Aurora Road Network Modeler

Alex A. Kurzhanskiy * **Jaimyoung Kwon** ** **Pravin Varaiya** *

* *University of California, Berkeley, CA 94720-1770, United States (e-mail: {akurzhan, varaiya}@eecs.berkeley.edu)*
** *California State University, East Bay, Hayward, CA 94542, United States (e-mail: jaimyoung.kwon@csueastbay.edu)*

Abstract: Aurora object-oriented framework is a set of generic Java classes and interfaces used as base objects in applications that model flow networks. A network consists of directed links, nodes and monitors. Nodes can be complex, being networks themselves. Controllers on nodes control flows coming from the input links. Monitors are special objects whose purpose is to monitor the state of given links and feed the observed data to controllers or generate events. Events change network configuration or inputs and are triggered at specified times. They can be user-defined, or generated by monitors based on the data observed.
Aurora Road Network Modeler (RNM) is a tool set for operational planning and management of travel corridors (freeways and surrounding arterials), which is based on Aurora object-oriented framework. Currently, it consists of three graphical applications with intuitive user interface: Simulator runs macroscopic traffic simulations, allowing the user to create simulation scenarios by means of events; Configurator is used to build road network configurations from scratch or edit the existing ones; and GIS Importer extracts road information from the GIS `.shp` and `.dbf` files and saves it in the XML format of Aurora configuration file.
Aurora RNM is open source and can be freely downloaded from the internet.

Keywords: traffic macro-simulation, corridor management, flows on graphs, ramp metering

1. INTRODUCTION

Aurora Road Network Modeler (RNM) [2] [1] is a software suite providing tools for quantative assessment of operational strategies designed to improve traffic conditions on congested freeways and surrounding arterials. The elements of such strategies are: *demand management*, which focuses on reducing excessive demand; *incident management*, which targets resources to alleviate accident hot spots; *traveler information*, which potentially reduces traveler buffer time; and *traffic control*, which implements aggressive ramp metering at locations where significant reductions in congestion are likely to occur. The quick quantative assessment helps *rank* a large set of operational strategies in terms of the benefits they will yield.

Aurora RNM includes *configuration*, *simulation* and *analysis* modules. Configuration module produces XML configuration files with network geometry, road parameters, demand profiles and controllers, that are used to run simulations. Basic objects, nodes and links of different types, allow the user to construct heterogeneous road networks. Aurora simulation is based on the macroscopic Cell Transmission Model (CTM) [4, 5], which, compared with microscopic simulators, requires negligible computational effort. CTM can be extended to freeway networks [3] and urban roads with signalized intersections [7, 1]. It appears well-suited for calibration with point measurements of aggregate traffic variables that are routinely available [6, 8]. The user can test different ramp metering and signal control strategies by turning on selected local or complex controllers. Various event classes make it possible to generate simulation scenarios, such as accidents, lane management, variable speed limit and demand change. The special objects, called *monitors*, can keep track of the state at selected links, coordinate control actions at nodes, or generate events at nodes or links when the monitored states reach certain thesholds. Monitors and events can model the impact of traveler information, incident management and the coordination of signal control on arterials with the ramp metering at freeway entries. The analysis module, which is currently under development, will compare the performance of different scenarios and control strategies, and generate reports.

The known similar open source project is SUMO [10], which has developed into a solid product since it started in 2001. SUMO, however, is a *micro-simulation* package and as such is suitable primarily for small size urban networks.

In the rest of the paper Section 2 presents Aurora framework, Section 3 describes Aurora RNM, and Section 4 provides some concluding remarks.

2. ARHITECTURE OF AURORA

The five foundation principles of Aurora framework are listed below.
1. *Multi-purpose* — basic structures and algorithms are generic and can be used for any flow network application, such as road traffic, irrigation canals, oil/gas pipelines. Application specific classes inherit from these basic struc-

[1] This research is conducted within the TOPL project, that is supported by the California Department of Transportation through the California PATH program.

tures.
2. *Scenario oriented* — the user can create scenarios: lists of events that change configuration or inputs, with times or conditions of their occurrence, and feed them to the simulator.
3. *Interactivity* — the simulator offers clear and simple user interface with good data visualization, allowing the user to modify simulation settings and scenarios at run time.
4. *Usability* — Aurora tools are easy to handle: the user interface is intuitive and the format of configuration files is convenient for maintaining a library of possible configurations and scenarios, and for creating new configurations out of existing ones.
5. *Scalability* — the user can seemlessly add new subnetworks to already existing network configurations, or connect two or more networks with each other.
Aurora RNM, the suite of applications for operational planning and management of travel corridors is built on top of Aurora framework.

