Installation and Testing

This document outlines the installation and testing of TRANSIMS software on Windows and Linux computer systems. Three case study applications are provided for testing the installation and providing an overview tutorial of basic TRANSIMS concepts.

Software Installation

The accompanying CD includes two root directories. One for Windows installations called "windows" and the second for Linux installations called "linux". Install the software and data files by copying all files and subdirectories from the CD to an appropriate location on the hard disk.

Windows

Double click "My Computer" on the Windows desktop and select the CD drive. You should see two directories labeled "windows" and "linux". Right click the "windows" directory and select "Copy". Click the "Back" button at the top of the screen and select the disk drive you wish to use for TRANSIMS applications. You may wish to create a new folder labeled "TRANSIMS" on this drive to avoid any confusion. Once the drive or directory is selected, right click the location and select "Paste". This will copy the "windows" directory and all of its contents to the selected location.

Linux

TRANSIMS will typically be installed on Linux in a "transims" directory under the user's home directory. To do this, change to your home directory and create the "transims" directory using the make directory command:

```
cd  $HOME
mkdir  transims
```

Then change to the CD directory. In most cases this can be done with one of the following commands:

```
cd  /mnt/cdrom
```

or

```
cd  /media/disk
```

If this does not work, it is likely that the CD is not mounted. Contact a system administrator to mount the CD drive.

Once you get access to the CD, copy all of the Linux files to the "transims" directory:

```
cd  –r  linux  $HOME/transims/.
```

Overview

This section provides a general overview of directory organization, program execution, and text editing.

Directory Structure

Both the Windows and Linux installations include three subdirectories. These are "bin" for binary programs, "doc" for documentation, and "test" for case study applications. Within the "test" directory are three more directories "case1", "case2", and "case3". Each of the case study directories includes four subdirectories. These are labeled "control", "demand", "network", and "results". The "control" directory includes a control file for each program and a batch file for running the program. The "demand" directory contains the input trip tables and diurnal distribution curves used to generate the trips. The "network" directory contains the input network link and node files.

After the programs are executed, the "control" directory will include printouts for each program; the "demand" directory will include household data and travel plans; the "network" directory will include a full set of TRANSIMS network files; and the "results" directory will include the output results from the microsimulation. The "network" directory includes a subdirectory call "arcview" were the ArcView Shape files are stored.

The control files have been setup to work in both Windows and Linux. In this document and in the control files we use the forward slash ("/") to represent directory levels. Windows typically uses a back slash ("\") for this purpose, but works equally well with a forward slash. Linux only recognizes the forward slash ("/"). The only exception to this rule is in the batch files (*.bat). In this case MS-DOS requires a back slash ("\") to identify directories.

To simplify these tests we have elected to use back directory syntax ("../../") to reference files in different directories. Many users prefer to include the full path name in the control files as a way of documenting file locations and processing steps more completely. We use back directory syntax in this case to enable the user to install the software any place they choose without invalidating the test setups.

Program Execution

The user runs all of the programs from the "control" directory. The programs are "console" applications. This means that they run from the command prompt. Windows will automatically open a command window when the user double clicks on the MS-DOS batch file (*.bat). Alternatively, the user can open an MS-DOS window, navigate to the control directory, and enter the name of the program or batch file at the command prompt. In most cases, Linux applications are run from the command line.

Note that all Linux applications are case sensitive and Windows applications are not. This means that Linux commands need to include the exact upper or lower case characters used for the program or file. We have elected to use all lower case for directory names; title case for programs and data files; and all upper case for field names.

Text Files

The data and control files used in TRANSIMS are unformatted or TAB-delimited text files. They can be edited and reviewed using standard text editors or spreadsheet software. Windows provides Notepad and WordPad for this purpose. Most Linux users have a favorite text editor they prefer (e.g., vi, pico, gedit, etc.). This document assumes the user knows how to use a text editor.

Note that text files in Windows are not the same as text files in Linux. The MS-DOS convention for text files is to end each line with a carriage return-line feed. In Linux only a new line code is used. Most TRANSIMS programs have been coded to work around these differences. Text editors tend to be more sensitive. For example, if Notepad is used to open a text file generated in Linux, it will fail to recognized the new line commands and display the results as one long text string. WordPad will read the file properly. If the file was created in Windows and copied to Linux, the Linux text editors will typically show an extra character at the end of each line. These characters can be deleted or ignored.

Testing

The installation can be tested by running "case1". Navigate to the "control" directory for the "case1" test. In Windows you can use the Windows Explorer or "My Computer" to change directories to "../windows/test/case1/control". In Linux you use the changed directory command from the installation directory.

```
cd  linux/test/case1/control
```

To test the installation, run the TransimsNet.bat file to start the network conversion. In Windows double click on the TransimsNet.bat file. In Linux type the following at the command prompt:

```
./TransimsNet.bat
```

If everything works as it should, the program will be finished in less than one second and a printout file called TransimsNet.prn will be added to the "control" directory. If the program is not found or can not be executed, the printout file will not exist. In this case the path to the program or file access permissions need to be changed. If the program executes, but finds an error in the control file, it will display an error message and pause execution until the user hits a key on the keyboard. At this point, the error message is written to the printout file and the program is terminated.

