Identifier:	Revision:	Effective Date:
101.001	**1.0**	**03/31/2008**

Document Catalog Number: **20080331AuroraRNM**

Authors: **Alex Kurzhanskiy, Jaimyoung Kwon**

Aurora
Road Network Modeler

User Guide

TOPL Group
At UC Berkeley

Visit Our Website at path.berkeley.edu/topl

Document History

Revision History

Revision Number	Revision Date	Summary of Changes	Author
1.0	3/22/2008	User Guide created.	Alex Kurzhanskiy, Jaimyoung Kwon

Approvals

This document requires following approvals:

Name	Title
Roberto Horowitz	
Jaimyoung Kwon	
Pravin Varaiya	

Distribution

This document has been distributed to:

Name	Title

Table of Contents

1. Introduction 8
1.1. Purpose 8
1.2. Scope 8
1.3. System Requirements 8
2. Package Description 9
3. Installation 10
4. Getting Started 13
5. Road Network Configuration 15
5.1. Network Elements 15
5.1.1. Link 15
5.1.2. Node 18
5.1.3. Network 20
5.2. ODs and Paths 22
6. Configurator 24
6.1. Menu Commands 25
6.2. Editing Network Elements 26
6.2.1. Network Editor 26
6.2.2. Node Editor 28
6.2.3. Link Editor 29
6.3. Adding, Reassigning and Deleting Network Elements 32
6.4. Filtering Displayable Links 35
6.5. Validating Network Configuration 36
6.6. Settings Window 36
7. Simulator 37
7.1. Menu Commands and Options 38
7.2. Settings Window 39
7.3. Network Window 41
7.4. Node Window 43
7.5. Link Window 45
7.6. Path Window 47

7.7. Events 49
7.7.1. Change of Network Control Mode 49
7.7.2. Change of Simple Controller 51
7.7.3. Change of Split Ratio Matrix 52
7.7.4. Change of Fundamental Diagram 54
7.7.5. Change of Queue Limit 55
7.7.6. Change of Demand Coefficient 56
7.8. Mainline Control 58
7.8.1. TOD 58
7.8.2. ALINEA 60
7.9. Queue Control 61
7.9.1. Queue Override 62
7.9.2. Proportional 63
7.10. Configuration File 64
8. GIS Importer 66
8.1. Starting GIS Importer 67
8.2. Opening GIS File 67
8.3. Geometric Filtering via OpenJUMP 67
8.3.1. Installing OpenJUMP 67
8.3.2. Open GIS Data in OpenJUMP 68
8.3.3. Geometrically Filter GIS data 68
8.3.3.1. Using "Select Features Tool" 68
8.3.3.2. Using "Masking Polygon" 69
8.3.4. Export Features in the Data to GIS File 70
8.4. Road Type Filtering 70
8.4.1. Using OpenJUMP to Select Road Types 70
8.5. Edge Simplification 71
8.6. Export to XML 71
8.7. Save as GIS 71
9. Appendix A – References 72
10. Appendix B – XML Schema for Aurora RNM Configuration File 73
11. Appendix C – Glossary of Terms 86

About this guide

This document is divided into the following chapters:

- Chapter 1, *Introduction*.
- Chapter 2, *Package Description*, gives an overview of system modules the key features.
- Chapter 3, *Installation*, explains how to download and install the software.
- Chapter 4, *Getting Started*, explains how to get started.
- Chapter 5, *Road Network Configuration*, describes the elements that compose a road network.
- Chapter 6, *Configurator*, explains the operation and features of Aurora RNM Configurator.
- Chapter 7, *Simulator*, explains the operation and features of Aurora RNM Simulator.
- Chapter 8, *GIS Importer*, explains how to build Aurora configuration files from GIS databases.
- Appendix A, *References*.
- Appendix B, *XML Schema for Aurora RNM Configuration File*.
- Appendix C, *Glossary of Terms*.

Who Should Use It

This guide is intended for engineers involved in modeling traffic on road networks, test operational scenarios and design control strategies.

Typographical Conventions

This document uses the following typographical conventions:

Command and option names appear in **bold type** in definitions and examples.

Specific terms appear in *italic type*.

The names of directories and files, XML code samples and Java class names appear in `mono-space` type.

In addition, the following symbols appear in command syntax definitions.

- Square brackets [ ] surround optional items and references to literature.
- The pipe symbol | in a command statement separates mutually exclusive values for an argument.

Icons used in Aurora RNM Applications

	Freeway node
	Highway node
	Node representing signalized intersection
	Node representing stop sign intersection
	Freeway link
	Highway link
	HOV link
	Interconnect
	On-ramp
	Off-ramp
	Street

1. Introduction

1.1. Purpose

The purpose of this document is to define the functionality and operation of *Aurora Road Network Modeler* (*Aurora RNM* ©) developed by the TOPL Group at UC Berkeley and used for modeling travel corridors (road networks comprised of freeways and urban arterials).

Currently, this document serves as the sole reference for the scope of Aurora RNM functionality delivered by the TOPL Group.

1.2. Scope

This document describes the user interface and functionality of Aurora RNM Simulator, Configurator and GIS Importer. It also explains the structure and format of the configuration files used by the Simulator.

This document applies to

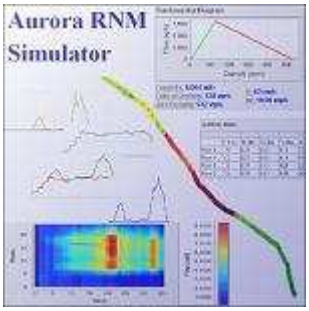
Title	Aurora Road Network Modeler
Abbreviation	Aurora RNM
Version number	1.0
Release number	1

1.3. System Requirements

Aurora RNM runs on Windows XP and Vista. It requires 20 MB of disk space and 512MB RAM.

2. Package Description

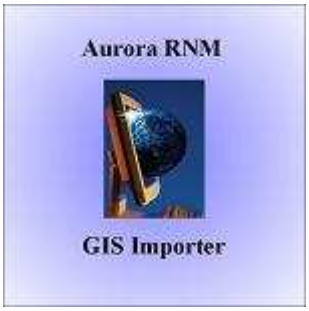
Aurora RNM consists of three modules.



Simulator is a graphical application with interactive user interface that runs traffic simulations on road networks. Traffic flow and density are computed using *macroscopic* Cell Transmission Model [1]. It also computes such traffic characteristics as speed, VMT, VHT, delay and productivity loss and generates flow, density and speed contour plots making it convenient to compare with the real traffic data collected by systems such as PeMS (pems.eecs.berkeley.edu).

The user can create simulation scenarios by means of events that change user-specified configuration parameters of the road network at user-specified times.

Configurator is a graphical application with interactive user interface whose purpose is to produce XML configuration files for the Simulator. It can be used to build road networks from scratch; edit existing road networks by deleting or reassigning network components or adding new network components; efficiently provision road parameters such as fundamental diagrams and split ratios; and input demand profiles.



GIS Importer is a graphical application with interactive user interface that extracts road information from the GIS .shp and .dbf files and saves it in the XML format of Aurora configuration file. This configuration file can be refined in the Configurator and then used by the Simulator.

It is recommended to use GIS Importer together with the open source OpenJUMP software used for geometric filtering of GIS data. OpenJUMP can be downloaded from sourceforge.net/projects/jump-pilot/.

3. Installation

To download and install Aurora RNM, follow the steps below.

1. Download `aurora_setup.exe` from the http://code.google.com/p/aurorarnm/downloads.
2. Run `aurora_setup.exe` after download is complete.
3. Read the License Agreement and press **I Agree** if you agree with the license agreement (Figure 3-1).
4. Choose which components to install. We suggest you install all components for the software to function properly (Figure 3-2).
5. Determine which folder to install Aurora (Figure 3-3). The default installation folder is `C:\Program Files\TOPL\Aurora`.
6. If you click **Install**, Aurora is installed on your computer (Figure 3-4).

To run Aurora, Java™ needs to be installed on the machine. To check whether Java™ is installed on your machine, follow the steps below:

1. Visit http://www.java.com/en/ using the web browser. (Figure 3-5)
2. Click **Do I have Java?** link on the page.
3. If you have Java™ version 1.6 or later, you don't need to do anything.
4. If you don't, go ahead and click **Download Java** link or visit http://www.java.com/en/download/index.jsp .

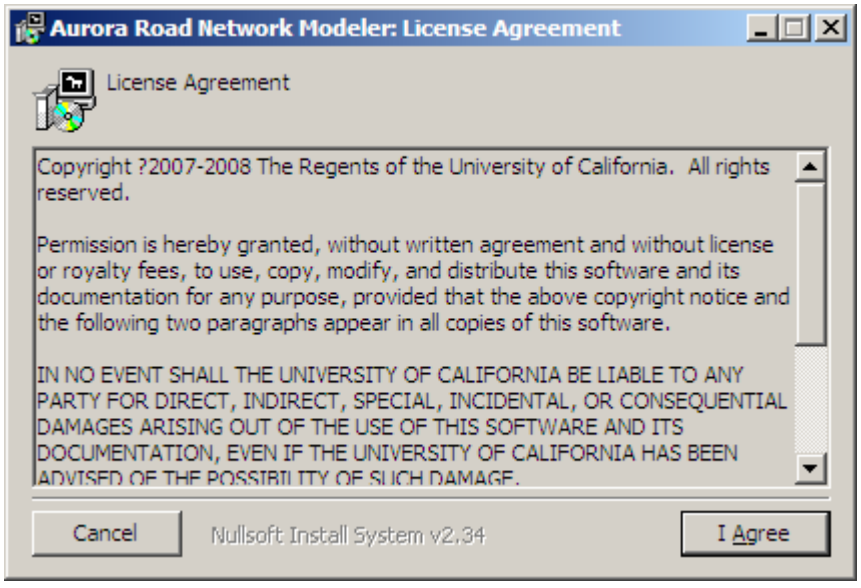


Figure 3-1: License agreement.

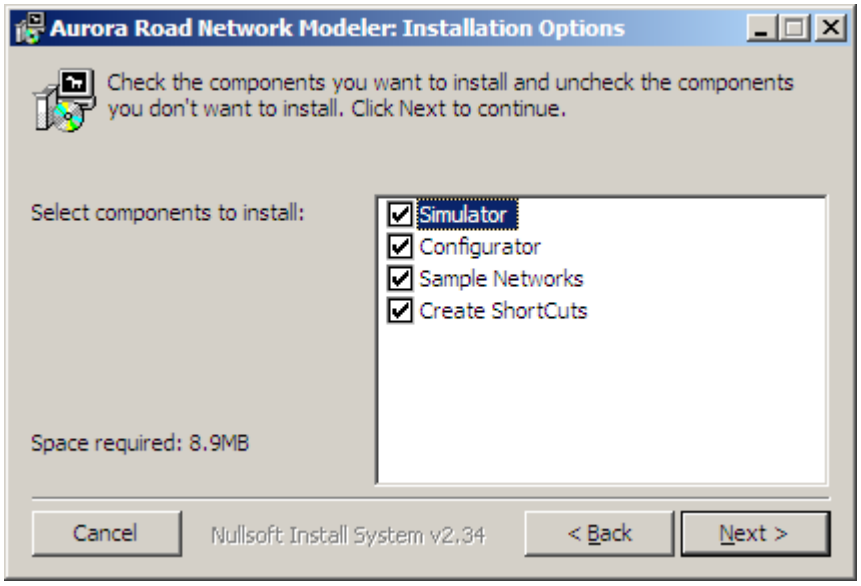


Figure 3-2: Installation options.

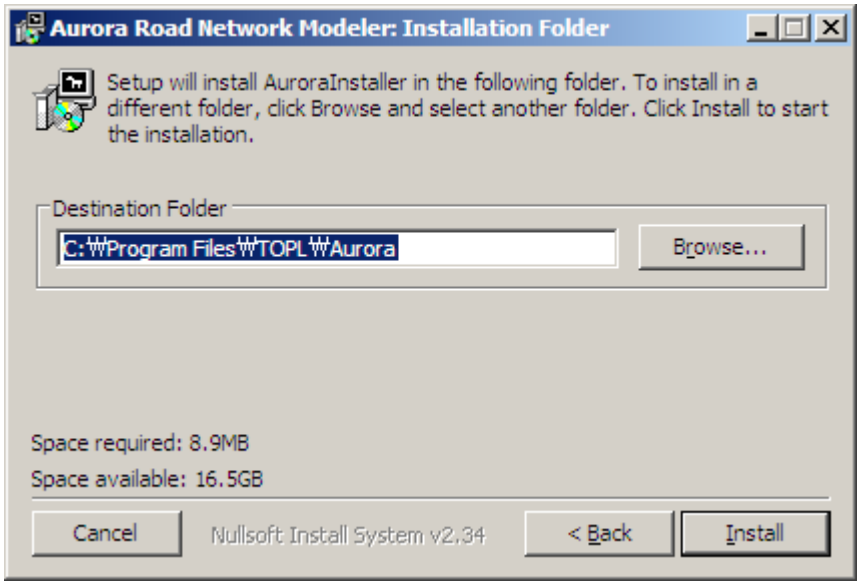


Figure 3-3: Installation folder.

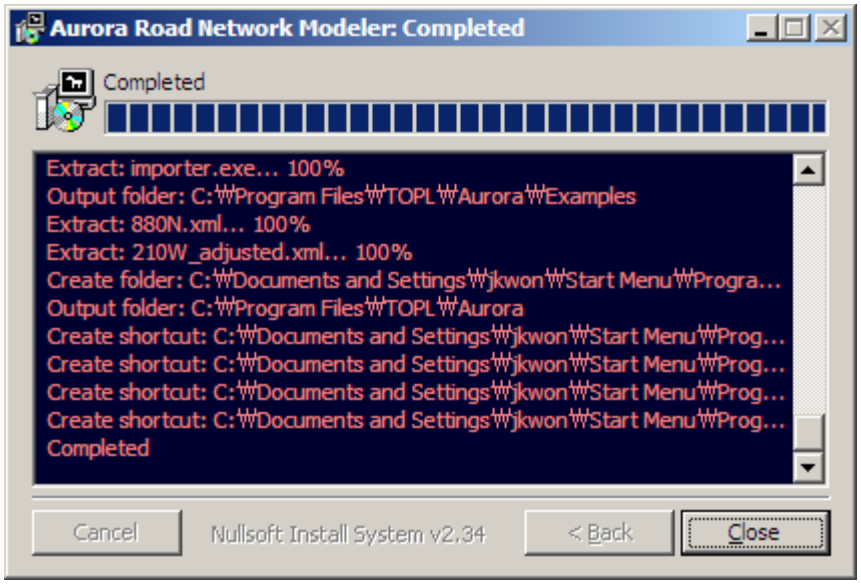


Figure 3-4: Installation is complete.

Figure 3-5: Sun Java™ webpage.

4. Getting Started

Start the Aurora Simulator by going to **Start → All Programs → TOPL → Aurora → Simulator**. Simulator starts as a blank window.

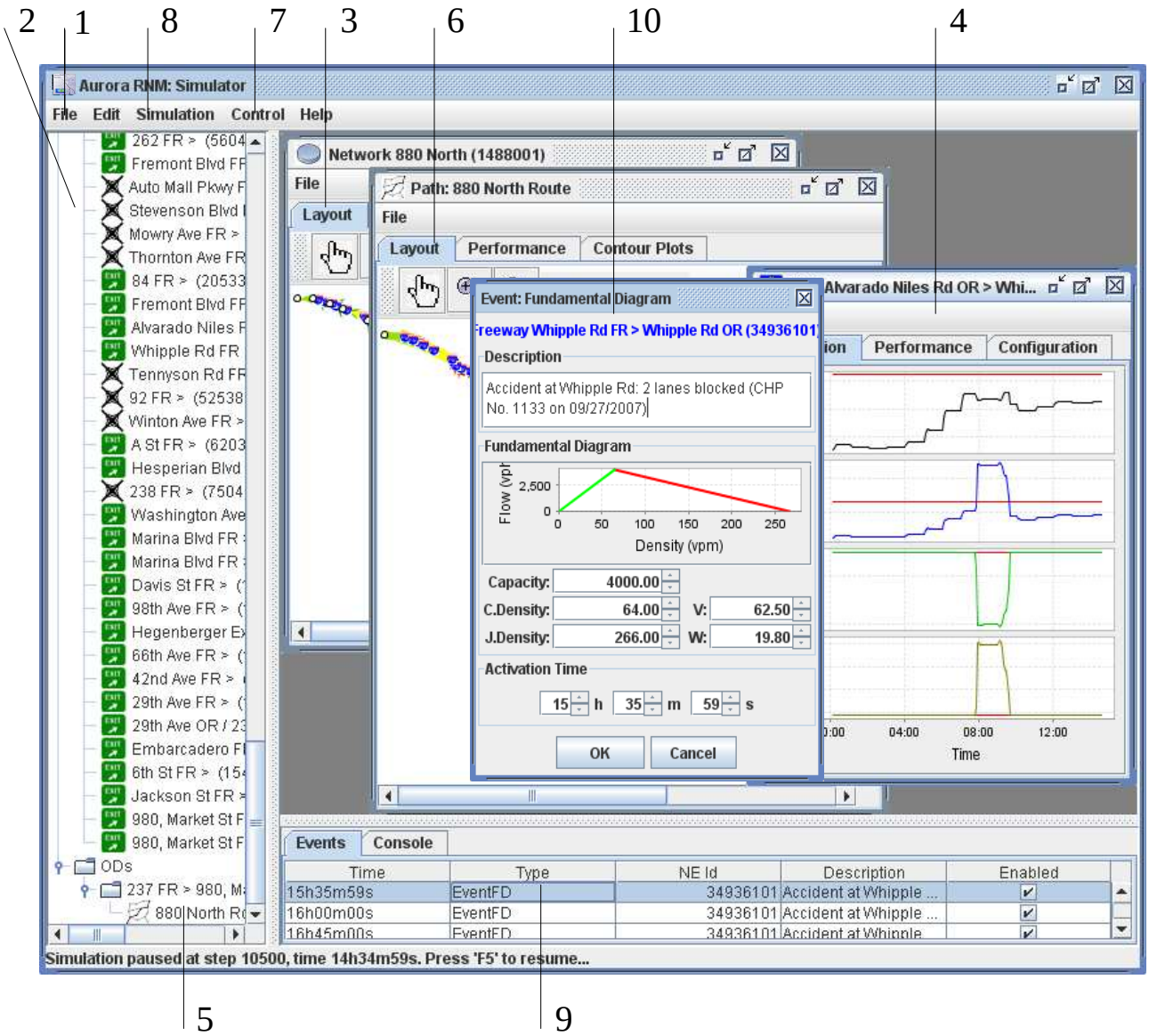


Figure 4-1: Aurora Simulator window – numbers 1-10 reflect the steps.