2.1 Basic Objects

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 (water canal, pipeline, etc.), *simple node* — point where links merge and/or diverge, *complex node* — network built out of network elements, or *monitor* — an object that monitors the state of specified links and nodes in a network.

A link has direction and length. It must have either of the two nodes, *begin node* or *end node*, or both of them, attached to it. Links with no begin nodes are *source links*. Source links provide input to a system. In Aurora RNM, associated with source links are demand values and queues. Links with no end nodes are *destination links*. In Aurora RNM, the user can specify the available capacity downstream of a destination link, which allows to model congested boundary conditions. If this capacity is not specified, it is assumed that the downstream of the destination link is in free flow. Link have types. In Aurora RNM admissible types are *freeway*, *highway*, *HOV*, *interconnect*, *on-ramp*, *off-ramp*, and *street*. Each link objects has associated with it fundamental diagram and dynamics. Dynamics is declared as Java interface. CTM is its particular implementation, currently, the only one available. Any other macroscopic traffic model can be used to compute the link state, namely, traffic density. Density is implemented as a numerical vector of variable size allowing to distinguish vehicles by their types, e.g. SOV, HOV, trucks, or by origin-destination information. For each link, Aurora computes travel time, VHT [2], VMT [3], delay [4] and productivity loss [5].

A simple node [6] must have one or more input and one or more output links. Admissible node types in Aurora RNM are *freeway*, *highway*, *signal junction*, and *stop junction*. Nodes can be connected only with links of compatible types. For example, an on-ramp may be an input link only for freeway and highway nodes, and street links can be only connected to signal or stop junctions. For details, see Aurora RNM User Guide available online at [2]. Local controllers, if any, reside on nodes and are assigned to given input links, potentially restricting flows coming from these links. When there are multiple output links, nodes also carry information about what portions of which input flows must be directed to which output. Currently, for m inputs and n outputs in the node, it is implemented as an $m \times n$ split ratio matrix, where elements are nonnegative and sum up to 1 in each row.

While links and nodes physically form a network, a monitor is a special object whose purpose is to monitor the state of specified links, and feed the observed data to system wide controllers. Such monitors are called *control monitors*. There are also *event monitors*, which are used to generate certain events (such as split matrix change — to simulate traveler information affecting traffic flow directions) based on observed conditions (more about events in 2.2).

All described network elements — links, nodes and monitors — 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, nodes and monitors belong. Network objects are nodes, 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, and opens a door to parallel computation when simulation steps for different subnetworks can be computed concurrently by different processors. Another benefit of using a hierarchical structure is that different subnetworks may have different sampling periods, that is, simulation steps of different duration. 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 subnetworks with different sampling periods reduces computation time.

There are two other basic objects. Object *path* describes route from a node to node as a sequence of adjacent links. For each path, Aurora RNM computes travel time, VHT, VMT, delay and productivity loss based on corresponding data from links forming the path. Object *OD* describes a pair of origin and destination nodes together with list of paths connecting the two. A complex node 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 [7].

[2] Vehicle Hours Traveled — for a given unit of time and a given link, the amount of time spent by all of the vehicles in the link.
[3] Vehicle Miles Traveled — for a given unit of time and a given link, the sum of the miles of the link driven by each vehicle.
[4] Difference between the actual VHT and the VHT that would be incurred if vehicles traveled at free flow speed.
[5] Number of lane-mile-hours on the freeway lost due to reduced flow, while operating under congested instead of free-flow conditions; positive only in congestion, otherwise zero.

[6] We refer to it as node from now on, while referring to a complex node as a network.
[7] Origin and destination nodes may belong to the same subnetwork, while links at connecting paths are part of a different subnetwork. Such paths belong to an upper level network.