If the program does not execute in Linux, it is likely it does not have execute permissions or the user permissions don't permit writing to the case study directories. The following command can be used to change the permissions on the programs in the "bin" directory:

```
cdmod  777  *
```

Test Case #1

Assuming the installation test was successful, the full Test Case #1 can be executed. A seven step process is used for this simple application. The steps should be executed in order to ensure that the required data files are available. All of these steps are executed from the "test/case1/control" directory. The control directory contains 7 control files (*.ctl) and 9 batch files (*.bat). There is one batch file to execute each program and two batch files for executing a series of programs. NetPrep.bat prepares the network files. RunAll.bat executes a model application starting with trip table conversion and ending with a microsimulation.

Step 1: Network Conversion

The first step is to convert the input link and node files found in the "network" directory to a full set of TRANSIMS network files. The TransimsNet program can be used to synthetically generate these files from basic network information. Begin by reviewing the TransimsNet control file (TransimsNet.ctl) found in the "control" directory using a text editor. This file looks like this:

```
TITLE                              Case1 Synthetic TRANSIMS Network
NET_DIRECTORY                      ../network/
NET_NODE_TABLE                     Input_Node
NET_LINK_TABLE                     Input_Link
INPUT_ZONE_FILE                    ../network/Input_Zone
KEEP_NODE_LIST                     ../network/Keep_Nodes
NEW_DIRECTORY                      ../network/
NEW_NODE_TABLE                     Node
NEW_LINK_TABLE                     Link
NEW_ACTIVITY_LOCATION_TABLE        Activity_Location
NEW_PARKING_TABLE                  Parking
NEW_PROCESS_LINK_TABLE             Process_Link
NEW_POCKET_LANE_TABLE              Pocket_Lane
NEW_LANE_CONNECTIVITY_TABLE        Lane_Connectivity
NEW_UNSIGNALIZED_NODE_TABLE        Sign_Warrants
NEW_SIGNALIZED_NODE_TABLE          Signal_Warrants
POCKET_RANGE_FOR_FACILITY_1        100, 400
POCKET_RANGE_FOR_FACILITY_2        60, 200
POCKET_RANGE_FOR_FACILITY_3        40, 100
POCKET_RANGE_FOR_FACILITY_4        30, 60
POCKET_RANGE_FOR_FACILITY_8        30, 60
SIGNAL_WARRANT_FOR_AREA_TYPE_1     COLLECTOR, LOCAL
SIGNAL_WARRANT_FOR_AREA_TYPE_2     COLLECTOR, COLLECTOR
SIGNAL_WARRANT_FOR_AREA_TYPE_3     MINOR, COLLECTOR
SIGNAL_WARRANT_FOR_AREA_TYPE_4     MINOR, MINOR
SIGNAL_WARRANT_FOR_AREA_TYPE_5     MAJOR, MINOR
SIGNAL_WARRANT_FOR_AREA_TYPE_6     MAJOR, MAJOR
SIGNAL_WARRANT_FOR_AREA_TYPE_7     PRINCIPAL, MAJOR
SIGNAL_WARRANT_FOR_AREA_TYPE_8     PRINCIPAL, PRINCIPAL
MAXIMUM_ACCESS_POINTS              3
MINIMUM_SPLIT_LENGTHS              100, 200, 300, 300, 300, 300, 300, 300
MINIMUM_LINK_LENGTH                37.5
MAXIMUM_LENGTH_TO_XY_RATIO         1.2
INTERSECTION_SETBACK_DISTANCE      0.0
```

TRANSIMS control files are unformatted text files that contain a list of control keys and parameters. In general the key order is not important. The only exception to this rule is

when the same key is entered multiple times. In this case, the last instance of that key will be used by the program. All keys are upper case. The key must be the first value on a line. A space or tab is used to separate the key from the parameters that follow. All parameters must be on the same line as the key. If a key can accept multiple parameters, it will end with a parameter number (e.g., `POCKET_RANGE_FOR_FACILITY_4`). In these cases, any number of parameters can be provided and the numbers do not need to be in sequential order. The control file can include comment statements and keys that the program does not require.

In this example, the Input_Link, Input_Node, Input_Zone, and Keep_Nodes files are read from the "network" directory. The output TRANSIMS network includes:

Node	node numbers and coordinates
Link	two-way link attributes
Activity_Location	locations on links where activities take place
Parking	locations on links where vehicles are parked
Process_Link	links connecting parking lots to activity locations
Pocket_Lane	turn and merge pockets at intersections
Lane_Connectivity	lane connections between links at intersections
Sign_Warrants	approaches where stop and yield signs are warranted
Signal_Warrants	intersections where traffic signals are warranted

The rest of the keys are used to control the conversion algorithm.