To get familiar with the Simulator, follow the steps using Figure 4-1, where numbers from 1 to 10 correspond to the steps, as illustration.

1. Open a sample configuration file `I880N_ALINEA.xml` by going to **File → Open** in the menu and choosing this file in the file dialog window. This loads I880 North freeway.
2. In the Explorer-like network tree open **880 North (1488001)** item by double-clicking on it.
3. Tabbed network window with the map of I-880 North freeway opens.
4. **Nodes** folder contains a list of nodes, **Links** folder contains the list of links. Double-clicking on a node or link opens a tabbed window for that node or link where the simulation data will be displayed. In this example, it is a link window.
5. ODs folder contains a list of origin-destination pairs, and each of them contains a list of paths. In this example, there is one OD – **237 Fr > 980, Market St FR**, and one path – **880 North Route**. Double-click on it.
6. Tabbed path window opens. **Layout** tab displays the map of I-880 North route, **Performance** and **Contour Plots** tabs display data computed as simulation runs.
7. **Control → Mainline** menu option is used to turn the control at nodes on or off. In this example, ALINEA ramp metering can be turned on or off.
8. To start the simulation, go to **Simulation → Start**. To pause the simulation, go to **Simulation → Stop**.
9. Events scheduled to fire at specific times define a scenario. They are listed at the bottom of the Simulator window. To disable an event, un-check the checkbox in the corresponding row of the event table. To modify an event, double-click on it.
10. Event editor window pops up when the user double-clicks on an event, allowing the user to modify the event properties.

5. Road Network Configuration

5.1. Network Elements

The basic building block of the Aurora system is a *network element* with a unique integer ID. A network element can be a *link* representing a stretch of road; *simple node* – point where links merge and/ or diverge that contains information about split portions of the output flow into outgoing links and may control input flow rates; *complex node* – network built out of network elements.

Throughout this document simple node is referred to just as *node*, and complex node – as *network*.

5.1.1. Link

A link is defined by:

- type – Table 5-1 lists possible link types and Java classes that implement them together with node types to which they can be connected;
- length (miles);
- number of lanes – allows fractional numbers to account for auxiliary lanes;
- dynamics – model used to compute the evolution of density and flow in time (currently, the only available is the Cell Transmission Model implemented by `aurora.hwc.DynamicsCTM` Java class – see Table 5.2);
- fundamental diagram – triangular density-flow relation specified by:
    - capacity (vehicles per hour) – maximum flow that can be achieved,
    - critical density (vehicles per mile) – density at which the capacity is achieved, and
    - jam density (vehicles per mile) – density at which traffic speed drops to zero;

Link must have either of the two nodes, *begin node* or *end node*, or both of them, attached to it. Links with no begin node are *source links*. Source links provide input to the system. Associated with them are the following additional parameters:

- demand profile (vehicles per hour) – array of desired input flow values;
- demand time period (hours) – specifies the frequency of switching between the values in the demand profile;
- demand coefficient – the number by which the demand value is multiplied before it is processed (its default value is 1) by the system, allowing the user to increase or decrease the demand from its profile values;

- queue limit – the maximum number of vehicles the source link queue can hold, which makes difference only if queue control is turned on.

Links with no end nodes are *destination links*. It is assumed that anything downstream of a destination link is in free flow.

Link type	Java class	Admissible begin nodes	Admissible end nodes
Freeway	`aurora.hwc.LinkFwML`	Freeway	Freeway
Highway	`aurora.hwc.LinkHw`	Highway, signal and stop junctions	Highway, signal and stop junctions
HOV	`aurora.hwc.LinkFwHOV`	Freeway, highway	Freeway, highway
Interconnect	`aurora.hwc.LinkIC`	Freeway, highway	Freeway, highway
On-ramp	`aurora.hwc.LinkOR`	Signal and stop junctions	Freeway, highway
Off-ramp	`aurora.hwc.LinkFR`	Freeway, highway	Signal and stop junctions
Street	`aurora.hwc.LinkStreet`	Signal and stop junctions	Signal and stop junctions
Dummy	`aurora.hwc.LinkDummy`	Any	Any

Table 5-1: Link types and node types to which they can be connected.

Table 5-2 presents an XML block of a configuration file describing a sample ordinary freeway link, a link that has both, begin and end nodes.

The `<density>` block in this description specifies the initial density, in this example, 15.88 vehicles per mile.

```
<link class="aurora.hwc.LinkFwML" id="100842441" length="0.37" lanes="4.0">
   <begin id="10084244" />
   <end id="10244258" />
   <dynamics class="aurora.hwc.DynamicsCTM" />
   <density>15.88</density>
   <fd densityCritical="128.0" densityJam="532.0" flowMax="8064.0" />
</link>
```

Table 5-2: Representation of an ordinary link in a configuration file.

Table 5-3 presents an XML block of a configuration file describing a sample source on-ramp link, a link with no begin node.

The `<density>` block in a source link description is used to specify the initial queue size. In the current example the initial queue is 18 vehicles long following the formula

$$QueueSize = Density \times LinkLength\ .$$

The `<demand>` block in this description specifies the demand profile – desired flow values (vehicles per hour); `tp="1.0"` means that demand sampling period is 1 hour; and `knob` parameter represents the demand coefficient, and `knob="0.9"` indicates that 90% of each of the demand values will be applied.

The `<qmax>` block in this description represents the queue limit – in this particular example, it is 100 vehicles.

The `<fd>` block defines the fundamental diagram with `densityCritical` being critical density, `densityJam` – jam density, and `flowMax` – the capacity. These values are *total* (***not*** per lane!).

```
<link class="aurora.hwc.LinkOR" id="135946303" length="0.18" lanes="1.0">
   <end id="14104636" />
   <dynamics class="aurora.hwc.DynamicsCTM" />
   <density>100.0</density>
   <demand tp="1.0" knob="0.9">330.0, 250.0, 221.0, 203.0, 329.0, 791.0, 1508.0,
2034.0, 2138.0, 2000.0, 1849.0, 1825.0, 1890.0, 1951.0, 1880.0, 2020.0, 2025.0,
2136.0, 1852.0, 1428.0, 1199.0, 890.0, 737.0, 521.0</demand>
   <qmax>100.0</qmax>
   <fd densityCritical="30.0" densityJam="150.0" flowMax="5000.0" />
   <position>
      <point x="-2.2330555555555556" y="-7.775" z="0.0" />
      <point x="0.0" y="0.0" z="0.0" />
   </position>
</link>
```

Table 5-3: Representation of a source link in a configuration file.

Source and destination links – those without either begin or end nodes – require `<position>` block in their configuration description, which defines only how these links will be displayed by Aurora Simulator and has no influence over simulation itself. Each `<position>` block contains

two `<point>` sub-blocks: the first describes where the link starts, the second – where the link ends. In the example in Table 5-3, only the starting point matters because this link has no begin node, and the end point of the link is defined by the position of its end node (see Section 5.1.2).

5.1.2. Node

A node must have one or more input and one or more output links. It is defined by:

- type – Table 5-4 lists possible node types and Java classes that implement them together with input and output link types they admit;
- name – string of ASCII characters to be displayed in the node lists and node labels by Aurora Simulator;
- description – optional information such as direction, post mile, etc.;
- input controllers – assigned to each in-link there may be a controller that, if activated, restricts the flow coming out of this link;
- split ratio matrix with the number of rows equal to the number of in-links and the number of columns equal to the number of out-links, its elements being nonnegative summing up to 1 in each row, it carries information about what portion of which input flow must be directed to which output.

Node type	Java class	Admissible input links	Admissible output links
Freeway	`aurora.hwc.NodeFreeway`	Freeway, HOV, interconnect, on-ramp, dummy	Freeway, HOV, interconnect, off-ramp, dummy
Highway	`aurora.hwc.NodeHighway`	Highway, HOV, interconnect, on-ramp, dummy	Highway, HOV, interconnect, off-ramp, dummy
Signal junction	`aurora.hwc.NodeUJSignal`	Street, off-ramp, dummy	Street, on-ramp, dummy
Stop junction	`aurora.hwc.NodeUJStop`	Street, off-ramp, dummy	Street, on-ramp, dummy

Table 5-4: Node types and link types to which they can be connected.

Table 5-5 presents an XML block of a configuration file describing a sample freeway node with 2 incoming and 2 outgoing links.

The `<outputs>` block contains references to the outgoing links.

The `<inputs>` block lists the incoming links, each described by `<input>` block that contains a reference to an incoming link and `<splitratios>` sub-block specifying how the flow coming from this in-link is distributed among the out-links – this represents the corresponding row of a split ratio matrix. In the current example, 91% of the flow from the link with ID 135246271 is directed to the out-link 141046361 and the remaining 9% – to the out-link 141046362.

```
<node class="aurora.hwc.NodeFreeway" id="14104636" name="29th OR / 23rd FR">
   <description>880N: On-, Off-ramp PM 39.81</description>
   <outputs>
      <output id="141046361" />
      <output id="141046362" />
   </outputs>
   <inputs>
      <input id="135246271">
         <splitratios>0.91, 0.09</splitratios>
      </input>
      <input id="135946303">
         <splitratios>0.99, 0.01</splitratios>
         <controller class="aurora.hwc.control.ControllerALINEA" tp="0.0167">
            <limits cmin="0.0" cmax="10000.0" />
            <parameter name="upstream" value="false" />
            <parameter name="gain" value="60.0" />
         </controller>
      </input>
   </inputs>
   <position>
      <point x="-2.236" y="-7.777" z="0.0" />
   </position>
</node>
```

Table 5-5: Representation of a node in a configuration file.

Optionally, the `<input>` block can contain `<controller>` sub-block that describes a controller for this link. See Section 7.7 for more details about controllers.

The `<position>` block describes how the node should be displayed in the Aurora Simulator. It contains single `<point>` sub-block specifying `x` and `y` coordinates on Aurora map, `z` coordinate is ignored.

5.1.3. Network

All network elements – links and nodes – are always part of a complex node, a network. There is at least one network in any Aurora system – the top level complex node, to which all other links and nodes belong. Network objects are nodes themselves. Hence, networks can contain networks just as they contain simple nodes. It makes the Aurora structure hierarchical, allowing to create configurations out of building blocks that are more complex than links and simple nodes, which is faster and more convenient. Another benefit of using a hierarchical structure is that different sub-networks may have different sampling periods, that is, simulation steps of different duration. The larger is the sampling period, the faster is the simulation. But it cannot be arbitrary large – its upper bound is determined by the relation:

$$UpperBound = \min\left(\frac{LinkLength}{FreeFlowSpeed}\right),$$

where

$$FreeFlowSpeed = \frac{Capacity}{CriticalDensity}.$$

It can save time if, for example, network consists of roads with long enough links that do not require a small sampling period, and roads with rather short links that do. Separating them into sub-networks with different sampling periods reduces computation time and makes simulation run faster.

NOTE: Sampling periods of sub-networks cannot be greater than sampling period of top level network.

A complex node is characterized by:

- type – there is only one possible type representing a complex node which is network and is implemented by `aurora.hwc.NodeHWCNetwork` Java class;

- name – string of ASCII characters to be displayed in the node lists and node labels by Aurora Simulator;
- indication whether it is a top level network or not;
- indication if the control is turned on or off;
- time period (hours) – sampling period for the model that runs on the links that belong to this network;
- description – optional information such as region this network covers;
- Node list containing descriptions of nodes and networks that belong to this network;
- Link list containing descriptions of links that belong to this network;
- List of user-defined origin-destination pairs together with user-defined paths.

```
<network class="aurora.hwc.NodeHWCNetwork" id="1488001" name="880 Corridor"
         top="true" controlled="true" tp="0.00139">
   <description>880 North freeway</description>
   <position>
      <point x="0.0" y="0.0" z="0.0" />
   </position>

   <NodeList>
      ...
   </NodeList>

   <LinkList>
      ...
   </LinkList>

   <ODList>
      ...
   </ODList>
</network>
```

Table 5-6: Representation of a network in a configuration file.

Table 5-6 presents an XML block of a configuration file describing a sample network.

Here, `top="true"` indicates that it is a top level network, `controlled="true"` means that the control is turned on and all the controllers on simple nodes assigned to input links must be invoked during the simulation, and `tp="0.00139"` specifies the sampling period as 5 seconds.

The `<position>` block describes the network position similar to that of a node when it is displayed as part of a larger network. It only makes sense for networks that are nodes of larger networks, and is ignored in top level networks.

The `<NodeList>` block lists the descriptions of nodes (see Section 5.1.2) and networks such as this network itself, which belong to this network.

The `<LinkList>` block lists the descriptions of links (see Section 5.1.1), which belong to this network.

The `<ODList>` block lists the descriptions user-defined origin-destination pairs together with user defined paths for each origin-destination pair (see Section 5.2 for more detail).

5.2. ODs and Paths

There are two other basic objects that build up Aurora configuration. Object *path* (Java class `aurora.hwc.PathHWC`) describes route from a node to node as a sequence of adjacent links. Object *OD* (Java class `aurora.hwc.ODHWC`) describes a pair of origin and destination nodes together with list of paths connecting the two.

A network may contain a list of origin-destination pairs. For consistency, it is required that every link in every path of every origin-destination pair belongs to the same complex node as ODs in the list. It may happen that both, origin and destination nodes, belong to the same sub-network, while some links at certain paths connecting them are part of a different sub-network. This is not a problem because top level network contains all the links present.

A path is described by:

- name – string of ASCII characters to be displayed in the path lists by Aurora Simulator;
- sequence of links that form the path referred to by their IDs.

An OD is described by:

- begin and end nodes referred to by their IDs;
- path list that contains possibly more than one path connecting begin with the end node.

OD as an object is meaningful only if there exists at least one path between its begin and end nodes.

Table 5-7 presents an XML block of a configuration file describing a sample OD with a single path in its path list.

The sequence of links forming the path is represented by a coma separated link IDs.

```
<od class="aurora.hwc.ODHWC" begin="54522514" end="16444800">
   <PathList>
      <path class="aurora.hwc.PathHWC" name="880 North Route">544825021,
545225141, 550025361, 550425461, 550625581, 551126261, 551726561, 552427261,
554128031, 560428491, 561829061, 563929221, 565929371, 570729441, 574630141,
580930321, 582030401, 585731111, 592331251, 593031281, 595331401, 1431511,
2531571, 10832311, 12532531, 13133001, 20533371, 22133521, 22933591,
24734271,25734541, 32835411, 34436031, 34936101, 41036391, 43637151, 45237381,
510380301, 525382401, 549385701, 555390801, 612393501, 620394901, 634401001,
701403501, 736405801, 750410801,804411801, 813412401, 827413401, 855415401,
95142321, 100842441, 102442581, 104243151, 111043441, 112143581, 112844061,
113744151, 115344341, 120544471, 124245291, 125845491, 132546191, 135246271,
141046361, 141646471, 142046551, 151647221, 154847361, 160247441, 162247521,
164448001</path>
   </PathList>
</od>
```

Table 5-7: Representation of an OD with path list in a configuration file.

6. Configurator

Run the Aurora Configurator by selecting **Start → All Programs → TOPL → Aurora → Configurator**. Figure 6-1 presents the face of the Configurator. The application window is partitioned into 4 areas.