2.2 Events

Aurora RNM-specific events, changes in controllers and split ratio matrices for nodes, changes in demands, queue limits and fundamental diagrams for links, are derived from the generic Aurora event object and are handled by the Aurora event manager. Events that change fundamental diagrams are used to simulate traffic incidents and lane management by reducing or increasing capacity, and variable speed limit by modifying the free flow speed. Changing demands and split ratio matrices help imitate special events, road closures, or effects of displaying traveler information. Controller and queue limit changes may be part of complex ramp control strategies.

The event object carries the following information: activation time (in terms of simulation hours), ID of a network element where it must occur, new parameter values for this network element, and an `activate()` method that changes those parameters at a network element while storing the old values coming from the network element in their place. An event list is an optional part of the configuration file, but the user can generate new events before or during the simulation run as well as disable those already in the list.

When the simulator reads a configuration file, it places all events listed there into the queue sorted by event activation time. User generated events are inserted in the queue also based on their activation time. Events are triggered by the event manager. Before each simulation step, it selects those with activation time smaller than the next simulation time step, and activates them by invoking their `activate()` methods. Each activated event records its activation time. Then, the event manager moves the activated events from the event queue to the end of the event history list in the order they occured. Later, when the user resets the simulation, events are rolled back, or activated in reverse sequence with reverse action, returning to network element parameters their original values. Such maintenance of event queue and history list potentially allowes us to "rewind" simulation to any given point of its execution.

2.3 Computational Model

An object representing a network element contains the `dataUpdate()` method. It performs simulation step computations specific to the particular type of network element. The Aurora RNM recursive algorithm of `dataUpdate()` in a network is described next.

1. Check if at this time step any action is needed:

$$(k - k_0)\Delta t_0 < \Delta t, \tag{1}$$

where Δt is sampling period for this network, Δt_0 is the sampling period for the top level network, k is the current time step, k_0 is the time step at which the last action was performed.
If $k > 1$ and inequality (1) holds, then return without doing anything. Else, proceed to step 2.

2. For every monitor in the monitor list, call `dataUpdate()`. If present, each monitor has its own specific task—it may assign controller parameters, or generate events to be activated before the next simulation step or later, at prescribed time.

3. For every node in the node list, call `dataUpdate()`. If the node is complex, start the algoritm from step 1 with respect to this node. Else (the node is simple), compute input and output flows based on demand from upstream and available capacity of downstream links. This can be done in many ways.
Daganzo in [5] introduces the concept of priorities for multiple input flows and the first in, first out (FIFO) rule for multiple output flows.
In the Aurora RNM priorities are the fractions of the input flows accepted by the node, in case the upstream demand exceeds the downstream capacity (if the upstream demand is below the downstream capacity, priorities do not matter since all the vehicles from the upstream links can be accommodated by the downstream links). Different priority choices result in different flow values for the next simulation step. In the current Aurora RNM implementation we assume that the input priorities are proportionate to the input demands.
The FIFO rule means that if one of the output links cannot accommodate its allocation of flow, the total output flow is restricted [8]. In the Aurora RNM the FIFO rule implies that the input-output flow relations defined by the split ratio matrix must be preserved.
To summarize, we compute the input and output flows based on the input demands, satisfying the downstream capacity restrictions by assuming the input priorities to be proportionate to the demands, while preserving the input-output flow relations defined by the split ratio matrix.
Given $m > 0$ input and $n > 0$ output links,

(1) Compute input demands

$$\tilde{d}_i(k) = \min\left(v_i \rho_i(k_0), \mathcal{C}(\rho_i(k_0))\right),\ i = 1..m, \tag{2}$$

where v_i is free flow speed, $\rho_i(k_0)$ is the density at the input link i, and $\mathcal{C}(\rho_i(k_0))$ denotes flow value suggested by a controller, if a controller is assigned to the input link i.