To execute the TransimsNet program the user can enter the path to the executable program (in the "bin" directory) and hit enter or follow the program name with the name or path to the program control file. If the user hits enter, the program will prompt the user for the control file name. A batch file is a short hand way of entering the path to the program and the control file name. Use a text editor to review the contents of one of the batch files. The batch file can be executed in Windows by double clicking the batch file name. In Linux the command line syntax is:

```
./TransimsNet.bat
```

In most applications, the analyst would review the printout and synthetic files to determine if the conversion process accurately replicates the network. You should open the printout file (TransimsNet.prn) and several of the output network files using a text editor to see the types of information that is generated.

Step 2: Intersection Traffic Controls

The network conversion is not complete until the intersection traffic controls are created. The TransimsNet program generates sign and signal warrants that the user can review and edit prior to executing the IntControl program to complete the conversion process. This program creates the following files:

Unsignalized_Node	approaches with stop or yield signs or no controls

Signalized_Node	nodes with traffic signals
Timing_Plan	timing plan for each signal phase
Phasing_Plan	lane connections included in each signal phase
Detector	location of detectors for actuated signals
Signal_Coordinator	signal coordination codes

The IntControl program is executed using the IntControl.bat file. Double click the batch file name in Windows or type the appropriate Linux command:

```
./IntControl.bat
```

In this case the program runs, but an error message is generated. The program pauses execution, beeps, and displays the message:

```
Program Terminated Due to Errors
```

The program printout and screen output includes warning messages for about 10 node approaches where intersection controls were not provided. Hit any key on the keyboard to terminate execution. You can then use a text editor to review the warning messages in the printout file (IntControl.prn) in more detail.

In this particular case these are not real error messages. The program automatically adds a "no control" record to the Unsignalized_Node file for each of these approaches. The program is simply making the user aware of a potential problem before proceeding.

Step 3: ArcView Network Files

The third step is optional, but recommended. This step uses the ArcNet program to convert the TRANSIMS network files to ArcView Shape files so they can be reviewed graphically using ArcView, ArcGIS, or other software that can display Shape files.

Run the ArcNet.bat file by double clicking the file name in Windows or entering the Linux command:

```
./ArcNet.bat
```

This program convert the network files found in the "network" directory to Shape files in the "network/arcview" directory. Each Shape file includes three files: binary shape and index files (*.shp and *.shx) and a dBase file (*.dbf). In addition, ArcNet creates a Definition file (*.def) for each dBase file. You will also notice that each of the TRANSIMS network files in the "network" directory now have an associated Definition file as well.

Definition files are new with Version 4 of the TRANSIMS software. Since ArcNet is the first Version 4 program included in this process, it creates Definition files for all of the files it reads and writes. Definition files are then used by subsequent programs to identify the fields and data types included in the file. Open one of the Definition files using a text

editor to see what it contains. Definition files enable the TRANSIMS software to work with a variety of file types including the dBase files in the "arcview" directory.

If you have access to ArcView or ArcGIS software, go ahead and start up the software and open an empty map. Select the "Add Layer" button, navigate to the "test/case1/network/arcview" directory and select all of the shape files you see. The program will warn you of an "unknown spatial reference". Select "OK" and continue. The coordinates used for this case study do not correspond to a real world coordinate system.

The graphic will show a straight freeway section with one on-ramp and one off-ramp. There are external stations at the ends of each facility. These are represented as an activity location and parking lot on both sides of the link. These locations are connected by a Process Link. If you increase the line width for the Pocket Lane features, it will be easier to see the merge and diverge pocket lanes that were added at the ramp junctions.

Step 4: Trip Table Conversion

The "demand" directory includes four simple trip tables in Origin-Destination format and four sets of trip time files that distribute these trips to specific times of day. These trips are converted to TRANSIMS Trip, Household, Population, and Vehicle files using the ConvertTrips program.

If you open the ConvertTrips control file (ConvertTrips.ctl) in the "control" directory, you can see an example of how comment statements and file groups are used within TRANSIMS. Below is an extract for the second file group. Notice that all of the keys in this file group end with the number "2". This is how TRANSIMS knows that all of these keys should be processed together.

```
#---- 2-3 demand ----

TRIP_TABLE_FILE_2             TripTable2
TRIP_TIME_FILE_2              TripTime2
TIME_CONTROL_POINT_2          ORIGIN
ORIGIN_WEIGHT_FIELD_2         USER1
DESTINATION_WEIGHT_FIELD_2    USER2
TRIP_PURPOSE_CODE_2           1
TRAVEL_MODE_CODE_2            2
AVERAGE_TRAVEL_SPEED_2        15
VEHICLE_TYPE_2                1
VEHICLE_SUBTYPE_2             0
```

Execute the ConvertTrips batch file and notice the new files that are created in the "demand" directory. ConvertTrips creates a separate household, population, and vehicle record for each trip in the trip tables. Each trip is assigned to a specific origin and destination activity location and a specific second of the day.