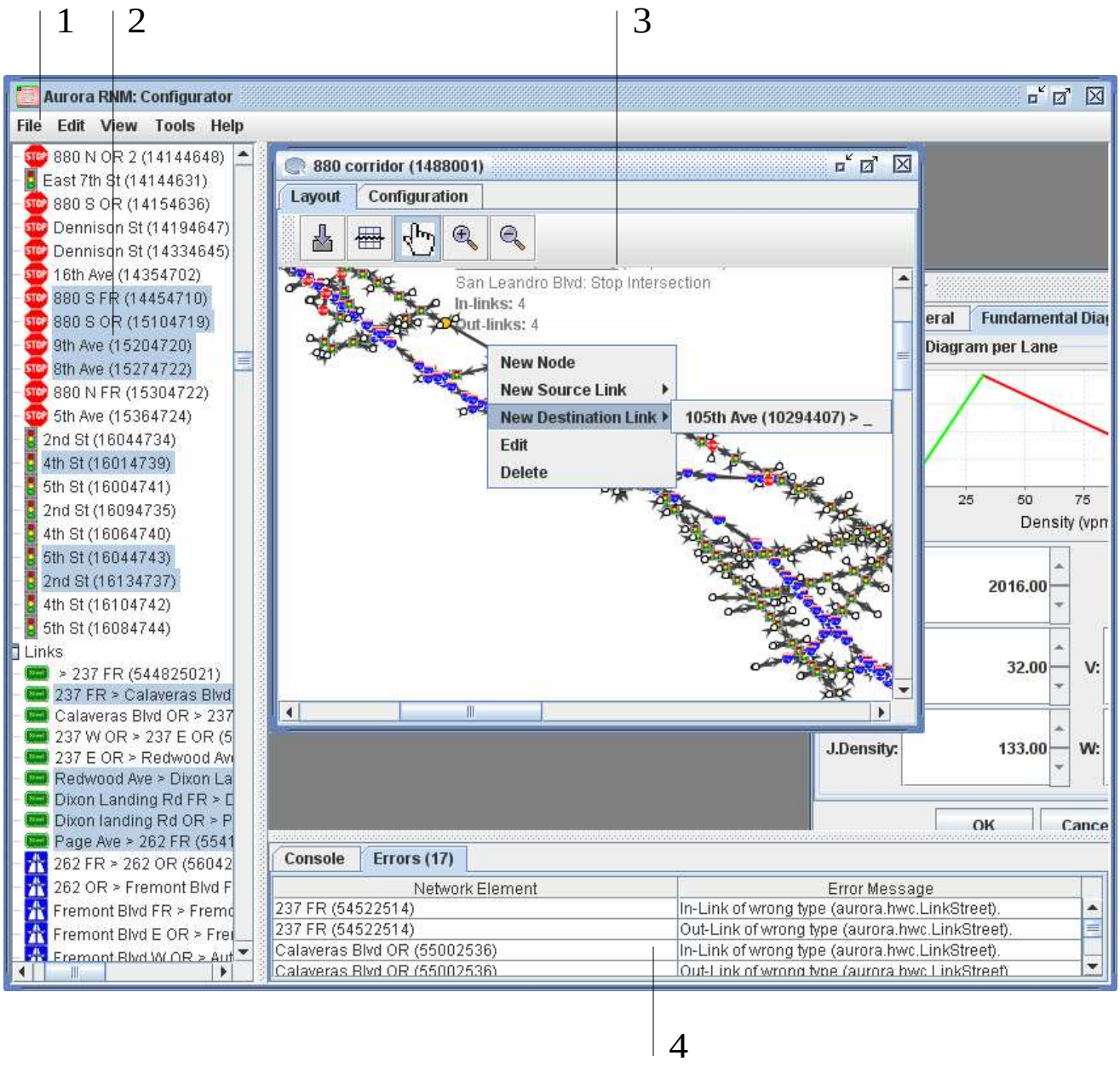


Figure 6-1: Aurora Configurator – look and feel.

1. Menu bar whose commands and options are described in detail in Section 6.1.
2. Network tree – displays network components – nodes and links in a hierarchical tree structure as in Windows Explorer. Special icons specify their types. Selecting multiple network elements in the tree and clicking enter, or double-clicking on a particular network element, brings up corresponding node and link editor sub- windows in the main frame.
3. Main frame – used to host editor sub-windows for selected network elements.
4. Error frame. This frame is divided into two tabs. One is used as a console for the Configurator output. The other one lists configuration errors after validation if there are any.

6.1. Menu Commands

Menu bar contains the following commands. In square brackets [] we indicate key combinations pressing which achieves the same result as selecting menu commands.

- **File**
  - **New** [`Alt-N`] – creates new configuration with top level network and default settings. This command is always enabled.
  - **Open** [`Ctrl-O`] – brings up a file dialog that allows the user to load either Aurora configuration file (.xml). Prior to this it brings up a confirmation dialog asking if the user wishes to save current configuration in case it has not been saved already. This command is always enabled.
  - **Save** [`Ctrl-S`] – brings up a file dialog that allows the user to save current configuration as is in an XML file. This command is enabled only if a configuration is loaded.
  - **Exit** [`Alt-X`] – closes the Configurator. Prior to exiting it brings up a confirmation dialog asking if the user wishes to save current configuration if it has not been saved already. This command is always enabled.
- **Edit**
  - **Settings** – brings up modal settings window. This command is enabled only if a configuration is loaded.
- **View**
  - **Filter** – brings up modal filter window. This command is enabled only if a configuration is loaded.
- **Tools**

- **Configuration Summary** – brings up a modal window with basic configuration statistics. This command is enabled only if a configuration is loaded.
- **Validate** – checks the validity of the configuration and display a modal information window with the results. Configuration errors if any are listed in the error frame (Figure 6-1#5). This command is enabled only if a configuration is loaded.

- **Help**
  - **About** – brings up a modal window with brief description of the application. This command is always enabled.
  - **Contact TOPL** – brings up a default email client with the TOPL group email address. This command is always enabled.

6.2. Editing Network Elements

The user can edit parameters of networks, nodes or links by means of network, node or link editors. Selecting the network elements of interest in the network tree (Figure 7-1#2) and pressing enter, or double-clicking on the particular network element opens the requested editor sub-windows in the main frame (Figure 7-1#3).

6.2.1. Network Editor

Network editor (Figure 6-2) opens individually for every network. It has two tabs.

- **Layout** tab (Figure 6-2a) displays the interactive network map. Placing the mouse cursor over a node or link makes a tooltip with information about that particular network element to appear. It also allows to add new or delete the existing network elements and to redirect the existing links (see Section 6.3). On the network map, nodes are depicted using icons corresponding to their types (freeway, highway, signal and stop junctions). White dots represent the ends of the *source* or *destination links* (links with either no begin or end node). Right-clicking on a node or link brings up a menu allowing the user to edit or delete this node or link.

  1. Fix button – fixes node positions as they are currently displayed. Node positions affect only how the nodes are displayed and nothing else.

2. Filter toggle button – if pressed, the filter is applied to the displayable links: only links of selected types together with the adjacent nodes are displayed. By default, all types are selected. To modify the selection, go to **Edit → Filter** in the menu.
3. Finger toggle button – when pressed, allows user to select nodes and links and move selected nodes around. To select multiple network elements, keep `SHIFT` key pressed while selecting these elements.
4. Zoom-in button. Each time the user presses this button the scale of the network map increases by 10%.
5. Zoom-out button. Each time the user presses this button the scale of the network map decreases by 10%.

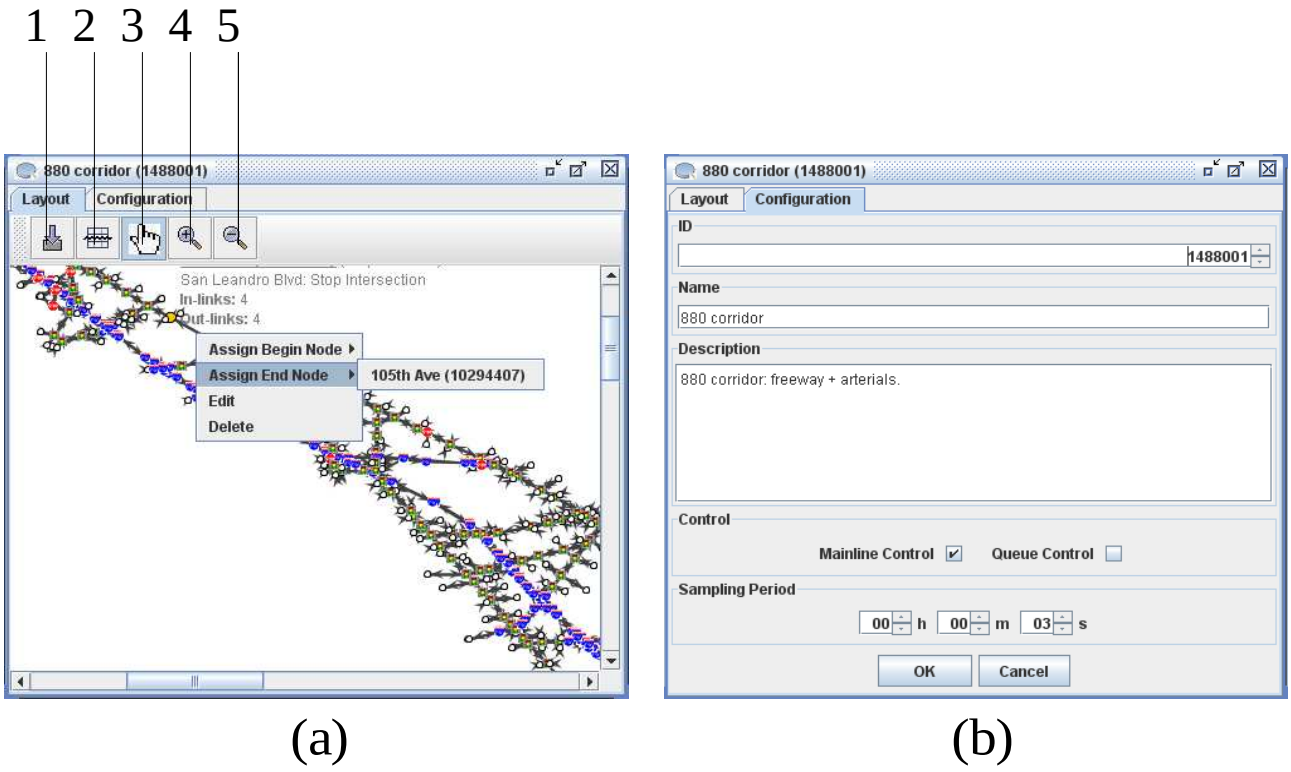


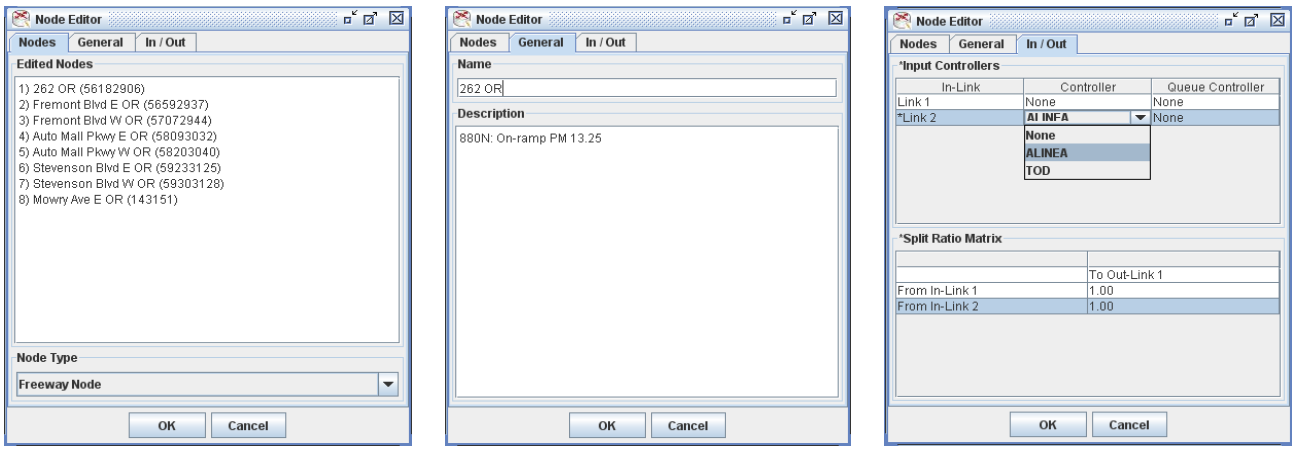
Figure 6-2: Network editor – (a) **Layout** tab; (b) **Configuration** tab.

- **Configuration** tab (Figure 6-2b) allows the user to edit network attributes.
  1. **ID** – network ID, which must be unique among the IDs of network elements in the loaded configuration.
  2. **Name** – network name.

3. **Description** – network description.
4. **Control** – allows the user to toggle the mainline and queue control modes to turn controllers on or off.
5. **Sampling Period** – simulation sampling period whose upper bound is determined as in Section 5.1.3. The user is responsible for making sure that the upper bound is not exceeded.

6.2.2. Node Editor

When the user selects multiple nodes and clicks enter, these nodes get grouped so that nodes with the same number of in- and out-links belong to one group. For every group of nodes an editor is opened (Figure 6-3). The node editor allows the user to set parameters for all nodes in the group at once. Only the parameters actually modified by the user will be provisioned to the nodes when the user clicks **OK**. Those parameters untouched will remain unchanged on all the edited nodes. If a certain parameter has been modified by the user, a '*' symbol appears in front of its name. To discard the any changes he/ she made, the user must click **Cancel**.



(a) (b) (c)

Figure 6-3: Node editor – (a) **Nodes** tab; (b) **General** tab; (c) **In/Out** tab.

The node editor has three tabs.

- **Nodes** tab (Figure 6-3a) has two areas.

1. **Edited Nodes** – lists the nodes in the group to remind the user what nodes are being edited. This list cannot be modified.
2. **Node Type** – allows the user to change the node type.

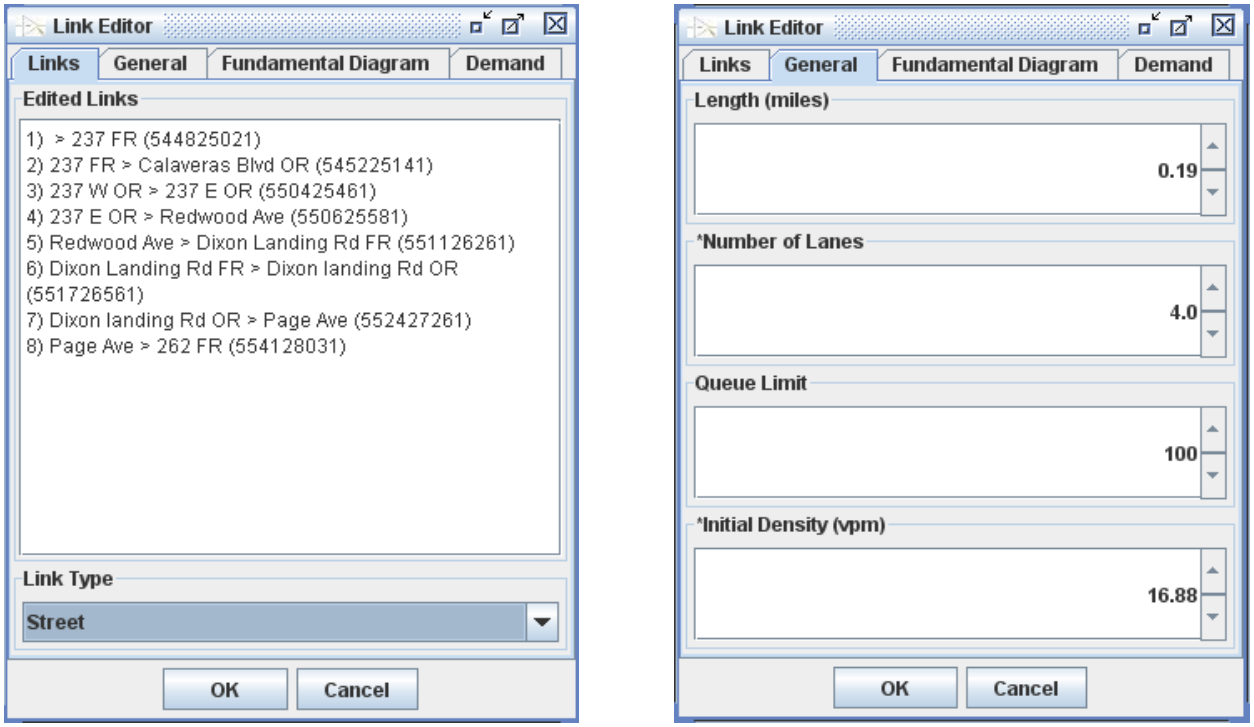
- **General** tab (Figure 6-3b) displays up to three parameters.
  1. **ID** – node ID. This parameter is displayed only if there is only one node in the edited group (you cannot see it in Figure 6-3b).
  2. **Name** – node name.
  3. **Description** – node description.
- **In/Out** tab (Figure 6-3c) has two parameters.
  1. **Input Controllers** lists the incoming links and types of the mainline and queue controllers assigned to them. The user can change the mainline controller for a particular in-link by double-clicking on the **Controller** column in the row corresponding to the desired in-link – then a combo box appears allowing the user to select different controller type or **None** to remove the controller from the in-link. If a controller is assigned to a particular in-link, the user can edit its parameters by double-clicking on the row corresponding to that in-link (just not on the **Controller** column) – controller editor window will pop up allowing the user to make necessary adjustments (see Section 7.8). To assign or change a queue controller the user needs to bring up the controller editor window. No queue controllers can be assigned if no mainline controller is assigned.
  2. **Split Ratio Matrix** displays which portions of incoming flows are directed to which out-links. The entries of this table can be edited by double-clicking on them. The user is responsible for the correctness of the values he/ she enters. They must be in the range from 0 to 1, and it is generally required that the values in each row sum up to one. In the current example there are two in-links and one out-link. Hence, the size of the split ratio matrix is 2 by 1, and both entries equal to 1 meaning that 100% of the flows from both in-links are directed to the out-link.

NOTE: when there is more than one node in the edited group, generic link titles such as *In-Link 1*, *Out-Link 1*, *In-Link 2*, etc. are displayed instead of specific ones formed from the name of the nodes they connect.

6.2.3. Link Editor

When the user selects multiple links and clicks enter, the single link editor sub-window (Figures 6-4 and 6-5) opens in the main frame allowing the user to modify the parameters on all these links at

once. Only the parameters actually modified by the user will be provisioned to the links when the user clicks **OK**. Those parameters untouched will remain unchanged on all the edited links. If a certain parameter has been modified by the user, a '*' symbol appears in front of its name. To discard any changes he/ she made, the user must click **Cancel**.