(2) From the $m \times n$ split ratio matrix $\mathcal{B}$, and the m-dimensional demand vector $\tilde{d}(k)$ we get the input-output $m \times n$ demand matrix $D(k)$,

$$D_{ij}(k) = \mathcal{B}_{ij}\tilde{d}_i(k),\ \ i = 1..m,\ j = 1..n, \tag{3}$$

and output demands

$$d_j(k) = \sum_{i=1}^{m} D_{ij}(k),\ \ j = 1..n. \tag{4}$$

(3) Compute available output capacities

$$c_j(k) = \min\left(w_j(\bar{\rho}_j - \rho_j(k_0)), F_j\right),\ \ j = 1..n, \tag{5}$$

where w_j is congestion wave speed, $\bar{\rho}_j$ is the jam density, and F_j is the capacity of the output link j.

(4) Compute input-output demand matrix adjusted by the output link capacity restrictions, assuming the input priorities to be proportionate to the demands,

$$\hat{D}_{ij}(k) = \frac{\min(d_j(k), c_j(k))}{d_j(k)} D_{ij}(k), \tag{6}$$

for $i = 1..m$, $j = 1..n$, (in case $d_j(k) = 0$, we set $\hat{D}_{ij}(k) = 0$), and adjusted input demands

$$\hat{d}_i(k) = \sum_{j=1}^{n} \hat{D}_{ij}(k),\ \ i = 1..m. \tag{7}$$

[8] Vehicles unable to exit from the upstream link prevent all those behind, regardless of their destination, from continuing.

This step ensures that the adjusted input demand does not exceed the downstream capacity. More precisely,

$$\sum_{i=1}^{m} \hat{D}_{ij}(k) \le c_j(k),$$

with equality being achieved if and only if $d_j(k) \ge c_j(k)$.

(5) Compute input flows

$$\tilde{f}_i = \hat{d}_i \min_j \left\{ \frac{\hat{D}_{ij}}{\hat{d}_i \mathcal{B}_{ij}} \right\}, \quad i = 1..m,\ j = 1..n. \tag{8}$$

In case $\hat{d}_i = 0$ or $\mathcal{B}_{ij} = 0$ for all $j = 1..n$, we set $\tilde{f}_i = 0$.

(6) Compute output flows

$$f_j(k) = \sum_{i=1}^{m} \mathcal{B}_{ij} \tilde{f}_i, \quad j = 1..n. \tag{9}$$

Steps (5) and (6) implement the FIFO rule.

4. For every link in the link list, call `dataUpdate()`.

(1) Compute density and speed using model specific equations. For CTM, these are

$$\rho(k) = \rho(k_0) + \frac{\Delta t}{\Delta x}(f_u(k) - f_d(k)), \tag{10}$$

and

$$V(k) = f_d(k)/\rho(k), \tag{11}$$

where Δx is the link length, f_u is the upstream flow (flow entering the link), f_d is the downstream flow (flow exiting the link), and V is the speed. If the link is a source link, $f_u(k)$ equals current demand, otherwise $f_u(k)$ is computed in the begin node of the link as one of its output flows. If the link is a destination link,

$$f_d(k) = \min\left(v\rho(k_0), F\right),$$

where v is the free flow speed, and F is the capacity; otherwise $f_d(k)$ is computed in the end node of the link as one of its input flows.

(2) Compute performance characteristics.

$$Travel\ Time(k) = \Delta x / V(k). \tag{12}$$

$$VHT(k) = \rho(k)\Delta x \Delta t. \tag{13}$$

$$VMT(k) = V(k)\rho(k)\Delta x \Delta t. \tag{14}$$

$$Delay(k) = \begin{cases} 0, & \text{if } \rho(k) \le \rho^c \\ VHT(k) - VMT(k)/v, & \text{if } \rho(k) > \rho^c \end{cases}, \tag{15}$$

where ρ^c denotes critical density.

$$Productivity\ Loss(k) = \begin{cases} 0, & \text{if } \rho(k) \le \rho^c \\ \left(1 - \dfrac{f_d(k)}{F}\right) \Delta x \Delta t, & \text{if } \rho(k) > \rho^c \end{cases}. \tag{16}$$

5. Set $k_0 = k$ and return.

The sampling period Δt associated with a network must satisfy

$$\Delta t \le \min\left(\frac{\Delta x}{v}\right), \tag{17}$$

where minimum is taken over all links in the network.