It is also helpful to review the ConvertTrips printout file to see if all of the trips were allocated. If a substantial number of trips are not converted, this generally implies that zones or zone interchanges in the input trip tables can not be associated with activity

locations in the TRANSIMS network. Each zone should contain at least two activity locations to allocate both intrazone and interzone trips. The ConvertTrips program lists zones that contain no activity locations and zones that contain only one activity location. The user should review these locations to determine if any adjustments or corrections can be made.

Step 5: Travel Plans

In TRANSIMS terminology, a travel path for a given trip is constructed using the Router. The output of the routing process is a Plan file. A Plan file specifies the time and duration of activities and the travel legs used to move between activities. Since all of the trips created for this application are automobile trips, the Plan file includes three travel records for each trip. The first record represents a walk trip from the origin activity location to the parking lot where the vehicle is parked. The second record is the vehicle trip between the origin parking lot and the destination parking lot. The third record is the walk trip from the parking lot to the destination activity location.

The Router is used to construct this travel plan based on a time-dependent shortest path algorithm. The Router control file specifies the path building methods and the link travel times by time of day. For this case study, a simple minimum time path based on free flow speeds is used. Since this network includes no path options, the Router is only converting files from one format to another. If there are trips that can not be constructed, they would be listed in the Problem file (Route_Problem) in the "results" directory and summarized at the end of the Router printout file.

Step 6: Plan Sorting

The Router constructs travel plans for a given household. The trips for a household are randomly assigned to a specific time of day during the trip table conversion process. The Microsimulator, on the other hand, process trips based on the time of day when the trip starts. A plan sorting step is used to reorganize the plans by time of day.

The PlanPrep program is used to sort the plans. It reads the Plan file, sorts the trip legs by time of day, and outputs a new Plan file (TimePlan). Run the PlanPrep by double clicking on the batch file in Windows or entering the following command at the Linux prompt:

```
./PlanPrep.bat
```

Take a few minutes to review the difference between the Plan file and the TimePlan file. In the Plan file it is relatively easy to identify the three legs of the travel plan for a given traveler. The first number in each group of records is the traveler ID. It is equal to the Household ID times one hundred plus the Person number within the household. In other words, traveler 201 is person 1 within household 2. Notice at the end of the second travel leg a variable length list of numbers. These are the node numbers the path travels through between the origin parking lot and the destination parking lot. The first number on the second line of each plan is the time of day when the travel leg is scheduled to start. This is the value used to sort the records in the TimePlan file.

Step 7: Microsimulation

The final step in the process is to simulate the network performance given the travel plans. TRANSIMS uses a second-by-second cell-based simulation procedure to track people and vehicles through the network. The Microsimulator identifies vehicle conflicts, performs lane changing maneuvers, cycles the traffic signals, and calculates traffic throughput and travel speeds. It also generates a variety of output files that document the status and performance of the system. The user defines the types of output, the summary criteria, and the file names in the Microsimulator control file. These files are stored in the "results" directory.

Eight types of output files can be generated by the Microsimulator. These include:

1. Problem File – Like the Router, the Microsimulator identifies problems that occur during the simulation, summarizes these problems by problem type, and outputs information about the problem trips to a problem file. The user can review this file to identify travelers or locations that need special attention.

2. Problem Link Files – For large applications, the Problem file can be huge. It contains individual records of each traveler that has a problem. A problem link file summarizes this information for selected time periods, locations, and problem types. This makes it much easier to focus on specific locations or times of day where significant numbers of problems exist.

3. Traveler Files – The traveler output files enables the user to track the second-by-second movements of selected travelers. The data include the time, link, lane, distance and speed of each vehicle movement.

4. Snapshot Files – Snapshots record the locations and speed of all vehicles in the network at specific increments of time. This information is typically used in the Output Visualizer to depict changes in speed and congestion levels over the course of a day. The user can control the snapshot frequency as well as the time duration and location where snapshots are recorded.

5. Link Delay Summary Files – Summary files aggregate information about network links at regular increments of time. A Link Delay file is used to record the volume and travel time of traffic on links in 15 minute time periods throughout the day. This information is then used by the Router to build time-dependent travel paths.

6. Performance Summary Files – The performance summary file provides an alternate way of presenting the link performance information by time of day. This format includes the average volume, the entering and exiting volume, the average speed, the average travel time, the average delay, the average density, the maximum density, the travel time ratio, the vehicle miles of travel, and the vehicle hours of travel for each links by time of day.

7. Event Files – Event files are used to record when things happen during the simulation. The current implementation permits the user to monitor the start time and end time of each trip. The report lists when the trip was scheduled by the Router to experience an event, the actual time when the event took place in the simulation, and the difference between the two. The output can be filtered to only include large variances between scheduled and actual times.

8. System Event Files – System events are changes to the network by time of day. These include traffic signal changes from one phase to another or from one timing plan to another. They also include changes in lane use or turn prohibitions by time of day. The user will typically output these events for selected time periods or locations.

