sample Printout

A sample printout file generated by the **Microsimulator** program is shown below. It is an ASCII text file with a maximum of 95 characters per line and 65 lines per page. The file can be viewed or printed using a variety of text editors. For best results in a word processor, use a 10-point Courier font and 0.5 inch margins on all sides.

```
*****************************************
|                                       |
|     Microsimulator - Version 4.0.75     |
|   Copyright (c) 2009 by AECOM Consult   |
|         Tue Apr 20 11:11:35 2010        |
|                                       |
*****************************************

Control File = Microsimulator.ctl
Report_File  = Microsimulator.prn (Create)

Case3 Microsimulation

Project Directory = ../

Network Directory = ../network/
Node File = ../network/Node
Link File = ../network/Link
Pocket Lane File = ../network/Pocket_Lane
Lane Connectivity File = ../network/Lane_Connectivity
Parking File = ../network/Parking
Activity Location File = ../network/Activity_Location
Process Link File = ../network/Process_Link
Unsignalized Node File = ../network/Unsignalized_Node
Signalized Node File = ../network/Signalized_Node
Phasing Plan File = ../network/Phasing_Plan
Timing Plan File = ../network/Timing_Plan
Detector File = ../network/Detector
Signal Coordinator File = ../network/Signal_Coordinator
Transit Stop File = ../network/Transit_Stop
Transit Route File = ../network/Transit_Route
Transit Route File Format = SPACE_DELIMITED
Transit Schedule File = ../network/Transit_Schedule
Transit Driver File = ../network/Transit_Driver

Vehicle Type File = ../demand/Vehicle_Type

Vehicle File = ../demand/Vehicle
Vehicle File will be Sorted by Vehicle ID

Plan File = ../demand/TimePlan
Plan Files contain Link List Paths

Cell Size = 7.50 meters
Time Steps per Second = 1

Time of Day Format = 24_HOUR_CLOCK
Simulation Start Time = 0:00
Simulation End Time = 1@3:00

Speed Calculation Method = CELL-BASED

Plan Following Distance = 525 meters
Look Ahead Distance = 260 meters
Look Ahead Lane Factor = 4.00
Look Ahead Time Factor = 1.00
Maximum Swapping Speed = 37.5 meters/second
Maximum Speed Difference = 7.5 meters/second
Parking Lanes will be Enforced
```

```
Driver Reaction Time = 0.70 seconds
Permission Probability = 50.0 percent
Slow Down Probability = 10.0 percent
Slow Down Percentage = 10 percent
Random Number Seed = 1122687672
Case3 Microsimulation
Tue Apr 20 11:11:35 2010  Microsimulator  page 2


Minimum Waiting Time = 2 seconds
Maximum Waiting Time = 10 seconds
Maximum Arrival Time Variance = 60 minutes
Maximum Departure Time Variance = 60 minutes

Output Traveler Range #1 = 1, 2
Output Traveler File #1 = ../results/Traveler
Output Traveler File #1 Format = TAB_DELIMITED

Output Snapshot File #1 = ../results/Snapshot
Output Snapshot File #1 Format = TAB_DELIMITED
Output Snapshot Time Format #1 = 24_HOUR_CLOCK
Output Snapshot Increment #1 = 0:15
Output Snapshot Time Range #1 = 0:00..1@3:00
Output Snapshot Link Range #1 = 2..10, 14..16, 18, 20

Output Summary File #1 = ../results/Performance
Output Summary File #1 Format = TAB_DELIMITED
Output Summary Time Format #1 = 24_HOUR_CLOCK
Output Summary Increment #1 = 0:15
Output Summary Time Range #1 = 0:00..1@3:00
Output Summary Link Range #1 = 2..10, 14..16, 18, 20

Output Summary File #2 = ../results/LinkDelay
Output Summary Increment #2 = 900 seconds
Output Summary Time Range #2 = 0..86400 seconds

Output Problem Type #1 = WAIT_TIME
Output Problem File #1 = ../results/Problem_Link
Output Problem File #1 Format = TAB_DELIMITED
Output Problem Filter #1 = 1 problem
Output Problem Time Format #1 = 24_HOUR_CLOCK
Output Problem Increment #1 = 0:15
Output Problem Time Range #1 = 0:00..1@3:00
Output Problem Link Range #1 = 2..10, 14..16

Output System Event Type #1 = TIMING_CHANGE, PHASE_CHANGE
Output System Event File #1 = ../results/Timing
Output System Event Time Format #1 = 24_HOUR_CLOCK
Output System Event Time Range #1 = 0:00..24:00

Output Event Type #1 = START_TIME, END_TIME
Output Event File #1 = ../results/Event
Output Event Filter #1 = 60 seconds


Number of Node File Records = 21

Number of Link File Records = 24
Number of Directional Links = 48

Number of Pocket Lane File Records = 72

Number of Lane Connectivity File Records = 240
Number of Lane Connectivity Data Records = 120

Number of Parking File Records = 96

Number of Activity Location File Records = 96
```

```
Number of Transit Stop File Records = 16
Case3 Microsimulation
Tue Apr 20 11:11:35 2010  Microsimulator  page 3


Number of Process Link File Records = 224

Number of Unsignalized Node File Records = 20
Number of Traffic Control Signs = 6

Number of Timing Plan File Records = 48

Number of Signal Coordinator File Records = 9

Number of Signalized Node File Records = 10

Number of Detector File Records = 72

Number of Phasing Plan File Records = 252

Number of Transit Route File Records = 20
Number of Transit Route Data Records = 2

Number of Transit Schedule File Records = 40

Number of Transit Driver File Records = 17
Number of Transit Driver Data Records = 2

Number of Vehicle Type File Records = 8

Number of Vehicle File Records = 6400
Number of Vehicle Cells = 6400

Number of Link Cells = 13848

Number of Signalized Intersections = 7
Number of Signal Controlled Movements = 112

Number of Traffic Control Signs = 6
Number of Sign Controlled Movements = 18

Number of Uncontrolled Left Turn Movements = 2

Transit Vehicle ID = 1xxx#
Number of Transit Vehicles = 5

Simulation Started at Time 0:00
Simulation Ended at Time 1:07:34

Number of Plan Files = 1
Number of Input Plans = 20798
Number of Input Records = 138806
Number of Input Travelers = 7279
Number of Input Trips = 6590

Number of Vehicle Trips Processed = 6405
Number of Vehicle Trips Started   = 6405
Number of Vehicle Trips Completed = 3655

Number of Transit Legs Processed = 190
Number of Transit Legs Started   = 41
Number of Transit Legs Completed = 1
Case3 Microsimulation
Tue Apr 20 11:11:37 2010  Microsimulator  page 4


Number of Priority Movements = 3138
Number of Reserved Movements = 168
```

```
Number of Required Lane Changes = 4378
Number of Swapping Lane Changes = 6
Number of Reserved Lane Changes = 107
Number of Look-Ahead Lane Changes = 12334

Number of Travelers with Problems = 2745 (37.7%)

Total Number of Problems = 2750
Number of Wait Time Problems = 2750 (100.0%)

Tue Apr 20 11:11:37 2010 -- Process Complete (0:00:02)
```