2.4 Configuration

Once the process of a freeway corridor study is established, the most time consuming task is putting together a configuration file with road network description. This task requires attention to details and patience. Therefore, efficient configuration management is one of the priorities in the development of Aurora.

General configuration file contains the following blocks.

- Information about the network, which includes list of nodes, describing their types, positions, split ratio matrices and local controllers; list of links, describing their types, lengths, widths, fundamental diagrams and nodes they connect or are attached to; list of monitors (if any are present), that refer to complex controllers or events; and list of origin-destination pairs, each with a list of feasible paths.
- Demand profile for source links.
- Capacity profile for destination links.
- Event scenario — list of events describing what occurs, where and when.

There is no single source for full network description that includes geometry, fundamental diagrams, demand profiles and split ratios. For California freeways, PeMS [9] provides this information, but only for those ramps and lanes where detectors are installed. Other data comes from regional planning agencies (e.g. MTC in Bay Area, SANDAG in San Diego), GIS databases and Google maps. Computation of fundamental diagrams and split ratios, and demand generation, with the shortage of measurement data is a challenge. Thus, complete configuration files have significant value, and so establishing a repository of configurations makes sense.

Aurora configuration files use XML format whose syntax, data sets and validation rules as well as the schema, are described in the Aurora RNM User Guide that can be obtained at [2]. Types of links, nodes, monitors, controllers, events and dynamics are defined in the `class` attribute, which specifies what classes Aurora must instantiate upon reading the configuration. Configuration is modular — origin-destination lists, demand profiles and event scenarios, are separate blocks that can be optionally added to a configuration file or stored on their own. This, plus the hierarchical structure of Aurora in which a network is just another complex node, make the manipulation of configuration building blocks relatively easy and efficient.

3. AURORA ROAD NETWORK MODELER

Currently, Aurora RNM consists of three applications: Simulator, Configurator and GIS Importer. Here we shall briefly describe them. For a detailed description of their user interface and functionality, we refer the reader to the Aurora RNM User Guide that is available online at [2].

In the typical workflow, first, a road network is extracted from GIS files and saved in the XML format understood by Aurora. This task is performed using GIS Importer. Then, the configuration is edited in the Configurator. Editing the configuration includes assigning fundamental diagrams and demand profiles to links, split ratios and controllers to nodes. Finally, the main application of the

suite, the Simulator, is used to run simulations with given configuration files. In the Simulator the user can create simulation scenarios by means of events that change user-specified configuration parameters of the road network at user-specified times.

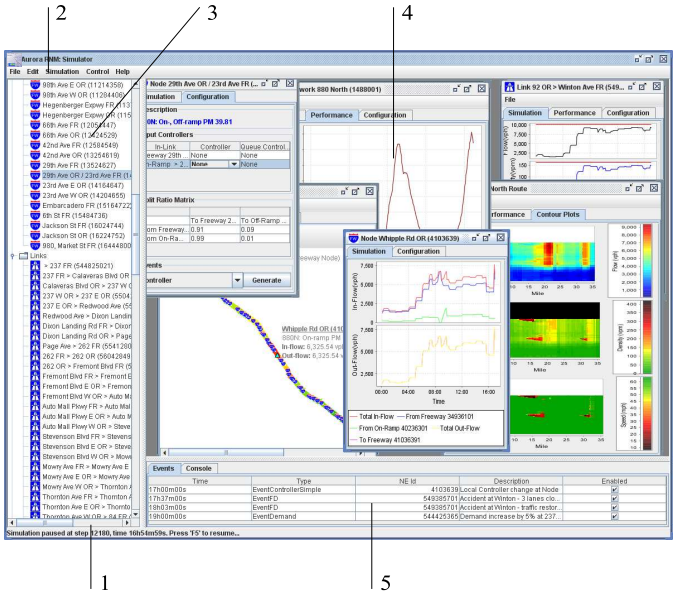

Fig. 1. Simulator

3.1 Simulator

Simulator is a graphical application that runs traffic simulations on road networks. Fig. 1 presents the face of Aurora Simulator. The application window is partitioned into five areas.