Batch Execution

As mentioned earlier the "control" directory includes two additional batch files: NetPrep.bat and RunAll.bat. These files demonstrate how a series of processing steps can be chained together to automate the execution process. The NetPrep.bat executes steps 1 through 3 of the case study while RunAll.bat executes steps 4 through 7. Go ahead and execute the NetPrep batch process. This is done by double clicking the NetPrep.bat file name in Windows or typing the Linux command:

```
./NetPrep.bat
```

As before the IntControl step generates an error message that requires the user to hit any key to continue the execution. Also execute the RunAll process.

Now try to make a change to the travel demand or change one of the Microsimulator control parameters and run the process again. Load the results of the original run and the modified run into a spreadsheet and identify the differences.

For example, load the original performance file into a spreadsheet. Since the file is a TAB-delimited text file with header records, it should load into a spreadsheet without any difficulty. Save the spreadsheet using a new file name. Now open one of the trip table files in the "demand" directory (e.g., TripTable1) and modify the number of trips on one of the interchanges. Save the changes and execute the RunAll batch file to generate the results. Load the new performance file into the spreadsheet and compare the differences in a graph or table. Now open the Microsimulator control file (Microsimulator.ctl) in the "control" directory. Change the `SLOW_DOWN_PROBABILITY` from 10 to 25, save the file, and re-run the Microsimulator. Load the new performance results into the spreadsheet and compare the differences in a graph or table.

[Note that a file called case1.zip is included in the "test" directory. This file can be used to restore the Test Case #1 directories to their original data files.]

Test Case # 2

This case study will demonstrate additional conversion features and display options. In this case we will be using a real world interchange from Arlington, Virginia. It is the intersection of US Route 50 and Washington Boulevard. It demonstrates how a network provided as an ArcView Shape file can be converted to a TRANSIMS network and used for microsimulation. All of the steps discussed below are executed from the "test/case2/control" directory.

Step 1: GIS Network Conversion

The "test/case2/network/arcview" directory contains the ArcView Shape file Input_Link.shp. It also includes the associated dBase, projection, and index files generated by a standard ArcGIS application. If you have access to ArcView or ArcGIS, open the program and load this shape file. You should see a curved freeway with cloverleaf ramps and a few local streets. If you review the attributes of one of the links, you will see that the field names are different from the standard TRANSIMS fields. This step will use the GISNet program to convert these data to a format that TransimsNet can read.

Begin this process by reviewing the GISNet control file (GISNet.ctl) located in the "control" directory. The GIS_LINK_FILE key is used to identify the input GIS file. The output of this process is three TRANSIM files called Input_Node, Input_Link, and Input_Shape. These will become the input files for the TransimsNet program.

The control file contains three additional concepts of special interest. These are the conversion script, the coordinate keys, and reports. Each of these are discussed below:

User Program Scripts

The GISNet program uses a user program script to convert the data in the GIS file to the data needed by TransimsNet. The Script.txt file is included in the "control" directory for this purpose. Open this file with a text editor and review its contents. This is a simple script that maps the field names found in the GIS file to the field names included in a standard TRANSIMS link file. The return code of "1" tells the GISNet program to save the link record.

The user program concept is capable of significantly more complex data processing. The UserProgram.pdf file provided in the "doc" directory describes these capabilities in more detail. User programs are the standard interface for custom processing within the TRANSIMS Version 4 environment.

Coordinate Conversion

The coordinate conversion keys in the GISNet control file are commented out for this particular application. If the GIS Shape file was not in UTM coordinates, the four keys shown in the control file could be used to convert Stateplane coordinates in feet for the Washington D.C. area to the corresponding UTM coordinates in meters. The ArcNet

program includes similar keys to enable the user to convert the TRANSIMS UTM coordinates back to the local Stateplane system.

Reports

Many of the TRANSIMS Version 4 programs include report options. In all cases the report keys use the same syntax. They start with the program name in upper case followed by _REPORT_# where the # is an arbitrary report number. The specific report is identified by an upper case text string listed in the program documentation. In this example, two reports were requested. The first lists the records found in the conversion script file in the program printout file. The second report shows how the script statements were converted to a user program command stack.

Execute the GISNet program using the batch file. In Windows this can be done by double clicking the batch file name (GISNet.bat). In Linux the following command is used:

```
./GISNet.bat
```

Open the printout file (GISNet.prn) to review the conversion results and the reports.

Step 2: Create the TRANSIMS Network

The results of the GISNet application are standard TRANSIMS link, node, and shape files. These files contain most, but not all of the information TRANSIMS needs. The TransimsNet program is used to add the additional information and generate the other required files.

Execute the TransimsNet batch file by double clicking the batch file name (TransimsNet.bat) in Windows or entering the Linux command:

```
./TransimsNet.bat
```

Step 3: ArcView Network Files

Use the ArcNet program to generate ArcView Shape files for the new TRANSIMS network. If you wish, you can display these files on top of the original input Shape files or aerial photos to confirm that the conversion is correct and accurate. For this to work properly, you should set the spatial referencing system for each of the new Shape files. The ArcToolbox in ArcGIS is typically used to define the coordinate system for each file. The coordinate system should be UTM 18N (in meters).