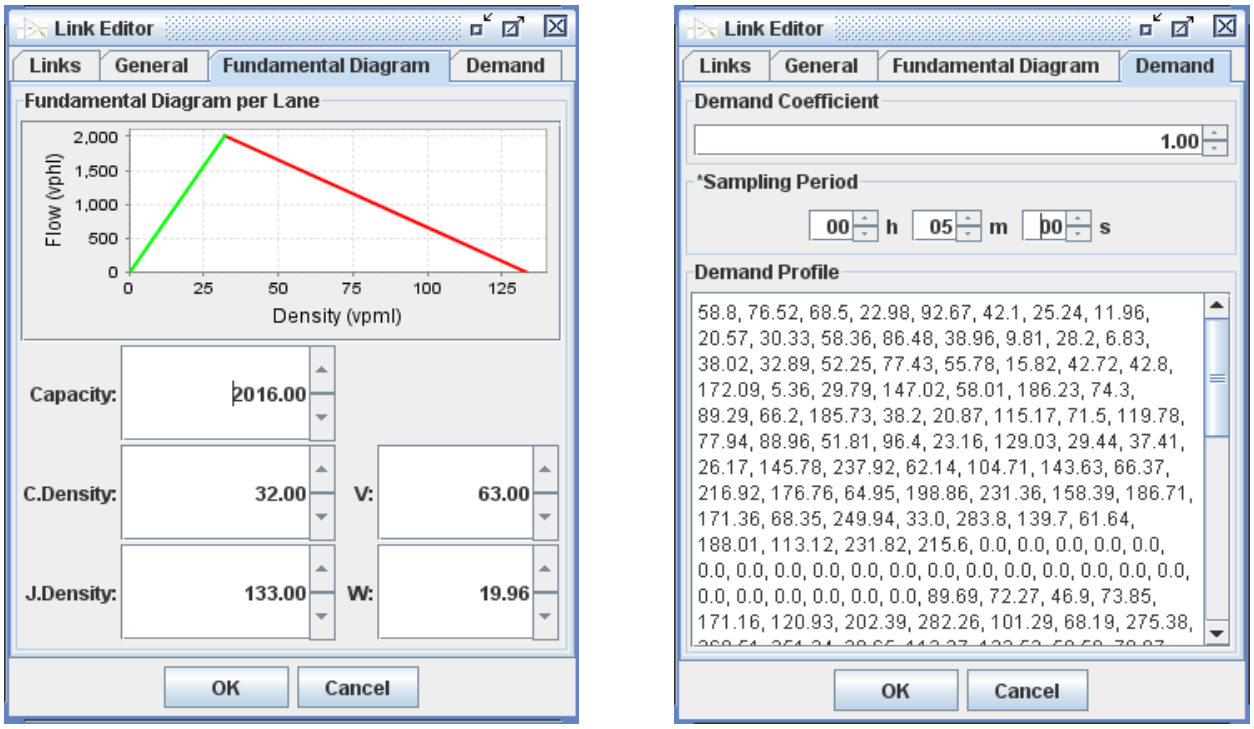
(a) (b)

Figure 6-4: Link editor – (a) **Nodes** tab; (b) **General** tab.

The link editor has four tabs.

- **Links** tab (Figure 6-4a) has two areas.
  1. **Edited Links** – lists the links that are edited to remind the user. This list cannot be modified.
  2. **Link Type** – allows the user to change the link type.
- **General** tab (Figure 6-4b) displays four parameters.
  1. **ID** – link ID. This parameter is displayed only if there is only a single link is edited (you cannot see it in Figure 6-4b).
  2. **Length** – link length in miles.

3. **Number of Lanes** – number of lanes. Fractional values represent the auxiliary lanes.
4. **Queue Limit** – queue limit. This parameter makes difference only for *source links*.
5. **Initial Density** – initial density value for the simulation in *vehicles per mile* (vpm).



(a) (b)

Figure 6-5: Link editor – (a) **Fundamental Diagram** tab; (b) **Demand** tab.

- **Fundamental Diagram per Lane** tab (Figure 6-5a) displays the fundamental diagram *per lane*, which the user can modify by adjusting **Capacity**, **C. Density** (critical density) and **J. Density** (jam density) parameters. Instead of critical and jam densities, the user can adjust free flow speed (parameter **V**) and congestion wave speed (parameter **W**) if it is more convenient. Capacity, critical density, jam density, free flow speed and congestion wave speed values are related as

$$FreeFlowSpeed = \frac{Capacity}{CriticalDensity},$$

and

$$CongestionWaveSpeed = \frac{Capacity}{JamDensity - CriticalDensity}.$$

The fundamental diagram parameters cannot be set arbitrarily. It is required that the fundamental diagram always remains a triangle. The editor would prevent the user from assigning negative values to any of these parameters or making jam density less than critical density. When the user would change capacity or critical density, free flow and congestion wave speeds would adjust automatically. Changing jam density triggers automatic adjustment of the congestion wave speed. And the other way around, if the user changes free flow speed, critical and jam densities adjust automatically. Change in the congestion wave speed results in automatic adjustment of the jam density. When provisioned to the link, the fundamental diagram (capacity, critical and jam densities) will be multiplied by the number of lanes.

- **Demand** tab (Figure 6-5b) makes difference only for the *source links*. It displays three parameters.
  1. **Demand Coefficient** – the number by which the demand value is multiplied before it is processed (its default value is 1) by the system, allowing the user to increase or decrease the demand from its profile values.
  2. **Sampling Period** – specifies the frequency of switching between the values in the demand profile.
  3. **Demand Profile** – array of coma separated desired input flow numbers measured in *vehicles per hour* (vph).

6.3. Adding, Reassigning and Deleting Network Elements

Adding and deleting network elements is done in the Layout tab of the network editor (Figure 6-2a).

To add a new node, follow the steps below.

1. Right-click on an empty space – a menu will pop up. If no network elements are selected, it will be a one choice menu (Figure 6-6a), otherwise, there will be a menu with additional options (Figures 6-6b and 6-6c).
2. Select **New Node** – this brings up a dialog window (Figure 6-7a) asking about the basic properties of the node: ID, name, description and type.
3. Fill in the node properties. Make sure to assign correct ID, which would be unique among all the network elements in the system – it is the user's responsibility to come up with node ID.

4. Click **OK**. The new node of given type will be created and displayed on the map (Figure 6-2a) and in the node list in the network tree (Figure 6-1#2).

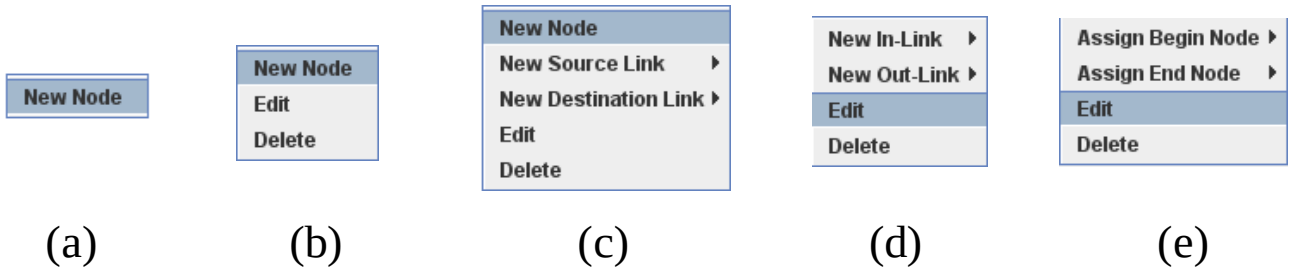


Figure 6-6: Popup menus in the network editor.

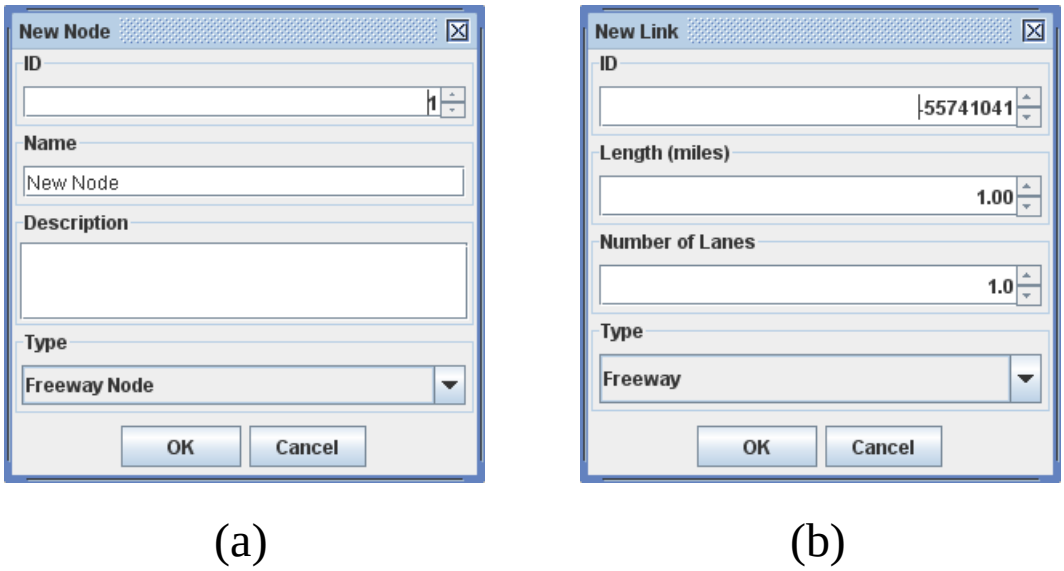


Figure 6-7: (a) new node dialog window; (b) new link dialog window.

To create a source or destination link, follow the steps below.

1. Select a node to which the source or destination link should be attached by pressing the finger toggle button (Figure 6-2a#3) and then clicking on the node. The selected node gets labeled.
2. Right-click on an empty spot – a menu will pop up (Figure 6-6c).
3. For source link, make a selection from the **New Source Link** sub-menu, for destination link – from the **New Destination Link** sub-menu – this brings up a dialog window (Figure 6-7b) asking about the basic properties of the link: ID, length, number of lanes and type.

4. Fill in the link properties. As opposed to node ID for which the user is responsible, the link ID is generated automatically. Most likely, the user does not need to modify it.
5. Click **OK**. The new source or destination link of given type will be created and displayed on the map (Figure 6-2a) and in the node list in the network tree (Figure 6-1#2).

To create an ordinary link, follow the steps below.

1. Select a node to which the link should be attached by pressing the finger toggle button (Figure 6-2a#3) and then clicking on the node. The selected node gets labeled.
2. Right-click on another node that should be attached to this link – a menu will pop up (Figure 6-6d).
3. For a link emanating from the labeled node, make a selection from the **New In-Link** sub-menu, for a link ending in the labeled node – from the **New Out-Link** sub-menu – this brings up a dialog window (Figure 6-7b) asking about the basic properties of the link: ID, length, number of lanes and type.
4. Fill in the link properties. As opposed to node ID for which the user is responsible, the link ID is generated automatically. Most likely, the user does not need to modify it.
5. Click **OK**. The new source or destination link of given type will be created and displayed on the map (Figure 6-2a) and in the node list in the network tree (Figure 6-1#2).

To reassign the begin or end node for an existing link, follow the steps below.

1. Select a node you wish to make a new begin or end node for the link by pressing the finger toggle button (Figure 6-2a#3) and then clicking on the node. The selected node gets labeled.
2. Right-click on the link whose begin or end node should be reassigned – a menu will pop up (Figure 6-6e).
3. For begin node, make a selection from the **Assign Begin Node** sub-menu, for end node – from the **Assign End Node** sub-menu. This will redirect the link and update the map (Figure 6-2a).

To delete network elements, follow the steps below.

1. Select nodes and links by pressing the finger toggle button (Figure 6-2a#3) and then clicking on the nodes and links of your choice while keeping the `SHIFT` key pressed.
2. Right-click anywhere on the map – a menu pops up (Figures 6-6b-e), allowing the user to select **Delete**. Alternatively, skip step 1 and right-click on the node or link you wish to delete, then in the popup menu select **Delete**.

3. Before deleting the selected network elements, the Configurator displays the confirmation dialog asking the user to confirm the intent to delete these network elements. Clicking OK button confirms the intent. The map (Figure 6-2a) and the network tree (Figure 6-1#2) get updated.

When some network elements are selected, right-clicking anywhere on the map brings up a menu (Figures 6-6b-e) which has **Edit** item. Selecting **Edit** in these menus opens corresponding editors for the selected network elements if these elements are not being edited already.

6.4. Filtering Displayable Links

The **Layout** tab of a network editor can display only those types of links that the user wishes to see together with nodes attached to these links. To apply the link type filter, press the filter toggle button (Figure 6-2a#2). By default, the filter allows all link types.

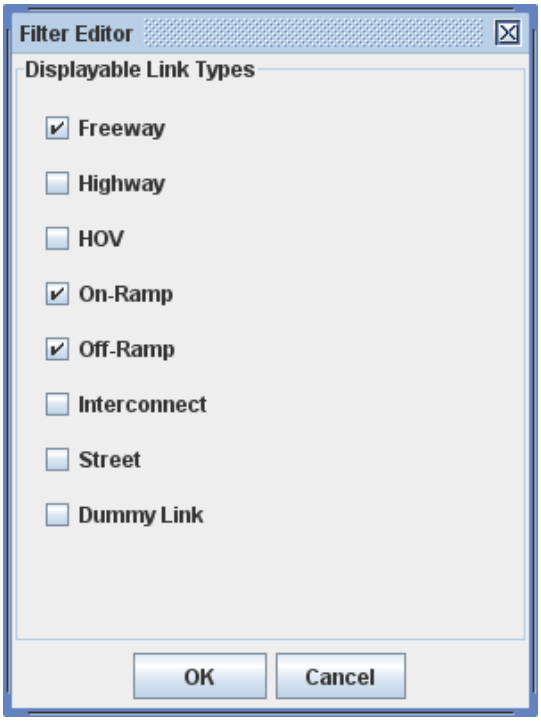


Figure 6-8: Displayable link type filter editor.

To edit the filter, go to **View → Filter** in the menu – this will open a modal filter editor window (Figure 6-8). The filter editor lists possible link types with a checkbox against each type allowing the user to check the types to be displayed and un-check those to be hidden.

Clicking **OK** sets up the filter.

6.5. Validating Network Configuration

To be accepted by the Simulator, the network configuration has to satisfy certain restrictions.

- Each node must have at least one in-link and at least one out-link.
- Each node must have a name.
- Each link must be attached to at least one node.
- Adjacent nodes and links must have compatible types as listed in Tables 5-2 and 5-4 in Section 5.1.

To check if the network configuration is correct, go to **Tools → Validate** in the menu – this will run the test whose results will be displayed in the information dialog.

If errors are found they are listed in the **Errors** tab of the error frame (Figure 6-1#4). Each entry in the list contains the reference to the affected network element and the error description. Double-clicking on the list entry opens an editor window for the affected network element if it is not being edited already.

If the test shows no errors, it means that the network configuration is ready for the Simulator.

6.6. Settings Window

See Section 7.2.

7. Simulator

Figure 7-1 presents the face of Aurora Simulator. The application window is partitioned into 5 areas.

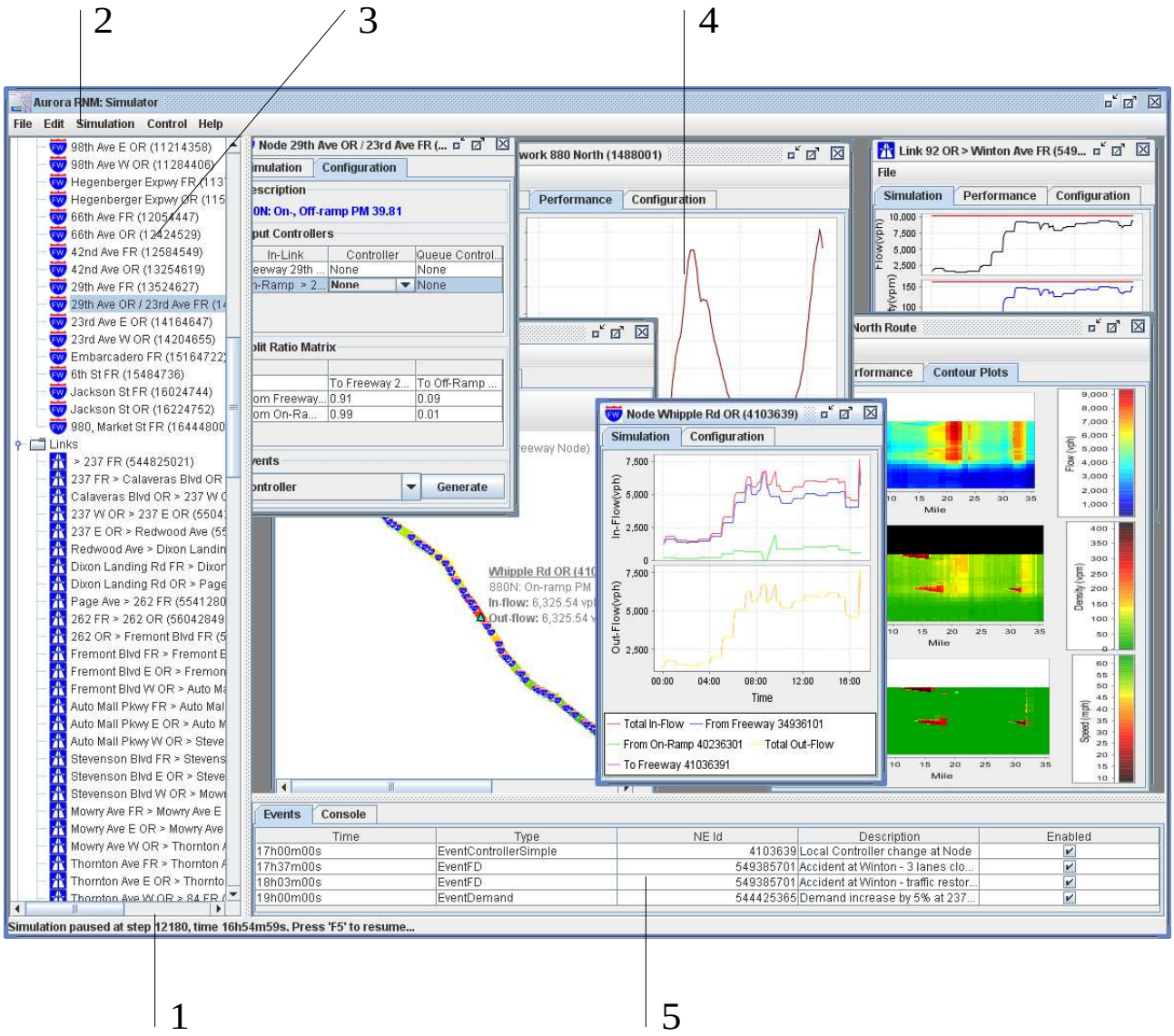


Figure 7-1: Aurora Simulator – look and feel.