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 allow to load road network from file, start and stop simulation, turn control on and off, modify simulation settings, and save simulation.

3. Network tree displays network components: nodes, links, monitors, origin-destination pairs and paths in a hierarchical tree structure a la Windows Explorer. Special icons specify their types. Double-clicking on a component brings up a subwindow in the main frame with details and simulation data of that particular network element or path.

4. Main frame is used to host subwindows for selected network elements or paths. These subwindows display simulation data and performance measures related to the particular network elements or pathsas well as their configuration parameters.
Network subwindow shows the network map, color coded based on density or speed, and plots the computed total network delay as simulation runs.
Node subwindow plots the input and output flow values that correspond to possibly multiple in- and out-links.
Link subwindow displays plots of density, out-flow, speed and travel time as well as such performance characteristics as VMT, VHT, delay and productivity loss for that particular link. For source links — demand values and queue sizes are plotted.
Path subwindow shows the map of the corresponding path, color coded similarly to the network, displays the same performance measures as for links, only computed for the whole path, and contour plots for flow, density and speed. All the displayed data can be saved by the user in a CSV [9] file.
These subwindows also contain user interfaces for generating events on the corresponding network elements.

5. Scenario frame lists the events of the current simulation scenario: their type, description, activation time. The user can edit an event by double-clicking on it. 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 or inputs at given network elements at user-specified times. A scenario is a list of events generated by the user. The user can generate events at any point before or during the simulation. If the activation time specified by the user is smaller than the current simulation time, then the event will be activated before the next simulation step. Events with activation time 0 fire before the first simulation step. Once generated, events show up in the event list and cannot be deleted, but they can be disabled.

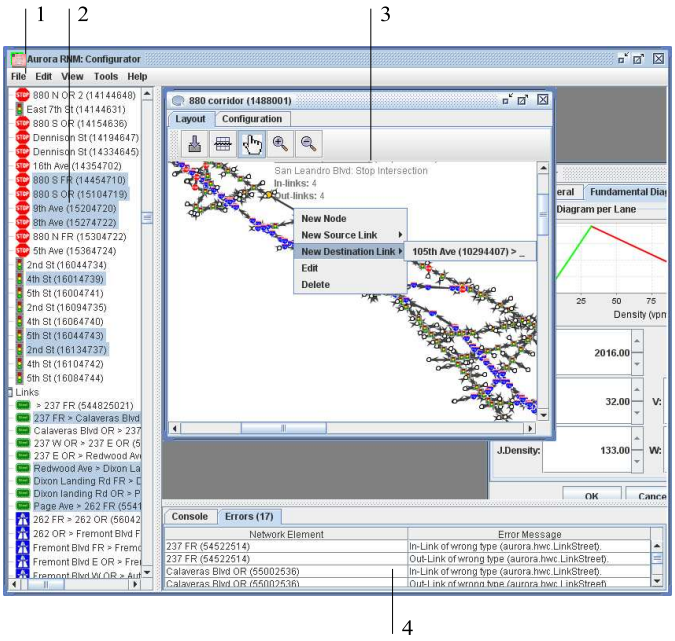

Fig. 2. Configurator

3.2 Configurator

Configurator is a graphical application that produces 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; provision road parameters such as fundamental diagrams and split ratios; and input demand

[9] Comma separated values file format that stores tabular data.

and output capacity profiles. The application window is partitioned into four areas (Fig. 2).

1. Menu bar whose commands allow the user to start new; open existing configuration files; filter links based on their type; validate the configuration to make sure that links are connected to nodes, nodes have both inputs and outputs, and that connected links and nodes have compatible types; edit simulation settings; and save the configuration in an XML file.

2. Network tree displays the network elements in the same way it does in the Simulator. The user can make multiple selections and open networks, nodes and links for editing.

3. Main frame is used to host editor subwindows for selected networks, nodes, links and monitors.
Network editor displays the network map and provides facility for deleting and reassigning the existing nodes and links as well as adding new ones and loading subnetworks from files. It also allows to add new and delete or edit the existing monitors. Setting sampling period for a network is also done in the network editor.
Node editor is used to modify node parameters — its ID, type, name, description, split ratios and local controllers. It is possible to edit multiple nodes at once with a single node editor.
Link editor is used to modify link parameters — its ID, type, length, number of lanes, initial density, fundamental diagram; for source links — queue limits and demand profiles; and for destination links — downstream capacity profiles. Similarly to the node editor, it is possible to edit multiple links at once with a single link editor.
Monitor editors are specific to particular monitors.