Step 4: Trip Table Conversion

The "demand" directory includes one trip table and trip time file for this case study. A moderate number of trips are defined between each of the external stations. The ConvertTrips program is used to convert these trips to TRANSIMS trip, household, population, and vehicle files.

Execute the ConvertTrips batch file (ConvertTrips.bat) as previously described.

Step 5: Travel Plans

The Router is then used to construct travel plans for each trip. Execute the Router batch file (Router.bat) to generate the Plan file. The program will construct the plans and terminate normally. This does not, however, necessarily mean that all of the plans were constructed successfully. It is always advisable to check the printout file (Router.prn) to understand the types of routing problems that may have occurred.

At the end of the Router printout file you should find the following information:

```
Number of Trip Records = 10700
Number of Trips Processed = 10700
Number of Households Processed = 10700
Number of Vehicle Trips Saved = 3742

Number of Output Plans = 18184
Number of Output Records = 116850
Number of Output Travelers = 10700
Number of Output Trips = 3742

Total Number of Problems = 6958
Number of Path Building Problems = 6958 (100.0%)
```

This means that 10,700 trips were read from the input trip file and all of these trips were processed by the Router. Unfortunately only 3,742 of these trips were saved in the Plan file. A total of 6,958 trips had routing problems. All of the problems were related to path building. In other words, a path could not be found between the origin and destination. The Route_Problem file in the "results" directory lists each of the problem trip records. The first one hundred or more problems have origin activity locations 1 or 2 and destination activity locations 5 or 6. Let's try to determine why paths could not be built for these trips.

For a small network like this it is relatively easy to identify the problem by looking at the network in ArcView or ArcGIS. Activate the link and activity location layers in the TRANSIMS network. The process can be further simplified by labeling the activity location dots using the ID field. You will then see that activity locations 1 and 2 are attached to the outbound link on US Route 50 (Arlington Boulevard) on the Southwest side of the screen. Activity locations 5 and 6 are on the two-way segment of Washington Boulevard to the Northwest. Since these trips start on a one-way link heading outbound and no U-Turn lane connectivity is provided at node 1, there is no way to build a path that will connect activity locations 1 or 2 to 5 or 6 using this network.

Now the question might be, how did these trips get created in the first place? The trips were created by the ConvertTrips program based on the trip table found in the "demand" directory. If you look at the trip table, you will see there are no trips listed between 1 or 2 and 5 or 6. That is because 1, 2, 5, and 6 are activity location numbers and the values in the trip table are zone numbers. The activity locations are mapped to zone numbers in the activity location file. In the activity location file you will find that activity locations 1, 2, 3, and 4 are mapped to TAZ 1 and activity locations 5 and 6 are mapped to TAZ 2.

The trip table includes 500 trips between zone 1 and 2. All of the trips that start at activity locations 3 and 4 were successfully routed to activity locations 5 and 6.

This demonstrates the need to review the network coding and correct situations that generate unexpected results. The ConvertTrips program randomly assigned trips to and from zone 1 to activity locations 1, 2, 3, and 4 without regard to the access issues caused by the one-way streets. This problem can be resolved in a number of ways. The simplest is to change the USER1 and USER2 fields for these locations in the activity location file.

If you look in the ConvertTrips control file, you will see that USER1 is used to distribute trip origins and USER2 is used to distribute trip destinations. By changing the USER1 field for activity locations 1 and 2 to zero, these locations will no longer be available for trips originating at zone 1. Likewise, the USER2 field for activity locations 3 and 4 can be set to zero to eliminate these locations as trip destinations. Just making these two changes will increase the number of routed trips to 5,551. Similar corrections can be implemented at the other external stations.

Step 6: ArcView Plan Files

Another approach to addressing routing problems is to visualize the paths using ArcPlan. ArcPlan constructs a shape record for each leg in a travel plan. The program provides a number of ways to select plans of interest including selecting plans by traveler, time of day, and link or node sequences. In this case the plan file is small enough that we will create graphics for all of the plans.

Execute the ArcPlan batch file (ArcPlan.bat) and load the results on top of the network shape files created by ArcNet. Increase the line width for the ArcPlan layer so the paths stand out. In this case you will see paths use most of the links. The exceptions are the tails of each link at the external stations and the two local streets in the Northwest quadrant. These links serve activity locations 15, 16, 17, and 18, but there are no trips going to or from these locations. If you review the data fields for these locations, you will see that the TAZ values are zero. Since the TAZ values are zero, no trips will be assigned to these locations.

Now open the attribute table (dBase file) for the ArcPlan layer and look at the data provided. In this case each traveler has three records: the initial walk to the parking lot, the drive between parking lots, and the walk to the final destination. Highlight the three records for traveler 401 and review the path that is highlighted on the map. The path shows the access links at both ends and the loop ramp that is required to travel between parking lots 3 and 6. In this way the user can review selected paths through the network and judge if the path looks reasonable or points to a potential network problem.