1. Status area – displays the status of simulation – running, paused or stopped together with simulation step and time, and issues short instructions to the user.
2. Menu bar whose commands and options are described in detail in Section 7.1.

3. Network tree – displays network components – nodes, links, origin-destination pairs and paths in a hierarchical tree structure as in Windows Explorer. Special icons specify their types. Double clicking on a component brings up a sub-window in the main frame with details of that particular network element or path.
4. Main frame – used to host sub-windows for selected network elements or paths.
5. Scenario frame. This frame is divided into two tabs. One lists the events of the current simulation scenario: their type, description, activation time. The user cannot delete an event from the list, only edit it, or disable it. The event editor window pops up when the user double clicks on an event. See Section 7.7 for more detail. The other one is a console for the Simulator output.

7.1. Menu Commands and Options

Menu bar contains the following commands and options. In square brackets [] we indicate key combinations pressing which achieves the same result as selecting menu commands.

- **File**
  - **New Simulation** [`Alt-N`] – initializes the simulation by resetting simulation time to 0 and the state of the system to the initial condition. Prior to this it brings up a confirmation dialog asking if the user wishes to save current simulation in case it has not been saved already. This command is enabled only if a configuration is loaded.
  - **Open** [`Ctrl-O`] – brings up a file dialog that allows the user to load either Aurora configuration file (.xml) or previously saved simulation (.dat). Prior to this it brings up a confirmation dialog asking if the user wishes to save current simulation in case it has not been saved already. This command is always enabled.
  - **Save Configuration** – brings up a file dialog that allows the user to save current configuration as is in an XML file. It may be useful, if the user has changed the configuration from within the Simulator by editing settings, editing/ adding new events, toggling control mode of the system, etc. This command is enabled only if a configuration is loaded.
  - **Save Simulation** [`Ctrl-S`] – brings up a file dialog that allows the user to save current state of simulation in a binary (.dat) file. This command is enabled only if a configuration is loaded.
  - **Exit** [`Alt-X`] – closes the Simulator. Prior to exiting it brings up a confirmation dialog asking if the user wishes to save current simulation if it has not been saved already. This command is always enabled.

- **Edit**
  - **Settings** – brings up modal settings window. This command is enabled only if a configuration is loaded.
- **Simulation**
  - **Start** [F5] – starts the simulation from the beginning if it is new, or from the point it has been stopped by the user. This command is enabled only if the simulation is either new or was previously stopped by the user.
  - **Stop** – stops the simulation, which can then be resumed by the **Simulation → Start** command. This command is enabled only when the simulation is running.
- **Control**
  - **Mainline** (checkbox) – when checked, sets the control mode of all networks in the system ON, which enables the assigned input controllers on the nodes. This option is enabled only if a configuration is loaded.
  - **Queue** (checkbox) – when checked, sets the queue control mode of all networks in the system ON, which forces the input controllers on the nodes to invoke queue controllers when those are assigned. This option is enabled only if **Control → Mainline** option is enabled and checked.
- **Help**
  - **About** – brings up a modal window with brief description of the application. This command is always enabled.
  - **Contact TOPL** – brings up a default email client with the TOPL group email address. This command is always enabled.

7.2. Settings Window

Settings window is an editor for the simulation settings. It is presented in Figure 7-2.

This window displays the following parameters.

- **Sampling Period** – simulation sampling period (see Section 5.1.3). This parameter cannot be modified from within Simulator. Use the Configurator to set the sampling period for each network in your configuration. Alternatively, edit the configuration file manually.
- **Display Update Period** – specifies the frequency with which the simulation data display is updated. This value makes sense only if it is not less than the sampling period. Total delay displayed in the network window (see Section 7.3); and VMT, VHT, delay and productivity

loss data displayed in the link and path windows (see Section 7.5 and Section 7.6) of the Simulator are computed per display period.

- **Maximum Simulation Time** and **Maximum Simulation Step** determine how long simulation should run. Although related,

$$MaximumSimulationTime = SamplingPeriod \times MaximumSimulationStep,$$

these two parameters can be set independently – the one resulting in the smaller value for the maximum simulation time applies.

- **Timeout** specifies the pause duration between display updates while the simulation runs. Increasing this value results in slowing down the simulation.

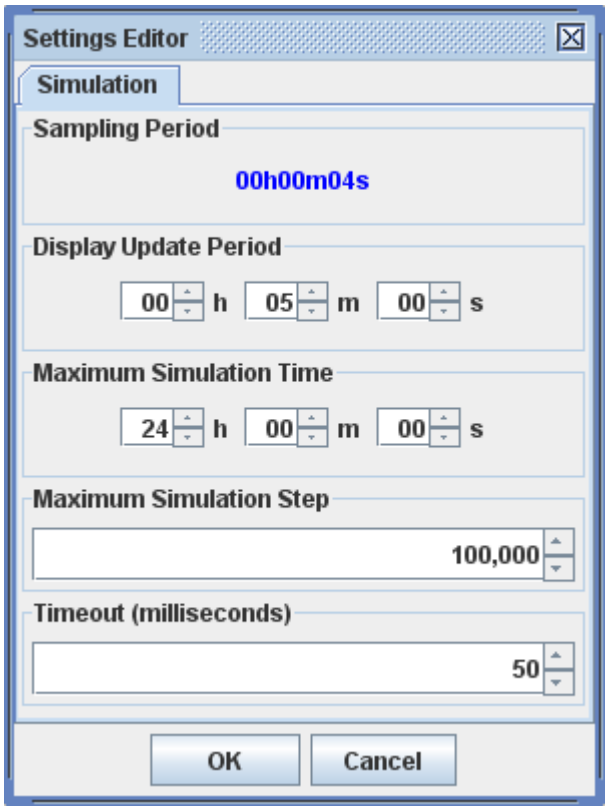


Figure 7-2: Settings window.

Table 7-1 presents an XML block of a configuration file describing the simulation settings.

The `<display>` block contains all the settable parameters from the settings window. Parameter `tp` specifies the display update period in *hours*. In this example, `tp="0.0833"` means that the displayed data will be sampled with 5 minute interval. Parameters `tsMax` and `timeMax`

represent maximum simulation step and maximum simulation time in *hours*, and parameter `timeout` defines the pause duration between display updates in *milliseconds*.

```
<settings>
   <display tp="0.0833" timeout="50" tsMax="100000" timeMax="24.0" />
</settings>
```

Table 7-1: Representation of the simulation settings in a configuration file.

7.3. Network Window

Network window (Figure 7-3) has three tabs.

- **Layout** tab (Figure 7-3) displays the interactive network map. Placing the mouse cursor over a node or link makes a tooltip with information about that particular network element to appear.
    1. Menu bar: **File → Save…** command brings up a file dialog that allows user to save data plotted in the **Performance** tab in a CSV file.
    2. Finger toggle button – when pressed, allows user to move nodes around and label nodes and links on the network map by clicking on them. To label multiple network elements, keep `SHIFT` key pressed while selecting these elements.
    3. Zoom-in button. Each time the user presses this button the scale of the network map increases by 10%.
    4. Zoom-out button. Each time the user presses this button the scale of the network map decreases by 10%.
    5. Color code mode checkbox – when checked, the links are colored using the speed values at each simulation step; otherwise, densities determine the link colors.

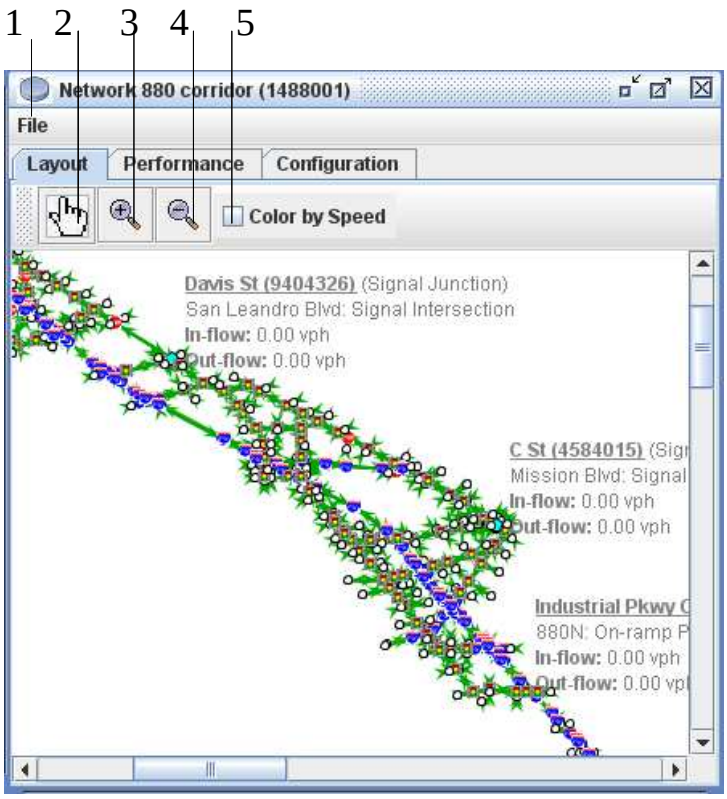
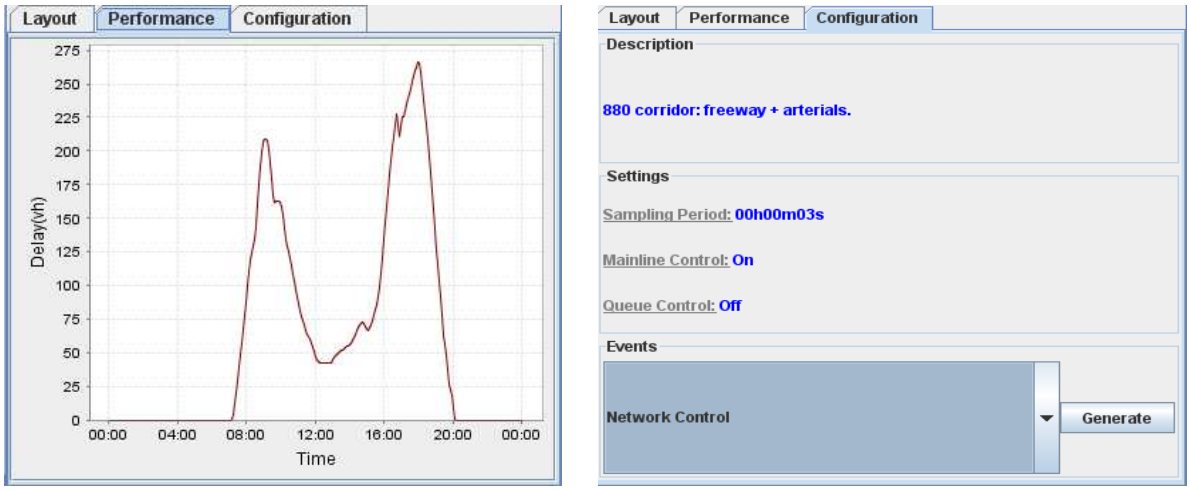


Figure 7-3: Network window – **Layout** tab.

On the network map, nodes are depicted using icons corresponding to their types (freeway, highway, signal and stop junctions). White dots represent the ends of the *source* or *destination links* (links with either no begin or end node). Links change their colors as simulation runs. By default, they are colored using density values – the color changes from green to yellow to orange as density goes from zero to critical; and from orange to red to black, as density goes from critical to jam. Alternatively, the user may choose to color the links using speed values – from green for maximum speed to yellow, to red, to black as speed drops to zero. Labels on selected network elements display the details together with computed simulation data for those particular network elements.

Right-clicking on a node or link brings up a menu with single item **Details**. Selecting **Details** brings up a node or a link window for that network element.



(a) (b)

Figure 7-4: Network window – (a) **Performance** tab; (b) **Configuration** tab.

- **Performance** tab (Figure 7-4a) shows the plot of total network delay, which is computed in vehicle-hours per display period.
- **Configuration** tab (Figure 7-4b) displays configurable network parameters and provides facility for generating network events. To generate an event, select an event to be generated and press **Generate** button – this will bring up an event editor window (see Section 7.6 for details). Currently, there is only one possible network event – change of the control mode (see Section 7.6.1).

  Sampling Period parameter displayed here is *generally not the same* as the one displayed in the settings window (see Section 7.2). They are the same only if current network is the top level one.

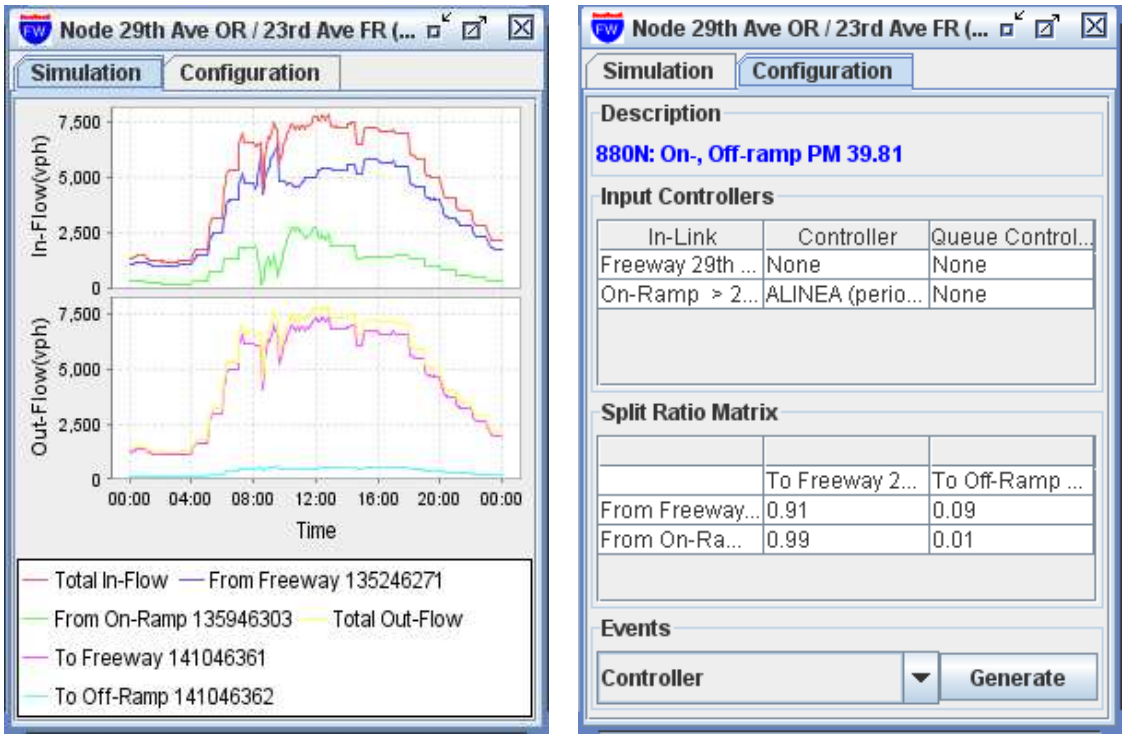
  Mainline Control and **Queue Control** settings can be toggled by **Control** → **Mainline** and **Control** → **Queue** commands accordingly (see Section 7.1).

7.4. Node Window

Node window (Figure 7-5) has two tabs.

- **Simulation** tab (Figure 7-5a) displays two plotting areas. The top one is used to plot the flows coming into the node from each input link and their sum – total input flow. The bottom one is used to plot the flows going out of the node to each of the output links and their sum –

total output flow. The legend at the bottom of this tab explains which color on the plots corresponds to which flow. Total input flow always equals the total output flow, as nodes do not have any storage capacity.



(a) (b)

Figure 7-5: Node window – (a) **Simulation** tab; (b) **Configuration** tab.

- **Configuration** tab (Figure 7-5b) is divided into 4 sections.
  1. **Description** – contains the node description taken from the configuration file (see Section 5.1.2).
  2. **Input Controllers** lists the incoming links and types of the mainline and queue controllers assigned to them. The user can change the mainline controller for a particular in-link by double-clicking on the **Controller** column in the row corresponding to the desired in-link – then a combo box appears allowing the user to select different controller type or **None** to remove the controller from the in-link. If a controller is assigned to a particular in-link, the user can edit its parameters by double-clicking on the row corresponding to that in-link (just not on the **Controller** column) – controller editor window will pop up allowing the user to make necessary adjustments (see Section 7.8).