4. Error 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. Clicking on an error directs the user to the node or link where the error was reported.

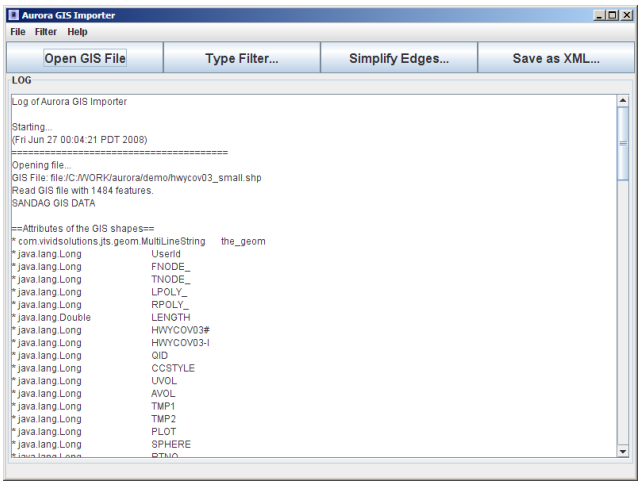

Fig. 3. GIS Importer

3.3 GIS Importer

GIS Importer is a simple graphical application (Fig. 3) that reads road network information in a GIS file and converts it to XML format understood by Aurora.

The application window has the following components
1. Menu bar whose commands allow the user to open GIS files; filter links based on their type; simplify network structure by removing unnecessary nodes; and save the road network as an aurora XML file.
2. Four buttons corresponding to the main commands.
3. Log window shows the helpful information as each command is executed.

The `Type Filter...` command lets the user export only road segments of certain types (e.g., freeway and major arterials) to Aurora xml files. To use this function, the database schema of the GIS file itself needs to have an attribute specifying the type (sometimes called `functional class`) of each road segment. In SANDAG GIS data, for example, such attribute is called `CCSTYLE` and in TeleAtlas GIS data for routing (Dynamap), the attribute is called `ACC`. The user needs to refer to the manual of the GIS file to find out the name of the road type/class attribute.

4. CONCLUSION

Aurora RNM is the first open-source *macro-simulation* package designed to help a traffic modeler in all stages of his/her job — starting from the creation of road networks from available GIS sources and efficiently editing their geometry and parameters, continuing with multiple macroscopic simulations under different scenarios, and finishing with simulation comparison and report generation.

Aurora RNM is freely available at [2].

REFERENCES

[1] E. Almasri and B. Friedrich. Online offset optimization in urban networks based on cell transmission model. *ITS Hanover*, 2005.
[2] Aurora RNM. http://code.google.com/p/aurorarnm.
[3] C. Buisson, J.-P. Lebacque, and J. B. Lesort. STRADA, a discretized macroscopic model of vehicular traffic flow in complex networks based on the Godunov scheme. *CESA'96 IMACS Multiconference*, 1996.
[4] C. F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research, B*, 28(4):269–287, 1994.
[5] C. F. Daganzo. The cell transmission model II: Network traffic. *Transportation Research, B*, 29(2):79–93, 1995.
[6] W.-H. Lin and D. Ahanotu. Validating the basic cell transmission model on a single freeway link. *Technical Note 95-03*, 1995.
[7] H. K. Lo. A cell-based traffic control formulation: Strategies and benefits of dynamic timing plans. *Transportation Science*, 35(2):148–164, 2001.
[8] L. Munõz, X. Sun, D. Sun, G. Gomes, and R. Horowitz. Methodological calibration of the cell transmission model. *Proc. of the ACC 2004*, pages 798–803, 2004.
[9] Performance Management System (PeMS). http://pems.eecs.berkeley.edu.
[10] Simulation of Urban MObility (SUMO). http://sumo.sourceforge.net.