Step 7: Plan Sorting

Once the user is satisfied that the plans are reasonable, the process of microsimulation can begin. This starts by sorting the travel plans by time of day using the PlanPrep program. Execute the PlanPrep batch file (PlanPrep.bat) to generate the TimePlan file expected by the Microsimulator.

Step 8: Microsimulation

The Microsimulator can now be applied. Executed the Microsimulator batch file (Microsimualtor.bat) and review the output results. If you did not correct the problems with the trip conversion or routing, the results will show significantly less traffic on the facilities than expected.

Test Case #3

The first two case studies used simple freeway-related networks with auto traffic from external stations. This case study focuses on a 16 block arterial grid. Each block is 1000 meters long and contains two activity locations on each side of each block. Each block also represents a separate zone in the trip table.

There are nine intersections in the network. Seven have traffic signals, one is two-way stop controlled, and the last is four-way stop controlled. Of the seven signalized intersections, one is fixed timed, four are four phase actuated, and two are eight phase dual actuated. One of the dual actuated signals also changes phasing plans every 15 minutes.

In addition to the complexity represented by the traffic control system, this case study includes two transit routes and a transit trip table. The routes wind through the arterial network and provide transfer opportunities. Transit stops are included on some blocks, but not others. One route makes three runs on a 15-minute headway. The other route makes two runs on a 15-minute headway.

This network was original generated using the TransimsNet program, but was subsequently hand adjusted to include the various traffic controls and the transit routes. As a result, this case study starts with a complete TRANSIMS network and proceeds through the analysis process. All of the steps discussed below are executed from the "test/case3/control" directory.

Step 1: ArcView Network Files

Start by creating an ArcView representation of the network. Execute the ArcNet batch file (ArcNet.bat) by double clicking the file in Windows or typing the Linux command:

```
./ArcNet.bat
```

Open ArcView or ArcGIS and load all of the shape files into the editor. You should see a simple grid network with two sets of activity locations and parking lots on each link. Make the pocket lane symbols thicker to see that turn pockets are included at each intersection. Now make the signalized node symbols larger to see where the traffic signals are located. Hide the pocket lane and signal layers to focus now on transit.

Select the transit stop layer and make the stop symbols somewhat larger. Now open the property page for the transit route layer, change the symbology to categories, and add all values for Route. Two route numbers will be shown: routes 100 and 200. Select a wider symbol width and a different color for each route, and apply the changes. The map

should now show the two transit routes more clearly. Zoom in on one of the intersections with multiple routes and stops to study the coding detail.

Notice that transit stops are setback from the intersection. The routes are displayed on the appropriate side of the street based on the direction of travel, and a small connection to each stop on the route is shown at the appropriate location. Transfers between routes will typically require a short walk in most TRANSIMS networks. Notice also that each stop is connected to a nearby activity location though a process link. This is not strictly necessary, but is a "traditional" TRANSIMS coding technique. We will see later in the case study that this coding technique has some drawbacks.

Step 2: Trip Table Conversion

Two trip tables and one trip time file are included in the "demand" directory. Each trip table includes trips from all 16 zones to all other zones, including intrazonal trips. A total of 6,400 automobile trips and 1,280 transit trips are generated by the ConvertTrips process. Execute the ConvertTrips batch file (ConvertTrips.bat) by double clicking the file name in Windows or entering the Linux command:

```
./ConvertTrips.bat
```

Review the printout file (ConvertTrips.prn) to ensure that all of the trips were allocated. There should be a total of 7,680 trips, households, and persons, but only 6,400 vehicles.

Step 3: Travel Plans

A number of changes were made to the Router control file for this application. Open the Router control file (Router.ctl) in the "control" directory with a text editor. The transit stop, route and schedule files are now included among the network files. Note that the `WALK_TIME_VALUE` is 20.0 and the `VEHICLE_TIME_VALUE` is 10.0. This means that time spent walking will be considered twice as onerous as time spent in a vehicle. Also note that the `MAX_WALK_DISTANCE` is 3000 meters (about two miles). The control file also tells the Router to ignore the time constraints, save walk path details, and record the path as a list of link numbers rather than a list of node numbers (i.e., `NODE_LIST_PATHS = NO`). There are many other parameters that can be defined to control the routing process. The Router.pdf file in the "doc" directory provides additional detail on all of these parameters.

Execute the Router batch file (Router.bat) and open the printout file (Router.prn). Notice that a number of other transit related parameters were included by default. The 60 minute wait time limit, a maximum of three transfers, the value of waiting time (first and transfer) is 20.0 impedance units, and the transfer penalty is zero. Since a transit fare file was not provided, the transit path is not affected by cost.

You will see at the end of the printout that 393 travelers had walk distance problems. If you look at the mode information included in the Route_Problem file in the "results" directory, you will see that all of these trips are mode 3 (i.e., transit) trips. This means that 393 transit trips required walks of 3000 meters or more to travel between their origin

and destination. In other words, the bus routes did not serve their trip interchange and the distance was too far to walk.