To assign or change a queue controller the user needs to bring up the controller editor window. No queue controllers can be assigned if no mainline controller is assigned.

3. **Split Ratio Matrix** displays which portions of incoming flows are directed to which out-links. The entries of this table can be edited by double-clicking on them. The user is responsible for the correctness of the values he/ she enters. They must be in the range from 0 to 1, and it is generally required that the values in each row sum up to one. In the current example (Figure 7-5b) the first out-link receives 91% of the first incoming flow and 99% of the second incoming flow, and the second out-link gets the remaining 9% from the first incoming flow and 1% from the second incoming flow.
4. **Events** section allows the user to select in the combo box and generate by clicking **Generate** button an event for this node. There are two possible node events – the change of controller (see Section 7.7.2) and the change of the split ratio matrix (see Section 7.7.3).

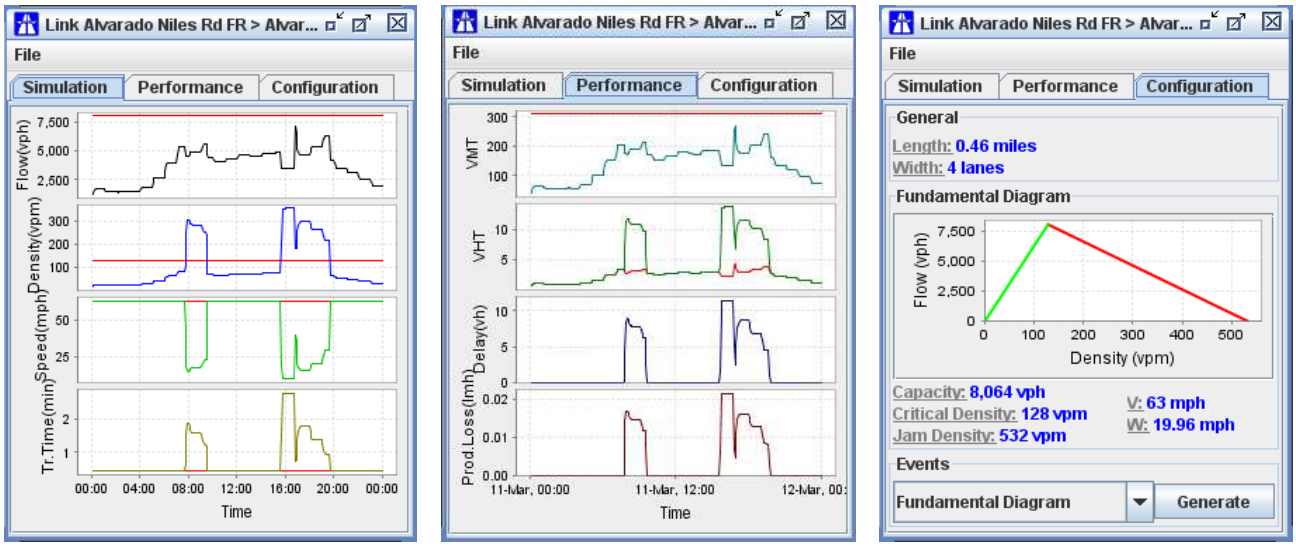
7.5. Link Window

Link window (Figure 7-6) has three tabs.

- **Simulation** tab (Figure 7-6a) with 4 plotting areas.
  1. **Flow** – plots the actual flow exiting this link in *black* versus the link capacity in *red*. Flow values are given in *vehicles per hour* (vph).
  2. **Density** (for ordinary and destination links) – plots the actual traffic density in the link in *blue* versus critical density in *red*. Density values are given in *vehicles per mile* (vpm).

     Queue (for source links) – plots the actual queue size in *blue* versus queue limit in *red*. Queue size and limit are measured in the number of vehicles.
  3. **Speed** – plots the actual traffic speed in the link in *green* versus free flow speed in *red*. Speed values are given in *miles per hour* (mph).
  4. **Travel Time** (for ordinary and destination links) – plots the instantaneous travel time through the link in *dark yellow* versus minimal travel time through this link in *red*. Travel time values are given in *minutes* (min).

     Demand (for source links) – plots the demand in *dark yellow* versus link capacity in *red*. Demand and capacity values are given in *vehicles per hour* (vph).



(a) (b) (c)

Figure 7-6: Link window – (a) **Simulation** tab; (b) **Performance** tab; (c) **Configuration** tab.

- **Performance** tab (Figure 7-6b) with 4 plotting areas.
  1. **VMT** – plots the actual vehicle-miles-traveled on the link in *cyan* versus maximum achievable vehicle-miles-traveled if the link operated at capacity in *red*.
  2. **VHT** – plots the actual vehicle-hours-traveled on the link in *dark green* versus critical vehicle-hours-traveled had all the vehicles present in the link traveled with free flow speed in *red*.
  3. **Delay** – plots delay caused by congestion in the link or by the queue if it is a source link. The delay values are given in *vehicle-hours* (vh).
  4. **Productivity Loss** – plots productivity loss due to congestion at the link. The productivity loss values are given in *lane-mile-hours* (lmh).

All the data plotted in this tab is computed per *display period* (see Section 7.2).

File → Save… menu command in the link window brings up a file dialog that allows the user to save the data displayed in the **Simulation** and **Performance** tabs in a CSV file.

- **Configuration** tab (Figure 7-6c) is divided into 3 sections.
  1. **General** displays link length and the number of lanes, in case of a source link it also displays the demand coefficient and queue limit (see Section 5.1.1).

2. **Fundamental Diagram** shows the plot of the fundamental diagram and displays the values of capacity, critical density and jam density as well as free flow speed **V** and the *congestion wave speed* **W**, which are related as

$$FreeFlowSpeed = \frac{Capacity}{CriticalDensity},$$

and

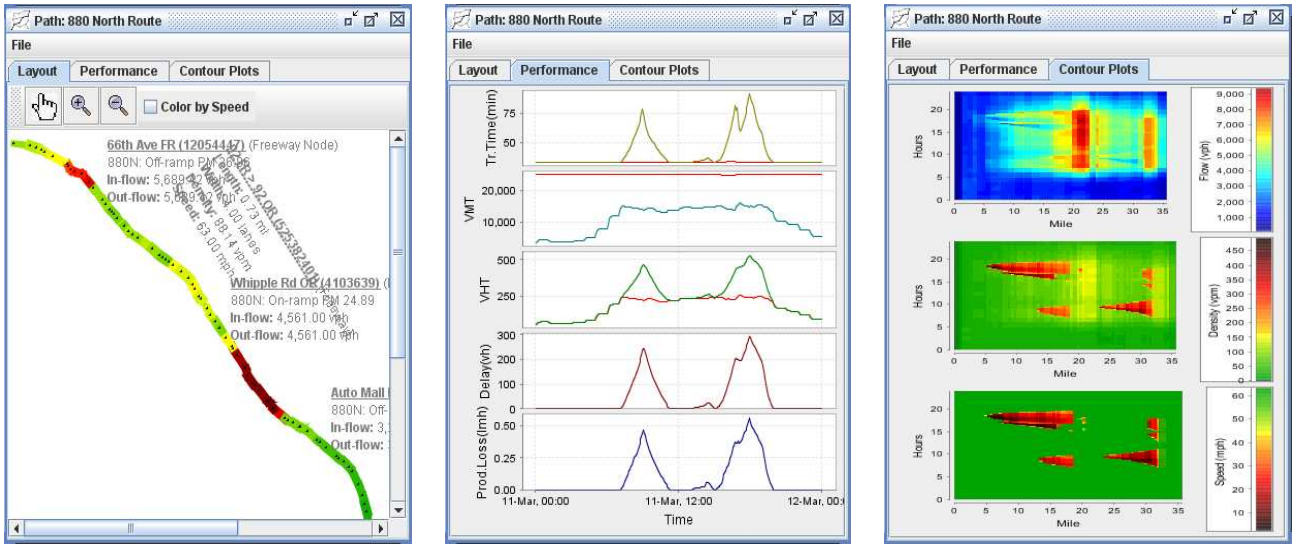
$$CongestionWaveSpeed = \frac{Capacity}{JamDensity - CriticalDensity}.$$

3. **Events** section allows the user to select in the combo box and generate by clicking **Generate** button an event for this link. For all links there is one possible event – the change of fundamental diagram (see Section 7.7.4). For source links there are two additional events – the change of queue limit (see Section 7.7.5) and the change of demand coefficient (see Section 7.7.6).

7.6. Path Window

Path window (Figure 7-7) has three tabs.

- **Layout** tab (Figure 7-7a) displays the interactive map of a path that connects given origin with given destination. This tab has the same functionality as the **Layout** tab in the network window (see Section 7.3).
- **Performance** tab (Figure 7-7b) with 5 plotting areas
  1. **Travel Time** – plots the instantaneous travel time through the path in *dark yellow* versus minimal travel time through this path in *red*. Travel time values are given in *minutes* (min).
  2. **VMT** – plots the actual vehicle-miles-traveled on the path in *cyan* versus maximum achievable vehicle-miles-traveled if all the links of the path operated at capacity in *red*.
  3. **VHT** – plots the actual vehicle-hours-traveled on the path in *dark green* versus critical vehicle-hours-traveled had all the vehicles present in the path traveled with free flow speed in *red*.
  4. **Delay** – plots delay caused by congestion in the path and/ or by the queue at the first link of the path. The delay values are given in *vehicle-hours* (vh).
  5. **Productivity Loss** – plots productivity loss due to congestion at the path. The productivity loss values are given in *lane-mile-hours* (lmh).



(a) (b) (c)

Figure 7-7: Link window – (a) **Simulation** tab; (b) **Performance** tab; (c) **Contour Plots** tab.

- **Contour Plots** tab (Figure 7-7c) with three contour plots.
  1. **Flow** computed in *vehicles per hour* (vph).
  2. **Density** computed in *vehicles per mile* (vpm).
  3. **Speed** computed in *miles per hour* (mph).

  On all of these plots the traffic direction is from left to right. Horizontal axis represents the length of the path in miles always starting at 0. Vertical axis represents time in hours increasing from bottom up, always starting at 0. To the right of each of the contour plots there is a color map with the scale indicating which color corresponds to which value.

File → Save… menu command in the path window brings up a file dialog that allows the user to save the data displayed in the **Performance** and **Contour Plots** tabs in a CSV file.

7.7. Events

One of the key features Aurora offers to the user is the ability to create simulation *scenarios*. This is done by means of events that change configuration parameters at given network elements at user-specified times. A scenario is a list of events generated by the user.

An event is characterized by its type, which determines the specific parameter change that occurs when the event fires, the ID of a network element whose parameter(s) must be changed and the time of activation. It also contains description whose sole purpose is to remind the user about the meaning of a particular event. Table 7-2 presents an XML block describing a sample event. Here, parameter `neid` specifies the ID of the network element where this event should occur, parameter `tstamp` defines the activation time in hours (in this particular example it is 14 hours 30 minutes), parameter `enabled` set to `true` indicates that this event will fire, `false` value would disable it. The `<description>` block contains the event description.

All the event editors described in the following sections have **Description** and **Activation Time** fields. Other fields depend on the event type.

The user can generate events at any point before or during the simulation. If the activation time specified by the user is smaller than current simulation time, then the event will be activated before the next simulation step. Events with activation time 0 (`0h0m0s`) fire before the first simulation step. Once generated, events show up in the **Events** tab of the scenario frame in the Simulator window (Figure 7-1#5). Events cannot be deleted – they can be disabled by un-checking the **Enabled** check box. Double-clicking on an event in that table brings up the event editor allowing the user to modify the event parameters.

NOTE: to remove an event from the list completely, the configuration file must be edited manually – the XML block describing this event has to be deleted.

7.7.1. Change of Network Control Mode

The change of network control mode event (Java class `aurora.hwc.EventNetworkControl`) may occur only at networks. It turns on or off the mainline and queue control for a given network. It means that actually mainline and queue controllers on all the nodes that belong to this network are turned on or off. If the network contains sub-networks, these sub-networks are also subject to the same control mode change. If the mainline control is turned off (its mode is set to `false`), then the mode of queue control is irrelevant as queue controllers can only be invoked from within mainline controllers. Thus, setting mainline control mode to `false` de facto turns off both, mainline and queue control.

Table 7-2 presents an XML block of a configuration file describing the change of network control mode event.

The `<control>` block describes the new control mode: `mainline="true"` turns on the mainline control and `queue="true"` turns on the queue control.

```
<event class="aurora.hwc.EventNetworkControl" neid="1488001"
                                              tstamp="14.5" enabled="true">
   <description>Control mode change: turn on queue control</description>
   <control mainline="true" queue="true" />
</event>
```

Table 7-2: Representation of the network control mode change event in a configuration file.

Figure 7-8 shows the editor for this event.

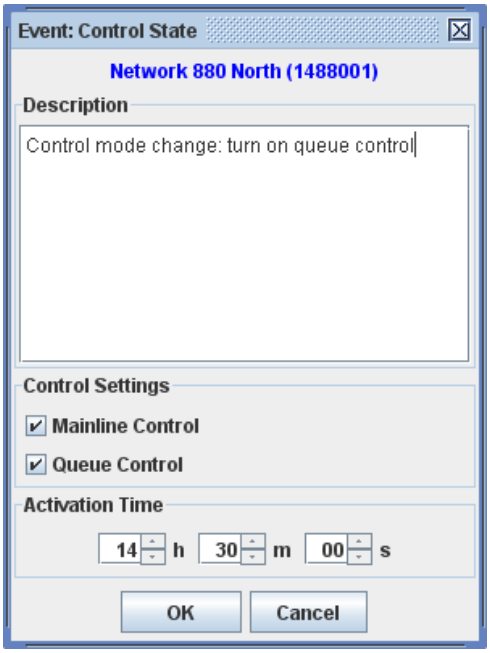


Figure 7-8: Network control mode change event editor.

7.7.2. Change of Simple Controller

The change of simple controller event (Java class `aurora.hwc.EventControllerSimple`) may occur only at nodes. It assigns user-specified controller to a user-selected link at the node. This event may be used to simulate control strategies that in the morning hours use different algorithms from the ones in the afternoon.

Table 7-3 presents an XML block of a configuration file describing the change of simple controller event.

The `<lkid>` block specifies the ID of an in-link to which the controller will be assigned.

The `<controller>` block describes the controller that is to be assigned. See Section 7.8 for more detail about controllers.

```
<event class="aurora.hwc.EventControllerSimple" neid="14104636"
                                                 tstamp="10.0" enabled="true">
   <description>Local Controller change at 29th St on-ramp</description>
   <lkid>135246271</lkid>
   <controller class="aurora.hwc.control.ControllerALINEA" tp="0.0834">
      <limits cmin="10.0" cmax="200.0" />
      <qcontroller class="aurora.hwc.control.QOverride">
         <parameter name="delta" value="120.0" />
      </qcontroller>
      <parameter name="upstream" value="false" />
      <parameter name="gain" value="60.0" />
   </controller>
</event>
```

Table 7-3: Representation of the simple controller change event in a configuration file.

Figure 7-9 shows the editor for this event. The user can select the input link to which the controller should be assigned in the **Input Links** section and specify the controller in the **Simple Controller** section. The combo box allows the user to choose a controller and clicking on the **Properties** button brings up the editor window for the chosen controller (see Section 7.8). The queue controller can be

assigned from the controller editor. The choice of **None** from the controller list disables the **Properties** button.

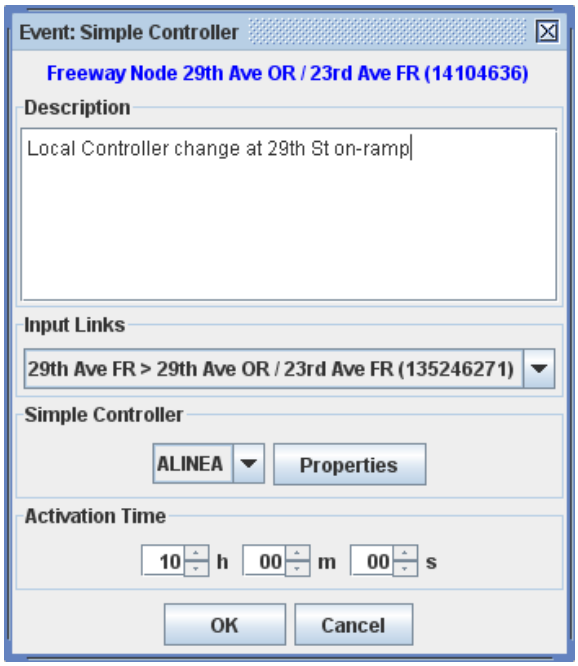


Figure 7-9: Simple controller change event editor.

7.7.3. Change of Split Ratio Matrix

The change of split ratio matrix event (Java class `aurora.hwc.EventSRM`) may occur only at nodes. It assigns new user-defined split ratio matrix to the node. This event may be used for managing time-variant split ratios.

Table 7-4 presents an XML block of a configuration file describing the change of split ratio matrix event.

The `<srm>` block defines the split ratio matrix. Each `<splitratios>` sub-block represents a row of the matrix. All `<splitratios>` sub-blocks must contain the same number of values corresponding to the number of columns of the matrix. It is generally required (but not enforced) that these values be in the range between 0 and 1 and sum up to 1 in each row. The number of rows must be the same as the number of input links for the node. The number of columns must be the same as the number of output links for the node.

```
<event class="aurora.hwc.EventSRM" neid="14104636" tstamp="0.0" enabled="true">
   <description>Split Ratio Matrix change:
                  increased output to 23rd Ave</description>
   <srm>
      <splitratios>0.81, 0.09</splitratios>
      <splitratios>0.99, 0.01</splitratios>
   </srm>
</event>
```

Table 7-4: Representation of the split ratio matrix change event in a configuration file.

Figure 7-10 shows the editor for this event.

The user can edit matrix values by double-clicking on them.

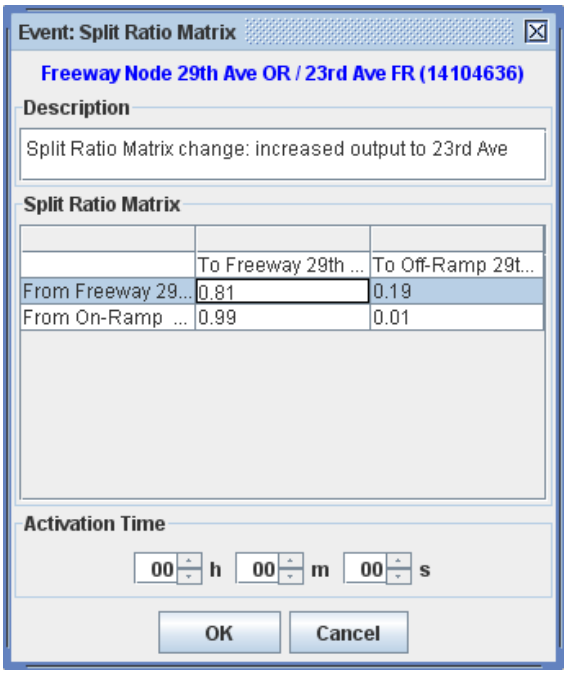


Figure 7-10: Split ratio matrix change event editor.

7.7.4. Change of Fundamental Diagram

The change of fundamental diagram event (Java class `aurora.hwc.EventFD`) may occur only at links. It sets new user-defined fundamental diagram for the link. This event may be used to simulate incidents that result in lane closures and hence, capacity drop or any other manipulation with the link capacity and free flow speed.

Table 7-5 presents an XML block of a configuration file describing the change of fundamental diagram event.

The `<fd>` block defines the fundamental diagram with `densityCritical` being critical density, `densityJam` – jam density, and `flowMax` – the capacity.

```
<event class="aurora.hwc.EventFD" neid="34936101" tstamp="15.6" enabled="true">
   <description>Accident at Whipple Rd: 2 lanes blocked
                            (CHP No. 1133 on 09/27/2007)</description>
   <fd densityCritical="64.0" densityJam="266.0" flowMax="4000.0" />
</event>
```

Table 7-5: Representation of the fundamental diagram change event in a configuration file.

Figure 7-11 shows the editor for this event.

Fundamental Diagram section displays the fundamental diagram, which the user can modify by adjusting **Capacity**, **C. Density** (critical density) and **J. Density** (jam density) parameters. Instead of critical and jam densities, the user can adjust free flow speed (parameter **V**) and congestion wave speed (parameter **W**) if it is more convenient. Capacity, critical density, jam density, free flow speed and congestion wave speed values are related as shown in Section 7.5.

The fundamental diagram parameters cannot be set arbitrarily. It is required that the fundamental diagram always remains a triangle. The editor would prevent the user from assigning negative values to any of these parameters or making jam density less than critical density. When the user would change capacity or critical density, free flow and congestion wave speeds would adjust automatically. Changing jam density triggers automatic adjustment of the congestion wave speed. And the other way around, if the user changes free flow speed, critical and jam densities adjust automatically. Change in the congestion wave speed results in automatic adjustment of the jam density.

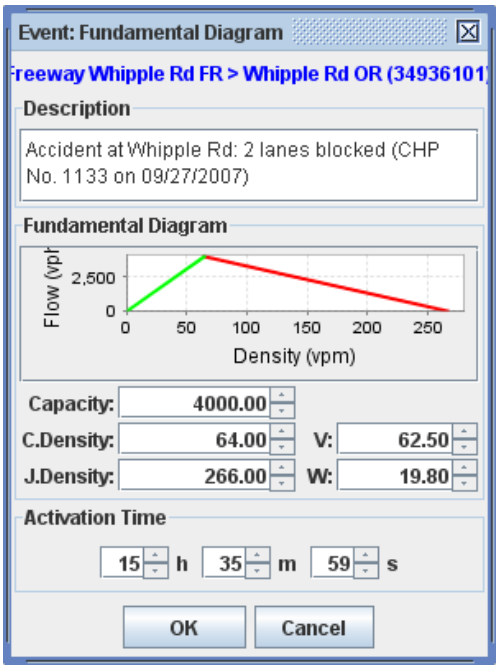


Figure 7-11: Fundamental diagram change event editor.

7.7.5. Change of Queue Limit

The change of queue limit event (Java class `aurora.hwc.EventQueueMax`) may occur only at *source links*. It sets new user-defined queue limit to be used by the queue controller (if a queue controller is assigned to this link). This event may be used to simulate the situation when different queue sizes are allowed at different times of day.

NOTE: queue limit parameter makes sense only for the source links. To instruct the queue controller that operates at an ordinary link about the queue size change, use the change fundamental diagram event (see Section 7.7.4), since queue controllers treat critical density as queue limit at such links. For more detail, see Section 7.9.

Table 7-6 presents an XML block of a configuration file describing the change of queue limit event.

The `<qmax>` block defines the queue limit. In this particular example the queue limit is set to 150 vehicles.

```
<event class="aurora.hwc.EventQueueMax" neid="624395803"
                                        tstamp="16.25" enabled="true">
   <description>Increase queue limit to 150 at A ST on-ramp</description>
   <qmax>150</qmax>
</event>
```

Table 7-6: Representation of the queue limit change event in a configuration file.

Figure 7-12 shows the editor for this event.

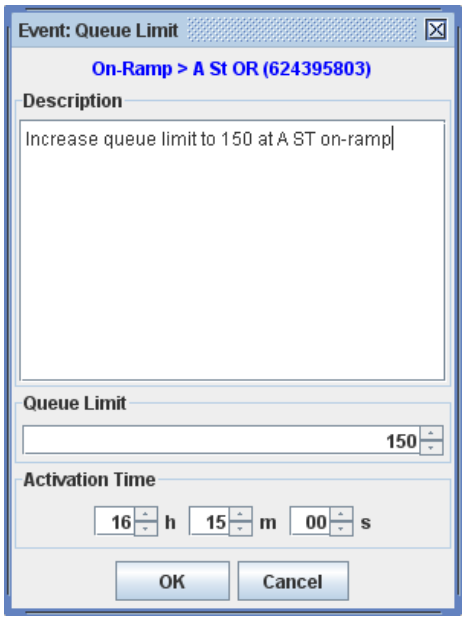


Figure 7-12: Queue limit change event editor.

7.7.6. Change of Demand Coefficient

The change of demand coefficient event (Java class `aurora.hwc.EventDemand`) may occur only at *source links*. It sets the coefficient by which the demand value is multiplied before being

processed by the Simulator. This event may be used to test scenarios such as: what if the afternoon demand would increase by X%?

Table 7-7 presents an XML block of a configuration file describing the change of demand coefficient event.

The `knob` parameter in the `<demand>` block defines the demand coefficient.

```
<event class="aurora.hwc.EventDemand" neid="131946151"
                                      tstamp="9.25" enabled="true">
   <description>Flow from 42nd Ave at 90%</description>
   <demand knob="0.9" />
</event>
```

Table 7-7: Representation of the demand coefficient change event in a configuration file.

Figure 7-13 shows the editor for this event.

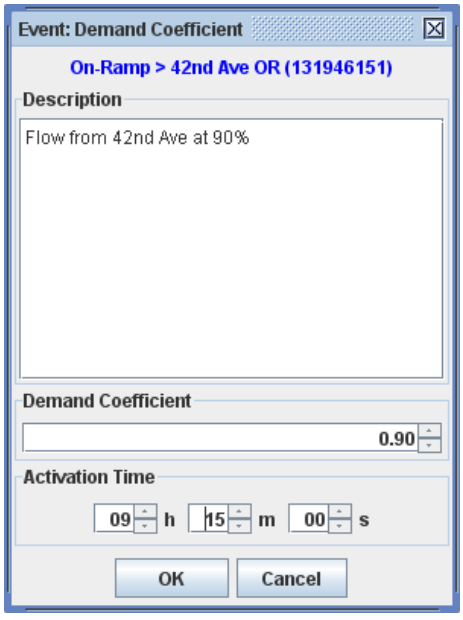


Figure 7-13: Demand coefficient change event editor.

7.8. Mainline Control

By mainline control we imply ramp metering. Mainline controllers reside at nodes and operate on input links for those nodes. Every input link may have a controller assigned to it. A controller determines the maximum flow the link may release (if no controller is assigned, the maximum flow is determined by the minimum of the link capacity and the available downstream capacity).

The common parameters of all mainline controllers are the **Time Period** (Figure 7-14) – parameter `tp` in Table 7-8, specified in hours – which defines how often the simulator requests the maximum flow rate from the controller; the **Rate Limits** (Figure 7-14) specified in *vehicles per hour* (vph) – the `<limits>` block in Table 7-8 – which sets the lower (**Min** parameter, `cmin` in configuration file) and upper (**Max** parameter, `cmax` in configuration file) bounds for the maximum flow the controller can allow; and **Queue Controller** (Figure 7-14) that can be optionally assigned to control the size of a queue stored at the link. When the selection of the queue controller is other than **None**, the **Properties** button is enabled – clicking on it brings up an editor for the selected queue controller. For more detail see Section 7.9. See Table 7-3 for an example of how queue controller is described within a mainline controller in a configuration file.

7.8.1. TOD

The Time Of Day – TOD – controller (Java class `aurora.hwc.control.ControllerTOD`) is the simplest mainline controller, which returns a fixed maximum flow rate depending on the time of day. It maintains a timetable of flow rates. Each time the controller is invoked, it looks at the current simulation time and finds the appropriate flow value in its rate schedule.

Table 7-8 an XML block of a configuration file describing the sample TOD controller.

The `<todrate>` block defines an entry in the rate schedule. Time is specified in *hours*, flow rate in *vehicles per hour* (vph). The first entry in the timetable `time="5.0" rate="360"` indicates that starting at 5 am until the next time in the schedule (in this example it is 12 noon) the maximum flow rate must be 360 vph. If the simulation time is less than 5 hours, then the controller returns its upper limit as the maximum flow rate. In this example it is 2500 vph (parameter `cmax` in the `<limits>` block).

```
<controller class="aurora.hwc.control.ControllerTOD" tp="0.5">
   <limits cmin="60.0" cmax="2500.0" />
   <todrate time="5.0" rate="360.0" />
```

```
   <todrate time="12.0" rate="2400.0" />
   <todrate time="14.0" rate="360.0" />
   <todrate time="20.0" rate="2400.0" />
</controller>
```

Table 7-8: Representation of TOD controller in a configuration file.

Figure 7-14 shows the editor for TOD controller.

Rate Schedule section displays the timetable of flow rates. Double-clicking on a particular rate value allows the user to edit this value. Double-clicking on a particular time brings up a dialog window in which the user may edit both, time and rate values. Similar dialog window pops up when the user clicks on the **Add** button, and the user can add a new entry to the timetable. To delete an entry, select the entry in the table then click on **Delete** button. If the table is empty, the controller will return the value of **Max** parameter as the maximum allowed flow rate.

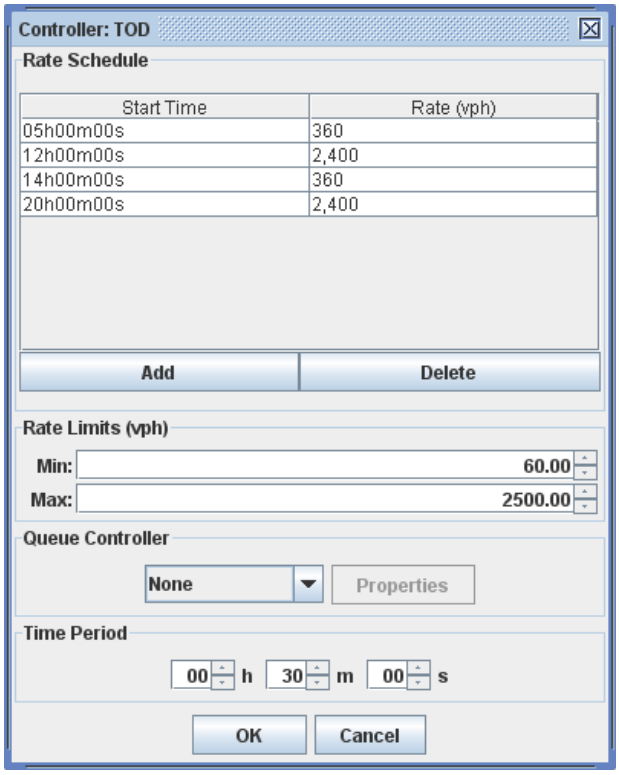


Figure 7-14: TOD controller editor.

7.8.2. ALINEA

The ALINEA controller [2] (Java class `aurora.hwc.control.ControllerALINEA`) uses the following formula for the rate computation:

$$Rate(t) = Rate(t-1) + Gain \times (CriticalDensity - Density(t)),$$

where *gain* is a user-settable parameter which is typically taken equal to the free flow speed at the link whose occupancy is measured. By default, it is the node's outgoing link. However, it may be a 'peer' link to the link on which ALINEA operates – it is anther input link for the node, if such exists.

Table 7-9 an XML block of a configuration file describing the sample ALINEA controller.

The `<parameter>` blocks describe ALINEA parameters. The `upstream` parameter indicates which link should be used for the occupancy measurement – the outgoing link if its value is set to `false`, or another incoming link otherwise. The value of `gain` parameter defines the controller gain.

```
<controller class="aurora.hwc.control.ControllerALINEA" tp="0.0167">
   <limits cmin="60.0" cmax="5000.0" />
   <parameter name="upstream" value="false" />
   <parameter name="gain" value="60.0" />
</controller>
```

Table 7-9: Representation of ALINEA controller in a configuration file.

Figure 7-15 shows the editor for ALINEA controller.

Parameters section displays the ALINEA parameters. **Feedback from upstream** checkbox when un-checked indicates that the outgoing link should be used for occupancy measurement, when checked – another incoming link. **Gain** parameter defines the controller gain.

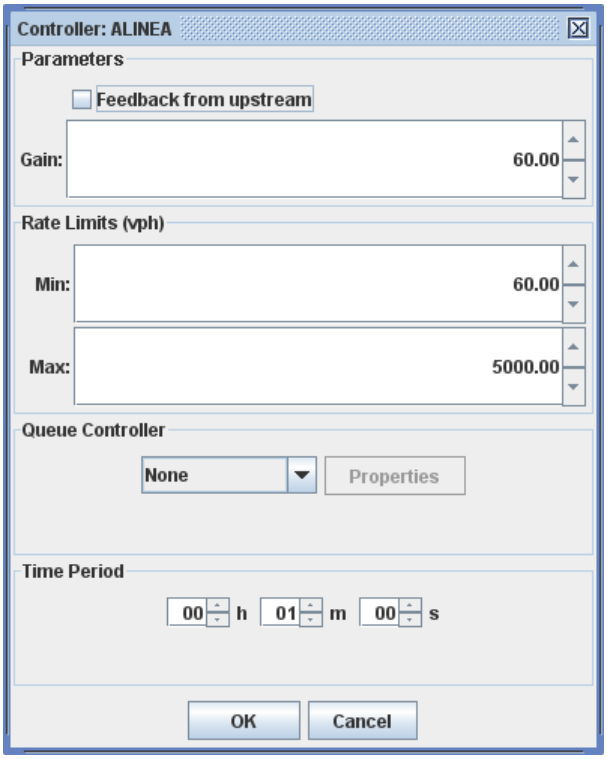


Figure 7-15: ALINEA controller editor.

7.9. Queue Control

The purpose of queue control is to keep track of the queues that result from the ramp metering and when those become too large, dispose of them by increasing the maximum flow rate allowed by the mainline controller. The queue becomes too large when its size exceeds the queue limit. Queue size is explicitly computed for *source links* every simulation step, and these links have queue limit parameter against which the queue size is compared. For ordinary links these values are obtained as follows:

$$QueueSize(t) = LinkLength \times Density(t), \text{ and } QueueLimit = LinkLength \times CriticalDensity.$$

Queue controllers operate as part of mainline controllers. The mainline controller chooses the largest value of the two for the maximum flow rate – one computed by itself, and the other one computed by a queue controller.

7.9.1. Queue Override

The simplest queue controller is Queue Override [3] (Java class `aurora.hwc.control.QOverride`). It compares the queue size with the queue limit, and if the former exceeds the latter, then it returns the flow produced by the link during the previous simulation step increased by a certain user-defined delta vehicles per hour for each lane of the link it operates on:

$$Rate(t) = LinkFlow(t-1) + Delta \times NumberOfLanes;$$

otherwise, it lets the mainline controller compute the maximum flow rate.

Table 7-10 an XML block of a configuration file describing the sample Queue Override queue controller.

The value of parameter `delta` specifies the amount per lane by which the link flow from the previous simulation step must be increased in *vehicles per hour* (vph).

```
<qcontroller class="aurora.hwc.control.QOverride">
   <parameter name="delta" value="120.0" />
</qcontroller>
```

Table 7-10: Representation of Queue Override queue controller in a configuration file.

Figure 7-16 shows the editor for Queue Override queue controller.

Parameter **Delta** specifies the amount per lane by which the link flow from the previous simulation step must be increased in *vehicles per hour* (vph).