Step 4: ArcView Plan Files

Since this Plan file includes transit trips, it is important to review the transit paths to confirm that the path building parameters used in the Router are generating the desired results. Execute the ArcPlan batch file (ArcPlan.bat) by double clicking the file name in Windows or entering the Linux command:

```
./ArcPlan.bat
```

Now open ArcView or ArcGIS and add the network and ArcPlan shape files. Plan records will be shown on all links. To review specific paths, you can query a given traveler or route ID or you can open the attribute table and select various traveler records for review. Use this method to review a number of paths. Open the attribute table attached to the ArcPlan layer. The transit-related trips will be located at the end of the table. Scroll down to the end of the table and highlight the records for selected travelers. Review the highlighted path on the map.

For example, the last trip in the ArcPlan file is for traveler 768001. Notice that this traveler has only one travel leg and it is a walk leg. Since there is no bus route that serves this trip, the traveler walked all the way from the origin to the destination (about 1000 meters). The trip took 16 minutes and 40 seconds.

It is important to understand that even though this traveler was told to take transit for this trip, the Router created an all-walk path. In other words, the Router does not guarantee that a given trip will be completed using a specified mode. If the Router finds that a walk trip is faster (less impedance), it will choose the walk path.

Now look at the path constructed for traveler 766901. This trip includes five travel legs. It starts with two walk legs, then boards transit route 100 (i.e., the Mode ID value), and completes the trip with two more walk legs. Highlight each leg of the trip one at a time to trace the sequence of events.

The traveler starts by walking north for 11 minutes until it arrives at the activity location that is connected to a bus stop with a process link. The traveler waits for 6 minutes and 20 seconds for the bus to arrive at the bus stop. The bus trip takes 5 minutes. The person then follows the process link from the bus stop back to the activity location near the middle of the link. The traveler than walks back along the link and turns the corner to complete the trip to the activity location to the south.

This is a prime example of the types of illogical paths that can be created when transit stops are attached to mid-block activity locations. Approximately 11 minutes of unnecessary walking was added to the trip due to the type of access coding that was used. The process links appear relatively long on the map, but in reality the process link file only includes 30 seconds of travel time to use this link. This coding technique also

significantly complicates cross street transfers, because the path will force the trip to back track to the activity location, walk from that activity location to the activity location attached to the next transit stop, and then use the process link to board the next bus.

These and other examples like it are the reason significant care should be taken in coding transit networks to avoid unintended consequences in the path building process.

Step 5: Plan Sorting

Execute the PlanPrep batch file (PlanPrep.bat) by double clicking the file name in Windows or entering the Linux command:

./PlanPrep.bat

Step 6: Microsimulation

Execute the Microsimulator batch file (Microsimulator.bat) by double clicking the file name in Windows or entering the Linux command:

./Microsimulator.bat

There are two things about this Microsimulator application that are different from previous case studies and worth noting. One is the fact that transit vehicles are included in the simulation and the other is the System Event file.

Transit Vehicle Simulation

In the Version 4 network, the Transit Driver Plans have been replaced by a Transit Driver File. Open the Transit_Driver file in the "network" directory to review this new file format. The driver file is a nested data file. It includes two record types. The first record identifies the route number, the number of links on the route, and the vehicle type. This is followed by a list of link numbers. In this case the LinkDir format is used. In this format a positive link number represents the A→B direction on the link and a negative link number represents the B→A direction on the link. This is the same convention used in the Plan file for link-based paths.

The Microsimulator uses the transit driver file to construct the vehicle path through the network. It uses the transit route and stop files to identify where the bus stops to serve passengers. The transit schedule file identifies the time of day when each run starts and the time when the vehicle is permitted to leave the time points specified in the route file. When a new run starts, the Microsimulator synthesizes a transit vehicle, loads it on the network at the first transit stop, and tracks the vehicle through the network. Each transit run is given a unique vehicle number. The vehicle number is constructed based on the maximum number of highway vehicles included in the simulation, the route number, and the run number. The Microsimulator printout file documents the vehicle offset used to start the transit vehicle numbering. Any vehicle number larger than this value will be a transit vehicle. If the bus is stuck in traffic, these vehicle IDs will be reported in the problem file.

Traffic Signal Events

For this application, a System Event file was generated. Open the Timing file in the "results" directory to see the type of information that is generated for traffic signal events. Event type 1 is the Timing Change event and event type 0 is the Phasing Change event. Timing change events are used to initialize the signal cycles and change the timing plans by time of day. Open the Signalized_Node file in the "network" directory using a text editor. Notice that the timing plan for node 16 changes four times during the simulation. See if you can find these timing change events in the System Event file.

The System Event file also shows the time and result of each signal phasing decision. This includes changing the signal for a given approach from green to yellow to red. It also includes green extension phases for traffic actuated signals. An actuated signal will start with a minimum green phase and extend the green phase by the extension time increments as long as traffic is detected by the corresponding detectors and the total green time is less than the maximum green phase. If the signal includes dual rings and barrier groups, the process is quite complicated. Selective output of the signal phasing events can be a useful way of checking if the signal timing and phasing plans are coded correctly.