Figure 7-16: Queue Override queue controller editor.

7.9.2. Proportional

Another queue controller implemented in Aurora is Proportional [3] (Java class `aurora.hwc.control.QProportional`). Again, if the queue size is below the limit, then it lets the mainline controller to compute the maximum flow rate. Otherwise, the rate is computed using formula

$$Rate(t) = Demand(t) + Kp \times \frac{QueueSize(t) - QueueLimit}{SamplingPeriod},$$

where *Kp* is proportionality coefficient whose default value is 1. Demand at given time is known for *source links*, and represents the desired flow coming from upstream for ordinary links.

Table 7-11 an XML block of a configuration file describing the sample Proportional queue controller.

The value of `kp` parameter defines the proportionality coefficient used in the formula above.

```
<qcontroller class="aurora.hwc.control.QProportional">
   <parameter name="kp" value="1.0" />
</qcontroller>
```

Table 7-11: Representation of Proportional queue controller in a configuration file.

Figure 7-17 shows the editor for Proportional queue controller.

Parameter **Kp** defines the proportionality coefficient used in the formula above.

Figure 7-17: Proportional queue controller editor.

7.10. Configuration File

Aurora uses XML format for its configuration files. The structure of a typical configuration file is presented in Table 7.12.

```
<?xml version="1.0" encoding="UTF-8"?>
<AuroraRNM>
   <network class="aurora.hwc.NodeHWCNetwork" id="1488001" name="880 North"
            top="true" controlled="true" tp="0.00138888">
      ...
   </network>

   <settings>
      ...
   </settings>

   <EventList>
      ...
   </EventList>

   <DemandProfile>
      <demand id="544425365" tp="1" knob="1">948.0, 888.0, 996.0, 936.0, 1020.0,
792.0, 864.0, 780.0, 816.0, 816.0, 732.0, 804.0, 756.0, 720.0, 708.0, 732.0,
720.0, 804.0, 876.0, 864.0, 840.0, 816.0, 828.0, 708.0</demand>
      ...
   </DemandProfile>
</AuroraRNM>
```

Table 7-12: Structure of Aurora configuration file.

The main XML block that contains everything else is `<AuroraRNM>`.

The `<network>` block covered in Section 5.1.3 contains network geometry – list of nodes including sub-networks, list of links and optionally list of ODs.

The optional `<settings>` block covered in Section 7.2 contains simulation settings.

The optional `<EventList>` block contains event scenario – list of events covered in Section 7.7.

The optional `<DemandProfile>` block is used to overwrite the demand profiles specified within the links – it contains the sequence of `<demand>` sub-blocks, each similar to the one described in Section 5.1.1 with additional parameter `id` referring to link for which the demand is specified.

8. GIS Importer

GIS Importer lets users read road network data in an existing GIS data (.shp and .dbf) files and export it to XML file used as configuration file by Aurora. The following information is extracted from GIS, processed, and exported to Aurora XML configuration file.

Aurora Information	Present in GIS	Notes on Processing
Link ID	Y	Most GIS data provide unique numeric IDs to links.
Link length	Y	
Link Java class	Y	`aurora.hwc.LinkFwML`, etc. determined from the road type of link (CCSTYLE field in SANDAG data; ACC field in TANA data)
(Link description)	Y	GIS data provides street and highway names for each link. Currently, Aurora doesn't allow link description.
Link from, to coordinates	Y	Only the first and last coordinates of default geometry (poly-lines) are kept.
Link from, to node ID		SANDAG only. If not present, nodes are constructed from link endpoints.
Link predecessors and successors		Nodes that connect to the specified link. Derived from network geometry.
Node ID		SANDAG provides numeric node ID. When not specified, importer uses the x-y coordinate as the node ID. E.g. node at coordinate -10.54 and 38.12 is named "-10.54 38.12".
Node Java class		`aurora.hwc.NodeFreeway` etc. determined using road type of links connected to the node.
Node coordinate		Inferred from from-to coordinates of links.
Node description		Generated as "from-link-desc to-link-desc".
Node predecessors and successors		Links that connect to the specified node. Derived from network information.
Link one-way vs. two-way information	Y	If two-way, 2 separate aurora links are created for each direction.
Link road type	Y	
Node split ratio		By default, all incoming traffic is split evenly to outgoing links.
Link number of lanes		SANDAG only. By default, 1-lane is assigned.

Table 8-1: Data exported from GIS databases to Aurora configuration file.

8.1. Starting GIS Importer

Run GIS Importer by selecting **Start → All Programs → TOPL → Aurora → GISImporter**.

8.2. Opening GIS File

1. Select **File → Open GIS File...**; file dialog appears.
2. Browse to the GIS file to use and select it.

8.3. Geometric Filtering via OpenJUMP

Geometric filtering of GIS network data can be best done in dedicated GIS software. Commercial GIS software such as *ESRI ArcView* as well as open source GIS tools could be used. This section focuses on an open source GIS platform, *OpenJUMP*.

OpenJUMP is a Java based open source Geographic Information System (GIS) platform. This is used for geometric filtering and other manipulation of GIS files. See http://openjump.org/wiki/show/HomePage for details.

8.3.1. Installing OpenJUMP

1. If Java is not installed already, download and install Java from http://www.java.com/en/download/windows_manual.jsp .

   This step may not be necessary if Java is installed already.

2. Download and install OpenJUMP from sourceforge.net/projects/jump-pilot/. Among many choices, download **OpenJUMP win32 installer** (Latest 1.2 D). Filename is **Setup-openjump12d.exe**.

8.3.2. Open GIS Data in OpenJUMP

1. Start OpenJUMP (**Start → Programs → OpenJUMP Folder → OpenJUMP**). An empty project (**Project 1**) is open by default.
2. Go to **File → Load Dataset(s)** menu.
3. At the bottom of "Load Dataset(s)" dialog, there is the "Format" dialog. Default is JUMP GML. Select **ESRI Shapefile** instead.
4. Browse to the directory where the shape file is stored. Select the shape file.
    - On the left of the screen is CATEGORY - LAYER tree. Under the "Working" category, a new layer (say "input") is added which is the shape file you just opened.

8.3.3. Geometrically Filter GIS data

There are two ways of doing this: using "Select Features Tool" and using "Masking Polygon".

8.3.3.1. Using "Select Features Tool"

1. Activate the "input" layer by right-clicking "input" layer node under "Working" category in the layer tree pane.
2. Select **Select Feature Tool** in the toolbar (arrow shape).
3. Select features of interest by drawing a box around them.
    - Pressing `Ctrl` key while making selection lets you keep previous selections.
4. Right-click within the selection; context menu appears.
5. Select **Replicate Selected Items...** in the context menu.
6. Leave **Replicate to new layer** checked; click **OK**.
7. A "new" layer is created. The features selected in "input" layer is now replicated in this new layer.

- To confirm whether the items are copied correctly, make only the “new” layer visible. You can do this by un-checking “input” layer and checking “new” layer in the layer tree pane.
- If only the selected features are visible, everything is OK.

8.3.3.2. Using "Masking Polygon"

1. Activate the “input” layer by right-clicking “input” layer node under “Working” category in the layer tree pane.
2. Draw convex hull or polygon or fence on a new layer, say “Polygon” layer.
3. Select **Tools** → **Queries** → **Spatial Queries...**; “Spatial Query” dialog opens.
4. In “Spatial Query” dialog specify:
   - Source layer: **input**;
   - Relation: is-covered-by or intersections;
   - Mask layer: **Polygon** layer;
   - Leave **Create a new layer for the result** checked.
5. A “new” layer is created with the selected (masked) features.
6. Same as above.

An advantage of this scheme is that

1. The masking polygons (“Polygon” layer) can be saved for later use.
2. More principled geo-filtering approach is possible.

8.3.4. Export Features in the Data to GIS File

1. Right-click the “new” layer in the tree pane. A context menu appears.
2. Select **Save Dataset As...** in the context menu. “Save Dataset As” dialog box opens.
3. Again, select **ESRI Shapefile** as the “Format” (bottom of the dialog box)
4. Save the shape file to the path/file name of your choice.
    - If successful, three files (*.dbf, *.shp, and *.shx) are created.

8.4. Road Type Filtering

1. Select **Filter** → **Type Filter...**; type selection dialog appears.
2. Select those types you wish to keep.
3. Click **OK**.

8.4.1. Using OpenJUMP to Select Road Types

Since different GIS files follow different conventions in specifying road types, the user needs to provide the name of the road type attribute and the set of road type codes to select. Selecting a few major roads in OpenJUMP with "Feature Info Tool" turned on gives the user an idea about the following information:

- Attribute name of the road type: could be *CCSTYLE* or *ROADTYPE* etc.
- Attribute values for the road type: which numeric values correspond to freeway and major arterials?

8.5. Edge Simplification

Select **Filter** → **Simplify Edges**.

8.6. Export to XML

1. Select **File** → **Save as XML...**; XML file selection dialog appears.
2. Specify the path and file name of the target XML file.

8.7. Save as GIS

GIS importer lets users save the current network data as a set of GIS files (.shp + .dbf). Road type filtering and edge simplification results are reflected in the output GIS files. This functionality lets users perform a sanity check of the processed network data via GIS mapping software.

9. Appendix A – References

1. Daganzo, C. F., “The cell transmission model II: Network traffic.” in Transportation Research B, Vol.29, No.2, pp.79-93, 1995.

2. Papageorgiou M.; Hadj-Salem H.; Blosseville, H. M., “ALINEA: A local feedback control law for onramp metering.” in Transportation Research Record, No.1320, 1991.

3. Papageorgiou M.; Smaragdis E., “Series of new local ramp metering strategies.” in Transportation Research Record, No.1856, pp.74-86, 2004.

10. Appendix B – XML Schema for Aurora RNM Configuration File

```
<?xml version="1.0" encoding="UTF-8" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="AuroraRNM">

    <xs:complexType>

      <xs:sequence>

        <xs:element ref="network" />

        <xs:element ref="settings" />

        <xs:element ref="DemandProfile" />

        <xs:element ref="EventList" />

      </xs:sequence>

    </xs:complexType>

  </xs:element>

  <xs:element name="network">

    <xs:complexType>

      <xs:sequence>

        <xs:element ref="description" />

        <xs:element ref="MonitorList" />

        <xs:element ref="NodeList" />

        <xs:element ref="LinkList" />

        <xs:element ref="ODList" />

      </xs:sequence>
```

```
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="controlled" type="xs:boolean" use="required" />
      <xs:attribute name="class" type="xs:string" use="required" />
      <xs:attribute name="top" type="xs:boolean" use="required"
                                              default="false" />
      <xs:attribute name="tp" type="xs:decimal" use="required" />
      <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="settings">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="display" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DemandProfile" minOccurs="0" maxOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="demand" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="EventList" minOccurs="0" maxOccurs="1">
```

```
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="event" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="description" type="xs:string" />
  <xs:element name="MonitorList" minOccurs="0" maxOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="monitor" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="NodeList">
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="node" maxOccurs="unbounded" />
          <xs:element ref="network" maxOccurs="unbounded" />
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name="LinkList">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="link" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ODList" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="od" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="monitor">
  <xs:complexType mixed="true">
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="class" type="xs:string" use="required" />
    <xs:attribute name="id" type="xs:integer" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="node">
  <xs:complexType>
    <xs:sequence>
```

```
        <xs:element ref="description" />
        <xs:element ref="outputs" />
        <xs:element ref="inputs" />
        <xs:element ref="position" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="class" type="xs:string" use="required" />
      <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="link">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="begin" minOccurs="0" maxOccurs="1" />
        <xs:element ref="end" minOccurs="0" maxOccurs="1" />
        <xs:element ref="fd" />
        <xs:element ref="density" />
        <xs:element ref="dynamics" />
        <xs:element ref="position" />
        <xs:element ref="demand" />
        <xs:element ref="qmax" />
      </xs:sequence>
      <xs:attribute name="lanes" type="xs:decimal" use="required" />
      <xs:attribute name="length" type="xs:decimal" use="required" />
```

```
      <xs:attribute name="class" type="xs:string" use="required" />
      <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="od">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="PathList" />
      </xs:sequence>
      <xs:attribute name="begin" type="xs:integer" use="required" />
      <xs:attribute name="end" type="xs:integer" use="required" />
      <xs:attribute name="class" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="outputs">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="output" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="inputs">
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="input" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="position">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="point" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="begin">
    <xs:complexType>
      <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="end">
    <xs:complexType>
      <xs:attribute name="id" type="xs:integer" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="fd">
    <xs:complexType>
      <xs:attribute name="densityCritical" type="xs:decimal"
```

```
                                        use="required" />
      <xs:attribute name="flowMax" type="xs:decimal" use="required" />
      <xs:attribute name="densityJam" type="xs:decimal" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="density" type="xs:decimal" />
  <xs:element name="dynamics">
    <xs:complexType>
          <xs:attribute name="class" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="demand">
    <xs:complexType mixed="true">
      <xs:attribute name="knob" type="xs:decimal" use="required" />
      <xs:attribute name="tp" type="xs:decimal" use="optional" />
      <xs:attribute name="id" type="xs:integer" use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:element name="qmax" type="xs:decimal" />
  <xs:element name="PathList">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="path" maxOccurs="unbounded" />
      </xs:sequence>
```

```
  </xs:complexType>
</xs:element>
<xs:element name="output">
  <xs:complexType>
    <xs:attribute name="id" type="xs:integer" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="input">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="splitratios" />
      <xs:element ref="controller" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:integer" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="path">
  <xs:complexType mixed="true">
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="class" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="display">
  <xs:complexType>
```

```
      <xs:attribute name="timeMax" type="xs:decimal" use="required" />
      <xs:attribute name="tsMax" type="xs:integer" use="required" />
      <xs:attribute name="timeout" type="xs:integer" use="required" />
      <xs:attribute name="tp" type="xs:decimal" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="event">
    <xs:complexType>
      <xs:choice>
        <xs:element ref="demand" />
        <xs:element ref="description" />
        <xs:element ref="fd" />
        <xs:element ref="srm" />
      </xs:choice>
      <xs:attribute name="tstamp" type="xs:decimal" use="required" />
      <xs:attribute name="neid" type="xs:integer" use="required" />
      <xs:attribute name="enabled" type="xs:boolean" use="required"
                                                  default="true" />
      <xs:attribute name="class" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="point">
    <xs:complexType>
      <xs:attribute name="x" type="xs:decimal" use="required" />
```

```
      <xs:attribute name="y" type="xs:decimal" use="required" />
      <xs:attribute name="z" type="xs:decimal" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="splitratios">
    <xs:complexType mixed="true" />
  </xs:element>
  <xs:element name="controller">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="todrate" minOccurs="0" maxOccurs="unbounded" />
        <xs:element ref="limits" />
        <xs:element ref="qcontroller" />
      </xs:sequence>
      <xs:attribute name="class" type="xs:string" use="required" />
      <xs:attribute name="tp" type="xs:decimal" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:element name="parameter">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string" use="required" />
      <xs:attribute name="value" type="xs:string" use="required" />
    </xs:complexType>
```

```
</xs:element>


<xs:element name="todrate">
  <xs:complexType>
    <xs:attribute name="time" type="xs:decimal" use="required" />
    <xs:attribute name="rate" type="xs:decimal" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="limits">
  <xs:complexType>
    <xs:attribute name="cmin" type="xs:decimal" use="required" />
    <xs:attribute name="cmax" type="xs:decimal" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="qcontroller">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="parameter" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="class" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="srm">
  <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="splitratios" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

11. Appendix C – Glossary of Terms

Term	Meaning
ACC	Arterial Classification Code (ACC). The set of Arterial Classification Codes (ACCs) is TeleAtlas' system for categorizing roads according to the level of travel mobility that they provide in the road network. http://www.spatialinsights.com/catalog/product.aspx?product=230
CSV format	The coma-separated values file format is a file type that stores tabular data. The format dates back to the early days of business computing. For this reason, CSV files are common on all computer platforms. http://en.wikipedia.org/wiki/Comma-separated_values
Capacity	The maximal hourly rate at which vehicles reasonably can be expected to traverse a point or a uniform section of a lane or roadway during given time period under prevailing roadway, traffic and control conditions.
Critical density	The density for which the flow reaches its maximum (the road operates at capacity).
Delay	Difference between the actual VHT and the VHT that would be incurred if vehicles traveled at free flow speed. Measured by vehicle-hours.
Demand	The number of vehicles that desire to use a given facility during a specified time period.
Density	The number of vehicles occupying a segment of road of a given length about a given point at a given time instant.
Flow	The total number of vehicles that pass by given point during a given time interval, divided by the length of the time interval.
Free flow speed	The average speed of traffic measured under conditions of low volume, when vehicles can move freely at their desired speed.
Fundamental diagram	Triangular density-flow relation specified by capacity, critical density and jam density.
GIS	Geographic Information System.

	http://en.wikipedia.org/wiki/Geographic_information_system
Jam density	The density for which the flow drops to zero (traffic stops).
Productivity loss	The number of lane-mile-hours on the freeway lost due to reduced flow, while operating under congested instead of free-flow conditions.
SANDAG	San Diego Association of Governments. San Diego's Regional Planning Agency. http://www.sandag.org/
TANA	Tele Atlas North America. http://www.teleatlas.com/
VHT	Vehicle Hours Traveled, for a given unit of time and a given section of freeway, the amount of time spent by all of the vehicles on the freeway.
VMT	Vehicle Miles Traveled, for a given unit of time and a given section of the freeway, the sum of the miles of freeway driven by each vehicle.
XML format	The Extensible Markup Language (XML) is a general-purpose *specification* for creating custom markup languages. http://en.wikipedia.org/wiki/XML